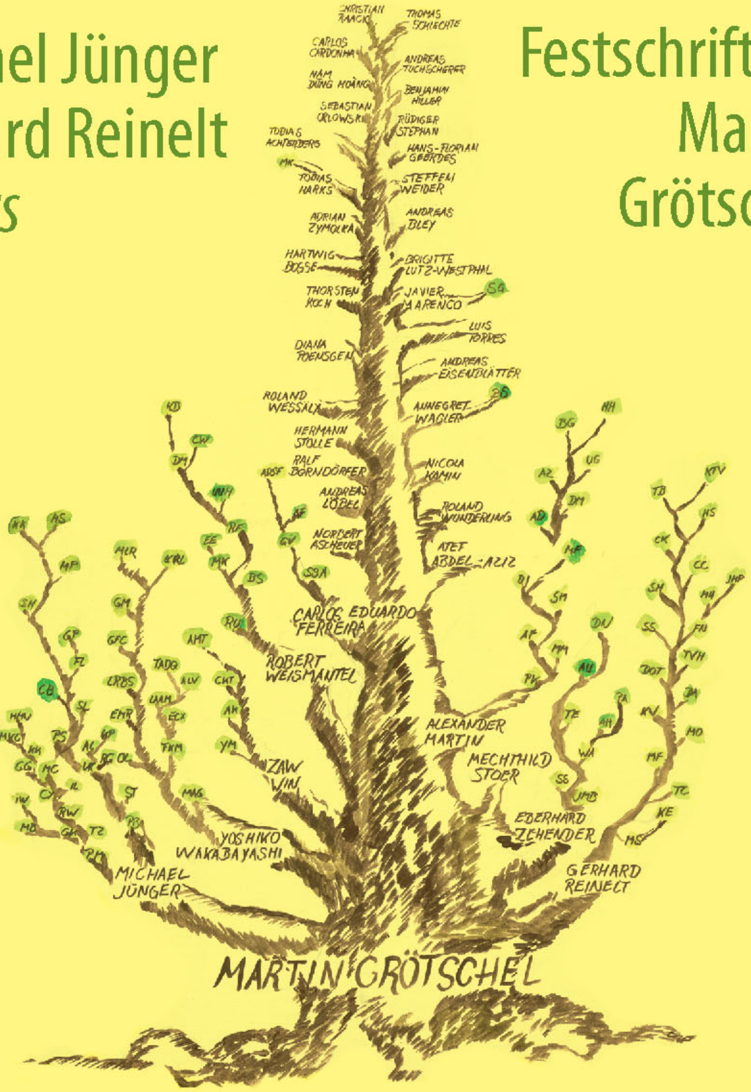
Facets of Combinatorial Optimization

Michael Jünger
Gerhard Reinelt
Editors

Festschrift for Martin Grötschel



Springer

Facets of Combinatorial Optimization

Michael Jünger • Gerhard Reinelt
Editors

Facets of Combinatorial Optimization

Festschrift for
Martin Grötschel

Springer

Editors
Michael Jünger
Dept. of Computer Science
University of Cologne
Cologne, Germany

Gerhard Reinelt
Dept. of Computer Science
University of Heidelberg
Heidelberg, Germany

ISBN 978-3-642-38188-1 ISBN 978-3-642-38189-8 (eBook)
DOI 10.1007/978-3-642-38189-8
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013942544

Mathematics Subject Classification (2010): 90, 90-06, 90C

© Springer-Verlag Berlin Heidelberg 2013
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

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

We dedicate this book to Martin Grötschel
on the occasion of his 65th birthday.

Preface

We received our doctoral degrees in the years 1983 and 1984; we are the oldest doctoral descendants of Martin Grötschel who celebrated his 65th birthday on September 10, 2013.

In the summer of 2011, with this happy occasion still more than two years in the future, we wondered what would be a nice birthday present for Martin. We knew that we were just the first two in a long list of doctoral descendants, but little did we know then how many there were. (Now we do, see Part III.)

We came up with the idea of organizing a celebration with his doctoral descendants and his closest scientific friends. We started serious work on the project during the Oberwolfach Workshop on Combinatorial Optimization in November 2011, where we also decided to organize a colloquium in his honor at the University of Cologne on September 13, 2013, and to edit this book that will be presented to him and all participants during the colloquium.

Many of Martin Grötschel's doctoral descendants still work in research, in and outside academia. When we issued a call for contributions on November 24, 2011, including the sentence "We aim at the highest quality, so please dedicate your best work in honor of Martin Grötschel!", we had an overwhelmingly positive response. The result is Part IV, the core of this book. It is preceded by a personal tribute by the editors to our mentor (Part I), a contribution by a very special "predecessor" (Part II), and the doctoral descendant tree 1983–2012 (Part III).

Cologne, Germany — Michael Jünger
Heidelberg, Germany — Gerhard Reinelt
September 2013

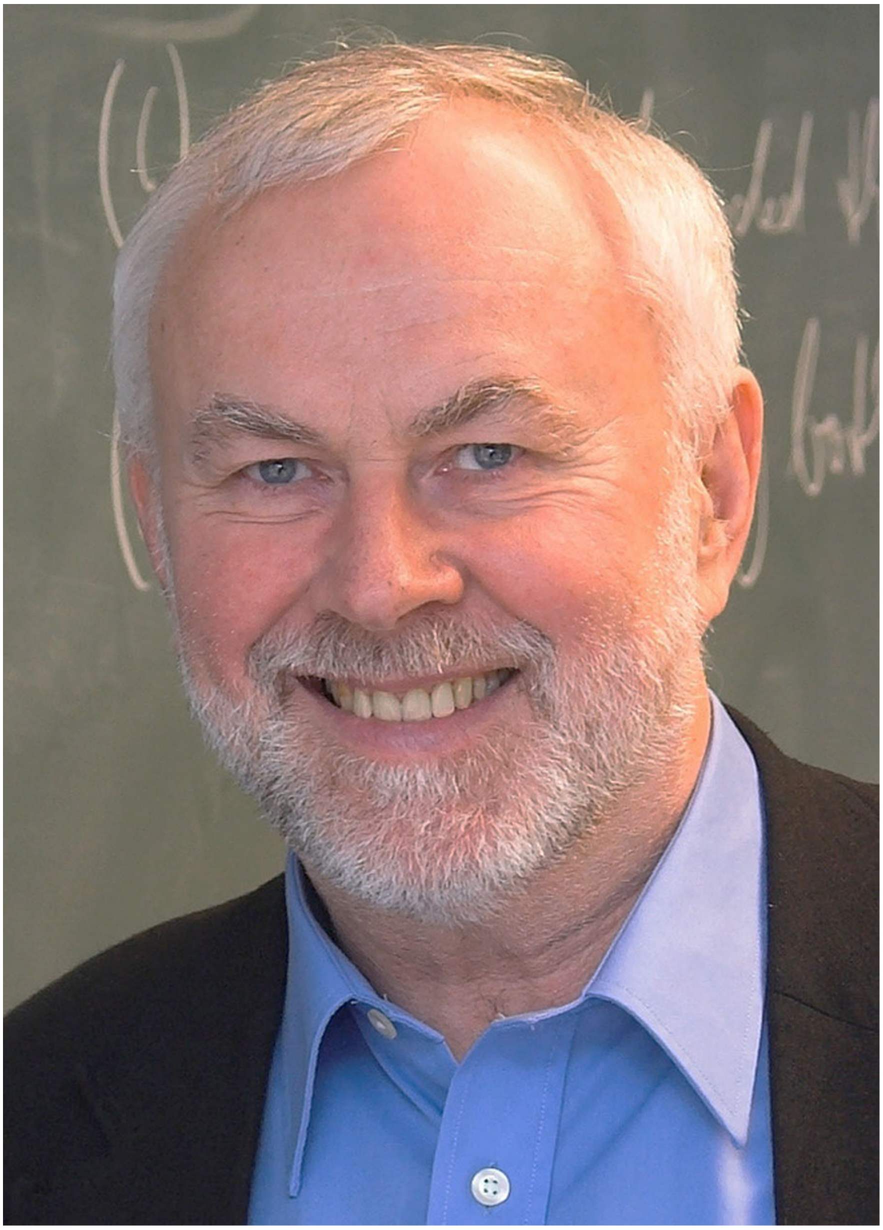
Acknowledgements

Preparing this book and organizing the Festkolloquium have been intertwined tasks that turned out to be much more demanding than we had anticipated when we sketched the ideas about two years ago. Many people helped us. We would like to express our gratitude to

- *Iris Grötschel* for advice, photos, and for keeping the secret,
- *Bernhard Korte* for valuable information on the early days at the Institut für Ökonometrie und Operations Research of the University of Bonn,
- *Dirkje Keiper* and *Simona Schöneweiß* for exploring the archives of the Forschungsinstitut für Diskrete Mathematik at the University of Bonn for us,
- *Teodora Ungureanu* for helping us refresh our memories of the days at the Mathematics Department of the University of Augsburg,
- *Uwe* and *Renate Zimmermann* for turning scans of blurred photographs into reasonable pictures,
- *Bill Pulleyblank* for reading a preliminary version of the first chapter and providing very helpful feedback,
- *Manfred Padberg* for continuous “grandparental” advice before and during the preparation of this book and his immediate agreement to contribute his latest article to this book,
- *Sven Mallach* for his help in typesetting Manfred Padberg’s contribution,
- the doctoral descendants who helped getting the doctoral descendant tree right,
- *Thomas Möllmann* who did not hesitate when we approached him with the demanding task of turning a mathematician’s tree into a real tree,
- the authors of Part II and Part IV, who patiently dealt with our many requests and strict deadlines,
- the anonymous referees who supported us with their expert knowledge and gave valuable advice to the authors,
- *Martin Peters* of Springer Verlag who supported this book project from the very beginning,
- *Ruth Allewelt* of Springer Verlag who accompanied us all the way from the early stages to the final book,
- *Frank Holzwarth* of Springer Verlag who provided “quick LaTeX hacks” whenever needed in the technical editing,
- *Göntje Teuchert* for coordinating the organization of the Festkolloquium, and
- *Martin Gronemann*, *Thomas Lange*, *Sven Mallach*, *Daniel Schmidt*, and *Christiane Spisla* for proofreading the book and for their help in the organization of the Festkolloquium.

Contents

Part I
Martin Grötschel—Activist in Optimization



Martin Grötschel

Breve Curriculum Vitae

Martin Grötschel holds the Chair for Information Technology at the Institute of Mathematics, Technische Universität Berlin, and is the current President of the Konrad-Zuse-Zentrum Berlin (ZIB), where he served as Vice President from 1991 until September 2012. From November 2002 until May 2008 he chaired the DFG Research Center MATHEON "Mathematics for key technologies," a scientific institution involving over 200 applied mathematicians from the three large Berlin universities and two research institutes.

Martin Grötschel's main areas of scientific interest are mathematics, optimization and operations research. His special focus over the last 15 years has been on the design of theoretically and practically efficient algorithms for hard combinatorial optimization problems. He is especially interested in real applications. He has been working with scientists from other disciplines and, in particular, with engineers and economists from the industry to mathematically model challenging problems in their respective fields of expertise. He has contributed to application areas such as telecommunication, chip design and very large scale integration, production planning and flexible manufacturing, logistics, and planning for public transportation systems. He has also spearheaded several projects on electronic information and communication.

Before coming to Berlin in 1991, Martin Grötschel was a Professor of Applied Mathematics at the University of Augsburg (1982–1991) and held several visiting positions abroad. He received his Master's Degree in Mathematics from Bochum University (1973), and his doctoral (1977) and postdoctoral degrees (1981) from Bonn University.

Martin Grötschel's scientific achievements have been recognized with several distinctions including the John von Neumann Theory Prize (2006), the EURO Gold Medal (2004), the Leibniz Prize (1995), the Dantzig Prize (1991), the Beckurts Prize (1990), and the Fulkerson Prize (1982). He is a member of the Berlin-Brandenburg Academy of Sciences and Humanities (BBAW), of acatech, the "German Academy of Sciences and Engineering," and of the German Academy of Sciences, Leopoldina. He is also a Foreign Associate of the National Academy of Engineering, USA and a SIAM Fellow. He has received honorary doctorates from the University of Karlsruhe (2006) and Magdeburg (2008) and the Vietnamese Academy of Science and Technology (2007). He has written and edited 13 books and published more than 150 scientific papers. He has been a member of the editorial boards of 16 scientific journals.

Martin Grötschel has served the academic community in many administrative functions. He was, e.g., on the Executive Committees of the Mathematical Programming Society (MPS) and of the German Mathematical Society (DMV), and in 1993–1994, he was President of the DMV. He has been on the Executive Board of BBAW since 2001 and has been an active member of the scientific advisory boards of various organizations and institutions (FWF-Austrian Science Foundation, University of Vienna, Imperial College London, KTH Stockholm, ILOG, CWI Amsterdam, CMM Santiago de Chile, BICM Beijing).

After having served as a member-at-large of the Executive Committee of the International Mathematical Union (IMU) from 1999 to 2006, Martin Grötschel was elected to be the IMU's Secretary for the period 2007 to 2010. At the IMU General Assembly in Bangalore in August 2010, he was confirmed in this function for the next four years. Moreover, he has been the Chairman of the Executive Board at the Einstein Foundation Berlin since June 2011.

(This is a slightly edited version of the "Short CV" taken from Martin Grötschel's professional homepage at www.zib.de/groetschel, where many more details can be found.)

Martin Grötschel—The Early Years in Bonn and Augsburg

Michael Jünger and Gerhard Reinelt

1 Introduction

Martin Grötschel is one of the most influential mathematicians of our time. He has received numerous honors and serves in a number of key positions in the international mathematical community. A summary is sketched in the preceding short curriculum vitae taken from his professional homepage. Numerous scientific and popular articles recount his achievements and prominent role. These include articles in respected papers such as *DIE ZEIT*, *Financial Times*, *Die Welt*, and *Der Tagesspiegel*. His "scientific facets" will be covered in detail in the rest of this book.

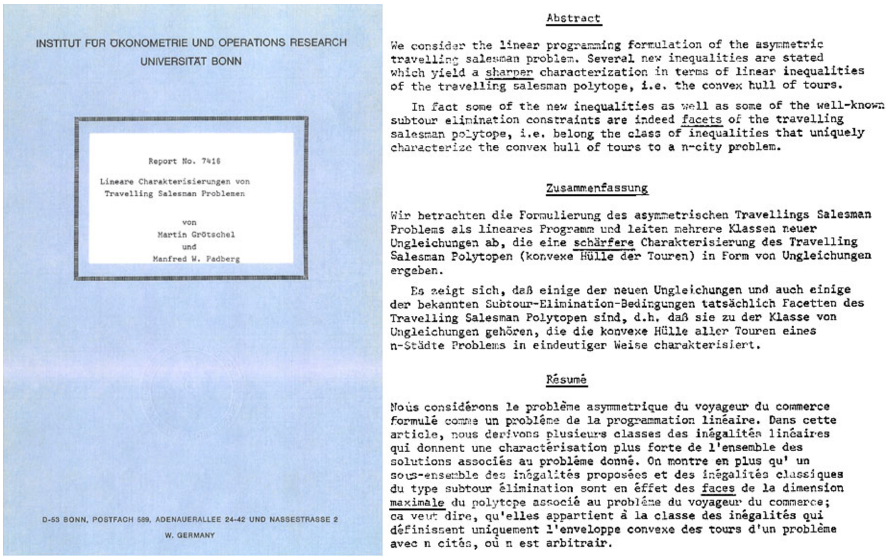
When we first conceived of this chapter, we had a different perspective in mind. As his first doctoral descendants, we can tell a different story, because we were actively involved in his early career. This was the time when he developed from a scientific assistant in Bonn to a full professor in Augsburg. In the following, we recount some of our memories of this time, with no guarantee for either accuracy or completeness. Instead, these are the stories that came to mind when we sat together in Heidelberg and Cologne, preparing this chapter.

2 Bonn

After obtaining his diploma in Mathematics at the University of Bochum in 1973, Martin started his doctoral studies in Mathematics and Economics at the Institut für Ökonometrie und Operations Research of the Rheinische Friedrich-Wilhelms-

M. Jünger (✉)
Institut für Informatik, Universität zu Köln, Albertus-Magnus-Platz, 50923 Cologne, Germany
e-mail: mjuenger@informatik.uni-koeln.de

G. Reinelt
Institut für Informatik, Ruprecht-Karls-Universität Heidelberg, Im Neuenheimer Feld 368, 69120 Heidelberg, Germany
e-mail: gerhard.reinelt@informatik.uni-heidelberg.de



INSTITUT FÜR ÖKONOMETRIE UND OPERATIONS RESEARCH
UNIVERSITÄT BONN

Report No. 7416

Lineare Charakterisierungen von
Travelling Salesman Problemen

von
Martin Grötschel
und
Manfred W. Padberg

D-53 BONN, POSTFACH 589, ADENAUERALLEE 24-42 UND NASSESTRASSE 2
W. GERMANY

Abstract

We consider the linear programming formulation of the asymmetric travelling salesman problem. Several new inequalities are stated which yield a sharper characterization in terms of linear inequalities of the travelling salesman polytope, i.e. the convex hull of tours.

In fact some of the new inequalities as well as some of the well-known subtour elimination constraints are indeed facets of the travelling salesman polytope, i.e. belong the class of inequalities that uniquely characterize the convex hull of tours to a n-city problem.

Zusammenfassung

Wir betrachten die Formulierung des asymmetrischen Travellings Salesman Problems als lineares Programm und leiten mehrere Klassen neuer Ungleichungen ab, die eine schärfere Charakterisierung des Travelling Salesman Polytopen (konvexe Hülle der Touren) in Form von Ungleichungen ergeben.

Es zeigt sich, daß einige der neuen Ungleichungen und auch einige der bekannten Subtour-Elimination-Bedingungen tatsächlich Facetten des Travelling Salesman Polytopen sind, d.h. daß sie zu der Klasse von Ungleichungen gehören, die die konvexe Hülle aller Touren eines n-Städte Problems in eindeutiger Weise charakterisiert.

Résumé

Noùs considérons le problème asymmetrique du voyageur du commerce formulé comme un probléme de la programmation linéaire. Dans cette article, nous derivons plusieurs classes des inégalités linéaires qui donnent une charactérisation plus forte de l'ensemble des solutions associés au probléme donné. On montre en plus qu' un sous-ensemble des inégalités proposées et des inégalités classiques du type subtour élimination sont en éffet des faces de la dimension maximale du polytcpe associé au probléme du voyageur du commerce; ca veut dire, qu'elles appartient à la classe des inégalités qui définissent uniquement l'enveloppe convexe des tours d'un problème avec n cités, où n est arbitrair.

Fig. 1 Report No. 7416, the first outcome of the collaboration of Martin Grötschel and Manfred Padberg: Cover (*left*) and abstract in English, German, and French (*right*)

Universität Bonn, which would prove to be an excellent choice. The Operations Research Department, under the leadership of Bernhard Korte, quickly developed into an internationally renowned center for Mathematics of Operations Research. Prominent researchers came for colloquium presentations, a visit of a couple of weeks, a few months, or even an entire sabbatical term. Bernhard Korte created a stimulating atmosphere in which young scientists could grow and bloom. He also had good taste when it came to selecting his scientific assistants, and Martin was one of them. (They first met at a summer school for members of the Deutsche Studienstiftung, where Bernhard Korte gave a series of lectures and Martin was one of the happy students to receive such a renowned stipend.)

We, the authors, were still in high school in 1973, too young to be part of what was going on in Bonn in 1973 and 1974. We entered the scene as students taking Operations Research as a minor in our studies of Computer Science in 1975, but this was early enough to know a little bit about the two preceding years.

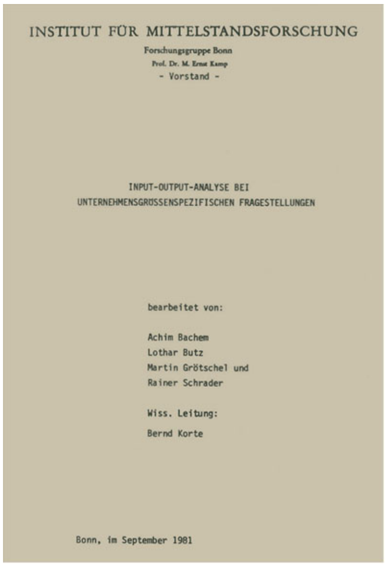
In 1974, Manfred Padberg spent six months at the OR Department, and one day, young Martin (no beard yet) appeared at the door to his office and asked: "Was ist eigentlich Kombinatorische Optimierung?" ("What is Combinatorial Optimization anyway?"), to which Manfred replied: "Come in." Manfred soon got Martin working on the traveling salesman problem. Everybody in Combinatorial Optimization knows what followed, but the reader may not know the first technical report on this unique and fruitful collaboration, namely Report No. 7416 of the Institut für Ökonometrie und Operations Research's famous blue preprint series, the cover and abstract of which are shown in Fig. 1. (For the published version, see [1].)

Our own early experience in the Bonn OR Department was dominated by our fascination with Bernhard Korte's unique, unusual, and entertaining teaching style. At some point, our minor subject of study interested us so much that we applied for student assistant positions ("Studentische Hilfskräfte"). The department had a data station including all the high-tech equipment that Computer Science students longed for back then. This included a punched card reader not shared by a hundred but only a dozen or so users, (comparatively) quick results from a noisy line printer, et cetera. Initially, we really had nothing to do with Martin. Our assignments, such as implementing basic computer routines like line search in nonlinear programming, came from Achim Bachem, another young scientific assistant who has also had an impressive career, but that is a different story. During this period, we tried to follow the seminars and colloquia offered by the department, sitting in the last row, without much success. What we clearly remember is that Martin was a critical member of the team, with pointed questions for those who gave presentations. This was our first encounter.

A later encounter was less enjoyable: One of our duties was to run around a table with piles of paper sheets, assemble them into scientific preprints of the blue Bonn series, and staple them. One issue we remember very clearly: Report No. 7770 (Ref. [2]) seemed to contain nothing but numbers. Later we realized that this was the distance matrix of the famous TSP instance that would later become known as "gr120.tsp," a 120-city instance of German towns, the distances between which had been taken from the ADAC (Allgemeiner Deutscher Automobil-Club) road map. Martin's optimum solution of the TSP for this set of cities set a new world record at the time, with the help of the aforementioned data station of the IBM mainframe at the University of Bonn. We later learned that it more or less went like this: Michael Hahmann (aka James) let IBM's LP-solver MPSX run on a relaxation and gave the output to Martin, who plotted it at home that night (armed with only a pencil and an eraser), did the separation by hand, and handed some new cutting planes (facet defining for the TSP polytope, of course) to James the next morning. James solved the stronger relaxation during the day, gave the fractional solution to Martin in the evening, ..., and, lo and behold, after 13 iterations (days) an optimum solution was found.

Then the real excitement started. In 1980, the department acquired an IBM 4331, and for the first time we could work at an IBM 3270 terminal in time sharing mode (TSO: "time sharing option" in IBM jargon). Prior to this, computing was done in batch mode. We had by now received our master/diploma degrees, and we had been promoted to "Wissenschaftliche Hilfskräfte," which doubled our (still meager) salaries, enough to be less dependent on our parents' support. In 1981, we had already written our first paper in graph theory, jointly with and under the supervision of Bill Pulleyblank, who was one of the department's visiting scientists then, and whose main collaboration at the same time was with Martin. We remember when Martin said "We have found plenty of new TSP facets," referring to the clique tree inequalities discovered in the course of this collaboration.

One day in 1981, Martin announced: "Jetzt habilitiere ich mich." ("Now I'll get my habilitation.") This postdoctoral degree was a prerequisite for a professorship



INSTITUT FÜR MITTELSTANDSFORSCHUNG
Forschungsgruppe Bonn
Prof. Dr. M. Ernst Kamp
- Vorstand -

INPUT-OUTPUT-ANALYSE BEI
UNTERNEHMENSGRÖSSENSPEZIFISCHEN FRAGESTELLUNGEN

bearbeitet von:

Achim Bachem
Lothar Butz
Martin Grötschel und
Rainer Schrader

Wiss. Leitung:
Bernd Korte

Bonn, im September 1981

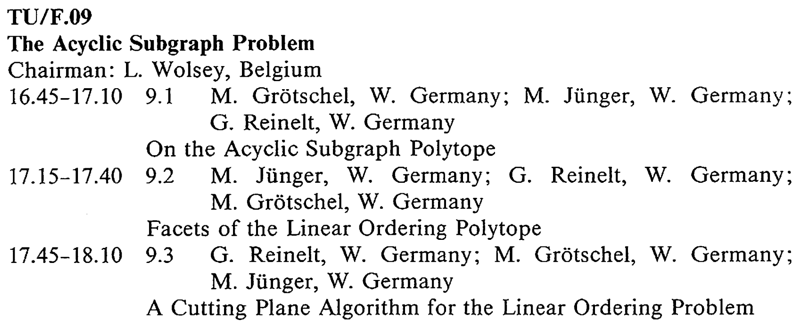
Fig. 2 A little book with an influential statement

then. He produced a monograph within a few weeks, and he got his habilitation soon afterwards.

Also in 1981, four of the department's scientific assistants had completed the little book [3], see Fig. 2, where the following statement is made about the triangulation problem for input-output tables: "Selbst beliebig verfeinerte Verfahrenstechniken werden vermutlich nicht dazu führen, daß man das Triangulationsproblem für wünschenswerte Größenordnungen von Input-Output-Matrizen optimal lösen kann." ("Even arbitrarily refined procedures are not likely to lead to the solvability of the triangulation problem for input-output matrices of desirable sizes.")

This economics problem is equivalent to the well known linear ordering problem. Martin gave us the integer programming relaxation using 3-cycle inequalities and let us hunt for facets of the linear ordering polytope (and the related acyclic subdigraph polytope), but apparently, he didn't expect much, stating simply: "It's not easy." Nevertheless, we came up with new facet classes rather quickly and he was impressed. Part of our success was, of course, due to the fact that, to us, the use of software assistance in the search came rather naturally. By chance, we had written a computer program that implemented Jack Edmonds' all-integer version of Gaussian elimination with high-precision arithmetic (an early assignment from Achim Bachem). Therefore we were able to accurately determine the rank of a 0/1-matrix, see also [4]. We embedded this into a user-friendly software package that, given an inequality, gave one of the following replies: "Sorry, the inequality is not valid." or "The inequality is valid and induces a face of dimension r," or "Congratulations, you have found a facet!".

This allowed experimentation and provided many conjectures of generalizations to facet classes. We also wrote branch&cut software that we called a "cutting plane algorithm," avoiding any reference to the "dirty" enumeration part of the software.



TU/F.09
The Acyclic Subgraph Problem
Chairman: L. Wolsey, Belgium

16.45–17.10	9.1	M. Grötschel, W. Germany; M. Jünger, W. Germany; G. Reinelt, W. Germany On the Acyclic Subgraph Polytope
17.15–17.40	9.2	M. Jünger, W. Germany; G. Reinelt, W. Germany; M. Grötschel, W. Germany Facets of the Linear Ordering Polytope
17.45–18.10	9.3	G. Reinelt, W. Germany; M. Grötschel, W. Germany; M. Jünger, W. Germany A Cutting Plane Algorithm for the Linear Ordering Problem

Fig. 3 Talks of Martin Grötschel, Michael Jünger, and Gerhard Reinelt at the ISMP 1982 in Bonn

The technical term “branch&cut” was coined much later by Manfred Padberg and Giovanni Rinaldi.

On the IBM 4331, we used the programming language PL/I and IBM’s MPSX software to solve the linear programming relaxations. And we found we could indeed solve these triangulation instances that had seemed so hopeless just a year earlier. This became our ticket to Martin’s world.

We were confident that we had produced enough material for a scientific paper, but Martin said “One paper? No. Three!” The paper production process was just amazing for us: We would sit in Martin’s office, equipped with a notepad, a pencil, an eraser, a pair of scissors, and a bottle of glue. Days of writing, cutting and pasting (in the original sense) followed, and the result was given to Helga Grimm, the department secretary, who typed it up on an IBM Selectric “golfball” typewriter. Eventually, these three papers ended up as references [5, 6], and [7]. They were also the basis of an entire session chaired by Laurence Wolsey on August 24, 1982, at the International Symposium on Mathematical Programming ISMP 1982 in Bonn, see Fig. 3.

This date was a turning point for the two of us: These were the first scientific talks we gave. Later, we produced another paper with computational studies for economists, the first of that series to appear in print [8].

Of course, for Martin, this was only one of the many findings in a period of great scientific creativity and success. Most notably, at that same ISMP 1982, he received the prestigious Fulkerson Prize, jointly with László Lovász and Alexander Schrijver, for the groundbreaking work on the Ellipsoid Method and its consequences in Combinatorial Optimization [9], see Fig. 4.

In late 1982, he accepted a chair in Applied Mathematics at the University of Augsburg’s newly founded Mathematics Department.

3 Augsburg

As a result of a number of wise decisions made by an external committee, the newly founded Mathematics Department at the University of Augsburg consisted of an excellent group of active young professors and their scientific assistants. We were

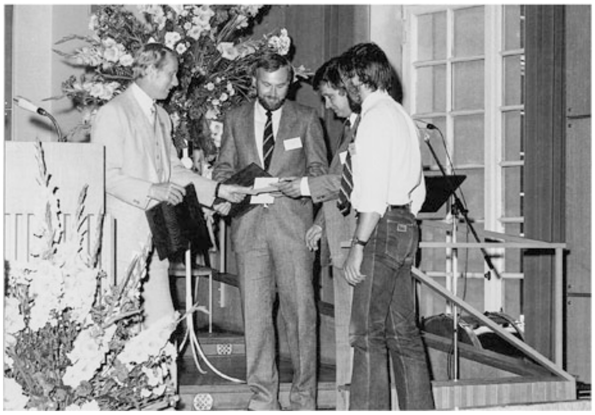


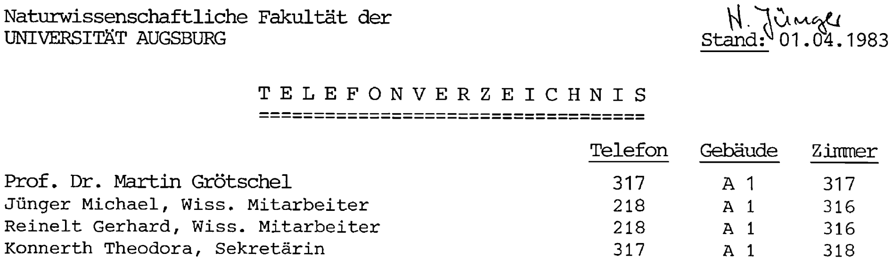
Fig. 4 Ron Graham presents the Fulkerson Prize to Martin Grötschel, László Lovász, and Alexander Schrijver at the ISMP 1982 in Bonn

lucky to be among the latter almost from the very beginning. When we followed Martin to Augsburg in early 1983, the student body consisted of about 45 beginners studying mathematics in their first semester. It is hard to imagine better studying conditions. In February of 1983, there was a party in the Rosenaustraße, where four scientific assistants shared a big apartment. Almost the entire Science Faculty (math only then) from the dean to the students showed up and fitted in.

The chair of Martin consisted of four people, see the first few lines of the faculty telephone directory of April 1, 1983, in Fig. 5.

"Konnerth Theodora" is now "Ungureanu Teodora." (Bavarians have the strange habit of listing the first name after the last name—for ordinary people, not for professors.) Teodora was the perfect addition for the one senior and the two junior scientists of the group. We worked hard in perfect harmony, and had a lot of fun. The following offers a glimpse into the atmosphere of the time: Both Martin and Teodora were unhappy with their weight: Teodora a bit too slim, Martin a bit overweight. So one day they agreed to work on this and made a contract to keep their weight sum constant.

While we were working on our doctoral dissertations, Martin's main scientific focus was the "GLS book," which would later become *Geometric Algorithms and Combinatorial Optimization* [10]. Consequently, among the many guests we had,


Naturwissenschaftliche Fakultät der
UNIVERSITÄT AUGSBURG

M. Jünger
Stand: 01.04.1983

TELEFONVERZEICHNIS

	Telefon	Gebäude	Zimmer
Prof. Dr. Martin Grötschel	317	A 1	317
Jünger Michael, Wiss. Mitarbeiter	218	A 1	316
Reinelt Gerhard, Wiss. Mitarbeiter	218	A 1	316
Konnerth Theodora, Sekretärin	317	A 1	318

Fig. 5 Excerpt of the Telephone Directory of the Science Faculty of the University of Augsburg dated April 1, 1983

two were regularly seen in Martin's office: Láci Lovász and Lex Schrijver, who would often come for quiet working sessions with Martin.

The lack of decent computer equipment at the University of Augsburg presented a problem. Martin fought hard to improve the situation, but initially we had no choice but to use the computers at the University of Bonn and at the Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt (DFVLR), now the Deutsches Zentrum für Luft- und Raumfahrt (DLR). The latter required frequent trips to Oberpfaffenhofen near Munich.

In addition to a computational platform, we wanted decent computer typesetting. TEX already existed in its second (and final) generation, but there were no easy to use custom distributions. So we ordered a tape and adapted the software for the IBM mainframe in Oberpfaffenhofen, and then wrote a driver for the big vector plotter that we used as an output device. This caught the attention of Hans-Martin Wacker, the director of the computing center, who asked us to make the installation available to all users, and we did. But we became more and more tired of all the traveling and cutting pages from plotter output, so Martin ordered a CADMUS, a MUNIX workstation from Periphere Computer-Systeme (PCS) in Munich, at a price of about 50 000 German marks. ("MUNIX" instead of "UNIX" is not a typo!) It was delivered in October 1983.

We still needed a decent DIN-A4 laser printer, and there was not much choice then. The Canon LBP-10 (the first desktop laser printer in the world) was the machine of choice, but its price (roughly 35 000 marks) significantly exceeded the chair's budget, so we talked the university computing center into buying one. We made sure it was ordered from PCS, and when it was delivered, it turned out that "unfortunately" its interface was only compatible with our CADMUS, so it ended up in our office.

The tiny CADMUS system (Motorola 68000 processor, 512 kilobytes of main memory, 60 megabytes of disk space) became our research workhorse. Again, we implemented TEX and wrote a new output driver, now with convenient DIN-A4 output, and it soon became the typesetting machine for many more mathematicians at the Augsburg department who were tired of the then still-common typewriters.

This machinery, plus a home-made TEX macro package for books, not only allowed our two doctoral dissertations to be typeset in TEX, but also made possible the first Springer book to have its manuscript submitted in TEX, namely the groundbreaking "GLS book" [10]. (LATEX came 2 years later.)

Martin took great care in teaching optimization courses; some of his lecture notes from the first few years of his tenure in Augsburg are still in circulation today. In addition to basic mathematics courses, he covered the state of the art in linear, nonlinear, and discrete optimization. His lectures were spiced up with reports on applications and software demonstrations. As an enthusiastic teacher, he motivated many students to specialize in Combinatorial Optimization.

In 1983, Yoshiko Wakabayashi, with whom Martin had already worked in São Paulo and Bonn, came to Augsburg in order to work on her doctoral thesis concerning the aggregation of binary relations, with applications to clustering. She received her doctoral degree in 1986. In 1984, Karl-Heinz Borgwardt joined the faculty as an

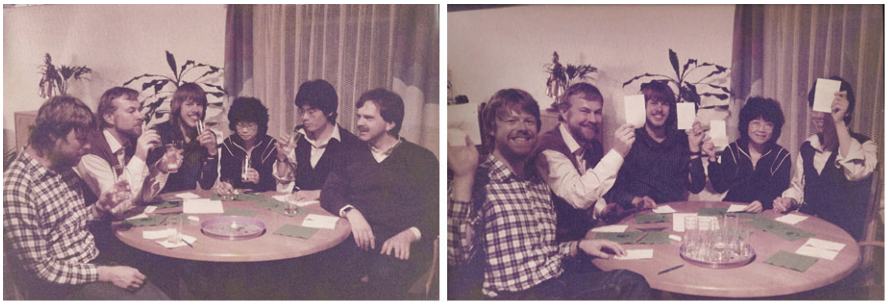


Fig. 6 Beer tasting at the home of Martin and Iris Grötschel: Tasting phase (*left*) and scoring phase (*right*), repeated 45 times. Sitting from left to right: Michael Jünger, Martin Grötschel, Jeff Edmonds, Yoshiko Wakabayashi, Mario Sakumoto, Gerhard Reinelt

associate professor of Mathematical Optimization. Zaw Win came as a doctoral student from Yangon, Myanmar, and Martin got him working on the Windy Postman Problem, a core problem in vehicle routing. His doctoral degree was conferred in 1987. Martin also supervised Eberhard Zehendner, a doctoral student of Computer Science in our faculty, but outside our working group. He also received his doctoral degree in 1987.

The atmosphere in our working group was not always entirely serious. The two pictures in Fig. 6 show a legendary beer tasting party at the home of Martin and Iris Grötschel. This has been covered in print before, but here is the full truth. With the ability to triangulate input-output tables, and knowing that marketing people considered "ranking by aggregation of individual preferences," which amounts to this very problem, we decided to determine a ranking of 10 Pilsner type beers. Pairwise comparison implied that we had to do 45 comparisons, in each of which the two glasses of each participant were (partially) filled with two different beers unknown to the drinkers. This meant 90 little portions of beer for each participant. We also had two cards each, one of which we simultaneously raised in order to indicate our preferences. See Fig. 6 for the tasting and the scoring phases.

In each round, one of us acted as the waiter/waitress; in the photographed round, the second author has just served on the left picture, and he is not on the right picture because he is taking scores. Figure 7 shows scores Martin took when he was the waiter.

Part of the magic of the Augsburg years was certainly the fact that this was one of many occasions when we met at the Grötschels' house. There were many parties with many visitors, in an atmosphere of warm hospitality.

Martin is well known as a driving force in applying state-of-the-art mathematics and computer science in industry and commerce and took on a leading position in such endeavors in the field of Combinatorial Optimization. This emerged during his time in Augsburg, where his first attempt was to become involved in the routing of the city garbage collection. Many letters were written and meetings were held, but in the end, no project was implemented due to conflicting interests among the city politicians. In retrospect, knowing that his big success in planning transport

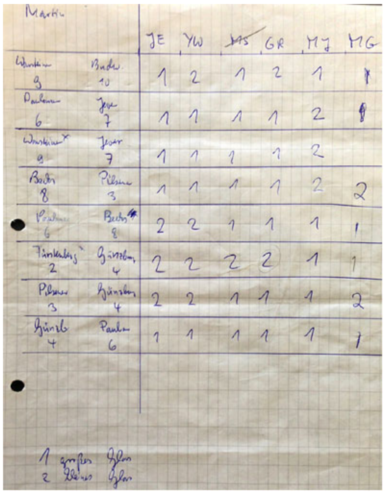


Fig. 7 Martin's beer tasting scoring sheet

Martin

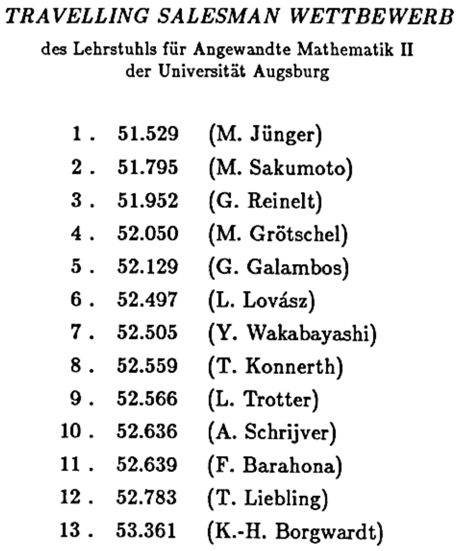
		JE	YW	~~MS~~	GR	MJ	MG
Warsteiner 9	Bader 10	1	2	1	2	1	1
Paulaner 6	Jever 7	1	1	1	1	2	1
Warsteiner* 9	Jever 7	1	1	1	1	2	
Becks 8	Pilsner 3	1	1	1	1	2	2
Paulaner 6	Becks* 8	2	2	1	1	1	1
Fürstenberg* 2	Günzburger 4	2	2	2	2	1	1
Pilsner 3	Günzburger 4	2	2	1	1	1	2
Günzb. 4	Paulaner 6	1	1	1	1	1	1

1 großes Glas
2 kleines Glas

for handicapped people in Berlin started about a decade later, we are confident that the Augsburg garbage project would have been implemented and successfully completed, if Martin had been more experienced in dealing with politicians. He certainly is now, but back then, he was a beginner. This would soon change, as we shall see later.

In 1983, Martin had completed a joint paper with Francisco Barahona and Ridha Mahjoub [11], building on the two other authors' previous related work. (Teodora has fond memories of her first assignment on her IBM Selectric.) In the mid eighties, we started serious work on the Maximum Cut Problem, together with Francisco Barahona, with whom we had established a joint German-Chilean project funded by the Volkswagen Foundation. Two applications drove us, namely ground state computations of spin glasses in solid state physics and via minimization in VLSI layout design. We still could not run branch&cut software on our university's computers; instead we used IBM mainframes at the Kernforschungsanlage Jülich (now the Forschungszentrum Jülich) and the University of Bonn, mainly because they had the LP solver MPSX installed (and let us use it).

We can't resist a digression on our university computer technology back in the day: We had no network connection and no email. Despite Martin's efforts, the computing center did not see a point in having such a thing. (They considered email a passing fad that would soon be forgotten.) So data was transferred on the big reels of tape that you see spinning whenever computers are shown in old movies. Lacking space in our temporary department home in the Memminger Straße, the computing center's tape reader was located in a nearby high school, and we had a "Studentische Hilfskraft," one of whose jobs it was to carry reels of tape to the high school and make sure that the data eventually made its way to our CADMUS. Incidentally, the studentische Hilfskraft in charge of this was Petra Mutzel, who would later become one of Martin's doctoral grandchildren.



TRAVELLING SALESMAN WETTBEWERB

des Lehrstuhls für Angewandte Mathematik II
der Universität Augsburg

1 .	51.529	(M. Jünger)
2 .	51.795	(M. Sakumoto)
3 .	51.952	(G. Reinelt)
4 .	52.050	(M. Grötschel)
5 .	52.129	(G. Galambos)
6 .	52.497	(L. Lovász)
7 .	52.505	(Y. Wakabayashi)
8 .	52.559	(T. Konnerth)
9 .	52.566	(L. Trotter)
10 .	52.636	(A. Schrijver)
11 .	52.639	(F. Barahona)
12 .	52.783	(T. Liebling)
13 .	53.361	(K.-H. Borgwardt)

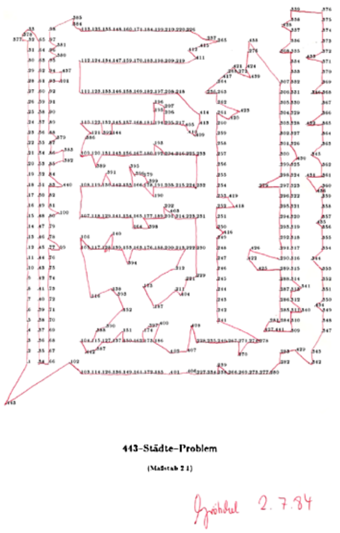


Fig. 8 TSP contest: Ranking by tour length (*left*), Martin Grötschel's tour (*right*)

Considering Martin's achievements, then and later in Berlin, it is clear that he not only has capabilities that go far beyond the normal, but he is also a person who enjoys challenges of all kinds, takes up new duties and drives new agendas, in science, and also in scientific administration. His extraordinary gifts as a scientist had already become apparent before he came to Augsburg; the first glimpses of his organizational talents emerged in his role as co-organizer of the ISMP 1982 in Bonn. But in Augsburg, he not only kept up his high scientific productivity, but also spearheaded the successful development of the new Mathematics Department. At the age of 35, he became its director and Dean of the Science Faculty.

All this requires a competitive character, and Martin always strove to be the best. One of the few examples where we (the authors) could beat him was a TSP competition we organized for members and visitors to our working group. It simply consisted of finding a good tour for drilling 443 holes in a printed circuit board owned by Martin. (It had been our task to determine the coordinates of the holes by hand. After correction of a mistake, this resulted later in the well-known TSP instance "pcb442.tsp".) Figure 8 shows the results of the competition, along with Martin's tour.

A second activity in which we could soundly beat him was computer typing. Martin was then (is still?) a very slow typist. We teased him by telling him the password for his account was "Handlungsreisender," knowing that he hated the German word for "traveling salesman." He didn't know that the first 8 letters were sufficient, and we grinned behind his back when we watched him slowly entering this long horrible word.

We know of no third such discipline. We are grateful that he never tried to convince us to compete with him in sports; we would have had no chance whatsoever.

Martin always made sure that we had enough money for traveling. In combination with the many visitors to the department this exposed us to the scientific community from the very beginning of our own productivity. In March 1986, he took the first author with him to a conference on "Computational Issues in Combinatorial Optimization" in Capri, Italy. After our presentation on spin glass ground state computations and VLSI via minimization, we were approached by Michel Balinski, who invited us to write an article for Operations Research, and this resulted in [12]. At this event, a young Italian approached the first author and introduced himself with the simple words: "My name is Giovanni Rinaldi, and I am the Italian version of Gerhard Reinelt." This marked the beginning of a life-long friendship with all three of us.

Via Minimization was the entry point into our first serious industry cooperation. Martin established valuable contacts with SIEMENS in Munich, where we focused on VLSI layout design. Although working for profit, the SIEMENS team maintained high scientific standards. This was ensured by scientists like Ulrich Lauther, Michael Kolonko, and Michael Hofmeister, who made sure that the best students were hired after graduation. There were regular meetings; we would visit the team in Munich, and they would come for colloquium talks in Augsburg. In addition, there were workshops where theory met practice, so this was an ideal starting point for our first excursion from the "ivory tower" into the "real world." We wrote a technical report on via minimization, mailed it to our scientific friends, but did not yet bother to submit it to a journal. We were surprised when we received a letter from East Germany in which we were informed that our "submission" had been accepted for publication in the *Zeitschrift für Angewandte Mathematik und Mechanik*! We gladly accepted, and the deal even included a complimentary Russian abstract, see [13].

In 1985, Gábor Galambos came from Szeged, Hungary, for a semester and worked on bin packing. In 1986, Klaus Truemper came from Dallas, Texas, as a Humboldt senior scientist, worked with Martin on cycles in binary matroids, and gave a lecture series on matroids. Also in 1986, Heinrich Puschmann came from Santiago de Chile and worked on transporting different kinds of oil in the same pipeline. In 1987, Leslie Trotter came as a Humboldt senior scientist from Cornell University, and Günter M. Ziegler joined us as a scientific assistant directly after receiving his Ph.D. at MIT. In 1988, Enrique Benavent came from Valencia, Spain, and worked on capacitated arc routing.

A distinctive feature of the mathematics curriculum in Augsburg was the inclusion of practical training in industry and commerce. This required close contacts between the department's professors and local industry. One day we went with a group of students to visit the personal computer production at SIEMENS Augsburg, where we saw the POSALUX drilling machine for printed circuit boards. While it was explained to us, Martin immediately realized that minimizing production time essentially boils down to solving a sequence of instances of the traveling salesman problem. So he boldly declared "We can do better." The answer was "No way, the machine's control software already includes an optimization module." But, of course, Martin insisted, and an agreement was reached that we would receive a tape

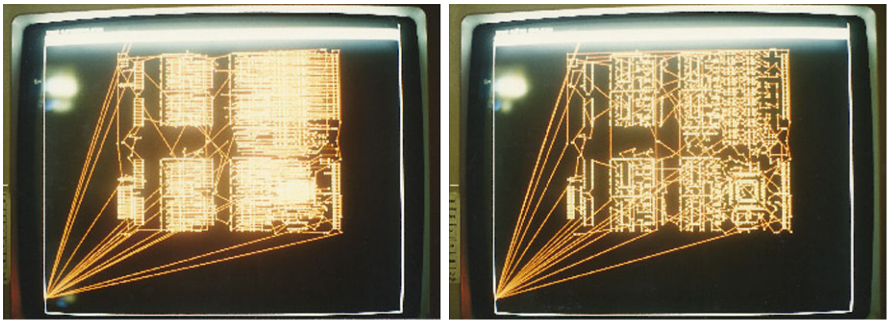


Fig. 9 The paths the POSALUX driller would take with the original control (*left*) and the improved control (*right*)

with the data of various printed circuit boards, and if we managed to get improvements of at least 5 %, we'd be invited to the *Eckestuben*, the only Augsburg restaurant with a star rating.

Meanwhile, our dear old CADMUS had been replaced by a network of SUN workstations, and Martin had succeeded in making email available at the University of Augsburg. Needless to say, we gained much more than 5 %, see Fig. 9 for the original and the improved control of the machine, photographed from our new SUN workstations, i.e., old-fashioned screenshots.

In addition, we were equally successful in designing a better control of a plotter for the printed circuit board wiring. Our experimental study can be found in [14]. So we were not only invited for a nice dinner at the *Eckestuben* (where our industry partners could not help smiling when they learned about university professors' salaries), but also to provide new software that would henceforth be used in the production at SIEMENS Augsburg.

In 1988, Giovanni Rinaldi came to work with us on maximum cut. Initially, Giovanni insisted on FORTRAN as the programming language, but eventually we convinced him that C would be a better choice. Bob Bixby was around, writing a C program, internally called BIXOPT, for the solution of linear programming problems. (BIXOPT later became LOPT, then PLEXUS, and finally CPLEX.) So the late eighties marked the starting point of our first branch&cut framework in C.

A problem with Bob's new LP software was the lack of an implementation of the dual simplex algorithm. The package contained only a two-phase primal method. But in branch&cut, most of the time, dual feasible bases are available. So we asked Bob about this, and his response was: "Transpose the matrix." We had no choice but do this, so the first version of this branch&cut framework contained a simulation of the dual simplex algorithm. But we kept insisting, of course, and eventually, Bob gave in. Not only could we replace our simulation with the real thing; his implementation of the dual simplex algorithm also turned out to be very successful on various LP benchmarks, and in his talks Bob was always sure to get a smile out of us when he told the story of its origin.

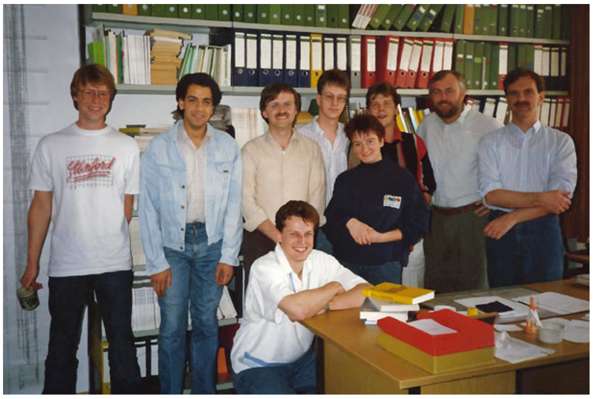


Fig. 10 Martin Grötschel's group in Augsburg 1989, from left to right: Michael Jünger, Atef Abdel-Aziz Abdel-Hamid, Karl-Heinz Borgwardt, Alexander Martin, Günter M. Ziegler, Doris Zepf, Robert Weismantel, Martin Grötschel, Gerhard Reinelt

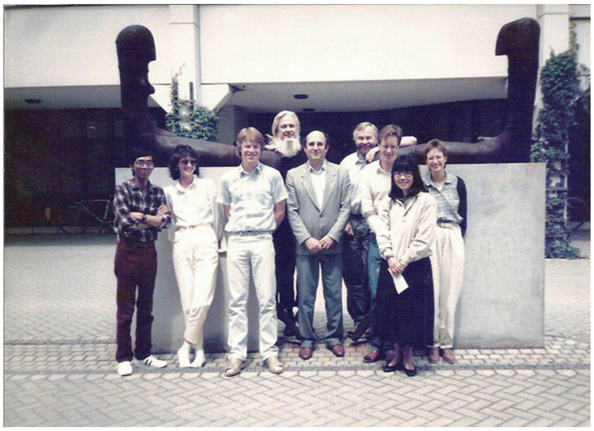


Fig. 11 Martin Grötschel's group in Augsburg with visitors 1989, from left to right: Zaw Win, Teodora Ungureanu, Michael Jünger, Jack Edmonds, Jean Fonlupt, Martin Grötschel, Günter M. Ziegler, Yoshiko Wakabayashi, Mechthild Stoer

With third-party funds from the German Science Foundation (DFG) in the context of a priority program on applied optimization and control, along with various other sources, the group continued to grow dynamically. Our first Augsburg students had graduated and joined the various activities: Mechthild Stoer, Robert Weismantel, Alexander Martin, Doris Zepf (now Tesch) and Petra Bauer. Atef Abdel-Aziz Abdel-Hamid joined us as a doctoral student from Egypt. The pictures shown in Fig. 10 and Fig. 11 were both taken in 1989, the second after we had moved to our new building on the new campus of the University of Augsburg in September 1989.

Martin had no difficulties supervising this sizeable group of fresh doctoral students. Unlike his previous doctoral students, they had been under his influence from the beginning of their mathematical studies, and had already embraced his special mixture of mathematics and applications.

Mechthild Stoer worked on the design of reliable networks, a joint project with Clyde Monma of AT&T Bell Labs. Robert Weismantel and Alexander Martin had already worked for SIEMENS in Munich as student interns before receiving their

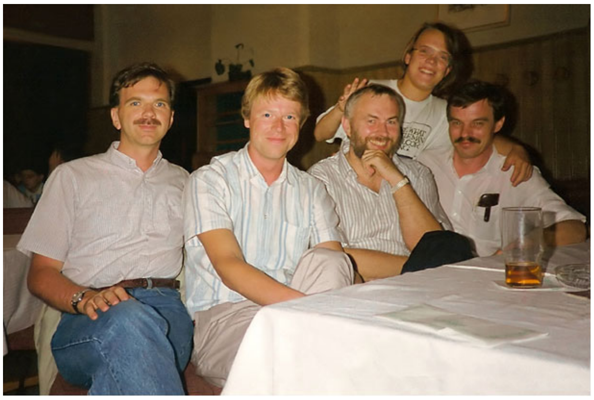


Fig. 12 Three scientific generations in *Auerbachs Keller* 1989, from left to right: Gerhard Reinelt, Michael Jünger, Martin Grötschel, Marc Oliver Padberg, Manfred Padberg

diplomas. Consequently, their doctoral thesis subjects were in VLSI layout design: placement and routing.

In July 1989, on the occasion of the 14th IFIP Conference on System Modelling and Optimization, there was a memorable evening in *Auerbachs Keller* in Leipzig (immortalized in Goethe's Faust and still behind the "iron curtain" back then), where the authors enjoyed a number of beers with their doctoral father Martin Grötschel and their (inofficial) doctoral grandfather Manfred Padberg who came as a Humboldt senior scientist, see Fig. 12. (The latter certainly claims grandfathership!)

In 1990, Peter Gritzmann joined the faculty as an associate professor, which strengthened the sizeable group of combinatorial optimizers in Augsburg even further. The group of doctoral students grew again, when Carlos Ferreira came from Brazil.

The last applied project we did with Martin involved robot control in cooperation with the Forschungsinstitut für Anwendungsorientierte Wissensverarbeitung (FAW) in Ulm. We modeled the task as a shortest Hamiltonian path problem with precedence constraints, and the work in this area would later turn out to be useful for various further applications. The project started in 1990, and with Norbert Ascheuer we got another original first-generation Augsburg student on board. It ended in 1991, by which time both authors had already left the University of Augsburg for professorships in Paderborn and Heidelberg.

But 1991 was also the year Martin left Augsburg for Berlin, taking all his remaining scientific assistants and doctoral students with him, including the freshly graduated students Ralf Borndörfer and Andreas Löbel. In collaboration with his colleagues Karl-Heinz Hoffmann and Hans Georg Bock, he had tried hard to establish working conditions in Augsburg that would reflect the success and growth of the application-oriented working groups of the Mathematics Department. An article in the *Augsburger Allgemeine* from March 2, 1991, describes the efforts to found a new institute called the "Institut für mathematische Optimierung und Steuerung (IMOS)" at the University of Augsburg. Yet this initiative (and formal application) sparked little or no interest among the authorities in the state capital, Munich. Martin is cited as having said "Noch diesen Monat oder ich gehe." ("Either it happens this month or I'm going."), though he still held out hope for at least a provisional start of such an institute near the university campus: "Wir könnten noch heuer anfangen." ("We could still start this year.") They couldn't, so they left Augsburg, as did Peter Gritzmann in the same year.

This was the end of an exceptionally dynamic and productive period for application-oriented mathematics in Augsburg. Subsequently the old and new professors spread out, and the field took off, especially with regard to Combinatorial Optimization, where Berlin is now one of the world's leading centers.

4 Conclusion

We were very lucky for the time we shared! Thanks, dear Martin!

References

1. Grötschel, M., Padberg, M.W.: Lineare Charakterisierungen von Travelling Salesman Problemen. Z. Oper.-Res. **21**, 33–64 (1977)
2. Grötschel, M.: On the symmetric travelling salesman problem: solution of a 120-city problem. Math. Program. Stud. **12**, 61–77 (1980)
3. Grötschel, M., Bachem, A., Butz, L., Schrader, R.: Input-Output-Analyse bei unternehmensgrößenspezifischen Fragestellungen. Institut für Mittelstandsforschung, Bonn (1981)
4. Padberg, M.W.: Facets and rank of integer polyhedra. In: Facets of Combinatorial Optimization: Festschrift for Martin Grötschel. Springer, Berlin (2013)
5. Grötschel, M., Jünger, M., Reinelt, G.: On the acyclic subgraph polytope. Math. Program. **33**, 28–42 (1985)
6. Grötschel, M., Jünger, M., Reinelt, G.: Facets of the linear ordering polytope. Math. Program. **33**, 43–60 (1985)
7. Grötschel, M., Jünger, M., Reinelt, G.: A cutting plane algorithm for the linear ordering problem. Oper. Res. **32**, 1195–1220 (1984)
8. Grötschel, M., Jünger, M., Reinelt, G.: Optimal triangulation of large real world input-output matrices. Stat. Hefte **25**, 261–295 (1984)
9. Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica **1**(2), 169–197 (1981)
10. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Algorithms and Combinatorics, vol. 2. Springer, Berlin (1988)
11. Barahona, F., Grötschel, M., Mahjoub, A.R.: Facets of the bipartite subgraph polytope. Math. Oper. Res. **10**, 340–358 (1985)
12. Barahona, F., Grötschel, M., Jünger, M., Reinelt, G.: An application of combinatorial optimization to statistical physics and circuit layout design. Oper. Res. **36**(3), 493–513 (1988)
13. Grötschel, M., Jünger, M., Reinelt, G.: Via minimization with pin preassignments and layer preference. Z. Angew. Math. Mech. **69**(11), 393–399 (1989)
14. Grötschel, M., Jünger, M., Reinelt, G.: Optimal control of plotting and drilling machines: a case study. Z. Oper.-Res. **35**, 61–84 (1991)

Part II
Contribution by a Very Special Predecessor of Martin Grötschel

Scene Institut für Ökonometrie und Operation Research, Rheinische Friedrich-Wilhelms-Universität Bonn, March 1974

Martin Grötschel "What is combinatorial optimization anyway?"

Manfred Padberg "Come in."

This scene, which we described in Part I of this book, was instrumental to Martin Grötschel's development as a combinatorial optimizer. Of course, he had already decided to work in this field when he joined Bernhard Korte as a scientific assistant, yet an initial spark was still needed. And, fortunately, Manfred Padberg was a visiting professor at the institute then, and more than ready to light that fire. So Manfred is indeed a very special predecessor of Martin.

Manfred Padberg received his diploma (master's degree) in Mathematics at the Westfälische-Wilhelms-Universität Münster in 1967, and his doctoral degree at Carnegie-Mellon University under the supervision of Egon Balas in 1971. Afterwards, while he was a Research Fellow at the Wissenschaftszentrum Berlin, he spent half a year at the Institut für Ökonometrie und Operation Research, Rheinische Friedrich-Wilhelms-Universität Bonn on Bernhard Korte's invitation. That set the stage for the above scene in March 1974.

Afterwards, Manfred Padberg continued his academic career at New York University, serving as an Associate Professor in 1974 and progressing to Full Professor, Research Professor, and finally Professor Emeritus since 2002, he still remains active in the field.

He is one of the most prolific researchers in combinatorial optimization. When the theory of NP-completeness shook up the world of Theoretical Computer Science in 1971, the reaction of many researchers was that trying to solve NP-hard combinatorial optimization problems to optimality was simply a waste of time. But not Manfred. His attitude was always: "Let's try anyway!". And he did so, most notably, for the traveling salesman problem, achieving major breakthroughs together with Martin Grötschel, but later also with Harlan Crowder and still later, with Giovanni Rinaldi. He has been instrumental in laying the foundations of today's state of the art. Certainly, Manfred can take pride in witnessing his visions come true, visions he had in the early 1970s when many people believed such a thing to be a fool's errand.

His many contributions to the field and the many honors he has received have been described elsewhere, most notably in a Festschrift in his honor that appeared in 2004 in the MPS-SIAM Series on Optimization, entitled "The Sharpest Cut," and edited by … Martin Grötschel.

When we (the editors) invited Manfred to contribute to *this* book, he was delighted, immediately accepted the challenge, and gave us his latest work on his lifetime scientific mission: "Facets and Rank of Integer Polyhedra," further proof that Manfred is always good for a surprise—enjoy!

Facets and Rank of Integer Polyhedra

Manfred W. Padberg

Dedicated to Martin Grötschel, my first and best former doctoral student and a personal friend for almost forty years now. Ad multos annos, Martin!

Abstract We discuss several methods of determining all facets of "small" polytopes and polyhedra and give several criteria for classifying the facets into different facet types, so as to bring order into the multitude of facets as, e.g., produced by the application of the double description algorithm (DDA). Among the forms that we consider are the normal, irreducible and minimum support representations of facets. We study symmetries of vertex and edge figures under permissible permutations that leave the underlying polyhedron unchanged with the aim of reducing the numerical effort to find all facets efficiently. Then we introduce a new notion of the rank of facets and integer polyhedra. In the last section, we present old and new results on the facets of the symmetric traveling salesman polytope Q_T^n with up to $n = 10$ cities based on our computer calculations and state a conjecture that, in the notion of rank $\rho(P)$ introduced here, asserts $\rho(Q_T^n) = n - 5$ for all $n \geq 5$. This conjecture is supported by our calculations up to $n = 9$ and, possibly, $n = 10$.

1 Introduction

Let $P \subseteq \mathbb{R}^n$ be any *pointed* rational polyhedron of $\mathbb{R}^n$ and $\mathbf{X} = (\mathbf{x}^1 \ \cdots \ \mathbf{x}^p)$, $\mathbf{Y} = (\mathbf{y}^1 \ \cdots \ \mathbf{y}^r)$ be a minimal generator of P. The columns $\mathbf{x}^1, \ldots, \mathbf{x}^p$ of $\mathbf{X}$ are a list of the extreme points of P and, likewise, the columns $\mathbf{y}^1, \ldots, \mathbf{y}^r$ of $\mathbf{Y}$ are a list of the direction vectors of the extreme rays of P and thus,

$$\mathbf{x} \in P \quad \Longleftrightarrow \quad \mathbf{x} = \sum_{i=1}^{p} \mu_i \mathbf{x}^i + \sum_{j=1}^{r} \lambda_j \mathbf{y}^j \quad \text{for some } \mu_i \geq 0 \text{ with } \sum_{i=1}^{p} \mu_i = 1 \\ \text{and } \lambda_j \geq 0,$$

M.W. Padberg, Professor Emeritus of New York University
OptCom, 17 rue Vendôme, 13007 Marseille, France

i.e., $P = \text{conv}(\mathbf{x}^1, \ldots, \mathbf{x}^p) + \text{cone}(\mathbf{y}^1, \ldots, \mathbf{y}^r)$ where $\text{conv}(\cdot)$ means the convex hull and $\text{cone}(\cdot)$ the conical hull of the respective point sets. For short, we denote by $\text{vert}\, P$ the set of all extreme points of P and by $\text{exray}\, P$ the set of the direction vectors of all extreme rays of P. Rationality of the polyhedron P means that we assume that both $\mathbf{X}$ and $\mathbf{Y}$ consist of rational numbers only and "pointed" means that we assume that $p \geq 1$. If $\mathbf{Y}$ is void, P is a *polytope* in $\mathbb{R}^n$ rather than a polyhedron. Given the *pointwise* description $\mathbf{X}$, $\mathbf{Y}$ of P we can determine for "small" n, p and r an ideal linear description of P by running the double description algorithm (DDA), see, e.g., [4] or [38], to find a basis $(\mathbf{v}^1, v_0^1), \ldots, (\mathbf{v}^s, v_0^s)$ of the lineality space L_C and a minimal generator $(\mathbf{v}^{s+1}, v_0^{s+1}), \ldots, (\mathbf{v}^{s+t}, v_0^{s+t})$ of the conical part of the polyhedral cone

$$C = \left\{(\mathbf{v}, v_0) \in \mathbb{R}^{n+1} : \mathbf{vX} - v_0\mathbf{e} \leq \mathbf{0}, \mathbf{vY} \leq \mathbf{0}\right\}, \tag{1}$$

where $\mathbf{e} = (1, \ldots, 1) \in \mathbb{R}^p$ and $\mathbf{0}$ the null vector. Defining $\mathbf{a}^i = \mathbf{v}^i$, $b_i = v_0^i$ for $1 \leq i \leq s$, $\mathbf{b}^T = (b_1, \ldots, b_s)$, $\mathbf{h}^i = \mathbf{v}^{s+i}$, $h_i = v_0^{s+i}$ for $1 \leq i \leq t$ and $\mathbf{h}^T = (h_1, \ldots, h_t)$ then

$$P = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}, \mathbf{Hx} \leq \mathbf{h}\right\} \tag{2}$$

is an *ideal*, i.e., a minimal and complete, linear description of P where $\mathbf{A}$ is the $s \times n$ matrix of rank s with rows $\mathbf{a}^1, \ldots, \mathbf{a}^s$ and $\mathbf{H}$ is the $t \times n$ matrix with rows $\mathbf{h}^1, \ldots, \mathbf{h}^t$. By the rationality assumption about $\mathbf{X}$ and $\mathbf{Y}$ both $(\mathbf{A}, \mathbf{b})$ and $(\mathbf{H}, \mathbf{h})$ are matrices of rational numbers and thus, after appropriate scaling, integer numbers. Moreover, the dimension of P satisfies $\dim P = n - s$. P is full dimensional if and only if $L_C = \emptyset$, i.e., if and only if the matrix $\mathbf{A}$ is void. If $\mathbf{A}$ is nonvoid, then P is a *flat* in $\mathbb{R}^n$, otherwise P is a *solid*. The system of equations $\mathbf{Ax} = \mathbf{b}$ is an ideal description of the affine hull $\text{aff}(P)$ of P. Every row $\mathbf{hx} \leq h_0$, say, of $\mathbf{Hx} \leq \mathbf{h}$ defines a *facet* of P, i.e., a face of dimension $\dim P - 1$ of P, and vice versa, for every facet F of P there exists some row $\mathbf{hx} \leq h_0$ of $\mathbf{Hx} \leq \mathbf{h}$ such that $F = \{\mathbf{x} \in P : \mathbf{hx} = h_0\}$. The linear description (2) of P is *quasi-unique*: if $\mathbf{hx} \leq h_0$ and $\mathbf{gx} \leq g_0$ are two different representations of some facet F of P then $\mathbf{g} = \lambda\mathbf{h} + \boldsymbol{\mu}\mathbf{A}$ and $g_0 = \lambda h_0 + \boldsymbol{\mu}\mathbf{b}$ for some $\lambda > 0$ and $\boldsymbol{\mu} \in \mathbb{R}^s$, while the system of equations $\mathbf{Ax} = \mathbf{b}$ is unique *modulo* nonsingular transformations of $\mathbb{R}^s$. $\lambda > 0$ follows because $\mathbf{gx} - g_0 = \lambda(\mathbf{hx} - h_0) < 0$ for all $\mathbf{x} \in P - F$. The preceding is well known and we refer to [38] for a detailed treatment of polyhedra in $\mathbb{R}^n$ including the double description algorithm and any unexplained notation used in this paper. There are other methods to pass from the pointwise description of polyhedra to their ideal linear description (2) and vice versa, like the Fourier-Motzkin elimination algorithm, see, e.g., [50]. Since I have written most of this paper and carried out the computational work reported here in Sect. 7—which was around 1996/7—some progress has occurred in the algorithms for passing from the point-wise description of rational polyhedra to an ideal linear one and vice versa. I am indebted to Gerd Reinelt [46] for bringing his work with Thomas Christof to my attention, see [5] and [6–8] as well as the many references contained therein.

If P is a solid, then the ideal description of P is unique up to multiplication of $(\mathbf{H}, \mathbf{h})$ by positive scalars. Thus by scaling the data to be relatively prime integer numbers a *unique* ideal linear description of P is obtained. In this case the only issue that we need to address is how to compute a possible enormous number of the facets of P. Much of the material in this paper addresses this question and the question of how to obtain "reasonable" representations by way of linear inequalities of the facets of flats in $\mathbb{R}^n$.

The polyhedra of interest to us are subsets of $\mathbb{R}^n_+$ in the polyhedral case or subsets of the unit cube of $\mathbb{R}^n$ in the polytopal case, but we do not use that here. In either case one is frequently interested in obtaining an ideal description $\mathbf{Hx} \leq \mathbf{h}$ of the facets of P such that either $\mathbf{H} \geq \mathbf{O}$ or $\mathbf{H} \leq \mathbf{O}$ provided that such descriptions exist for P. No matter what *sign* convention is used in the linear description, one is frequently interested in finding *minimal support* representations of the facets of P. If P is flat, running the DDA to find an ideal linear description (2) of P does not automatically provide facets of this form. Rather we get a representation of any facet of P *modulo* some linear combination of the equations $\mathbf{Ax} = \mathbf{b}$. Thus the output of the DDA for a simple nonnegativity constraint $x_j \geq 0$ (if it defines a facet of P) can have many nonzeros and even a nonzero right-hand side. Intersecting the cone (1) with the nonnegativity constraints $\mathbf{v} \geq \mathbf{0}$ does not work: the result of running the DDA on the smaller cone gives all facets of P (*if* P has a description $\mathbf{Hx} \leq \mathbf{h}$ with $\mathbf{H} \geq \mathbf{O}$) plus typically considerably more inequalities that are valid, but not facet defining.

When we wish to analyze 1,000,000 or more "extreme rays" of the cone (1), we need to classify the 1,000,000 or more representations of facets produced by DDA *automatically* into equivalence classes. Here an "equivalence class" is understood to be a set of facets of P that are identical in some linear representation of them *modulo* linear combinations of the equations $\mathbf{Ax} = \mathbf{b}$ defining the affine hull of P and *modulo* permutations of the indices $1, \ldots, n$ of the components of $\mathbf{x} \in P$ which leave the polyhedron unchanged. Since every permutation of $1, \ldots, n$ can be described by some $n \times n$ permutation matrix $\boldsymbol{\Pi}$, a permutation $\boldsymbol{\Pi}$ leaves P unchanged if $P = \{\boldsymbol{\Pi}\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \in P\}$. We call such index permutations *permissible* for P. Denote by $\Pi(P)$ the set of all permissible index permutations for P. $\Pi(P) \neq \emptyset$ since $\mathbf{I}_n \in \Pi(P)$, where $\mathbf{I}_n$ is the $n \times n$ identity matrix. Let $\boldsymbol{\Pi} \in \Pi(P)$ and $\boldsymbol{\Pi}^T$ be the transpose of $\boldsymbol{\Pi}$. Since $\boldsymbol{\Pi}^T\boldsymbol{\Pi} = \boldsymbol{\Pi}\boldsymbol{\Pi}^T = \mathbf{I}_n$, $\boldsymbol{\Pi}^T \in \Pi(P)$ for every $\boldsymbol{\Pi} \in \Pi(P)$. Moreover, $\boldsymbol{\Pi}_a, \boldsymbol{\Pi}_b \in \Pi(P)$ implies that $\boldsymbol{\Pi}_a\boldsymbol{\Pi}_b \in \Pi(P)$ and $\boldsymbol{\Pi}_a, \boldsymbol{\Pi}_b, \boldsymbol{\Pi}_c \in \Pi(P)$ implies that $(\boldsymbol{\Pi}_a\boldsymbol{\Pi}_b)\boldsymbol{\Pi}_c = \boldsymbol{\Pi}_a(\boldsymbol{\Pi}_b\boldsymbol{\Pi}_c) = \boldsymbol{\Pi}_a\boldsymbol{\Pi}_b\boldsymbol{\Pi}_c \in \Pi(P)$. But $\boldsymbol{\Pi}_a\boldsymbol{\Pi}_b \neq \boldsymbol{\Pi}_b\boldsymbol{\Pi}_a$ is possible for $\boldsymbol{\Pi}_a, \boldsymbol{\Pi}_b \in \Pi(P)$ and thus $\Pi(P)$ is a (nonabelian) group of order at most $n!$ in general.

Definition 1 Let $F \neq F'$ be any two distinct *facets* of P. If $F' = \{\mathbf{x} \in P : \boldsymbol{\Pi}^T\mathbf{x} \in F\}$ for some $\boldsymbol{\Pi} \in \Pi(P)$, then F and F' are *equivalent* under $\Pi(P)$. $\kappa(P)$ is the *class number* of distinct facets of P that are *pairwise not equivalent* under $\Pi(P)$.

Every permissible index permutation corresponds to a linear transformation of the polyhedron P. More general, affine transformations exist that leave a polyhedron unchanged. This is the case, e.g., for the *Boolean quadric polytope* QP^n, see

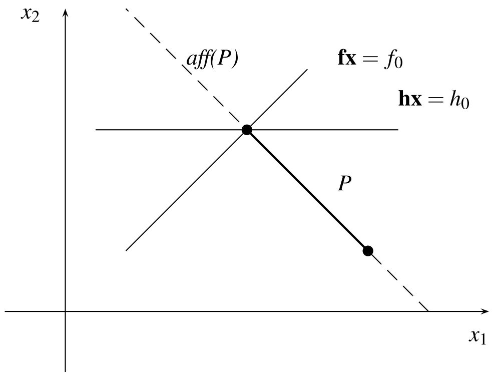


Fig. 1 Normal form representation of facets of a flat polyhedron $P \subseteq \mathbb{R}^2$

Theorem 6 of [37]. Here we study permissible index permutations only, even though several of the properties that we establish remain true mutatis mutandis for more general transformations of P.

2 Normal Form and Classification of Facets

The first task of the analysis is to find a "normal form" for the facets of flat polyhedra $P \subseteq \mathbb{R}^n$ that takes care of the multiplicity of the facet representations due to the equations defining the affine hull of P and that permits us to determine $\kappa(P)$ and some unique member of each equivalence class under $\Pi(P)$.

An inequality $\mathbf{fx} \le f_0$ is *valid* for P if $P \subseteq \{\mathbf{x} \in \mathbb{R}^n : \mathbf{fx} \le f_0\}$. Let $F \subseteq P$ be any facet of the polyhedron P. Every valid inequality $\mathbf{fx} \le f_0$ for P with

$$F = P \cap \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{fx} = f_0\right\}$$

is a *representation* of F.

Definition 2 ([28]) A representation $\mathbf{fx} \le f_0$ of a facet $F = \{\mathbf{x} \in P : \mathbf{fx} = f_0\}$ of P is in *normal form* if $\mathbf{Af}^T = \mathbf{0}$ where $\text{aff}(P) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}\}$.

Given any representation $\mathbf{hx} \le h_0$ of a facet F of P we can calculate its normal form representation by projecting $\mathbf{h}$ onto the subspace $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{0}\}$ and adjusting the right-hand side accordingly; see Fig. 1.

Claim 1 *Let* $\mathbf{hx} \le h_0$ *be any representation of a facet* F *of* P. *Then* $\mathbf{fx} \le f_0$ *with*

$$\mathbf{f} = \mathbf{h}\big(\mathbf{I}_n - \mathbf{A}^T\left(\mathbf{AA}^T\right)^{-1}\mathbf{A}\big), \qquad f_0 = h_0 - \mathbf{hA}^T\left(\mathbf{AA}^T\right)^{-1}\mathbf{b} \tag{3}$$

is a representation of F *in normal form.*

Claim 2 *The normal form representation of a facet F of P is unique up to scaling.*

Proof Let $\mathbf{gx} \le g_0$ be any representation of F and $\mathbf{fx} \le f_0$ be a normal form representation of F. Then we know from the quasi-uniqueness that $\mathbf{g} = \lambda\mathbf{f} + \boldsymbol{\mu}\mathbf{A}$, $g_0 = \lambda f_0 + \boldsymbol{\mu}\mathbf{b}$ where $\lambda > 0$ is a positive scalar and $\boldsymbol{\mu} \in \mathbb{R}^s$ is arbitrary. Suppose that $\mathbf{gx} \le g_0$ is also in normal form. Multiplying the equation for $\mathbf{g}$ by $\mathbf{A}^T$ we get $\mathbf{0} = \mathbf{gA}^T = \lambda\mathbf{fA}^T + \boldsymbol{\mu}\mathbf{AA}^T$ and thus $\boldsymbol{\mu} = \mathbf{0}$ since $\mathbf{fA}^T = \mathbf{0}$ and $r(\mathbf{AA}^T) = s$. Thus $(\mathbf{g}, g_0) = \lambda(\mathbf{f}, f_0)$ for some $\lambda > 0$. □

The normal form of the representation of some facet of the polyhedron $P \subseteq \mathbb{R}^n$ takes care—in essence—of the multiplicity of the representations of that facet that results from the flatness of the polyhedron. For rational polyhedra the remaining ambiguity can be eliminated by bringing the normal form representations of the facets of P into integer coefficient form with relatively prime integers.

Definition 3 A normal form representation $(\mathbf{f}, f_0)$ of a facet of P is in *primitive normal form* if the components of $(\mathbf{f}, f_0)$ are relatively prime integers.

Let $\mathbf{fx} \le f_0$ and $\mathbf{gx} \le g_0$ be any two facet-defining inequalities for P in primitive normal form. Then the facets F and F' of P defined by $\mathbf{fx} \le f_0$ and $\mathbf{gx} \le g_0$ are equivalent under $\Pi(P)$ if and only if $f_0 = g_0$ and $\mathbf{f} = \mathbf{g}\boldsymbol{\Pi}$ for some $\boldsymbol{\Pi} \in \Pi(P)$. To find the class number $\kappa(P)$ of distinct facet "types" of P we must check all primitive normal form representations pairwise for equivalence under $\Pi(P)$. Testing each pair of inequalities for isomorphism is computationally expensive. This effort can be reduced by a "preclassification" using criteria that are necessary for the equivalence under $\Pi(P)$ of distinct facets of P.

Claim 3 *If* $\mathbf{hx} \le h_0$ *and* $\mathbf{gx} \le g_0$ *define two distinct facets of P that belong to the same equivalence class with respect to some* $\boldsymbol{\Pi} \in \Pi(P)$, *then*

(i) $n^v_{\mathbf{h}} = n^v_{\mathbf{g}}$ *and* $n^{ex}_{\mathbf{h}} = n^{ex}_{\mathbf{g}}$, *where*

$$n^v_{\mathbf{h}} = \left|\{\mathbf{x} \in \operatorname{vert} P : \mathbf{hx} = h_0\}\right| \quad \textit{and} \quad n^{ex}_{\mathbf{h}} = \left|\{\mathbf{y} \in \operatorname{exray} P : \mathbf{hy} = 0\}\right| \tag{4}$$

and $n^v_{\mathbf{g}}$ *and* $n^{ex}_{\mathbf{g}}$ *are defined correspondingly for* $(\mathbf{g}, g_0)$.

(ii) $d_h = d_g$, *where d_h are d_g are the distances of the center of gravity* $\mathbf{x}^C$ *of P, i.e.,*

$$\mathbf{x}^C = \frac{1}{p}\sum_{i=1}^{p} \mathbf{x}^i, \tag{5}$$

from $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}, \mathbf{hx} = h_0\}$ *and* $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}, \mathbf{gx} = g_0\}$, *respectively.*

(iii) $a_0 = b_0$ *and* $\mathbf{a}^{ord} = \mathbf{b}^{ord}$, *where* $\mathbf{ax} \le a_0$, $\mathbf{bx} \le b_0$ *are the primitive normal forms of* $\mathbf{hx} \le h_0$, $\mathbf{gx} \le g_0$ *and* $\mathbf{a}^{ord}$, $\mathbf{b}^{ord}$ *are the vectors obtained from* $\mathbf{a}, \mathbf{b}$ *by ordering their components by increasing value.*

Proof

(i) Extreme points of P are mapped into extreme points of P for $\boldsymbol{\Pi} \in \Pi(P)$. For let $\mathbf{x} \in \operatorname{vert} P$, $\boldsymbol{\Pi} \in \Pi(P)$ and suppose that $\mathbf{z} = \boldsymbol{\Pi}\mathbf{x} \notin \operatorname{vert} P$. Then there exist $\mathbf{z}^1 \neq \mathbf{z}^2 \in P$ and $0 < \mu < 1$ such that $\mathbf{z} = \mu\mathbf{z}^1 + (1-\mu)\mathbf{z}^2$. But $\boldsymbol{\Pi}^T \in \Pi(P)$ and $\mathbf{x} \in \operatorname{vert} P$. Hence $\mathbf{x} = \boldsymbol{\Pi}^T\mathbf{z} = \mu\boldsymbol{\Pi}^T\mathbf{z}^1 + (1-\mu)\boldsymbol{\Pi}^T\mathbf{z}^2$ implies $\boldsymbol{\Pi}^T\mathbf{z}^1 = \boldsymbol{\Pi}^T\mathbf{z}^2$ and thus $\mathbf{z}^1 = \mathbf{z}^2$, which is a contradiction. We prove likewise that extreme rays of P are mapped into extreme rays of P for $\boldsymbol{\Pi} \in \Pi(P)$. Thus, since the facets defined by $\mathbf{hx} \leq h_0$ and $\mathbf{gx} \leq g_0$ are equivalent under $\Pi(P)$, the respective counts are equal.

(ii) To calculate d_h^2 we have to solve

$$\min\{\|\mathbf{x} - \mathbf{x}^C\|^2 : \mathbf{x} \in \mathcal{A}\}, \tag{6}$$

where $\mathcal{A} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}, \mathbf{hx} = h_0\}$. It follows that $\mathcal{A} \subseteq \{\mathbf{x} \in \mathbb{R}^n : \mathbf{f}^1\mathbf{x} = f_0^1\}$ where $(\mathbf{f}^1, f_0^1)$ is the normal form of $(\mathbf{h}, h_0)$. Solving $\min\{\|\mathbf{x} - \mathbf{x}^C\|^2 : \mathbf{f}^1\mathbf{x} = f_0^1\}$ by the Lagrangean multiplier technique we calculate $\mathbf{x}^* = \mathbf{x}^C - \frac{(\mathbf{f}^1\mathbf{x}^C - f_0^1)}{\|\mathbf{f}^1\|^2}(\mathbf{f}^1)^T$. Since $\mathbf{hx} - h_0 = \mathbf{f}^1\mathbf{x} - f_0^1$ for all $\mathbf{x} \in \mathbb{R}^n$ with $\mathbf{Ax} = \mathbf{b}$ we calculate $\mathbf{x}^* \in \mathcal{A}$, because

$$\mathbf{f}^1\mathbf{h}^T = \|\mathbf{h}\|^2 - \mathbf{hA}^T\left(\mathbf{AA}^T\right)^{-1}\mathbf{Ah}^T = \|\mathbf{f}^1\|^2. \tag{7}$$

Since $\mathcal{A} \subseteq \{\mathbf{x} \in \mathbb{R}^n : \mathbf{f}^1\mathbf{x} = f_0^1\}$ and $\mathbf{x}^* \in \mathcal{A}$ it follows that

$$d_h = \frac{|\mathbf{hx}^C - h_0|}{\|\mathbf{f}^1\|} \quad \text{and likewise} \quad d_g = \frac{|\mathbf{gx}^C - g_0|}{\|\mathbf{f}^2\|}, \tag{8}$$

where $(\mathbf{f}^2, f_0^2)$ is the normal form of $(\mathbf{g}, g_0)$. Since $\mathbf{hx} \leq h_0$ and $\mathbf{gx} \leq g_0$ belong to the same equivalence class and the normal form is unique up to scaling, there exists $\lambda > 0$ such that $\mathbf{f}^1 = \lambda\mathbf{f}^2\boldsymbol{\Pi}$ and $f_0^1 = \lambda f_0^2$ for some $\boldsymbol{\Pi} \in \Pi(P)$. Consequently,

$$\begin{aligned} p\left|\mathbf{hx}^C - h_0\right| &= \left|\sum_{i=1}^{p}\left(\mathbf{f}^1\mathbf{x}^i - f_0^1\right)\right| = \lambda\left|\sum_{i=1}^{p}\left(\mathbf{f}^2\boldsymbol{\Pi}\mathbf{x}^i - f_0^2\right)\right| \\ &= \lambda\left|\sum_{i=1}^{p}\left(\mathbf{f}^2\mathbf{x}^i - f_0^2\right)\right| = p\lambda\left|\mathbf{gx}^C - g_0\right|. \end{aligned}$$

Since $\|\mathbf{f}^1\| = \lambda\|\mathbf{f}^2\|$ it follows that $d_h = d_g$.

(iii) By assumption $\mathbf{hx} \leq h_0$ and $\mathbf{gx} \leq g_0$ belong to the same equivalence class with respect to $\boldsymbol{\Pi}$ and their primitive normal forms are unique. Thus $a_0 = b_0$ and $\mathbf{a} = \mathbf{b}\boldsymbol{\Pi}$. Hence if $a_0 \neq b_0$ or $\mathbf{a}^{ord} \neq \mathbf{b}^{ord}$ then $\boldsymbol{\Pi} \in \Pi(P)$ cannot exist, contradiction. □

Applying Claim 3 to the 1,000,000 or more inequalities $(\mathbf{H}, \mathbf{h})$ produced by the DDA we can partition the system $(\mathbf{H}, \mathbf{h})$ into q, say, disjoint subsystems

$$\mathbf{H}_1\mathbf{x} \le \mathbf{h}_1, \qquad \mathbf{H}_2\mathbf{x} \le \mathbf{h}_2, \qquad \dots, \qquad \mathbf{H}_q\mathbf{x} \le \mathbf{h}_q, \tag{9}$$

such that for each row of $(\mathbf{H}_i, \mathbf{h}_i)$ the criteria of Claim 3 are met. This breaks the 1,000,000 or more rows of $(\mathbf{H}, \mathbf{h})$ typically down into "chunks" of 5,000, 10,000, etc. rows that we need to analyze further.

Example 1 The distance and normal form calculation can be effectively shortened in most cases when a particular structure is present. To illustrate this for the symmetric traveling salesman problem, we can ignore the basis of the lineality space as calculated by, e.g., the DDA and let $\mathbf{A}$ be the node versus edge incidence matrix of the complete graph having m nodes, say, so that $n = m(m-1)/2$ is the number of variables in the problem. Every *permutation* of the m nodes of the graph induces a permissible permutation of the indices $1, \dots, n$ because re-indexing the nodes of the graph leaves the associated symmetric traveling salesman (STS) polytope unchanged. We wish to find the equivalence classes of its facets for such permutations. We know, see [19], that the affine hull for the symmetric traveling salesman polytope is given by the system of equations $\mathbf{Ax} = \mathbf{2}$ where $\mathbf{2}$ is a vector of m entries equal to 2. Thus

$$\mathbf{AA}^T = (m-2)\mathbf{I}_m + \mathbf{e}_m\mathbf{e}_m^T, \qquad \left(\mathbf{AA}^T\right)^{-1} = \frac{1}{m-2}\left(\mathbf{I}_m - \frac{1}{2(m-1)}\mathbf{e}_m\mathbf{e}_m^T\right),$$

where $\mathbf{e}_m$ is a column vector of m entries equal to 1. It follows that (3) and the calculation of $\|\mathbf{f}\|^2$ simplify to

$$\begin{aligned} \mathbf{f} &= \mathbf{h} - \frac{1}{m-2}\mathbf{h}^*\mathbf{A} + \frac{2(\mathbf{he}_n)}{(m-1)(m-2)}\mathbf{e}_n^T, \\ f_0 &= h_0 - \frac{2}{m-1}(\mathbf{he}_n), \end{aligned} \tag{10}$$

$$\|\mathbf{f}\|^2 = \|\mathbf{h}\|^2 - \frac{1}{m-2}\left\|\mathbf{h}^*\right\|^2 + \frac{2}{(m-1)(m-2)}(\mathbf{he}_n)^2, \tag{11}$$

where $\mathbf{h}^* = \mathbf{hA}^T$ is the vector with components $h_v^* = \sum_{e\in\delta(v)} h_e$ for all nodes $1 \le v \le m$ of the graph, $\delta(v)$ is the set of edges e of the graph meeting the node v and h_e for $1 \le e \le n$ are the components of $\mathbf{h}$. Thus the numerical inversion of $\mathbf{AA}^T$ can be avoided. Moreover, the gravity center $\mathbf{x}^C$ of the STS polytope is given by $x_e^C = \frac{2}{m-1}$ for $1 \le e \le n$ and so the formula (8) for the squared distance calculation simplifies to

$$d_h^2 = \frac{(m-2)(2\mathbf{he}_n - (m-1)h_0)^2}{(m-1)((m-1)(m-2)\|\mathbf{h}\|^2 - (m-1)\|\mathbf{h}^*\|^2 + 2(\mathbf{he}_n)^2)}. \tag{12}$$

If the components of $\mathbf{h}$ are integer, d_h^2 can thus be computed exactly in rational form. Multiplying $(\mathbf{f}, f_0)$ as defined by (10) by $(m-1)(m-2)$ and clearing the greatest common divisor we get for integer $(\mathbf{h}, h_0)$ the *unique* representation in primitive normal form of the facet of the STS polytope defined by $\mathbf{hx} \le h_0$.

Having obtained the partitioning (9) we see at present no other way than to check the remaining members in each class of (9) pairwise for equivalence under $\Pi(P)$, unless there are other affine linear transformations of P that leave P unchanged. The remaining isomorphism test can be done, e.g., *enumeratively* on the primitive normal form representation of the facets of P and is computationally expensive. But for relatively "small" polyhedra it can be done in reasonable computing times. This way we can find the class number $\kappa(P)$ of the distinct types of facets of flat or solid polyhedra $P \subseteq \mathbb{R}^n$ as well as a particular representation for each facet class.

3 Irreducible Representations of Facets

Denote f_j for $j = 1, \ldots, n$ the n first components of $(\mathbf{f}, f_0) \in \mathbb{R}^{n+1}$.

Definition 4 A representation $(\mathbf{f}, f_0)$ of a facet of P is in *irreducible form* if either

(i) $\mathbf{f} \ge \mathbf{0}$ and $|\{j \in \{1, \ldots, n\} : f_j > 0\}|$ is as small as possible or
(ii) $\mathbf{f} \le \mathbf{0}$ and $|\{j \in \{1, \ldots, n\} : f_j < 0\}|$ is as small as possible or
(iii) $|\{j \in \{1, \ldots, n\} : f_j \ne 0\}|$ is as small as possible.

In case (i) of the definition $(\mathbf{f}, f_0)$ has minimum positive, in case (ii) it has minimum negative and in case (iii) minimum support. This concept of irreducibility of facet representations does not imply uniqueness, but it reduces substantially the number of ways in which we can represent the facets of *flat* polyhedra which is, a priori, a continuum. See Fig. 5 of Sect. 7 where we show 20 different facet types of the 194,187 facets of the traveling salesman polytope on eight cities in irreducible nonnegative form (except the four remaining types given by the nonnegativity and subtour elimination constraints), i.e., $\kappa(P) = 24$ in this case. The question is how to find them (if they exist at all). To do so we have to use the DDA again.

We will discuss only the polytopal case, i.e., when $\mathbf{Y}$ is void, but the following applies mutatis mutandis to the polyhedral case as well. Let $(\mathbf{h}, h_0)$ be a representation of some facet F of P and partition $\mathbf{X}$ into two matrices $\mathbf{X}_1$, $\mathbf{X}_2$ such that

$$\mathbf{hX}_1 - h_0\mathbf{e}_1 = \mathbf{0}, \qquad \mathbf{hX}_2 - h_0\mathbf{e}_2 < \mathbf{0}, \tag{13}$$

where $\mathbf{e}_1$, $\mathbf{e}_2$ are compatible vectors of ones. Consider the polyhedral cone

$$C_h^+ = \left\{(\mathbf{w}, w_0) \in \mathbb{R}^{n+1} : \mathbf{wX}_1 - w_0\mathbf{e}_1 = \mathbf{0}, \mathbf{wX}_2 - w_0\mathbf{e}_2 \le \mathbf{0}, \mathbf{w} \ge \mathbf{0}\right\} \tag{14}$$

and denote by $F_\mathbf{h}$ the facet of P defined by $\mathbf{hx} \le h_0$, i.e.,

$$F_\mathbf{h} = P \cap \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{hx} = h_0\right\}. \tag{15}$$

If $F_{\mathbf{h}}$ has a representation $\mathbf{fx} \le f_0$ with $\mathbf{f} \ge \mathbf{0}$, then $(\mathbf{f}, f_0) \in C_h^+$. Thus any extreme ray of the conical part of C_h^+ with minimum positive support is an irreducible representation of $F_{\mathbf{h}}$ and can be found by running the DDA with (14) as input. To find a minimum negative support representation of the facet $F_{\mathbf{h}}$ of P we use the polyhedral cone C_h^- which is (14) with the constraints $\mathbf{wX}_2 - w_0\mathbf{e}_2 \le \mathbf{0}$ replaced by $\mathbf{wX}_2 - w_0\mathbf{e}_2 \ge \mathbf{0}$ and run the DDA.

Since the polyhedron $P \subseteq \mathbb{R}^n$ may or may not admit representations of its facets satisfying either sign restriction, we are lead to consider the cone $C_h^\pm$ which is (14) but without the constraints $\mathbf{w} \ge \mathbf{0}$. It is not difficult to show that a minimal generator for $C_h^\pm$ consists of a basis of its lineality space, i.e., of the rows of the matrix $(\mathbf{A}, \mathbf{b})$ defining the affine hull of the polyhedron P, plus the direction vector given by the (unique) normal form of the facet of P defined by $\mathbf{hx} \le h_0$. So this construction does not help us to find "minimal support" representations of the facets of flat polyhedra.

Consider instead

$$C_h = \big\{(\mathbf{u}, \mathbf{v}, w_0) \in \mathbb{R}^{2n+1} : \\ \mathbf{uX}_1 - \mathbf{vX}_1 - w_0\mathbf{e}_1 = \mathbf{0}, \mathbf{uX}_2 - \mathbf{vX}_2 - w_0\mathbf{e}_2 \le \mathbf{0}, \mathbf{u} \ge \mathbf{0}, \mathbf{v} \ge \mathbf{0}\big\}. \quad (16)$$

Since the extreme rays of C_h are defined by submatrices of the constraint set of (16) of rank $2n$, it follows from a rank consideration that every extreme ray $(\mathbf{u}, \mathbf{v}, w_0)$ of C_h with $u_j \ne v_j$ satisfies either $u_j = 0$ or $v_j = 0$ for all $1 \le j \le n$. Moreover, it follows also from a rank consideration that every extreme ray of C_h^+ and every extreme ray of C_h^- defines an extreme ray of C_h. Let $\mathbf{fx} \le f_0$ be any representation of the facet $F_{\mathbf{h}}$ of *minimal support*. Setting $\mathbf{u} = \max\{\mathbf{0}, \mathbf{f}\}$, $\mathbf{v} = \max\{\mathbf{0}, -\mathbf{f}\}$ and $w_0 = f_0$ it follows that $(\mathbf{u}, \mathbf{v}, w_0) \in C_h$. Consequently, every minimum support representation of $F_{\mathbf{h}}$ defines an extreme ray of the conical part of C_h. The cone C_h exhibits a lot of "symmetry" and has many extreme rays.

We can thus find the *irreducible representations* of any facet $F_{\mathbf{h}}$ of any flat P by running the DDA with the respective cones C_h^+, C_h^-, and C_h as input.

In a computer implementation we translate the polyhedron P so that $F_{\mathbf{h}}$ contains the origin of $\mathbb{R}^n$. Consequently, the "homogenizing" variable w_0 in (14) and (16) can be dropped. Moreover, we need to generate only a single row for the changed $\mathbf{X}_2$ part of the constraint set of, e.g., (16). For the changed $\mathbf{X}_1$ part of it we need only a submatrix of maximal rank. This reduces the size of the input to the DDA substantially. More precisely, in all three cases exactly $\dim P$ constraints plus the nonnegativity conditions suffice to find an irreducible representation of $F_{\mathbf{h}}$.

4 Symmetry of Vertex Figures

Given a pointwise description of a polyhedron $P \subseteq \mathbb{R}^n$ the numerical effort to find all facets of P consists of running the DDA, or some similar algorithm, with the cone (1) as input. E.g., for the symmetric traveling salesman problem with m cities on a complete graph the number of inequalities in (1) equals $p = \frac{1}{2}(m-1)!$. For

$m = 8$ we have $p = 2{,}520$, for $m = 9$ we have $p = 20{,}160$, for $m = 10$ we have $p = 181{,}440$ and so forth. Given the current state of computing machinery it is out of the question to attack this problem directly by analyzing the cone (1) for $m \geq 9$. In this section and the next we discuss ways of reducing the computational effort for general polyhedra and polytopes for which $\Pi(P) \neq \{\mathbf{I}_n\}$.

Definition 5 For $\mathbf{x}^0 \in \operatorname{vert} P$ let $\mathbf{x}^1, \ldots, \mathbf{x}^a$ be all of its *adjacent vertices* and $\mathbf{y}^1, \ldots, \mathbf{y}^b$ represent all the extreme rays P such that $\mathbf{x}^0 + \lambda\mathbf{y}^i$ for $\lambda \geq 0$ is a 1-dimensional face of P. The displaced cone with apex at $\mathbf{x}^0$

$$OC(\mathbf{x}^0, P) = \left\{ \mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \mathbf{x}^0 + \sum_{i=1}^{a} \lambda_i (\mathbf{x}^i - \mathbf{x}^0) + \sum_{i=1}^{b} \mu_i \mathbf{y}^i, \lambda_i \geq 0, \mu_i \geq 0 \right\} \quad (17)$$

is the *vertex figure* (or the *outer cone*) for P at $\mathbf{x}^0$.

Note that our definition of a vertex figure is different from the one frequently found in the literature on polytopes, see, e.g., [22] or [50]. By construction, $OC(\mathbf{x}^0, P) \supset P$, $\dim OC(\mathbf{x}^0, P) = \dim P$ and every facet of $OC(\mathbf{x}^0, P)$ is a facet of P, but not vice versa since all facets of $OC(\mathbf{x}^0, P)$ contain $\mathbf{x}^0$. Let $\boldsymbol{\Pi} \in \Pi(P)$ and $\mathbf{x}^* = \boldsymbol{\Pi}\mathbf{x}^0$. Then $\mathbf{x}^* \in \operatorname{vert} P$ and we claim that the vertex figure $OC(\mathbf{x}^*, P)$ for P at $\mathbf{x}^*$ is given by

$$OC(\mathbf{x}^*, P) = \left\{ \mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \mathbf{x}^* + \sum_{i=1}^{a} \lambda_i (\boldsymbol{\Pi}\mathbf{x}^i - \mathbf{x}^*) + \sum_{i=1}^{b} \mu_i \boldsymbol{\Pi}\mathbf{y}^i, \lambda_i \geq 0, \mu_i \geq 0 \right\}.$$

Claim 4 *For any* $\mathbf{x}^0 \neq \mathbf{x}^1 \in P$ *let* $F(\mathbf{x}^0, \mathbf{x}^1)$ *be the face of* minimal dimension *of* P *containing both* $\mathbf{x}^0$ *and* $\mathbf{x}^1$. *Then* $F(\boldsymbol{\Pi}\mathbf{x}^0, \boldsymbol{\Pi}\mathbf{x}^1) = \{\mathbf{x} \in \mathbb{R}^n : \boldsymbol{\Pi}^T\mathbf{x} \in F(\mathbf{x}^0, \mathbf{x}^1)\}$ *and* $\dim F(\mathbf{x}^0, \mathbf{x}^1) = \dim F(\boldsymbol{\Pi}\mathbf{x}^0, \boldsymbol{\Pi}\mathbf{x}^1)$ *for all* $\boldsymbol{\Pi} \in \Pi(P)$.

Proof Let $F = F(\mathbf{x}^0, \mathbf{x}^1)$ and likewise $\Pi F = F(\boldsymbol{\Pi}\mathbf{x}^0, \boldsymbol{\Pi}\mathbf{x}^1)$. Since F is a face of P, there exists $(\mathbf{f}, f_0) \in \mathbb{R}^{n+1}$ such that $\mathbf{fx} = f_0$ for all $\mathbf{x} \in F$ and $\mathbf{fx} < f_0$ for all $\mathbf{x} \in P$, $\mathbf{x} \notin F$. Since $\mathbf{x}^0, \mathbf{x}^1 \in F$ and $\boldsymbol{\Pi}\mathbf{x}^0, \boldsymbol{\Pi}\mathbf{x}^1 \in \Pi F$ it follows from the minimality of ΠF that $\Pi F \subseteq \{\mathbf{x} \in \mathbb{R}^n : (\mathbf{f}\boldsymbol{\Pi}^T)\mathbf{x} = f_0\}$. Consequently, we have $\Pi F \subseteq \{\mathbf{x} \in \mathbb{R}^n : \boldsymbol{\Pi}^T\mathbf{x} \in F\}$ since $\boldsymbol{\Pi}^T\mathbf{x} \in P$ for all $\mathbf{x} \in \Pi F$. We conclude likewise that $F \subseteq \{\mathbf{x} \in \mathbb{R}^n : \boldsymbol{\Pi}\mathbf{x} \in \Pi F\}$. Consequently, if $\boldsymbol{\Pi}^T\mathbf{x} \in F$ for some $\mathbf{x} \in \mathbb{R}^n$ then $\boldsymbol{\Pi}(\boldsymbol{\Pi}^T\mathbf{x}) = \mathbf{x} \in \Pi F$. Thus $\Pi F = \{\mathbf{x} \in \mathbb{R}^n : \boldsymbol{\Pi}^T\mathbf{x} \in F\}$ and the first part follows. Let $P = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Hx} \leq \mathbf{h}\}$ be any linear description of P. Since P is pointed we have $r(\mathbf{H}) = n$. Denote by $(\mathbf{H}_F, \mathbf{h}_F)$ the *largest* submatrix of $(\mathbf{H}, \mathbf{h})$ such that $\mathbf{H}_F\mathbf{x} = \mathbf{h}_F$ for all $\mathbf{x} \in F$. Thus $\dim F = n - r(\mathbf{H}_F)$. Likewise, let $(\mathbf{H}_{\Pi F}, \mathbf{h}_{\Pi F})$ be the largest submatrix $(\mathbf{H}, \mathbf{h})$ such that $\mathbf{H}_{\Pi F}\mathbf{x} = \mathbf{h}_{\Pi F}$ for all $\mathbf{x} \in \Pi F$. By the preceding argument it follows that—up to row permutations—$\mathbf{H}_{\Pi F} = \mathbf{H}_F\boldsymbol{\Pi}^T$ and thus $\dim \Pi F = \dim F$ since $\boldsymbol{\Pi}$ is nonsingular. $\square$

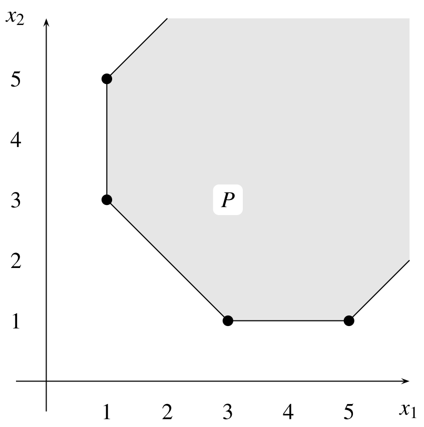


Fig. 2 Vertex classes of polyhedra

Thus the face of minimal dimension of P containing $\boldsymbol{\Pi}\mathbf{x}^0$ and $\boldsymbol{\Pi}\mathbf{x}^1$ is the image under $\boldsymbol{\Pi}$ of the face of minimal dimension of P that contains $\mathbf{x}^0$ and $\mathbf{x}^1$. Thus permissible index permutations for P preserve the adjacency of extreme points of P and hence $OC(\mathbf{x}^*, P)$ is the vertex figure for P at $\mathbf{x}^* = \boldsymbol{\Pi}\mathbf{x}^0$ for any $\boldsymbol{\Pi} \in \Pi(P)$ as claimed.

Claim 5 *Let* $\mathbf{x}^0 \neq \mathbf{x}^* \in \operatorname{vert} P$ *be such that* $\mathbf{x}^* = \boldsymbol{\Pi}\mathbf{x}^0$ *for some* $\boldsymbol{\Pi} \in \Pi(P)$. *Then* $\mathbf{fx} \leq f_0$ *defines a facet of* $OC(\mathbf{x}^0, P)$ *if and only if* $(\mathbf{f}\boldsymbol{\Pi}^T)\mathbf{x} \leq f_0$ *defines a facet of* $OC(\mathbf{x}^*, P)$.

Proof Let $\mathbf{fx} \leq f_0$ define a facet of $OC(\mathbf{x}^0, P)$. Consequently, $\mathbf{fx}^0 = f_0$ and $\mathbf{fx} \leq f_0$ for all $\mathbf{x} \in P$. Hence $f_0 = \mathbf{fx}^0 = (\mathbf{f}\boldsymbol{\Pi}^T)\boldsymbol{\Pi}\mathbf{x}^0 = (\mathbf{f}\boldsymbol{\Pi}^T)\mathbf{x}^*$ and $(\mathbf{f}\boldsymbol{\Pi}^T)\mathbf{y} = \mathbf{f}\boldsymbol{\Pi}^T\boldsymbol{\Pi}\mathbf{x} = \mathbf{fx} \leq f_0$ for all $\mathbf{y} \in P$ because $\mathbf{y} \in P$ implies $\mathbf{y} = \boldsymbol{\Pi}\mathbf{x}$ for some $\mathbf{x} \in P$. Let $\widehat{\mathbf{x}} \in P$ be such that $\mathbf{f}\widehat{\mathbf{x}} = f_0$ and $\dim F(\mathbf{x}^0, \widehat{\mathbf{x}}) = \dim P - 1$. Since $\mathbf{fx} \leq f_0$ defines a facet of P such an $\widehat{\mathbf{x}} \in P$ exists. Then $\boldsymbol{\Pi}\widehat{\mathbf{x}} \in P$ and $(\mathbf{f}\boldsymbol{\Pi}^T)\boldsymbol{\Pi}\widehat{\mathbf{x}} = \mathbf{f}\widehat{\mathbf{x}} = f_0$. From Claim 4 it follows that $\dim F(\mathbf{x}^*, \boldsymbol{\Pi}\widehat{\mathbf{x}}) = \dim P - 1$. Thus the inequality $(\mathbf{f}\boldsymbol{\Pi}^T)\mathbf{x} \leq f_0$ defines a facet of $OC(\mathbf{x}^*, P)$. The rest follows by symmetry. □

It follows from Claim 5 that we know all facets of $OC(\mathbf{x}^*, P)$ if we know all facets of $OC(\mathbf{x}^0, P)$ where $\mathbf{x}^* = \boldsymbol{\Pi}\mathbf{x}^0$ for some $\boldsymbol{\Pi} \in \Pi(P)$. Consequently, if for *every pair* $\mathbf{x}^0 \neq \mathbf{x}^* \in \operatorname{vert} P$ there exists some $\boldsymbol{\Pi} \in \Pi(P)$ such that $\mathbf{x}^* = \boldsymbol{\Pi}\mathbf{x}^0$, then all vertex figures of P are identical *modulo* $\Pi(P)$. The task of finding all facets of P is thus reduced to finding all facets of the vertex figure $OC(\mathbf{x}^0, P)$, where $\mathbf{x}^0$ is some extreme point of P. This observation makes the task of finding an ideal linear description of P easier from a computational point of view: the number of the facets of $OC(\mathbf{x}^0, P)$ is typically considerably smaller than the number of the facets of P.

In general, we cannot expect that for *every* pair $\mathbf{x}^0 \neq \mathbf{x}^* \in \operatorname{vert} P$ there exists some $\boldsymbol{\Pi} \in \Pi(P)$ such that $\mathbf{x}^* = \boldsymbol{\Pi}\mathbf{x}^0$. In Fig. 2 we show a "symmetric" polyhedron where this is not the case. The index permutation

$$\boldsymbol{\Pi} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

is permissible for the polyhedron P of Fig. 2 and the extreme points of P fall into the *two* vertex classes $\{(1,5),(5,1)\}$ and $\{(1,3),(3,1)\}$, of which it suffices to analyze one representative in each class. More generally, $\Pi(P)$ partitions the set of all extreme points into equivalence classes of which it suffices to analyze one representative in each class in order to find all the facets of P. If the number of such equivalence classes is relatively small—as compared, e.g., to the total number of extreme points of P—then substantial savings in the computational effort for the problem of finding all the facets of P result. In the case of the Boolean quadric polytope QP^n $\Pi(\mathrm{QP}^n)$ induces precisely n vertex classes. However, in this case there is also a "symmetry theorem", see [37], that permits one to reduce the study of the facial structure of QP^n to a single vertex figure as well.

For simplicity of exposition we will assume that all extreme points of P fall into a single equivalence class with respect to $\Pi(P)$ and that P is a polytope rather than a polyhedron. The following applies, however, mutatis mutandis to the general case of pointed polyhedra having several vertex classes as well.

We can thus replace the problem of finding all facets of P by the problem of finding all the facets of the vertex figure $OC(\mathbf{x}^0, P)$ at *any* extreme point $\mathbf{x}^0$ of P. To do so numerically we translate the displaced cone $OC(\mathbf{x}^0, P)$ with apex at $\mathbf{x}^0 \in P$ to the origin of $\mathbb{R}^n$ and consider instead of $OC(\mathbf{x}^0, P)$ the cone

$$CC(\mathbf{x}^0, P) = \left\{ \mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \sum_{i=1}^{a} \lambda_i (\mathbf{x}^i - \mathbf{x}^0), \lambda_i \geq 0 \right\}. \tag{18}$$

Now we use, e.g., the double description algorithm DDA to find a linear description of $CC(\mathbf{x}^0, P)$. To do so we find a minimal generator of the "polar" cone

$$PCC(\mathbf{x}^0, P) = \{ \mathbf{f} \in \mathbb{R}^n : \mathbf{f}(\mathbf{x}^i - \mathbf{x}^0) \leq 0 \text{ for } 1 \leq i \leq a \}, \tag{19}$$

where $\mathbf{f} \in \mathbb{R}^n$ is a row vector and a is the number of extreme points adjacent to $\mathbf{x}^0$. We write $PCC_0 = PCC(\mathbf{x}^0, P)$, for short. The following two facts are well known. If $\mathbf{f} \in PCC_0$ belongs to the lineality space of PCC_0, then $\mathbf{fx} = \mathbf{fx}^0$ belongs to the system of equations describing the affine hull of the polytope P. If $\mathbf{f} \in PCC_0$ belongs to the conical part of the minimal generator of PCC_0, then $\mathbf{fx} \leq \mathbf{fx}^0$ defines a facet of P that is tight at $\mathbf{x}^0$.

These considerations bring about a substantial reduction in the number of rows of the constraint matrix for the cone PCC_0 (19) that is the input for DDA which reduces the computations to find all facets of P. We illustrate the reduction for the symmetric traveling salesman problem on m cities: for $m = 8$ the input cone (1) has $p = 2{,}520$ rows while (19) has $a = 730$ rows, for $m = 9$ we have $p = 20{,}160$ while $a = 3{,}555$, and for $m = 10$ we have $p = 181{,}440$ and $a = 19{,}391$. Of course, we must *find* all extreme points of the polyhedron that are adjacent to $\mathbf{x}^0 \in P$ and this is difficult in general. However, it can be done, e.g., enumeratively for "small" polyhedra and all of the previous numbers were computed by a computer program that we have written for this purpose.

From the numbers that we give for the symmetric traveling salesman polytope it is clear that with our current computing machinery we can find the linear description of $CC(\mathbf{x}^0, P)$ for $m = 8$ at best. We have indeed succeeded to compute the corresponding linear description on a SunSparc 4 computer this way. However, for $m = 9$ this is again out of the question—at least at present.

5 Symmetry of Edge Figures

To generalize the previous device that we have used to reduce the computational effort let us state it as follows: we replace the problem of finding all facets of P by the problem of finding all facets of P that contain some 0-dimensional face of P, namely $\mathbf{x}^0 \in \operatorname{vert} P$. If P has several vertex classes then we do likewise by choosing some representative in each class. This brings about a substantial reduction in the size of the problem that we need to solve using the DDA. A further reduction can be expected if we *increase* the dimension of the face of P that we require the facets to contain. So we want to find all facets of P that contain a k-dimensional face of P that contains $\mathbf{x}^0$ where $k \geq 0$ is relatively small. For $k = 0$ we retrieve the previous trick, for $k = 1$ we ask for all facets of P that contain an edge $\mu\mathbf{x}^0 + (1-\mu)\mathbf{x}^i$ of P, where $0 \leq \mu \leq 1$ and $\mathbf{x}^i$ is some extreme point of P that is adjacent to $\mathbf{x}^0$. We can do likewise for any $k \geq 2$. Let us consider the case $k = 1$ explicitly, since for $k \geq 2$ the notation becomes a bit messy.

For $k = 1$ we replace (17) in the polytopal case by the a displaced cones

$$EC_P\left(\mathbf{x}^0, \mathbf{x}^\ell\right) = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \mathbf{x}^0 + \mu\left(\mathbf{x}^\ell - \mathbf{x}^0\right) + \sum_{\ell \neq i = 1}^{a} \lambda_i\left(\mathbf{x}^i - \mathbf{x}^0\right), \lambda_i \geq 0\right\}, \quad (20)$$

where $1 \leq \ell \leq a$ and $\mu \in \mathbb{R}$ is arbitrary because we want all facets of P containing the edge $\mu\mathbf{x}^0 + (1-\mu)\mathbf{x}^\ell$ for $0 \leq \mu \leq 1$ of P and thereby, necessarily, the entire line $\mu\mathbf{x}^0 + (1-\mu)\mathbf{x}^\ell$ for all $\mu \in \mathbb{R}$. By construction, $\dim EC_P(\mathbf{x}^0, \mathbf{x}^\ell) = \dim P$ and every facet of $EC_P(\mathbf{x}^0, \mathbf{x}^\ell)$ defines a facet of P, but not vice versa since the facets of $EC_P(\mathbf{x}^0, \mathbf{x}^\ell)$ all contain both $\mathbf{x}^0$ and $\mathbf{x}^\ell$. Of course, to find all the facets of P we must a priori analyze the displaced cones $EC_P(\mathbf{x}^0, \mathbf{x}^\ell)$ for all values of $\ell = 1, \ldots, a$, but as we shall see this number can be reduced substantially if $\Pi(P) \neq \{\mathbf{I}_n\}$.

Let us replace $\sum_{\ell \neq i=1}^{a} \lambda_i(\mathbf{x}^i - \mathbf{x}^0)$ in the definition of $EC_P(\mathbf{x}^0, \mathbf{x}^\ell)$ by a list N_ℓ^0 of *all* extreme points of P other than $\mathbf{x}^0$ and $\mathbf{x}^\ell$. The $\mathbf{x}^i - \mathbf{x}^0$ with $\mathbf{x}^i$ *not adjacent* to $\mathbf{x}^0$ are not extremal in $OC(\mathbf{x}^0, P)$ nor in $EC(\mathbf{x}^0, \mathbf{x}^\ell)$. This does not change the displaced cone (20) because $EC_P(\mathbf{x}^0, \mathbf{x}^\ell) \supseteq OC(\mathbf{x}^0, P) \supseteq P$. We calculate

$$\begin{aligned} &EC_P\left(\mathbf{x}^0, \mathbf{x}^\ell\right) \\ &\quad = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \mathbf{x}^0 + \mu\left(\mathbf{x}^\ell - \mathbf{x}^0\right) + \sum_{i \in N_\ell^0} \lambda_i\left(\mathbf{x}^i - \mathbf{x}^0\right), \lambda_i \geq 0\right\} \end{aligned}$$

$$= \left\{ \mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \mathbf{x}^\ell + \left(1 - \mu - \sum_{i \in N_\ell^0} \lambda_i\right)(\mathbf{x}^0 - \mathbf{x}^\ell) + \sum_{i \in N_\ell^0} \lambda_i (\mathbf{x}^i - \mathbf{x}^\ell), \lambda_i \geq 0 \right\}$$

$$= EC_P(\mathbf{x}^\ell, \mathbf{x}^0),$$

because $\nu = 1 - \mu - \sum_{i \in N_\ell^0} \lambda_i$ runs through all reals when $\mu \in \mathbb{R}$ is arbitrary. This calculation reflects the fact that the direction in which we traverse the edge of P defined by $\mathbf{x}^0$ and $\mathbf{x}^\ell$ is immaterial for the "shape" of $EC_P(\mathbf{x}^0, \mathbf{x}^\ell)$. So the displaced cone (20) is well defined by the edge given by the pair of adjacent extreme points $\mathbf{x}^0$ and $\mathbf{x}^\ell$ of P and we call it the *edge figure* of P relative to $\mathbf{x}^0$ and $\mathbf{x}^\ell$.

Shifting the displaced cone (20) to contain the origin we get the cone

$$ECC_P(\mathbf{x}^0, \mathbf{x}^\ell) = \left\{ \mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \mu(\mathbf{x}^\ell - \mathbf{x}^0) + \sum_{\ell \neq i = 1}^{a} \lambda_i (\mathbf{x}^i - \mathbf{x}^0), \lambda_i \geq 0, \mu \in \mathbb{R} \right\}, \tag{21}$$

which is an infinite "wedge" in $\mathbb{R}^n$, i.e., its lineality space has a dimension of 1. Moreover, while (21) is a valid pointwise description of $ECC_\ell = ECC_P(\mathbf{x}^0, \mathbf{x}^\ell)$, it is typically far from minimal. By testing each direction vector $\mathbf{x}^i - \mathbf{x}^0$ of ECC_ℓ with $1 \leq i \neq \ell \leq a$ for extremality in the cone ECC_ℓ we can reduce the pointwise description of ECC_ℓ to a minimal one. To keep the notation simple, let us assume that—after reindexing if necessary—the direction vectors $\mathbf{x}^i - \mathbf{x}^0$ for $1 \leq i \leq a(\ell)$ remain. Forming the polar cone like in (19) we thus get

$$PEC_P(\mathbf{x}^0, \mathbf{x}^\ell) = \left\{ \mathbf{f} \in \mathbb{R}^n : \mathbf{f}(\mathbf{x}^\ell - \mathbf{x}^0) = 0, \mathbf{f}(\mathbf{x}^i - \mathbf{x}^0) \leq 0 \text{ for } 1 \leq i \leq a(\ell) \right\} \tag{22}$$

as the input cone for the double description algorithm. Its number of rows $a(\ell) + 1$ is typically considerably smaller than the corresponding number for the cone (19), though $a(\ell)$ can vary significantly with ℓ. As mentioned above, all of this can be extended to k-dimensional faces of P that contain $\mathbf{x}^0$, where $k \geq 0$ is arbitrary. The cost of the "input preparation" for the corresponding cones to the analyzed by the DDA increases, but the mechanics of carrying out the analysis are clear.

Using this new trick we can thus replace the single problem (19) by a total of a typically smaller problems (22) to be analyzed by the DDA and this is the way it was done by Christof, Jünger and Reinelt [9] who used this methodology for $k = 1$ to determine all facets for the symmetric traveling salesman polytope on a complete graph with $m = 8$ nodes. But rather than solving the $a = 730$ problems that result from the number of extreme points that are adjacent to any given one of this polytope, they reduced the number of edge figures to be analyzed to a total of 59 and this is how.

Claim 6 *Let $\mathbf{x}^a$, $\mathbf{x}^b$ and $\mathbf{x}^c$, $\mathbf{x}^d$ be any two pairs of adjacent extreme points of P and suppose that $\mathbf{x}^c = \boldsymbol{\Pi}\mathbf{x}^a$, $\mathbf{x}^d = \boldsymbol{\Pi}\mathbf{x}^b$ for some $\boldsymbol{\Pi} \in \Pi(P)$. Then $\mathbf{f}\mathbf{x} \leq f_0$ defines a facet of $EC_P(\mathbf{x}^a, \mathbf{x}^b)$ if and only if $(\mathbf{f}\boldsymbol{\Pi}^T)\mathbf{x} \leq f_0$ defines a facet of $EC_P(\mathbf{x}^c, \mathbf{x}^d)$.*

Proof Let $\mathbf{fx} \le f_0$ define a facet of $EC_P(\mathbf{x}^a, \mathbf{x}^b)$. Then $\mathbf{fx}^a = \mathbf{fx}^b = f_0$ and $\mathbf{fx} \le f_0$ for all $\mathbf{x} \in P$. Consequently, $(\mathbf{f}\boldsymbol{\Pi}^T)\mathbf{x}^c = \mathbf{fx}^a = f_0$, $(\mathbf{f}\boldsymbol{\Pi}^T)\mathbf{x}^d = \mathbf{fx}^b = f_0$ and for any $\mathbf{y} \in P$ we get $(\mathbf{f}\boldsymbol{\Pi}^T)\mathbf{y} = \mathbf{fx} \le f_0$ since $\mathbf{y} = \boldsymbol{\Pi}\mathbf{x}$ for some $\mathbf{x} \in P$. Let $\widehat{\mathbf{x}} \in P$ be such that $\mathbf{f}\widehat{\mathbf{x}} = f_0$ and $\dim F(\mathbf{x}^a, \widehat{\mathbf{x}}) = \dim P - 1$, where $F(\mathbf{x}^a, \widehat{\mathbf{x}})$ is the face of smallest dimension of P containing both $\mathbf{x}^a$ and $\widehat{\mathbf{x}}$. Since $\mathbf{fx} \le f_0$ defines a facet of P such an $\widehat{\mathbf{x}} \in P$ exists. Then $\boldsymbol{\Pi}\widehat{\mathbf{x}} \in P$, $(\mathbf{f}\boldsymbol{\Pi}^T)\boldsymbol{\Pi}\widehat{\mathbf{x}} = f_0$ and by Claim 4, $\dim F(\boldsymbol{\Pi}\widehat{\mathbf{x}}, \mathbf{x}^c) = \dim P - 1$. Thus $(\mathbf{f}\boldsymbol{\Pi}^T)\mathbf{x} \le f_0$ defines a facet of $EC_P(\mathbf{x}^c, \mathbf{x}^d)$ and the rest follows by symmetry. □

We do not require distinctness of the extreme points $\mathbf{x}^a$, $\mathbf{x}^b$, $\mathbf{x}^c$ and $\mathbf{x}^d$ in Claim 6. So it may be that $\mathbf{x}^a = \mathbf{x}^c = \mathbf{x}^0$, say, and let

$$\Pi\left(\mathbf{x}^0\right) = \left\{\boldsymbol{\Pi} \in \Pi(P) : \mathbf{x}^0 = \boldsymbol{\Pi}\mathbf{x}^0\right\}, \tag{23}$$

be the permissible permutations for P that leave the extreme point $\mathbf{x}^0 \in P$ invariant. Since $\mathbf{I}_n \in \Pi(\mathbf{x}^0)$ this set is always nonempty and it forms evidently a subgroup of $\Pi(P)$. E.g., for the symmetric traveling salesman polytope every subgroup $\Pi(\mathbf{x}^0)$ has precisely $2m$ elements where m is the number of nodes of the graph. The $2m$ elements in $\Pi(\mathbf{x}^0)$ come about by reindexing the nodes of the graph—in both a forward and backward sense—while maintaining the same order of the nodes as in the tour $\mathbf{x}^0$.

The reduction in the number of cones (22) to be analyzed by the DDA that results from an application of Claim 6 with the special permutations in $\Pi(\mathbf{x}^0)$ can be enormous: for the symmetric traveling salesman problem with $m = 8$ nodes 59 cones (edge figures) suffice rather than the 730 original ones (which is exactly the number analyzed in [9]), for $m = 9$ we get 216 instead of 3,555 and for $m = 10$ we get 1,032 cones to analyze instead of the 19,391 original ones. By a complete application of Claim 6 these numbers can be reduced even further; see Table 5 of Sect. 7.

6 Rank of Facets and Integer Polyhedra

To bring order into the facial structure of polyhedra related to combinatorial optimization problems we introduce next the notions of the "rank" of a facet and of a polyhedron.

Let $\mathcal{F}$ be the family of row vectors $(\mathbf{h}, h_0) \in \mathbb{R}^{n+1}$ of the matrix $(\mathbf{H}, \mathbf{h})$ of an ideal description (2) of a pointed rational polyhedron $P \subseteq \mathbb{R}^n$. So every $(\mathbf{h}, h_0) \in \mathcal{F}$ defines a facet $F_{\mathbf{h}}$ of P via (15) and vice versa, for every facet of P there is some row of $(\mathbf{H}, \mathbf{h})$ that defines it. For simplicity of notation we say that $\mathcal{F}$ is ideal for P. The set

$$\operatorname{relint} P = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}, \mathbf{fx} < f_0 \ \forall (\mathbf{f}, f_0) \in \mathcal{F}\right\}$$

is the *relative interior* of P. As before we denote by vert P the set of extreme points of P and by exray P the set of the direction vectors of all extreme rays of P.

To make the notion of rank precise we need some knowledge about the facets of the facets of a polyhedron, i.e., the *ridges* of the polyhedron P. For $(\mathbf{f}, f_0) \in \mathcal{F}$ the facet

$$F_{\mathbf{f}} = \{\mathbf{x} \in P : \mathbf{fx} = f_0\}$$

of P defined by $\mathbf{fx} \leq f_0$ is itself a *pointed* polyhedron of dimension $\dim P - 1$ in $\mathbb{R}^n$ satisfying $\operatorname{vert} F_{\mathbf{f}} \subseteq \operatorname{vert} P$ and $\operatorname{exray} F_{\mathbf{f}} \subseteq \operatorname{exray} P$. Let

$$\mathcal{H}^f = \left\{(\mathbf{h}, h_0) \in \mathcal{F} : \dim F_{\mathbf{f}} \cap F_{\mathbf{h}} = \dim P - 2\right\}, \tag{24}$$

i.e., $(\mathbf{h}, h_0) \in \mathcal{H}^f$ if and only if the facet-defining inequality $\mathbf{hx} \leq h_0$ of P defines a facet of the polyhedron $F_{\mathbf{f}}$, and $(\mathbf{H}^f, \mathbf{h}^f)$ be the matrix of all $(\mathbf{h}, h_0) \in \mathcal{H}^f$. If $\dim P < 2$ then $\mathcal{H}^f = \emptyset$ and $P = \emptyset$, or $P = \{\mathbf{x}\}$ is a singleton, or P is a line segment or a halfline in $\mathbb{R}^n$. We define the rank of P to equal -1 in these cases and assume throughout that $\dim P \geq 2$ and $\operatorname{relint} P \neq \emptyset$.

Lemma 1 *For every $(\mathbf{f}, f_0) \in \mathcal{F}$ $\mathcal{H}^f$ is ideal for $F_{\mathbf{f}}$ and*

$$F_{\mathbf{f}} = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}, \mathbf{fx} = f_0, \mathbf{H}^f\mathbf{x} \leq \mathbf{h}^f\right\}$$

is an ideal linear description of $F_{\mathbf{f}}$.

Proof Suppose $\dim P \geq 2$ and that $\mathcal{H}^f$ is not complete. Then there exists some facet H of $F_{\mathbf{f}}$ such that $H \neq \{\mathbf{x} \in F_{\mathbf{f}} : \mathbf{hx} = h_0\}$ for all $(\mathbf{h}, h_0) \in \mathcal{H}^f$. Since H is a facet of $F_{\mathbf{f}}$ there exists some $(\mathbf{g}, g_0) \in \mathbb{R}^{n+1}$ such that $\mathbf{gx} \leq g_0$ for all $\mathbf{x} \in F_{\mathbf{f}}$ and $H = \{\mathbf{x} \in F_{\mathbf{f}} : \mathbf{gx} = g_0\}$. Let

$$\nu_{\mathbf{x}} = \max\left\{\frac{\mathbf{gx} - g_0}{f_0 - \mathbf{fx}} : \mathbf{x} \in \operatorname{vert} P \text{ with } \mathbf{fx} < f_0\right\} \tag{25}$$

and $\nu_{\mathbf{x}} = -\infty$ if (25) does not exist.

Let $\operatorname{vert} P = \{\mathbf{x}^1, \ldots, \mathbf{x}^p\}$ and $\operatorname{exray} P = \{\mathbf{y}^1, \ldots, \mathbf{y}^r\}$. Since $\mathbf{fx} \leq f_0$ defines a facet of P we have $\mathbf{fy}^i \leq 0$ for all $1 \leq i \leq r$. Moreover, $\mathbf{fy}^i = 0$ implies $\mathbf{gy}^i \leq 0$ for any $i \in \{1, \ldots, r\}$ since $\mathbf{gx} \leq g_0$ defines the facet H of $F_{\mathbf{f}}$. If $\nu_{\mathbf{x}} > -\infty$ and $\mathbf{gy}^i + \nu_{\mathbf{x}}\mathbf{fy}^i \leq 0$ for $1 \leq i \leq r$ then we claim that $\mathbf{hx} \leq h_0$ where $\mathbf{h} = \mathbf{g} + \nu_{\mathbf{x}}\mathbf{f}$ and $h_0 = g_0 + \nu_{\mathbf{x}} f_0$ defines a facet F' of P which is different from $F_{\mathbf{f}}$. By (25)

$$\left(\mathbf{gx}^i - g_0\right) \leq \nu_{\mathbf{x}}\left(f_0 - \mathbf{fx}^i\right) \quad \forall \mathbf{x}^i \in \operatorname{vert} P \text{ with } \mathbf{fx}^i < f_0$$

and thus $\mathbf{hx}^i = (\mathbf{g} + \nu_{\mathbf{x}}\mathbf{f})\mathbf{x}^i \leq g_0 + \nu_{\mathbf{x}} f_0 = h_0$ for $\mathbf{x}^i \in \operatorname{vert} P$ with $\mathbf{fx}^i < f_0$. If $\mathbf{x}^i \in \operatorname{vert} P$ with $\mathbf{fx}^i = f_0$ then $\mathbf{gx}^i \leq g_0$ and thus $\mathbf{hx}^i \leq h_0$ as well. Let $\mathbf{x} \in P$. Then $\mathbf{x} = \sum_{i=1}^{p} \mu_i \mathbf{x}^i + \sum_{j=1}^{r} \lambda_j \mathbf{y}^j$ for some $\mu_i \geq 0$ with $\sum_{i=1}^{p} \mu_i = 1$ and $\lambda_j \geq 0$. Thus $\mathbf{hx} \leq h_0$ for all $\mathbf{x} \in P$. Since H is a facet of $F_{\mathbf{f}}$ there exist $\ell = \dim P - 1$ affinely independent $\mathbf{x}^1, \ldots, \mathbf{x}^\ell$ with $\mathbf{x}^i \in F_{\mathbf{f}}$ and $\mathbf{hx}^i = h_0$ for $1 \leq i \leq \ell$. Let $\mathbf{x}^0$ be a vertex of P for which the maximum in (25) is attained. Then $\mathbf{hx}^0 = h_0$ and $\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^\ell$ are affinely independent because $\mathbf{fx}^0 < f_0$ and $\mathbf{fx}^i = f_0$ for $i = 1, \ldots, \ell$. Thus $\mathbf{hx} \leq h_0$

defines a facet of P which is different from $F_{\mathbf{f}}$ and $\dim F_{\mathbf{f}} \cap F_{\mathbf{h}} = \dim P - 2$. Since $\mathcal{F}$ is ideal for P, there exists some $(\mathbf{f}', f_0') \in \mathcal{F}$ such that $F' = \{\mathbf{x} \in P : \mathbf{f}'\mathbf{x} = f_0'\} = \{\mathbf{x} \in P : \mathbf{h}\mathbf{x} = h_0\}$. But

$$\begin{aligned} F_{\mathbf{f}} \cap F' &= \left\{\mathbf{x} \in P : \mathbf{f}\mathbf{x} = f_0, \mathbf{f}'\mathbf{x} = f_0'\right\} = \{\mathbf{x} \in P : \mathbf{f}\mathbf{x} = f_0, \mathbf{h}\mathbf{x} = h_0\} \\ &= \{\mathbf{x} \in P : \mathbf{f}\mathbf{x} = f_0, \mathbf{g}\mathbf{x} = g_0\} = H, \end{aligned}$$

contradicting $H \neq \{\mathbf{x} \in F_{\mathbf{f}} : \mathbf{h}\mathbf{x} = h_0\}$ for all $(\mathbf{h}, h_0) \in \mathcal{H}^f$. Thus $\mathcal{H}^f$ is complete for $F_{\mathbf{f}}$. Suppose that $\mathbf{g}\mathbf{y}^i + \nu_{\mathbf{x}}\mathbf{f}\mathbf{y}^i > 0$ for some $1 \le i \le r$ or that $\nu_{\mathbf{x}} = -\infty$. If $\nu_{\mathbf{x}} = -\infty$ then $\mathbf{f}\mathbf{x} = f_0$ for all $\mathbf{x} \in \operatorname{vert} P$. If $\mathbf{f}\mathbf{y} = 0$ for all $\mathbf{y} \in \operatorname{exray} P$, then $F_{\mathbf{f}} = P$ which contradicts $\dim F_{\mathbf{f}} < \dim P$ and thus $\mathbf{f}\mathbf{y}^i < 0$ for some $i \in \{1, \dots, r\}$. So suppose $\nu_{\mathbf{x}} > -\infty$ and $\mathbf{g}\mathbf{y}^i + \nu_{\mathbf{x}}\mathbf{f}\mathbf{y}^i > 0$ for some $i \in \{1, \dots, r\}$. If $\mathbf{f}\mathbf{y}^i = 0$ for all i then $0 < \mathbf{g}\mathbf{y}^i + \nu_{\mathbf{x}}\mathbf{f}\mathbf{y}^i \le 0$ for some $i \in \{1, \dots, r\}$ is a contradiction because $\mathbf{g}\mathbf{y}^i \le 0$ for all i in this case as well. Thus $\mathbf{f}\mathbf{y}^i < 0$ for at least one $i \in \{1, \dots, r\}$ in both subcases and the scalar

$$\nu_{\mathbf{y}} = \max\left\{-\frac{\mathbf{g}\mathbf{y}^i}{\mathbf{f}\mathbf{y}^i} : \mathbf{f}\mathbf{y}^i < 0, 1 \le i \le r\right\}$$

is well defined in the second case. We set $\nu = \max\{\nu_{\mathbf{x}}, \nu_{\mathbf{y}}\}$ and claim $\mathbf{h}\mathbf{x} \le h_0$ with $\mathbf{h} = \mathbf{g} + \nu\mathbf{f}$ and $h_0 = g_0 + \nu f_0$ defines a facet F' of P with $F' \neq F_{\mathbf{f}}$. By the definition of ν

$$\mathbf{h}\mathbf{y}^i = \mathbf{g}\mathbf{y}^i + \nu\mathbf{f}\mathbf{y}^i \le 0 \quad \forall i \in \{1, \dots, r\} \text{ with } \mathbf{f}\mathbf{y}^i < 0$$

and $\mathbf{h}\mathbf{y}^i \le 0$ for all $i \in \{1, \dots, r\}$ with $\mathbf{f}\mathbf{y}^i = 0$, since $\mathbf{g}\mathbf{y}^i \le 0$ in this case. Thus $\mathbf{h}\mathbf{y} \le 0$ for all $\mathbf{y} \in \operatorname{exray} P$. Like above, using $\nu \ge \nu_{\mathbf{x}}$, we show that $\mathbf{h}\mathbf{x} \le h_0$ for all $\mathbf{x} \in \operatorname{vert} P$ and thus $\mathbf{h}\mathbf{x} \le h_0$ for all $\mathbf{x} \in P$. If $\nu = \nu_{\mathbf{x}}$ then the claim follows like in the first case and we are done. If $\nu = \nu_{\mathbf{y}}$ let $\mathbf{y} \in \operatorname{exray} P$ be such that equality in the definition of $\nu_{\mathbf{y}}$ is attained. Since H is a facet of $F_{\mathbf{f}}$ there exist $\ell = \dim P - 1$ affinely independent $\mathbf{x}^1, \dots, \mathbf{x}^\ell$ with $\mathbf{x}^i \in F_{\mathbf{f}}$ and $\mathbf{h}\mathbf{x}^i = h_0$ for $1 \le i \le \ell$. Moreover, by construction $\mathbf{h}(\mathbf{x}^i + \alpha\mathbf{y}) = h_0$ for $1 \le i \le \ell$ and $\alpha \ge 0$. Thus $\mathbf{x}^1, \dots, \mathbf{x}^\ell, \mathbf{x}^1 + \mathbf{y}$ are all in P and affinely independent. For suppose not. Then $\lambda_0(\mathbf{x}^1 + \mathbf{y}) + \sum_{i=1}^{\ell} \lambda_i \mathbf{x}^i = \mathbf{0}$ for some nonnull $\lambda_0, \lambda_1, \dots, \lambda_\ell$ with $\lambda_0 + \sum_{i=1}^{\ell} \lambda_i = 0$ and thus $\lambda_0 \neq 0$ since $\mathbf{x}^1, \dots, \mathbf{x}^\ell$ are affinely independent. Thus $\mathbf{x}^1 + \mathbf{y} = \sum_{i=1}^{\ell} \lambda_i' \mathbf{x}^i$ for some $\lambda_1', \dots, \lambda_\ell'$ with $\sum_{i=1}^{\ell} \lambda_i' = 1$ and $\mathbf{f}\mathbf{y} = 0$ because $\mathbf{f}\mathbf{x}^i = f_0$ for $1 \le i \le \ell$. This contradicts $\mathbf{f}\mathbf{y} < 0$ and we conclude like in the first case that $\mathcal{H}^f$ is complete for $F_{\mathbf{f}}$.

Suppose that $\mathcal{H}^f$ is not minimal. Since $\mathcal{H}^f$ is complete for $F_{\mathbf{f}}$ there exists $(\mathbf{h}, h_0) \in \mathcal{H}^f$ and a facet $H_{\mathbf{h}}^{\mathbf{f}}$ of $F_{\mathbf{f}}$ such that

$$H_{\mathbf{h}}^{\mathbf{f}} = \left\{\mathbf{x} \in P : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{f}\mathbf{x} = f_0, \mathbf{h}\mathbf{x} = h_0, \mathbf{g}^1\mathbf{x} = g_0^1, \dots, \mathbf{g}^k\mathbf{x} = g_0^k\right\}$$

for $(\mathbf{g}^1, g_0^1), \dots, (\mathbf{g}^k, g_0^k) \in \mathcal{H}^f - (\mathbf{h}, h_0)$ and some $k \ge 1$. Since $\dim H_{\mathbf{h}}^{\mathbf{f}} = \dim P - 2 = n - r(\mathbf{A}) - 2$ it follows that $\mathbf{g}^i = \boldsymbol{\lambda}^i\mathbf{A} + \alpha^i\mathbf{f} + \beta^i\mathbf{h}$ and $g_0^i = \boldsymbol{\lambda}^i\mathbf{b} + \alpha^i f_0 + \beta^i h_0$ for some vectors $\boldsymbol{\lambda}^i$ and scalars α^i and β^i, i.e., the $(\mathbf{g}^i, g_0^i)$ are linearly dependent on $(\mathbf{A}, \mathbf{b})$, $(\mathbf{f}, f_0)$ and $(\mathbf{h}, h_0)$. Let $\mathbf{x}^{\mathbf{f}}$ and $\mathbf{x}^{\mathbf{h}}$ be any points in $\operatorname{relint} F_{\mathbf{f}}$ and $\operatorname{relint} F_{\mathbf{h}}$, respectively. It follows from $F_{\mathbf{h}} \neq F_{\mathbf{f}}$ that $\mathbf{hx^h} = h_0$, $\mathbf{fx^h} < f_0$, $\mathbf{g}^i\mathbf{x^h} < g_0^i$ and $\mathbf{fx^f} = f_0$, $\mathbf{hx^f} < h_0$, $\mathbf{g}^i\mathbf{x^f} < g_0^i$, for all $i = 1, \ldots, k$. Thus, e.g.,

$$\mathbf{g}^i\mathbf{x^h} = \boldsymbol{\lambda}^i\mathbf{Ax^h} + \alpha^i\mathbf{fx^h} + \beta^i\mathbf{hx^h} = \boldsymbol{\lambda}^i\mathbf{b} + \alpha^i\mathbf{fx^h} + \beta^i h_0 < g_0^i = \boldsymbol{\lambda}^i\mathbf{b} + \alpha^i f_0 + \beta^i h_0.$$

Consequently, $\alpha^i(f_0 - \mathbf{fx^h}) > 0$ and $\alpha^i > 0$. Multiplying by $\mathbf{x^f}$ we find likewise that $\beta^i > 0$. Consequently, $\mathbf{g}^i\mathbf{x} = g_0^i$ if and only if $\mathbf{fx} = f_0$ and $\mathbf{hx} = h_0$ for all $\mathbf{x} \in P$. Thus none of the $\mathbf{g}^i\mathbf{x} \leq g_0^i$ defines a facet of P. This contradicts the assumption that $\mathcal{F}$ is ideal for P. □

The last part of the proof of the lemma shows also that every ridge of P is the intersection of two *unique* facets of P. A shorter, nonconstructive proof of this fact is possible using the methods of Chap. 7.2.2 of [38]. The proof given here is useful: given a pointwise generator of P it describes a *lifting procedure* to compute facets of P from the facets of a facet of P.

Definition 6 For every $(\mathbf{f}, f_0) \in \mathcal{F}$ we call the polyhedron

$$P(\mathcal{H}^f) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}, \mathbf{fx} \leq f_0, \mathbf{H}^f\mathbf{x} \leq \mathbf{h}^f\} \tag{26}$$

the *facet figure* of P at the facet $F_{\mathbf{f}}$.

Clearly, $P(\mathcal{H}^f) \neq \emptyset$, $P(\mathcal{H}^f)$ is pointed, the linear description of $P(\mathcal{H}^f)$ is ideal,

$$F_{\mathbf{f}} \subsetneq P \subsetneq P(\mathcal{H}^f) \quad \text{and} \quad P = \bigcap_{(\mathbf{f}, f_0) \in \mathcal{F}} P(\mathcal{H}^f).$$

Let $P \subseteq \mathbb{R}^n$ be an *integer polyhedron*, i.e., every $\mathbf{x} \in \operatorname{vert} P$ satisfies $\mathbf{x} \in \mathbb{Z}^n$, and moreover, relint $P \neq \emptyset$. For any such polyhedron let $\mathcal{F}_{\min} \subseteq \mathcal{F}$ be such that

$$P_{\min} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}, \mathbf{fx} \leq f_0 \ \forall (\mathbf{f}, f_0) \in \mathcal{F}_{\min}\} \tag{27}$$

meets the following two requirements

(i) $P = \operatorname{conv}(P_{\min} \cap \mathbb{Z}^n)$
(ii) $P \cap \mathbb{Z}^n \subsetneq P_{\min}^{\mathbf{h}} \cap \mathbb{Z}^n \ \forall (\mathbf{h}, h_0) \in \mathcal{F}_{\min}$, where

$$P_{\min}^{\mathbf{h}} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}, \mathbf{fx} \leq f_0 \ \forall (\mathbf{f}, f_0) \in \mathcal{F}_{\min} - (\mathbf{h}, h_0)\}$$

and the containment in (ii) is proper. Given $\mathcal{F}$ we can construct $\mathcal{F}_{\min}$ e.g. as follows: Initially we set $\mathcal{F}_{\min} = \mathcal{F}$. If dropping $(\mathbf{h}, h_0)$ does not change the convex hull (i), we drop it from $\mathcal{F}_{\min}$ and continue to do so until every remaining element in $\mathcal{F}_{\min}$ satisfies (ii). Since $|\mathcal{F}| < \infty$ this procedure is finite. By construction we have $\dim P_{\min} = \dim P$, $P_{\min}$ is pointed, $\mathcal{F}_{\min} \neq \emptyset$ since relint $P \neq \emptyset$.

We call any subset $\mathcal{F}_{\min} \subseteq \mathcal{F}$ satisfying (i) and (ii) a *minimal formulation* for the integer polyhedron. Minimal formulations of integer polyhedra need not be unique.

An example of nonuniqueness is given by the *set packing* or *vertex packing polytope*

$$VP(G) = \text{conv}\big(\{\mathbf{x} \in \{0,1\}^{|V|} : x_u + x_v \le 1 \ \forall e = (u,v) \in E\}\big),$$

where $G = (V, E)$ is a finite undirected graph on $n = |V|$ nodes; see [32, 33, 35–37]. In this case every minimal "covering" of the edge set E of G by some of its "cliques" (= maximal complete vertex-induced subgraphs) together with the non-negativity conditions provides a minimal formulation for the integer polytope and a *unique* minimal formulation simply does not exist in the general case. On the other hand, in many cases of interest to us, such as, e.g., in the case of the symmetric and asymmetric traveling salesman polytopes, the corresponding minimal formulations appear to be unique (see Sect. 7).

Given an ideal family $\mathcal{F}$ for an integer polyhedron $P \subseteq \mathbb{R}^n$ we call

$$\mathcal{F}_0 = \big\{(\mathbf{f}, f_0) \in \mathcal{F} : (\mathbf{f}, f_0) \text{ belongs to } \textit{some} \text{ minimal formulation } \mathcal{F}_{\min} \text{ of } P\big\}$$

the *rank zero facets* of P. It follows that $\mathcal{F}_0$ is a uniquely defined subset of $\mathcal{F}$. We call

$$P_0 = \big\{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{f}\mathbf{x} \le f_0 \ \forall (\mathbf{f}, f_0) \in \mathcal{F}_0\big\}$$

the *rank zero formulation* of P. In the case of the set packing or vertex packing polytope

$$\begin{aligned} VP(G) &= \text{conv}\big(\{\mathbf{x} \in \mathbb{Z}^n : \mathbf{A}_C\mathbf{x} \le \mathbf{e}_C, \mathbf{x} \ge \mathbf{0}\}\big) \\ &= \big\{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}_C\mathbf{x} \le \mathbf{e}_C, \mathbf{A}_F\mathbf{x} \le \mathbf{f}^0, \mathbf{x} \ge \mathbf{0}\big\}, \end{aligned}$$

where $\mathbf{A}_C\mathbf{x} \le \mathbf{e}_C$ are all *clique* constraints, $\mathbf{x} \ge \mathbf{0}$ are the nonnegativity constraints and $\mathbf{A}_F\mathbf{x} \le \mathbf{f}^0$ all other facet-defining inequalities of $VP(G)$ in primitive normal form. Every clique constraint is of the form $\sum_{j \in K} x_j \le 1$ where $K \subseteq V$ is the node set of a clique in G. Every row $\mathbf{f}\mathbf{x} \le f_0$ of $\mathbf{A}_F\mathbf{x} \le \mathbf{f}^0$ consists of relatively prime integers $0 \le f_j \le f_0$ for all $j \in V$ and $f_0 \ge 2$. $\mathcal{F}_0$ consists of *all* clique inequalities of G plus *all* nonnegativity constraints. [The referee pointed out that a formal proof is needed that no facet-defining inequality with $f_0 \ge 2$ belongs to $\mathcal{F}_0$. I agree with the referee, but have at present no such proof. This makes the notion of rank proposed here, possibly, formulation-dependent. However, I continue to think that this is not the case.]

If $\mathcal{F} = \mathcal{F}_0$ then all facets of P have rank zero and we define the rank $\rho(P)$ of the polyhedron P to be zero. Examples of integer polyhedra of rank zero are the n-dimensional simplex S_n, the unit hypercube C_n

$$S_n = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \ge \mathbf{0}, \sum_{j=1}^{n} x_j \le 1\right\}, \qquad C_n = \big\{\mathbf{x} \in \mathbb{R}^n : 0 \le x_j \le 1 \text{ for } 1 \le j \le n\big\}$$

and the vertex packing polytope $VP(G) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}_C\mathbf{x} \le \mathbf{e}_C, \mathbf{x} \ge \mathbf{0}\}$ of a perfect graph G where $\mathbf{A}_C$ is defined above, see [33, 34]. There are other examples, e.g., polyhedra that are defined with respect to totally unimodular, ideal 0–1 matrices, etc.

If $\mathcal{F}_0 \neq \mathcal{F}$ then we proceed inductively as follows. Given $\rho + 1$ nonempty, pairwise disjoint subsets $\mathcal{F}_0, \ldots, \mathcal{F}_\rho$ of $\mathcal{F}$ we let

$$P_\rho = \left\{ \mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}, \mathbf{hx} \leq h_0 \ \forall (\mathbf{h}, h_0) \in \bigcup_{\ell=0}^{\rho} \mathcal{F}_\ell \right\}. \tag{28}$$

A facet of P defined by $(\mathbf{f}, f_0) \in \mathcal{F}_\ell$ is a facet of rank ℓ and $\mathbf{fx} \leq f_0$ is a rank ℓ inequality where $0 \leq \ell \leq \rho$. If $\mathcal{F} = \bigcup_{\ell=0}^{\rho} \mathcal{F}_\ell$ then $\rho(P) = \rho$, i.e., the integer polyhedron P has rank ρ. Otherwise, let

$$\mathcal{F}_{\rho+1} = \left\{ (\mathbf{f}, f_0) \in \mathcal{F} - \bigcup_{\ell=0}^{\rho} \mathcal{F}_\ell : \right.$$

$$\left. \exists (\mathbf{h}, h_0) \in \bigcup_{\ell=0}^{\rho} \mathcal{F}_\ell \text{ s.t. } \dim F_{\mathbf{f}} \cap F_{\mathbf{h}} = \dim P - 2 \right\} \tag{29}$$

Theorem 1 *If $\mathcal{F} \neq \bigcup_{\ell=0}^{\rho} \mathcal{F}_\ell$, then $\mathcal{F}_{\rho+1} \neq \emptyset$ and $\mathbf{x}^* \notin P_{\rho+1}$ for all $\mathbf{x}^* \in \operatorname{vert} P_\rho - \mathbb{Z}^n$.*

Proof If $\mathcal{F} \neq \bigcup_{\ell=0}^{\rho} \mathcal{F}_\ell$, then $P \neq P_\rho$ and there exists $\mathbf{x}^* \in \operatorname{vert} P_\rho - \mathbb{Z}^n$. Since $\mathbf{x}^* \in \operatorname{vert} P_\rho$ there exists $(\mathbf{h}, h_0) \in \bigcup_{\ell=0}^{\rho} \mathcal{F}_\ell$ such that $\mathbf{hx}^* = h_0$. If $\mathbf{x}^* \in F_{\mathbf{h}}$, then $\mathbf{x}^* \in P$ since $F_{\mathbf{h}} \subset P$, which contradicts $\mathbf{x}^* \notin P$. Consequently, $\mathbf{x}^* \notin F_{\mathbf{h}}$ and by the lemma there exists a $(\mathbf{f}, f_0) \in \mathcal{H}^h$ such that $\mathbf{fx}^* > f_0$ and thus $(\mathbf{f}, f_0) \notin \bigcup_{\ell=0}^{\rho} \mathcal{F}_\ell$, i.e., $(\mathbf{f}, f_0) \in \mathcal{F}_{\rho+1}$. Since $\mathbf{x}^* \in \operatorname{vert} P_\rho - \mathbb{Z}^n$ is arbitrary, the theorem follows. □

Since $|\mathcal{F}| < \infty$ and since we augment the set $\bigcup_{\ell=0}^{\rho} \mathcal{F}_\ell$ at every step of the above inductive process by at least one new element of $\mathcal{F}$, the inductive process is finite. Hence the rank $\rho(P)$ of every integer polyhedron $P \subseteq \mathbb{R}^n$ is some well-defined finite number. Moreover, the process produces a finite sequence of polyhedra $P_0, P_1, \ldots$ satisfying

$$P_0 \supsetneq P_1 \supsetneq \cdots \supsetneq P_{\rho(P)} = P$$

with the property that all *noninteger* extreme points $\mathbf{x}^* \in P_\ell$ are eliminated from $P_{\ell+1}$ where $0 \leq \ell < \rho(P)$ is arbitrary.

The notion of rank introduced here remains correct for *mixed-integer rational polyhedra* $P \subseteq \mathbb{R}^n$. In the case of mixed-integer polyhedra we have a partitioning of $\mathbf{x} \in \mathbb{R}^n$ into n_1, say, "integer" variables $\boldsymbol{\xi} \in \mathbb{R}^{n_1}$ and $n - n_1$ "flow" variables $\boldsymbol{\phi} \in \mathbb{R}^{n-n_1}$. The requirement $\mathbf{x} \in \mathbb{Z}^n$ is replaced by $\boldsymbol{\xi} \in \mathbb{Z}^{n_1}$. With the corresponding notational changes we define the notions of the rank of facets as well as of a mixed-integer rational polyhedron like above; see [40] and Chaps. 10.2–10.3 of [38] for details.

In Fig. 3 we show two families of polytopes in $\mathbb{R}^2$ of rank one and two, respectively. The integer polytope on the top is given by

$$P = \operatorname{conv}\left\{ \mathbf{x} \in \mathbb{Z}^2 : -Mx_1 + x_2 \leq 1, Mx_1 + x_2 \leq 3M + 1, x_2 \geq 0 \right\},$$

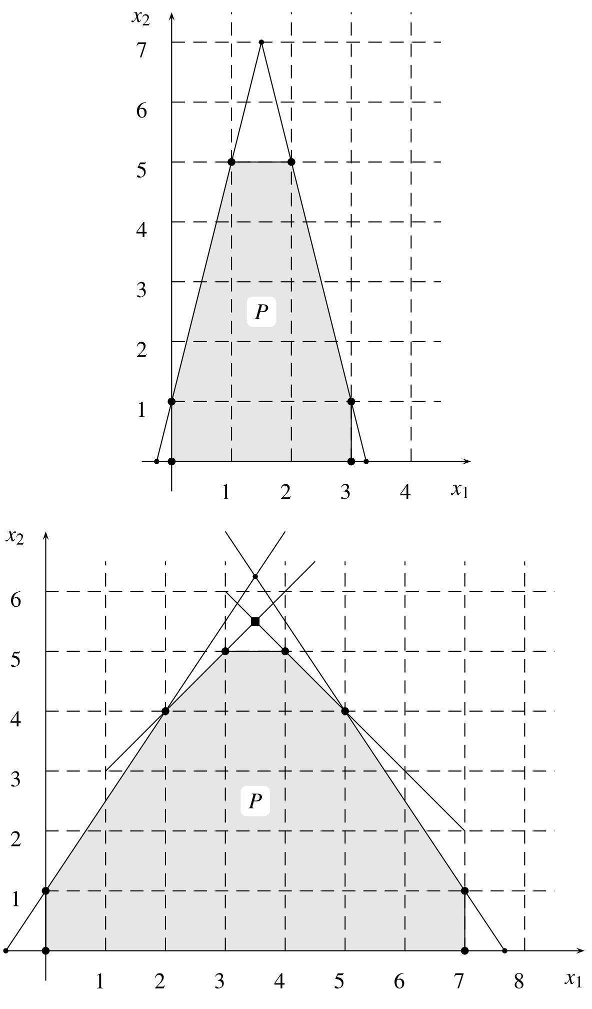


Fig. 3 Polytopes in $\mathbb{R}^2$ of rank one and two

where $M \geq 2$ is an arbitrary integer number, and the one on the bottom by

$$P = \operatorname{conv}\{\mathbf{x} \in \mathbb{Z}^2 : -Mx_1 + 2x_2 \leq 2,\ Mx_1 + 2x_2 \leq 7M + 2,\ x_2 \geq 0\},$$

where $M \geq 3$ is an arbitrary *odd* integer number. The corresponding minimal formulations are the polytopes P_0 given by the respective linear programming relaxations of the two constraint sets shown here and the corresponding subfamily $\mathcal{F}_0$ of $\mathcal{F}$ is unique in both cases for the permitted values of the parameter M. The polytope on the top has three facets of rank 0 and three facets of rank 1, namely $x_1 \geq 0$, $x_1 \leq 3$, and $x_2 \leq M + 1$. The polytope on the bottom has three facets of rank 0 as well. The inequalities $x_1 \geq 0$, $x_1 \leq 7$, $-\lfloor M/2 \rfloor x_1 + x_2 \leq 2$, and $\lfloor M/2 \rfloor x_1 + x_2 \leq 7\lfloor M/2 \rfloor + 2$ define the four facets of rank 1 of the corresponding polytope while the facet defined by $x_2 \leq 3\lfloor M/2 \rfloor + 2$ has a rank of two. The bottom figure also shows the polytopes

P_0, P_1 and $P_2 = P$ and their respective containment as an illustration of the second part of the theorem.

Note 1 Different notions of the rank of an integer polyhedron can be found, e.g., in the books by Nemhauser and Wolsey (Chap. II.1.2 in [30]) and Schrijver (Chap. 23.4 in [48]). Their concepts are based on the algorithm for integer programming due to Gomory [16]; see also [15] and [10]. If one uses the Nemhauser and Wolsey notion, the rank of a facet of a *flat* integer polyhedron may differ according to its different representations by linear inequalities which is absolutely undesirable. An example to this effect are the following three inequalities

$$\begin{aligned}
\mathbf{a}^1\mathbf{x} &:= x_{12} + x_{13} + x_{14} + x_{23} + x_{25} + x_{36} \leq 4 \\
\mathbf{a}^2\mathbf{x} &:= x_{12} + x_{13} + 2x_{14} + x_{23} + 2x_{25} + 2x_{36} + x_{45} + x_{46} + x_{56} \leq 8 \\
\mathbf{a}^3\mathbf{x} &:= x_{12} - 4x_{13} + 6x_{14} - 4x_{14} + x_{16} + 6x_{23} - 4x_{24} - 4x_{25} \\
&\quad\; + x_{26} + x_{34} + x_{35} - 4x_{36} + x_{45} - 4x_{46} + 6x_{56} \\
&\leq 16
\end{aligned}$$

which define the same facet of $\mathcal{Q}_T^6$, i.e., the symmetric traveling salesman polytope on 6 cities. Since $\max\{\mathbf{a}^1\mathbf{x} : \mathbf{x} \in \mathcal{Q}_S^6\} = 4.5$, where $\mathcal{Q}_S^6$ is the relaxation with all subtour elimination constraints, the inequality $\mathbf{a}^1\mathbf{x} \leq 4$ gets a rank of 1 in this notion, since $\max\{\mathbf{a}^2\mathbf{x} : \mathbf{x} \in \mathcal{Q}_S^6\} = 9$, the inequality $\mathbf{a}^2\mathbf{x} \leq 8$ has probably a rank of 2 and since $\max\{\mathbf{a}^3\mathbf{x} : \mathbf{x} \in \mathcal{Q}_S^6\} = 21$ the inequality $\mathbf{a}^3\mathbf{x} \leq 16$ has a rank of at least two according to this notion. Yet they all define the same facet of $\mathcal{Q}_T^6$. Thus the rank of $\mathcal{Q}_T^6$ that we get from this concept depends entirely upon the *representation* of the facets of $\mathcal{Q}_T^6$ that we work with. Such an outcome is not possible with the notion of rank that we propose here because the inductive process does not work with any particular representation of the facets of P. Schrijver's notion of rank gets formally around the representation dependency by choosing a formulation of P and determining the rank of P relative to the given formulation. In the case of *flat* polyhedra many different formulations of an integer polyhedron exist and it is not known in what way the resulting "rank" depends on the formulation. Like the previous one his notion employs Gomory's 1958 algorithm which has been known from its beginnings to be a bad algorithm for integer programs. In either case, if one calculates, for instance, the rank of the top integer polytopes of Fig. 3 from the textbooks, one finds a rank of something like M which is absurd and easily explained by the poor convergence properties of Gomory's algorithm; see [33]. Except in rare instances, the textbook notions and the concept of the rank of integer P introduced here will lead to radically different conclusions about the rank and the complexity of the facial structure of integer polyhedra; see [40] for more on this.

The reason for the irrelevancy of the textbook notions for the rank of integer polyhedra and especially of those related to combinatorial problems can be explained as follows: Let $\mathbf{A}$ be any matrix of rationals of size $m \times n$ and of rank m, $\mathbf{b}$ a vector of

m rationals and

$$P = \mathrm{conv}\left\{\mathbf{x} \in \mathbb{Z}^n : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\right\} \subseteq P_{LP} = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\right\}.$$

For any $\boldsymbol{\mu} \in \mathbb{R}^m$ it follows that $\lfloor \boldsymbol{\mu}\mathbf{A} \rfloor \leq \boldsymbol{\mu}\mathbf{A}$ where $\lfloor \alpha \rfloor$ is the largest integer less than or equal to α and $\lfloor \boldsymbol{\mu}\mathbf{A} \rfloor$ is the same componentwise. Since $\mathbf{x} \geq \mathbf{0}$ we get $\lfloor \boldsymbol{\mu}\mathbf{A} \rfloor \mathbf{x} \leq \boldsymbol{\mu}\mathbf{A}\mathbf{x}$, $\boldsymbol{\mu}\mathbf{A}\mathbf{x} = \boldsymbol{\mu}\mathbf{b}$ for all for all $\mathbf{x} \in \mathbb{R}^n$ with $\mathbf{A}\mathbf{x} = \mathbf{b}$ and thus $P \subseteq P_{LP} \subseteq \{\mathbf{x} \in \mathbb{R}^n : \lfloor \boldsymbol{\mu}\mathbf{A} \rfloor \mathbf{x} \leq \boldsymbol{\mu}\mathbf{b}\}$. But $\lfloor \boldsymbol{\mu}\mathbf{A} \rfloor \mathbf{x} \in \mathbb{Z}$ and $\lfloor \boldsymbol{\mu}\mathbf{A} \rfloor \mathbf{x} \leq \lfloor \boldsymbol{\mu}\mathbf{b} \rfloor$ for all $\mathbf{x} \in \mathbb{Z}^n$, and *any* $\mathbf{x} \in \mathbb{Z}^n$ with $\lfloor \boldsymbol{\mu}\mathbf{A} \rfloor \mathbf{x} \leq \boldsymbol{\mu}\mathbf{b}$ stops the rounddown of $\boldsymbol{\mu}\mathbf{b}$ to $\lfloor \boldsymbol{\mu}\mathbf{b} \rfloor$. Thus, coincidentally, $\lfloor \boldsymbol{\mu}\mathbf{A} \rfloor \mathbf{x} \leq \lfloor \boldsymbol{\mu}\mathbf{b} \rfloor$ for all $\mathbf{x} \in P$ as well. Hence no particularities of P or of a specific combinatorial problem are used in the derivation of these "cuts". To get Gomory's classical "fractional" cut, just choose $\boldsymbol{\mu} = \mathbf{u}^i\mathbf{B}^{-1}$ where $\mathbf{B}$ is a (feasible) basis for the associated linear program, $\mathbf{u}^i \in \mathbb{R}^m$ any i^{th} unit vector such that $\mathbf{u}^i\mathbf{B}^{-1}\mathbf{b} \notin \mathbb{Z}$ and do the algebra or consult [39] or do both. The fact that Gomory's 1958 cuts worked for Edmonds' (1965) matching polytope [13] is absolutely no reason to expect the same for other combinatorial problems like for vertex packing or traveling salesman polytopes and even less for general integer polyhedra.

7 The Facial Structure of "Small" STS Polytopes

We have applied the foregoing methodology to analyze the facial structure of symmetric traveling salesman (STS) polytopes where the underlying graph is complete and has up to 10 nodes. As it is usual we denote from now on by n the number of nodes of the graph and thus the dimension of the space that we work in equals $n(n-1)/2$.

The analysis of "small" STS polytopes has quite some history which begins apparently with work done in the 1950s—see [23, 26] and [31]. These early works were thoroughly forgotten, though, until Martin Grötschel uncovered them during an extensive literature search while writing his dissertation around 1975. In the case of the STS polytope complete descriptions of Q_T^n for $n \leq 6$ and a partial description for $n = 7$ were known in the fifties. The case $n = 7$ was completed by Boyd and Cunningham [3]. For $n = 8$ Naddef and Rinaldi [28] obtained among other results all but 60,480 of the 194,187 facets of this polytope, the complete set was computed and published by Christof, Jünger, and Reinelt [9]. Until very recently, I believed that my calculations for $n = 9$ and $n = 10$ were new. However, as Gerd Reinelt [46] pointed out to me, identical results for these two cases, see Table 1, were obtained independently by him and Christof using a similar methodology, see, e.g., [6, 8].

Similar work has been carried out for the asymmetric traveling salesman polytope, see [2, 14, 27], the multicut polytope, see [12], the linear ordering polytope, see [21], the minimum cut polyhedron, see [1], the quadratic assignment polytope, see [25] and [43], and possibly other polytopes or polyhedra as well. All of these works were concerned with finding explicit "lists" of all facet-defining inequalities for the respective problems. Here we study the STS polytope with up to 10 nodes.

Table 1 The facial structure of the polytopes Q_T^n for $3 \le n \le 10$

Nodes	3	4	5	6	7	8	9	10	n
Variables	3	6	10	15	21	28	36	45	$\frac{1}{2}n(n-1)$
$\dim Q_T^n$	0	2	5	9	14	20	27	35	$\frac{1}{2}n(n-3)$
$\pi(n)$	1	3	12	60	360	2,520	20,160	181,440	$\frac{1}{2}(n-1)!$
$\pi_0(n)$	0	2	10	41	168	730	3,555	19,391	?
$\phi(n)$	0	3	20	100	3,437	194,187	42,104,442	$\geq$51,043,900,866	?
$\phi_0(n)$	0	2	10	27	196	2,600	88,911	$\geq$13,607,980	?
$\kappa(Q_T^n)$	0	1	2	4	6	24	192	$\geq$15,379	?
$\rho(Q_T^n)$	-1	0	0	1	2	3	4	?	?
$\operatorname{diam} Q_T^n$	0	1	2	2	2	2	2	2	$\leq$4

It remains to bring some order into the resulting information about this incredibly complex family of polytopes; see Table 1.

Let $\mathbf{A}$ be the node *versus* edge incidence matrix of the complete graph $G = (V, E)$, say, having n nodes and $n(n-1)/2$ edges. As usually, we denote by

$$Q_A^n = \left\{\mathbf{x} \in \mathbb{R}^{n(n-1)/2} : \mathbf{Ax} = \mathbf{2}, \mathbf{0} \le \mathbf{x} \le \mathbf{1}\right\}$$

the *assignment* or *2-matching relaxation* of the STS problem. Q_A^n has integer and noninteger extreme points. The incidence vector of every tour in G is an extreme point of Q_A^n, but Q_A^n also has integer extreme points that correspond to subtours. These are ruled out by the subtour elimination constraints (SECs) of Dantzig, Fulkerson, and Johnson [11]. They give rise to the *subtour polytope relaxation* of the STS problem

$$Q_S^n = \left\{\mathbf{x} \in \mathbb{R}^{n(n-1)/2} : \mathbf{Ax} = \mathbf{2}, \mathbf{x}(S) \le |S| - 1 \text{ for } S \subset V \text{ with } |S| \ge 2, \mathbf{x} \ge \mathbf{0}\right\},$$

where $\mathbf{x}(S)$ is the sum of all components x_e of $\mathbf{x}$ such that e is an edge of G with both endpoints in the set $S \subset V = \{1, \dots, n\}$.

The polytope Q_S^n provides a formulation for the STS problem: every integer extreme point of Q_S^n corresponds to the incidence vector of some tour in G and vice versa. Thus the STS polytope, i.e., the convex hull of the incidence vectors of all tours in G, satisfies

$$Q_T^n = \operatorname{conv}\left(Q_S^n \cap \mathbb{Z}^{n(n-1)/2}\right).$$

The STS problem consists of minimizing some linear function over the polytope Q_T^n which has a dimension of $n(n-3)/2$ for all $n \ge 3$; see, e.g., [17].

There are $2^n - n - 2$ possible SECs in a complete graph and not all of them are needed because $\mathbf{x}(S) \le |S| - 1$ if and only if $\mathbf{x}(V - S) \le |V - S| - 1$ for all $\mathbf{x} \in Q_A^n$ and nonempty $S \subset V$. It was shown among other results by Grötschel and Padberg [17–20] that the SECs define precisely $2^{n-1} - n - 1$ distinct facets of Q_T^n for all $n \ge 5$. Moreover, the corresponding smaller set of SECs together with the equations $\mathbf{Ax} = \mathbf{2}$ and the nonnegativity $\mathbf{x} \ge \mathbf{0}$ suffice to formulate the STS problem correctly.

We also showed that the nonnegativity conditions define distinct facets of $\mathcal{Q}_T^n$ for all $n \geq 5$ that are distinct from those defined by the SECs.

For the purpose of a rank analysis of the facets of $\mathcal{Q}_T^n$, we take the reduced set of SECs plus the nonnegativity constraints as the rank zero facets of $\mathcal{Q}_T^n$. They constitute the family $\mathcal{F}_0$ and the subtour polytope $\mathcal{Q}_S^n$ is a minimal formulation $\mathcal{F}_{\min}^S$ for $\mathcal{Q}_T^n$. It is readily shown using [18, 19] that $\mathcal{F}_{\min}^S$ satisfies the requirements (27) of Sect. 6 and thus $\mathcal{F}_{\min}^S \subseteq \mathcal{F}_0$ as defined there. I state that $\mathcal{F}_{\min}^S = \mathcal{F}_0$ and thus that the STS problem has a *unique* minimal formulation. [The referee pointed out that $\mathcal{F}_{\min}^S = \mathcal{F}_0$ needs a formal proof and I agree. However, I will leave such a formal proof or (or disproof ?) as "food for thought" for the younger talents in our field.]

In Table 1 we summarize the key characteristics of the facial structure of the STS polytopes $\mathcal{Q}_T^n$ for $3 \leq n \leq 10$. We use the following notation:

$$\begin{aligned}
\pi(n) &= \text{the number of tours (extreme points) of } \mathcal{Q}_T^n,\\
\pi_0(n) &= \text{the number of tours } \textit{adjacent} \text{ to any given tour of } \mathcal{Q}_T^n,\\
\phi(n) &= \text{the number of facets of } \mathcal{Q}_T^n,\\
\phi_0(n) &= \text{the number of facets that are tight at any given tour of } \mathcal{Q}_T^n,\\
\kappa\big(\mathcal{Q}_T^n\big) &= \text{the class number of different facet types of } \mathcal{Q}_T^n,\\
\rho\big(\mathcal{Q}_T^n\big) &= \text{the rank of } \mathcal{Q}_T^n.
\end{aligned}$$

diam $\mathcal{Q}_T^n$ is the diameter of the STS polytope $\mathcal{Q}_T^n$, i.e., the maximum over all ordered pairs $(\mathbf{x}, \mathbf{y})$ of extreme points of $\mathcal{Q}_T^n$ of the minimum number of edges of $\mathcal{Q}_T^n$ that must be traversed to reach $\mathbf{x}$ from $\mathbf{y}$. For *asymmetric* traveling salesman polytopes we know that the diameter equals two for all $n \geq 6$, i.e., it is in particular independent upon the number n of nodes of the underlying directed graph; see [42]. In [20] we conjectured that diam $\mathcal{Q}_T^n = 2$ for $n \geq 5$. For $5 \leq n \leq 12$ this conjecture has been verified by way of a computer program; see [49]. Rispoli and Cosares [47] have shown that diam $\mathcal{Q}_T^n \leq 4$ for all $n \geq 5$, thus establishing a small upper bound on the diameter of this family of polytopes that does not depend on the number n of nodes of the graph.

Facet defining inequalities other than the SECs are known for the STS polytope $\mathcal{Q}_T^n$. See [24] for an excellent survey. From a computational point of view the following *r-comb inequalities* are the ones that are most frequently used. Let $H \subseteq V$ and $T_i \subseteq V$ for $1 \leq i \leq k$ be any subsets of nodes satisfying

$$\begin{aligned}
&|H \cap T_i| \geq 1, \qquad |T_i - H| \geq 1 \quad \text{for } 1 \leq i \leq k,\\
&T_i \cap T_j = \emptyset \quad \text{for } 1 \leq i < j \leq k,
\end{aligned}$$

where $k \geq 3$ is an odd integer. The configuration $C = (H, T_1, \ldots, T_k)$ in the graph G is called a *comb* in G with H being the "handle" and $T_1, \ldots, T_k$ the "teeth" of the comb. We partition each T_i into $r_i \geq 1$ nonempty sets T_i^j such that

$$\begin{aligned} &\left|H \cap T_i^j\right| \geq 1, \qquad \left|T_i^j - H\right| \geq 1 \quad \text{for } 1 \leq j \leq r_i, \\ &T_i^j \cap T_i^\ell = \emptyset \quad \text{for } 1 \leq j < \ell \leq r_i, \end{aligned}$$

for all $1 \leq i \leq k$. Then the r-comb inequality, see [38],

$$\begin{aligned} &\mathbf{x}(H) + \sum_{i=1}^{k}\left(\sum_{j=1}^{r_i}\mathbf{x}\big(T_i^j\big) + \sum_{1\leq \ell < j \leq r_i}\big(\mathbf{x}\big(T_i^\ell \cap H : T_i^j - H\big) + \mathbf{x}\big(T_i^\ell - H : T_i^j \cap H\big)\big)\right) \\ &\quad \leq |H| + \sum_{i=1}^{k}\big(|T_i| - r_i - 1\big) + \left\lfloor \frac{k}{2} \right\rfloor \end{aligned}$$

is a facet defining inequality for $\mathcal{Q}_T^n$. If $r_i = 1$ for $1 \leq i \leq k$, then we have the *comb inequalities* that were shown to define facets of $\mathcal{Q}_T^n$ by Grötschel and Padberg [19]. If $r_1 = 2$, $r_i = 1$ otherwise then we have the *chain inequalities* of Padberg and Hong [41]. The assertion that r-comb inequalities are facets defining for $\mathcal{Q}_T^n$ follows, e.g., from the work of Naddef and Rinaldi [29]. The prototype inequalities for r-combs can be seen in Fig. 5 and Fig. 6. They were known to us in early 1975 including the only chain inequality that exists for $\mathcal{Q}_T^8$. They were found by us empirically by trying to cut off "fractional" extreme points of LP relaxations that we encountered in small-scale numerical experimentation. When the comb and chain inequalities were tested by Martin Grötschel (September 1975) for their facet defining properties by way of a computer program at the University of Bonn, it turned out that the dimension of the face of $\mathcal{Q}_T^8$ defined by the chain inequality (facet type 7 in Fig. 5) was smaller than required, whereas the comb inequality (in particular the right-most one in Fig. 6) turned out to be facet defining for $\mathcal{Q}_T^8$. As a result we concentrated our effort on proving comb inequalities to be facet defining for $\mathcal{Q}_T^n$, but ignored the chain constraints. By the time it was found that the computer program used by Grötschel in the rank calculation had a "bug"—to fix the bug was one of Michael Jünger's and Gerd Reinelt's first tasks as research assistants at Bonn University—we had indeed more or less "forgotten" the chain constraints. They were shown to be facet defining by Sylvia Boyd and Mark Hartmann around 1990; see [3].

In Table 2 we summarize the knowledge about the number of facet defining inequalities for $\mathcal{Q}_T^n$ where $6 \leq n \leq 10$ up to about 1980. The exact number $\nu^C(n)$ of comb facets of $\mathcal{Q}_T^n$, see [19], is given by

$$\begin{aligned} \nu^C(n) = &\sum_{q=3}^{n-3}\sum_{j=3}^{n-q}\sum_{\substack{k=3\\k\text{ odd}}}^{\min\{j,q\}}\sum_{p=k}^{q}\sum_{\ell=0}^{k}\sum_{h=0}^{k}\frac{(-1)^{\ell+h}}{2k!} \\ &\times \binom{n}{q}\binom{n-q}{j}\binom{q}{p}\binom{k}{\ell}\binom{k}{h}(k-\ell)^j(k-h)^p. \end{aligned}$$

Table 2 Known and unknown facets of Q_T^n for $6 \le n \le 10$ as of 1980

Nodes	6	7	8	9	10	n
Vars	15	21	28	36	45	$\frac{1}{2}n(n-1)$
SECs	25	56	119	246	501	$2^{n-1}-n-1$
Combs	60	2,100	42,840	667,800	8,843,940	$\nu^C(n)$
$\phi(n)$	100	3,437	194,187	42,104,442	$\ge$51,043,900,866	?
	100 %	63 %	22 %	1.6 %	$\le$0.017 %	?

The formula for $\nu^C(n)$ was computed—with the help of Rabe von Randow of Bonn University—around 1976. Given the enormous number of facet defining inequalities that were known to us we began to optimize larger scale instances of symmetric traveling salesman problems around that time. With hindsight, it is fair to say that our ignorance of the true numbers $\phi(n)$ for Q_T^n was beneficial in this endeavor because, otherwise, we might have been discouraged to try out numerical experimentation. On the other hand, today instances with 10,000 cities or more have been optimized using essentially only comb inequalities. It is therefore fair to conjecture that some facets of Q_T^n, such as, e.g., those defined by the r-comb inequalities, are relatively more important than other ones. Whence our desire to bring some order into the facial structure of Q_T^n by "ranking" its facets.

To arrive at the numbers displayed in all tables (except $\dim Q_T^n$, $\pi(n)$ and $\operatorname{diam} Q_T^n$), we have written several computer programs. To find the number $\pi_0(n)$ of adjacent extreme points to a given one, e.g., the one corresponding to the tour $1, \ldots, n$, we use the CPLEX subroutines in a straightforward enumerative way. Then each edge of Q_T^n is subjected to the symmetry tests of Sect. 4 and Sect. 5 to reduce the number of edge figures to be analyzed. The output of this part of the overall program is a "problem file" of edge figures to be analyzed by the double description algorithm DDA. The facets for each edge figure from the problem file are calculated in a second program and classified like discussed in Sect. 2. This gives the entries for $\phi(n)$, $\phi_0(n)$, and $\kappa(Q_T^n)$ of Table 1. Only one representative for each facet type is generated and stored, the counting of the respective totals $\phi(n)$ and $\phi_0(n)$ is done in a separate subroutine. This program also calculates the normal form representation and the minimum positive support representations of each facet type. The rank analysis is done in a third program that implements the mathematics of Sect. 6 in an enumerative way.

As always in computer-based calculation we must leave a margin for error due to possibly remaining "bugs" in the programs. The numbers for Q_T^n—except for the rank $\rho(Q_T^n)$ which is a new concept—agree, however, exactly with the previously published results for $3 \le n \le 8$ and the same programs were used unchanged for $7 \le n \le 9$ to produce all numbers of this paper. For $n \le 6$ all edges of Q_T^n (after the corresponding reductions) were used, while for $7 \le n \le 9$ all edges of Q_T^n that are edges of Q_A^n were also excluded from consideration, which has a mathematical justification.

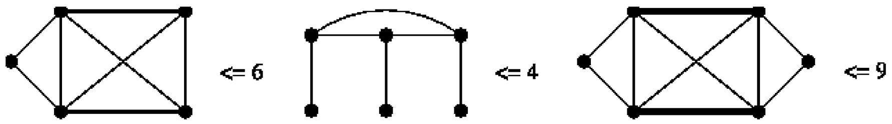


Fig. 4 Irreducible representations of facet types 4, 5, 6 of Q_T^7

For $n = 10$ we computed an initial set of facet types by this procedure as well. The corresponding edge figures, however, turned out to have more facets than we could calculate with our computing equipment at the time and so we changed the procedure as follows. For the initial set of about 2,000 facet types we compute all facets of each facet and classify them as known and unknown ones. For "unknown" facets we store a representative which is added to the list of facet types and later subjected to the same procedure as before. This procedure is then iterated until the program finds no new unknown facets. Clearly, this procedure need not find *all* facets of Q_T^{10} and we believe that Q_T^{10} has indeed more facets than we have found.

In Fig. 4 and Fig. 5 we depict the graphs of the minimal positive support representations of the facets of Q_T^n having rank 1 or higher for $n = 7$ and $n = 8$. The thickness of each arc corresponds to the numerical value of the respective inequality in less-than-or-equal-to form and the nonzero coefficients range from 1 to 4. Coefficients of 4 are drawn as heavy dotted lines. Thus Fig. 4 depicts 3,360 and Fig. 5 194,040 distinct facets of Q_T^n for $n = 7$ and $n = 8$. Figure 6 shows two "comb" forms of the respective facet types of Q_T^8. Note that the facet types 4, 5 and 6 of Q_T^7 are "inherited" by Q_T^8 (types 5, 6 and 12) and that, e.g., the facet type 15 of Q_T^8 is obtained by "lifting" facet type 5 of Q_T^7.

A more detailed breakdown of the facets of Q_T^n for $6 \le n \le 10$ is described below and the numerical experiments (see Table 1 and Table 8) suggest the following conjecture.

Conjecture 1 $\rho(Q_T^n) = n - 5$ *for all* $n \ge 5$.

Let $\mathbf{x}^0$ be an extreme point of Q_T^n and $\mathbf{x}^1 \in Q_T^n$ be an extreme point of Q_T^n that is adjacent to $\mathbf{x}^0$ on the polytope Q_T^n, i.e., $\mathbf{x}^1$ is a *neighbor* for $\mathbf{x}^0$. The face of minimal dimension of Q_A^n, of Q_S^n containing both $\mathbf{x}^0$ and $\mathbf{x}^1$ is denoted by $F_A(\mathbf{x}^0, \mathbf{x}^1)$ and $F_S(\mathbf{x}^0, \mathbf{x}^1)$, respectively. Table 3 and Table 4 show the breakdown of the neighbors of any extreme point of Q_T^n according to the dimensions $1, 2, 3, \ldots$ of $F_A(\mathbf{x}^0, \mathbf{x}^1)$ and $F_S(\mathbf{x}^0, \mathbf{x}^1)$ for $5 \le n \le 10$.

Common to the numbers of the two tables is a "linear" growth of the maximum dimension of $F_A(\mathbf{x}^0, \mathbf{x}^1)$ and $F_S(\mathbf{x}^0, \mathbf{x}^1)$, respectively.

Conjecture 2 *Let* $\mathbf{x}^0 \neq \mathbf{x}^1 \in Q_T^n$ *be any two extreme points and* $F_A(\mathbf{x}^0, \mathbf{x}^1)$, $F_S(\mathbf{x}^0, \mathbf{x}^1)$ *be the face of smallest dimension of* Q_A^n, *of* Q_S^n, *respectively, that contains both* $\mathbf{x}^0$ *and* $\mathbf{x}^1$. *If* $n \ge 5$ *and* $\dim F_A(\mathbf{x}^0, \mathbf{x}^1) \ge n - 3$ *or* $n \ge 7$ *and* $\dim F_S(\mathbf{x}^0, \mathbf{x}^1) \ge n - 5$, *then* $\mathbf{x}^0$ *and* $\mathbf{x}^1$ *are not adjacent on* Q_T^n.

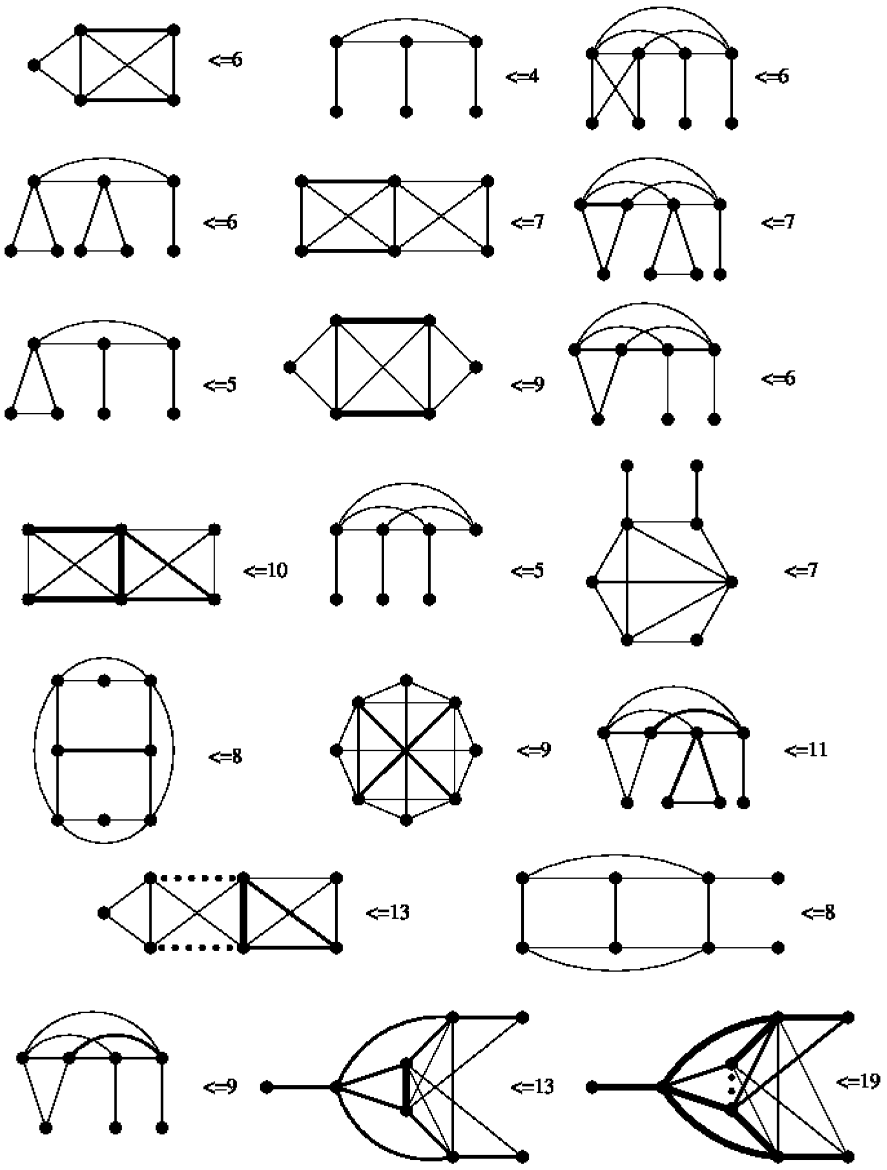


Fig. 5 Irreducible representations of facet types 5, . . . , 24 of Q_T^8

Note 2 I have included Conjecture 2 for the simple reason that it is suggested by my calculations (see Table 3 and Table 4) and that Papadimitriou's 1978 result [45] on the NP-completeness of the problem of checking the adjacency of tours on Q_T^n just goes against my intuition. Ting-Yi Sung and I [44] showed a related negative statement about the non-polytime solvability of certain TSPs called "traps" to be

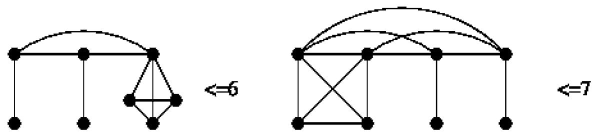


Fig. 6 Comb forms of facet types 5 and 9 of Q_T^8

Table 3 Adjacency on Q_T^n and Q_A^n

n	5	6	7	8	9	10
1	10	29	91	252	894	2,851
2		12	49	318	1,125	6,755
3			28	96	1,224	4,285
4				64	168	4,860
5					144	320
6						320
$\pi_0(n)$	10	41	168	730	3,555	19,391

Table 4 Adjacency on Q_T^n and Q_S^n

n	5	6	7	8	9	10
1	10	41	168	714	3,213	15,531
2				16	234	2,300
3					108	1,340
4						220
$\pi_0(n)$	10	41	168	730	3,555	19,391

Table 5 Reduction of the number of edge figures of Q_T^n using Claim 6

n	5	6	7	8	9	10
$\pi_0(n)$	10	41	168	730	3,555	19,391
$\sigma_0(n)$	2	7	16	59	216	1,032
$\iota_0(n)$	2	4	8	20	42	123

wrong. By refining the work on Conjecture 2 you may be able to prove $P = NP$ or to invalidate Papadimitriou's claim. [The referee pointed out that adjacency on Q_T^n is coNP-complete rather than NP-complete.]

Table 5 summarizes the effect of applying Claim 6 of Sect. 5 to reduce the number of edge figures of Q_T^n that have to be analyzed to determine the class number $\kappa(Q_T^n)$ of distinct facet types of Q_T^n. Like above $\pi_0(n)$ is the number of neighbors of any extreme point of Q_T^n and thus the number of edge figures to be analyzed a priori, $\sigma_0(n)$ is the number of edge figures that results when Claim 6 is applied for all permissible index permutations that leave $\mathbf{x}^0$ invariant, see (23), and $\iota_0(n)$ the number of "nonisomorphic" graphs that result from a full application of Claim 6.

Table 6 Classification of the facets of $\mathcal{Q}_T^n$ for $5 \le n \le 8$

n	κ	$\nu_\kappa(n)$	$\nu_0^\kappa(n)$	$d_\kappa^2(n)$	$d_\kappa(n)$	$n_\kappa^\pi(n)$	$\rho_\kappa(n)$
5	1	10	5	$\frac{1}{2}$	0.707	6	0
	2	10	5	$\frac{1}{2}$	0.707	6	0
6	1	15	9	$\frac{4}{15}$	0.516	36	0
	2	15	6	$\frac{3}{5}$	0.775	24	0
	3	10	3	$\frac{32}{45}$	0.843	18	0
	4	60	9	$\frac{128}{105}$	1.104	9	1
7	1	21	14	$\frac{1}{6}$	0.408	240	0
	2	21	7	$\frac{2}{3}$	0.816	120	0
	3	35	7	$\frac{5}{6}$	0.913	72	0
	4	1,260	70	$\frac{245}{156}$	1.253	20	1
	5	840	42	$\frac{5}{3}$	1.291	18	1
	6	1,260	56	$\frac{605}{372}$	1.275	16	2
8	1	28	20	$\frac{4}{35}$	0.338	1,800	0
	2	28	8	$\frac{5}{7}$	0.845	720	0
	3	56	8	$\frac{32}{35}$	0.956	360	0
	4	35	4	$\frac{27}{28}$	0.982	288	0
	5	3,360	88	$\frac{12}{7}$	1.309	66	1
	6	3,360	72	$\frac{256}{133}$	1.387	54	1
	7	2,520	52	$\frac{486}{259}$	1.370	52	1
	8	5,040	88	$\frac{1452}{749}$	1.392	44	1
	9	2,520	44	$\frac{27}{14}$	1.389	44	1
	10	5,040	88	$\frac{529}{280}$	1.375	44	1
	11	10,080	160	$\frac{361}{182}$	1.408	40	1
	12	5,040	80	$\frac{2883}{1498}$	1.387	40	1
	13	10,080	152	$\frac{600}{301}$	1.412	38	1
	14	10,080	152	$\frac{256}{133}$	1.387	38	1
	15	3,360	48	$\frac{289}{140}$	1.437	36	1
	16	20,160	264	$\frac{1587}{742}$	1.462	33	1
	17	10,080	124	$\frac{2}{1}$	1.414	31	1
	18	2,520	36	$\frac{729}{364}$	1.415	36	2
	19	20,160	272	$\frac{4107}{2114}$	1.394	34	2
	20	10,080	128	$\frac{1849}{952}$	1.394	32	2
	21	10,080	120	$\frac{900}{427}$	1.452	30	2
	22	20,160	240	$\frac{2883}{1400}$	1.435	30	2
	23	20,160	176	$\frac{4563}{1988}$	1.515	22	3
	24	20,160	176	$\frac{3025}{1344}$	1.500	22	3

Table 7 Facets-of-facets analysis of $\mathcal{Q}_T^n$ for $6 \leq n \leq 8$

κ	1	2	3	4	TT
1	14	14	6	24	58
2	14	6	4	12	36
3	9	6			15
4	6	3			9
$\rho_\kappa(6)$	0	0	0	1	

κ	1	2	3	4	5	6	TT
1	20	20	30	720	480	600	1,870
2	20	10	15	180	120	120	465
3	18	9	4	36			67
4	12	3	1			2	18
5	13	3				3	18
6	10	2		2	2		16
$\rho_\kappa(7)$	0	0	0	1	1	2	

κ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	TT
5	19	5	1	1			3					3		3		12									47
6	21	4												3		12	6			6	6	6	6		70
7	20	6		1	4				4			4				16		4			4			8	71
8	18	3	2														4		4						31
9	22	4		1			4		4					4		16					4				59
10	19	3	2																4						28
11	18	3	1									1					4		2			2			31
12	19	3			2		2				2									2					30
13	19	3	1											1					2			2			28
14	18	3			1	1			1				1			4				2	2				33
15	19	3																		3		6			31
16	15	4		1	2	2	2		2					2		4		1			3		2	2	42
17	17	5	2			2		2			4								4			4			40
18	16	4					4									8							8	8	48
19	17	2	1					1		1	1		1				2								26
20	17	2				2						1		2	1						1	2			28
21	16	4				2	1		1					2		6				1			4	2	39
22	17	2				1					1		1		1		2			1					26
23	13	4				1										2		1			2			2	25
24	11	4					1									2		1			1		2		22
$\rho_\kappa(8)$	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	3	3	

Table 8 Breakdown of the facet types and the facets of $\mathcal{Q}_T^n$ by rank for $5 \le n \le 10$

ρ/n	5	6	7	8	9	10
0	2	3	3	4	4	5
1		1	2	13	64	604
2			1	5	83	4,506
3				2	39	6,176
4					2	?
5						?
$\kappa(\mathcal{Q}_T^n)$	2	4	6	24	192	≥15,379

ρ/n	5	6	7	8	9	10
0	20	40	77	147	282	546
1		60	2,100	90,720	6,163,920	?
2			1,260	63,000	22,150,800	?
3				40,320	13,063,680	?
4					725,760	?
5						?
$\phi(n)$	20	100	3,437	194,187	42,104,442	≥51,043,900,866

Table 6 provides more detail on the classification of the facets of $\mathcal{Q}_T^n$ into equivalence classes for $5 \le n \le 8$. The additional notation used is as follows:

$$\nu_\kappa(n) = \text{the number of facets of } \mathcal{Q}_T^n \text{ of type } \kappa,$$

$$\nu_0^\kappa(n) = \text{the number of facets of type } \kappa \text{ that are tight at any given tour},$$

$$\rho_\kappa(n) = \text{the rank of facet of type } \kappa,$$

$$d_\kappa(n) = \text{the Euclidean distance from the center of facet of type } \kappa,$$

$$n_\kappa^\pi(n) = \text{the number of tours on facet of type } \kappa,$$

where $1 \le \kappa \le \kappa(\mathcal{Q}_T^n)$. The $\lfloor n/2 \rfloor$ first types of the facets in each table correspond to the nonnegativity constraints (type 1) and the upper bounds (type 2) followed by the SECs with increasing $|S| \ge 3$.

Table 7 and Table 8 give the results of our rank analysis of the facets of $\mathcal{Q}_T^n$ for $5 \le n \le 9$. The top and middle parts of Table 7 give a complete picture for $6 \le n \le 7$, while for $n = 8$ we left out the facets-of-facets analysis for the rank zero facets of $\mathcal{Q}_T^8$. Thus the facets defined by the nonnegativity constraints $x_e \ge 0$ have a total of 58 facets for $n = 6$ (1,870 facets for $n = 7$). For $n = 6$ there are 14 facets of the same type, 14 are upper bounds, 6 are SECs with $|S| = 3$ and 24 are "matching constraints" (see the picture in the middle of Fig. 4). The remaining entries in these tables are interpreted accordingly. TT are the respective totals, i.e., they are equal to the sum of the entries in each row. In Table 8 the rows correspond to rank $0, \dots, 5$ and the columns to the number of nodes $5, \dots, 10$ of the graph.

Work on ideal linear descriptions of "small" problems related to combinatorial optimization problems has become much easier with the arrival of software for the double description algorithm or the Fourier-Motzkin elimination algorithm. In the early 1970s we had to "guess" some linear inequality by trial and error and then prove it to be facet-defining (if it was). Now we can use the computer to experiment with educated guesses from the linear description of small instances. As in the work of Christof and Reinelt [7] we can use the descriptions of small instances for the optimization of larger instances. We can form conjectures about the problem in question—like the conjectures above. The observed linearity of the rank $\rho(\mathcal{Q}_T^n)$ for $n = 5, \dots, 9$ suggests to study the facial structure of $\mathcal{Q}_T^n$ *inductively*, i.e., to try to determine an ideal linear description of $\mathcal{Q}_T^{n+1}$ from the one of $\mathcal{Q}_T^n$, to prove or disprove Conjecture 1, to find an expression, e.g., for $\phi(n)$ of the form $\phi(n+1) = g(\phi(n))$ for $n \geq 5$ and some suitable function $g()$ and to continue the work that Martin Grötschel and I began so many years ago.

Acknowledgements Supported in part by ONR grant N00014-96-0327 and by visits to Cologne U (1996) and IASI-CNR Rome (1997). This paper was presented in preliminary form in a plenary session at the XVIth ISMP in August 1997 at the EPFL in Lausanne, Switzerland, under the title "Facets, Rank of Integer Polyhedra and Other Topics".

I wish to thank the referee for his constructive criticism.

References

1. Alevras, D.: Small min-cut polyhedra. Math. Oper. Res. **24**, 35–49 (1999)
2. Bartels, H.G., Bartels, S.G.: The facets of the asymmetric 5-city traveling salesman problem. ZOR, Z. Oper.-Res. **33**, 193–197 (1989)
3. Boyd, S.C., Cunningham, W.H.: Small traveling salesman polytopes. Math. Oper. Res. **16**, 259–271 (1991)
4. Burger, E.: Über homogene lineare Ungleichungssysteme. Z. Angew. Math. Mech. **36**, 135–139 (1956)
5. Christof, T.: Low-dimensional 0/1 polytopes and branch-and-cut in combinatorial optimization. Ph.D. thesis, University of Heidelberg, Heidelberg, Germany (1997)
6. Christof, T., Reinelt, G.: Combinatorial optimization and small polytopes. Top **4**, 1–64 (1996)
7. Christof, T., Reinelt, G.: Algorithmic aspects of using small instance relaxations in parallel branch-and-cut. Algorithmica **30**, 597–629 (2001)
8. Christof, T., Reinelt, G.: Decomposition and parallelization techniques for enumerating the facets of 0/1 polytopes. Int. J. Comput. Geom. Appl. **11**, 423–437 (2001)
9. Christof, T., Jünger, M., Reinelt, G.: A complete description of the traveling salesman polytope on 8 nodes. Oper. Res. Lett. **10**, 497–500 (2001)
10. Chvátal, V.: Edmonds polytopes and a hierarchy of combinatorial problems. Discrete Math. **4**, 305–337 (1973)
11. Dantzig, G.B., Fulkerson, D.R., Johnson, S.M.: Solution of a large-scale travelling salesman problem. Oper. Res. **2**, 393–410 (1954)
12. Deza, M., Grötschel, M., Laurent, M.: Complete descriptions of small multicut polytopes. In: Applied Geometry and Discrete Mathematics, pp. 221–252. Am. Math. Soc., Providence (1991)

13. Edmonds, J.: Maximum matching and a polyhedron with 0–1 vertices. J. Res. Natl. Bur. Stand. B, Math. Math. Phys. **69**, 67–92 (1965)
14. Euler, R., Verge, H.L.: A complete and irredundant linear description of the asymmetric traveling salesman polytope on 6 nodes. Discrete Appl. Math. **62**, 193–208 (1995)
15. Fleischmann, B.: Duale und primale Schnitthyperebenenverfahren in der ganzzahligen linearen Optimierung. Ph.D. thesis, University of Hamburg, Hamburg, Germany (1970)
16. Gomory, R.: An algorithm for integer solutions to linear programs. In: Graves, R.L., Wolfe, P. (eds.) Recent Advances in Mathematical Programming, pp. 269–302. McGraw-Hill, New York (1963)
17. Grötschel, M., Padberg, M.W.: Zur Oberflächenstruktur des Travelling Salesman Polytopen. In: Zimmermann, H.J., Schub, A., Späth, H., Stoer, J. (eds.) Proc. in Operations Research, vol. 4, pp. 207–211. Physica-Verlag, Würzburg (1974)
18. Grötschel, M., Padberg, M.W.: On the symmetric traveling salesman problem. Technical report 7536, OR—Inst. f. Ökonometrie und Operations Research, Bonn, Germany (1975)
19. Grötschel, M., Padberg, M.W.: On the symmetric traveling salesman problem: parts I and II. Math. Program. **16**, 265–302 (1979)
20. Grötschel, M., Padberg, M.W.: Polyhedral theory. In: Lawler, E.L., Lenstra, J.K., Rinnoy Kan, A.H.G., Shmoys, D.B. (eds.) The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, Chichester (1985)
21. Grötschel, M., Jünger, M., Reinelt, G.: Facets of the linear ordering polytope. Math. Program. **33**, 43–60 (1985)
22. Grünbaum, B.: Convex Polytopes. Wiley, New York (1967)
23. Heller, I.: On the problem of shortest paths between points. Bull. Am. Math. Soc. **59**, 551–552 (1953)
24. Jünger, M., Reinelt, G., Rinaldi, G.: The traveling salesman problem. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (eds.) Network Models. Handbooks in Operations Research and Management Science, vol. 7, pp. 225–330. North-Holland, Amsterdam (1995)
25. Kaibel, V.: Polyhedral combinatorics of the quadratic assignment problem. Ph.D. thesis, University of Cologne, Cologne, Germany (1997)
26. Kuhn, H.: On certain convex polyhedra. Bull. Am. Math. Soc. **61**, 557–558 (1955)
27. Kuhn, H.: On asymmetric traveling salesman polytopes. Presented at the 14th Intern. Symp. on Mathematical Programming (1991)
28. Naddef, D., Rinaldi, G.: The crown inequalities for the symmetric traveling salesman polytope. Math. Oper. Res. **17**, 308–326 (1992)
29. Naddef, D., Rinaldi, G.: The graphical relaxation: a new framework for the symmetric traveling salesman polytope. Math. Program. **58**, 53–88 (1993)
30. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, New York (1988)
31. Norman, R.: On the convex polyhedra of the symmetric traveling salesman problem. Bull. Am. Math. Soc. **61**, 559 (1955)
32. Padberg, M.W.: Essays in integer programming. Ph.D. thesis, Carnegie-Mellon University, Pittsburgh, PA (1971)
33. Padberg, M.W.: On the facial structure of set packing polyhedra. Math. Program. **5**, 199–215 (1973)
34. Padberg, M.W.: Perfect zero-one matrices. Math. Program. **6**, 180–196 (1973)
35. Padberg, M.W.: On the complexity of set packing polyhedra. Ann. Discrete Math. **1**, 421–434 (1977)
36. Padberg, M.W.: Covering, packing and knapsack problems. Ann. Discrete Math. **4**, 265–287 (1979)
37. Padberg, M.W.: The boolean quadric polytope: some characteristics, facets, and relatives. Math. Program. **45**, 139–172 (1989)
38. Padberg, M.W.: Linear Optimization and Extensions. Springer, Berlin (1999)
39. Padberg, M.W.: Classical cuts for mixed-integer programming and branch-and-cut. Math. Methods Oper. Res. **53**, 173–203 (2001). Reprinted in Ann. Oper. Res. **139**, 321–352 (2005)

40. Padberg, M.W.: The rank of (mixed-) integer polyhedra. Math. Program., Ser. A **137**, 593–599 (2011)
41. Padberg, M.W., Hong, S.: On the symmetric traveling salesman problem: a computational study. Math. Program. **12**, 78–107 (1980)
42. Padberg, M.W., Rao, M.: The traveling salesman problem and a class of polyhedra of diameter two. Math. Program. **7**, 32–45 (1980)
43. Padberg, M.W., Rijal, M.: Location, Scheduling, Design and Integer Programming. Kluwer Academic, Boston (1996)
44. Padberg, M.W., Sung, T.: A polynomial-time solution to Papadimitriou and Steiglitz's "traps". Oper. Res. Lett. **7**, 117–125 (1988)
45. Papadimitriou, C.H.: The adjacency relation on the traveling salesman polytope is NP-complete. Math. Program. **14**, 312–324 (1978)
46. Reinelt, G.: Personal communication (2011)
47. Rispoli, F., Cosares, S.: A bound of 4 for the diameter of the symmetric traveling salesman polytope. SIAM J. Appl. Math. **11**, 373–380 (1998)
48. Schrijver, A.: Theory of Linear and Integer Programming. Wiley, Chichester (1986)
49. Sierksma, G., Tijssen, G.: Faces with large diameter on the symmetric traveling salesman polytope. Oper. Res. Lett. **12**, 73–77 (1992)
50. Ziegler, G.: Lectures on Polytopes. Springer, Berlin (1995)

Part III
Martin Grötschel's Doctoral Descendants

It is a cherished academic tradition to honor one's ancestors as follows: If C is a doctoral student of P, then P is called the (doctoral) parent and C is called the (doctoral) child. In this sense, the editors are Martin Grötschel's oldest children.

We have compiled the tree of Martin Grötschel's doctoral descendants from 1983 to 2012, i.e., the first 30 years.

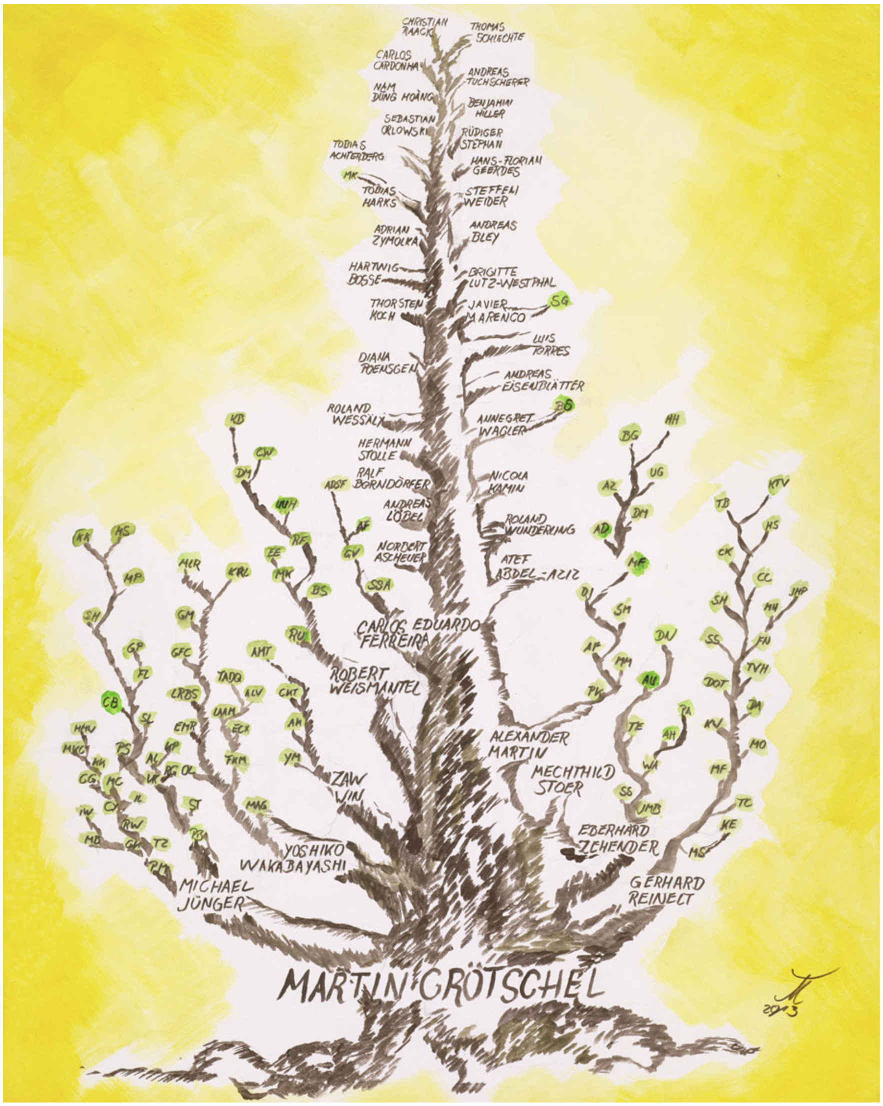
It is a special pleasure to start this part of the book with Thomas Möllmann's artistic interpretation of this tree. The original painting will be a gift for Martin Grötschel from his doctoral descendants, and will be presented to him during the Festkolloquium on September 13, 2013. It shows Martin Grötschel's

39 children,
74 grandchildren,
24 great-grandchildren, and
2 great-great-grandchildren,

a total of 139 doctoral descendants.

We then break down the tree precisely, just like a computer scientist would.

Martin Grötschel's Doctoral Descendant Tree 1983–2012



Painting by Thomas Möllmann

Martin Grötschel's Descendants and Their Doctoral Theses 1983–2012

Michael Jünger and Gerhard Reinelt

Abstract We present the tree of Martin Grötschel's doctoral descendants along with their doctoral theses from 1983 to 2012, i.e., the first 30 years. Our format should be self-explanatory: Depth in the hierarchy is indicated by indentation and color: blue for the 39 children, red for the 74 grandchildren, cyan for the 24 great-grandchildren, and magenta for the 2 great-great-grandchildren. The doctoral thesis titles of these 139 descendants show the many fruits of Martin Grötschel's original impetus.

Martin Grötschel (1977)
Polyedrische Charakterisierungen kombinatorischer Optimierungsprobleme

- Michael Jünger (1983)
  Polyhedral Combinatorics and the Acyclic Subdigraph Problem
  - Petra Mutzel (1994)
    The Maximum Planar Subgraph Problem
    - Thomas Ziegler (2001)
      The Constrained Crossing Minimization Problem
    - Gunnar Klau (2001)
      Optimal Compaction and Labelling
      - Markus Bauer (2008)
        A Combinatorial Approach to RNA Sequence-Structure Alignments

M. Jünger (✉)
Institut für Informatik, Universität zu Köln, Albertus-Magnus-Platz, 50923 Cologne, Germany
e-mail: mjuenger@informatik.uni-koeln.de

G. Reinelt
Institut für Informatik, Ruprecht-Karls-Universität Heidelberg, Im Neuenheimer Feld 368, 69120 Heidelberg, Germany
e-mail: gerhard.reinelt@informatik.uni-heidelberg.de

Inken Wohlers (2012)
Exact Algorithms for Pairwise Protein Structure Alignment

René Weiskircher (2002)
New Applications of SPQR-Trees in Graph Drawing

Ivana Ljubić (2004)
Exact and Memetic Algorithms for Two Network Design Problems

Canan Yildiz (2006)
Knickminimales Orthogonales Zeichnen planarer Graphen im Kandinsky Modell

Markus Chimani (2008)
Computing Crossing Numbers

Carsten Gutwenger (2010)
Application of SPQR-Trees in the Planarization Approach for Drawing Graphs

Karsten Klein (2011)
Interactive Graph Drawing with Constraints

Maria Kandyba-Chimani (2011)
Exact Algorithms for Network Design Problems Using Graph Orientations

Hoi-Ming Wong (2011)
Upward Planarization and Layout

Petra Bauer (1994)
A Polyhedral Approach to the Weighted Girth Problem

Stefan Thienel (1995)
ABACUS—A Branch-And-CUt System

Volker Kaibel (1997)
Polyhedral Combinatorics of the Quadratic Assignment Problem

Rafael Gillmann (2006)
Typical and Extremal 0/1-Polytopes

Andreas Loos (2011)
Describing Orbitopes by Linear Inequalities and Projection Based Tools

Kanstantsin Pashkovich (2012)
Extended Formulations for Combinatorial Polytopes

Peter Störmer (1998)
A Parallel Branch&Cut Algorithm for the Solution of Large-Scale Traveling Salesman Problems

Sebastian Leipert (1998)
Level Planarity Testing and Embedding in Linear Time

Christoph Buchheim (2003)
An Integer Programming Approach to Exact and Fuzzy Symmetry Detection

Frauke Liers (2004)
Contributions to Determining Exact Ground-States of Ising Spin-Glasses and to Their Physics

Gregor Pardella (2011)
Efficient Polynomial-Time Algorithms for Special Graph Partitioning Problems

Stefan Hachul (2005)
A Potential-Field-Based Multilevel Algorithm for Drawing Large Graphs

Merijam Percan, now Gotzes (2007)
Constrained Planarity and Augmentation Problems

Katerina Keunecke (2008)
Produktionsplanung in der Textil verarbeitenden Industrie

Michael Schulz, now Knüver (2008)
Simultaneous Graph Drawing

Gerhard Reinelt (1984)
The Linear Ordering Problem: Algorithms and Applications

Marc Steinbach (1995)
Fast Recursive SQP Methods for Large-Scale Optimal Control Problems

Klaas Eggert (2012)
Numerische Verfahren für nichtlineare gemischt-ganzzahlige Optimierungsprobleme im Wassermanagement

Thomas Christof (1997)
Low-Dimensional 0/1-Polytopes and Branch-and-Cut in Combinatorial Optimization

Meinrad Funke (1997)
Allgemeine Feedback Vertex Set Probleme

Marcus Oswald (2003)
Weighted Consecutive Ones Problems

Klaus Wenger (2003)
Generic Cut Generation Methods for Routing Problems

Dino Ahr (2004)
Contributions to Multiple Postmen Problems

Dirk Oliver Theis (2005)
Polyhedra and Algorithms for the General Routing Problem

Tran Van Hoai (2005)
Solving Large Scale Crew Pairing Problems

Sebastian Sager (2006)
Numerical Methods for Mixed-Integer Optimal Control Problems

Frank Noe (2006)
Transition Networks: Computational Methods for the Comprehensive Analysis of Complex Rearrangements in Proteins

Martin Held (2012)
Novel Concepts to Study Conformation and Association Dynamics of Biomolecules

Jan-Hendrik Prinz (2012)
Advanced Estimation Methods for Markov Models of Dynamical Systems

Silvia Harmsen (2006)
Algorithmic Computer Reconstructions of Stalactite Vaults—Muqarnas—in Islamic Architecture

Cara Cocking (2008)
Solutions to Facility Location-Network Design Problems

Christian Kirches (2010)
Fast Numerical Methods for Mixed-Integer Nonlinear Model Predictive Control

Hanna Seitz (2010)
Contributions to the Minimum Linear Arrangement Problem

Thorsten Bonato (2011)
Contraction-Based Separation and Lifting for Solving the Max-Cut Problem

Khoa Tan Vo (2011)
Exact and Heuristic Solutions to the Bandwidth Minimization Problem

Yoshiko Wakabayashi (1986)
Aggregation of Binary Relations: Algorithmic and Polyhedral Investigations

Maria Angela Gurgel (1992)
Poliedros de Grafos Transitivos

Flávio K. Miyazawa (1997)
Algoritmos de Aproximação para Problemas de Empacotamento

Eduardo Cândido Xavier (2006)
Algoritmos para Problemas de Empacotamento

Luís Augusto Angelotti Meira (2007)
Algoritmos para Problemas de Classificação e Particionamento em Grafos

André Luis Vignatti (2010)
Tempo de Convergência para o Equilíbrio de Nash nos Jogos de Empacotamento de Itens e Balanceamento

Thiago Alves de Queiroz (2010)
Algoritmos para Problemas de Corte e Empacotamento

Orlando Lee (1999)
Cobertura por Circuitos em Grafos Mistos

Estela Maris Rodrigues (2003)
Algoritmos para Construção de Árvores Filogenéticas e o Problema dos Pontos de Recombinação

Liliane R.B. Salgado (2004)
Algoritmos de Aproximação para Partições Conexas em Grafos

Glauber F. Cintra (2004)
Algoritmos para Problemas de Corte de Guilhotina Bidimensional

Gordana Manić (2006)
Empacotamento de Subgrafos em Grafos

Karla Roberta Lima (2011)
Recoloração Convexa de Caminhos

Mario Leston Rey (2012)
Um Arcabouço Generalizado para o Empacotamento de Ramificações e outras Estruturas Combinatórias

Eberhard Zehendner (1986)
Methoden der Polyedertheorie zur Herleitung von oberen Schranken für die Mächtigkeit von Blockcodes

Jens Martin Beck (1997)
Architekturabhängige Partitionierung von Datenflußgraphen

Sebastian Schmidt (1997)
Befehlsanordnung für superskalare Prozessoren

Wolfram Amme (1998)
Datenabhängigkeitsanalyse in Programmen mit Zeigern

Andreas Hartmann (2006)
Eine Methode zur effizienten und verifizierbaren Programmannotation für den Transport von Escape-Informationen

Philipp Adler (2012)
Spekulative Optimierung in interpretertativen virtuellen Umgebungen

Thomas Etzrodt (1998)
Automatische Speicherfreigabe in Mehrprozessorsystemen – Verfahren und Leistungsbewertung

Andreas Unger (2002)
Simultan spekulatives Scheduling

David Neuhäuser (2012)
Design and Evaluation of Computer Arithmetic Based on Carry-Save and Signed-Digit Redundant Number Representations

Zaw Win (1987)
Contributions to Routing Problems

Yi Myint (2000)
Investigation on Some Centrality Concepts in Graphs

Aung Kyaw (2004)
Investigations on Maximal Trees, with Prescribed Upper Bound for the Maximum Degree, in a Graph

Cho Kyi Than (2005)
Optimal Locations of Paths and Trees of a Given Size in a Given Tree Network

Aye Myat Thwe (2007)
Medians and Branch Weight Centroids in Graphs

Mechthild Stoer, now Opperud (1991)
Design of Survivable Networks

Robert Weismantel (1992)
Plazieren von Zellen: Theorie und Lösung eines quadratischen 0/1 Optimierungsproblems

Regina Urbaniak (1998)
Decomposition of Generating Sets and of Integer Programs

Bianca Spille (2001)
Primal Characterizations of Combinatorial Optimization Problems

Matthias Köppe (2002)
Exact Primal Algorithms for General Integer and Mixed-Integer Linear Programs

Elke Eisenschmidt (2009)
Integer Minkowski Programs and an Application in Network Design

Robert Firla (2002)
The Design of Exponential Neighborhoods: A Primal Approach to Integer Programming

Utz-Uwe Haus (2004)
An Augmentation Framework for Integer Programming

Dennis Michaels (2007)
Discrete Optimization Techniques for Nonlinear Mixed-Integer Optimization Problems Arising from Chemical Engineering

Christian Wagner (2011)
Maximal Lattice-Free Polyhedra in Mixed-Integer Cutting Plane Theory

Kathrin Ballerstein (2012)
Logical Interaction Networks in Biology: Theory and Application

Alexander Martin (1992)
Packen von Steinerbäumen: Polyedrische Studien und Anwendung

Petra Kopp (2004)
Mathematische Modellierung der Konsistenz und konsistenzerhaltender Erweiterungen von Vererbung in objektorientierten Sprachen

Markus Möller (2004)
Mixed Integer Models for the Optimisation of Gas Networks in the Stationary Case

Armin Fügenschuh (2005)
The Integrated Optimization of School Starting Times and Public Transport

Susanne Moritz (2006)
A Mixed Integer Approach for the Transient Case of Gas Network Optimization

Daniel Junglas (2006)
Optimised Grid-Partitioning for Block Structured Grids in Parallel Computing

Marzena Fügenschuh (2007)
Relaxations and Solutions for the Minimum Graph Bisection Problem

Agnes Dittel (2008)
Protein Folding and Self-Avoiding Walks—Polyhedral Studies and Solutions

Debora Mahlke (2010)
A Scenario Tree-Based Decomposition for Solving Multistage Stochastic Programs with Application in Energy Production

Andrea Zelmer (2010)
Designing Coupled Energy Carrier Networks by Mixed-Integer Programming Methods

Ute Günther (2010)
Integral Sheet Metal Design by Discrete Optimization

Björn Geißler (2011)
Towards Globally Optimal Solutions for MINLPs by Discretization Techniques with Applications in Gas Network Optimization

Henning Homfeld (2012)
Consolidating Car Routes in Rail Freight Service by Discrete Optimization

Carlos Eduardo Ferreira (1994)
On Combinatorial Optimization Problems Arising in Computer System Design

Said Sadique Adi (2005)
Identificação de genes por comparação de sequências

Gerardo Vianna (2007)
Técnicas para construção de árvores filogenéticas

André Fujita (2007)
Análise de dados de expressão gênica: normalização de microarrays e modelagem de redes regulatórias

Alexandre da Silva Freire (2012)
Empacotamento de bicliques em grafos bipartidos

Atef Abdel-Aziz (1994)
Combinatorial Optimization Problems Arising in the Design and Management of an Automatic Storage System

Norbert Ascheuer (1995)
Hamiltonian Path Problems in the On-Line Optimization of Flexible Manufacturing System

Roland Wunderling (1997)
Paralleler und objektorientierter Simplex-Algorithmus

Andreas Löbel (1997)
Optimal Vehicle Scheduling in Public Transit

Nicola Kamin (1998)
On-Line Optimization of Order Picking in an Automated Warehouse

Ralf Borndörfer (1998)
Aspects of Set Packing, Partitioning, and Covering

Hermann Stolle (2000)
Mathematische Modellierung und Lösung von Optimierungsproblemen bei der Planung von Telefonnetzen

Annegret Wagler (2000)
Critical Edges in Perfect Graphs

Balamurali Sreedhar (2010)
Preparative Chromatographic Separation of Ternary Mixtures—Analysis of Fractionation Times and Novel Concepts

Roland Wessäly (2000)
DImensioning Survivable Capacitated NETworks

Andreas Eisenblätter (2001)
Frequency Assignment in GSM Networks: Models, Heuristics, and Lower Bounds

Diana Poensgen (2003)
Facets of Online Optimization: Online Dial-a-Ride Problems and Dynamic Configuration of All-Optical Networks

Luis Torres (2003)
Online Vehicle Routing

Thorsten Koch (2004)
Rapid Mathematical Programming

Javier Marenco (2005)
Chromatic Scheduling Polytopes Coming from the Bandwidth Allocation Problem in Point-to-Multipoint Radio Access Systems

Sebastián Guala (2010)
Estudios del Juego de las Minoras

Hartwig Bosse (2005)
Representing Polyhedra by Few Polynomial Inequalities

Brigitte Lutz-Westphal (2006)
Kombinatorische Optimierung – Inhalte und Methoden für einen authentischen Mathematikunterricht

Adrian Zymolka (2006)
Design of Survivable Optical Networks by Mathematical Optimization

Andreas Bley (2007)
Routing and Capacity Optimization for IP Networks

Steffen Weider (2007)
Integration of Vehicle and Duty Scheduling in Public Transport

Tobias Harks (2007)
Multicommodity Routing Problems: Selfish Behavior and Online Aspects

Max Klimm (2012)
Competition for Resources—The Equilibrium Existence Problem in Congestion Games

Tobias Achterberg (2007)
Constraint Integer Programming

Hans-Florian Geerdes (2008)
UMTS Radio Network Planning: Mastering Cell Coupling for Capacity Optimization

Sebastian Orlowski (2009)
Optimal Design of Survivable Multi-layer Telecommunication Networks

Rüdiger Stephan (2009)
Polyhedral Aspects of Cardinality Constrained Combinatorial Optimization Problems

Benjamin Hiller (2009)
Online Optimization: Probabilistic Analysis and Algorithm Engineering

Nam Dũng Hoàng (2010)
Algorithmic Cost Allocation Games: Theory and Applications

Andreas Tuchscherer (2010)
Local Evaluation of Policies for Discounted Markov Decision Problems

Carlos Cardonha (2011)
Applied Methods for the Vehicle Positioning Problem

Thomas Schlechte (2011)
Railway Track Allocation: Models and Algorithms

Christian Raack (2012)
Capacitated Network Design: Multi-commodity Flow Formulations, Cutting Planes, and Demand Uncertainty

Part IV
Contributions by Martin Grötschel's Doctoral Descendants

This book is intended as a present for Martin Grötschel from his doctoral descendants, many of whom are still active researchers in academic and industry positions. When we put out a call for contributions for this book, we issued strict quality guidelines and tight deadlines.

The response has been delightful. The main and final part of this book contains these contributions. Every article is coauthored by at least one doctoral descendant. They could just as well be articles in optimization journals, except the authors were encouraged to relate their work to Martin, and they were free to acknowledge Martin's influence.

The sequence of the articles starts with contributions on the theory of Mathematical Optimization: Volker Kaibel and Kanstantsin Pashkovich present a framework for the construction of extended formulations via projections, with an emphasis on reflection relations. Using convex optimization techniques, Michel Baes, Timm Oertel, Christian Wagner, and Robert Weismantel reduce the hard core of mixed-integer convex optimization problems to a certain improvement oracle. Arnaud Pêcher and Annegret K. Wagler present superclasses of perfect graphs that still allow for the polynomial time computation of the clique number and the chromatic number, using the theta number. Zaw Win and Cho Kyi Than introduce the notions of the subtree-centroid and subtree-telecenter, and present an efficient algorithm for computing a subtree-telecenter of a tree. Carlos E. Ferreira and Alvaro J.P. Franco use a characterization of junctions in acyclic graphs to derive efficient algorithms for listing target pairs that have a given vertex as a junction. Rafael da Ponte Barbosa and Yoshiko Wakabayashi study non-preemptive and preemptive versions of the restricted strip cover problem and present both an improved polynomial time approximation algorithm for the former and an exact polynomial time algorithm for the latter.

The following articles combine new theoretical insights with algorithms and experiments: Ralf Borndörfer, Nam-Dũng Hoang, Marika Karbstein, Thorsten Koch, and Alexander Martin consider the Steiner connectivity problem and the Steiner tree packing problem and present new results concerning complexity, algorithms, and computational results. Also in the area of network design, Eduardo Álvarez-Miranda, Ivana Ljubić, and Petra Mutzel consider the maximum weight connected subgraph problem; they propose and analyze a new mixed-integer model and outperform previous computational experiments on benchmark sets. Frank Baumann, Sebastian Berckey and Christoph Buchheim present branch&bound algorithms for combinatorial optimization problems with submodular objective functions, alternatively using a linearization technique and Lagrangean decomposition, and put forward experimental evidence of superiority in wireless network design and mean-risk optimization. Martin Schmidt, Marc C. Steinbach, and Bernhard M. Willert address nonsmooth mixed-integer optimization problems and provide approximate smooth reformulations with complementarity constraints, and present numerical results for the validation of nominations in gas networks. Björn Geißler, Antonio Morsi, and Lars Schewe develop a new algorithm for mixed-integer nonlinear optimization based on the adaptive refinement of a new class of mixed-integer linear relaxations and demonstrate its potential for gas transport energy cost minimization.

We continue with computational studies: Miguel F. Anjos, Bissan Ghaddar, Lena Hupp, Frauke Liers, and Angelika Wiegele present a computational study of a semidefinite branch&cut approach for the max k-cut problem based on the bundle approach that outperforms previous approaches on certain instance classes. Michael N. Jung, Christian Kirches, and Sebastian Sager deal with mixed-integer nonlinear optimal control, survey various modeling approaches, and give computational results for a truck cruise control problem with logical implications due to gear constraints. Armin Fügenschuh, George Nemhauser, and Yulian Zeng present a mixed-integer linear optimization formulation of flow-over-flow models driven by the problem of scheduling and routing fly-in safari planes, along with a heuristic based on randomized local search, and present an extensive computational study.

The two closing articles are devoted to computational advances in general mixed-integer linear optimization, the first by scientists working in industry, the second by scientists working in academia: Tobias Achterberg and Roland Wunderling develop an unbiased way to analyze benchmark results and apply it to assess the contributions of the main components in CPLEX 12.5. Thorsten Koch, Alexander Martin, and Marc E. Pfetsch focus on the reproducibility of computational experiments, investigate the performance of competing solvers, and demonstrate the development of the academic solvers SIP and SCIP.

The contributions reflect the “scientific facets” of Martin Grötschel, who has set standards in theory, computation and applications.

Constructing Extended Formulations from Reflection Relations

Volker Kaibel and Kanstantsin Pashkovich

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 the 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 by iteratedly forming convex hulls of polytopes and (slightly modified) reflections of them 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 Goemans' extended formulation of the permutahedron of size $O(n \log n)$ and Ben-Tal and Nemirovski's extended formulation with $O(k)$ inequalities for the regular 2^k-gon) and for Huffman-polytopes (the convex hulls of the weight-vectors of Huffman codes). This work is an extension of an extended abstract presented at IPCO XV (2011).

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,

V. Kaibel (✉)
Fakultät für Mathematik, Otto-von-Guericke Universität Magdeburg, Universitätsplatz 2, 30106 Magdeburg, Germany
e-mail: kaibel@ovgu.de

K. Pashkovich
Dipartimento di Matematica, Universitá del Padova, Via Trieste 63, 35121 Padova, Italy
e-mail: kanstantsin.pashkovich@gmail.com

such extensions Q can have way less facets than the polyhedron P has. For a nice survey on extended formulations we refer to [5].

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 [11, 18]). 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.

There are several ways to build extended formulations of polytopes from linear descriptions or from extended formulations of other ones (see, e.g., [9, 14]). 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

$$\mathrm{P}^p_{\mathrm{ct}} = \mathrm{conv}\big(\big\{\mathrm{ct}(p, \sigma) : \sigma \in \mathfrak{S}(n)\big\}\big).$$

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 $\varrho : \mathbb{R}^{2n-2} \to \mathbb{R}^n$ defined via

$$\varrho(x) = \big(x' + p_n x'', \langle \tilde{p}, \mathbb{1} - x'' \rangle + p_n\big)$$

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 1980s [17]). This may appear surprisingly simple viewing the fact that $\mathrm{P}^p_{\mathrm{ct}}$ has exponentially many facets (see [16]). For the special case of the *permutahedron*

$$\mathrm{P}^n_{\mathrm{perm}} = \mathrm{P}^{\mathbb{1}_n}_{\mathrm{ct}} = \mathrm{conv}\big\{\big(\gamma(1), \ldots, \gamma(n)\big) \in \mathbb{R}^n : \gamma \in \mathfrak{S}(n)\big\},$$

Goemans [7] 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

$$R = \big\{(x, y) \in \mathbb{R}^{2n-2} \times \mathbb{R}^n : y = \varrho(x)\big\} \tag{1}$$

the extension derived in such a step reads

$$\mathrm{P}^p_{\mathrm{ct}} = \big\{y \in \mathbb{R}^n : (x, y) \in R \text{ for some } x \in P\big\}. \tag{2}$$

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 Sect. 2, and we are going to elaborate on one particular type of those, called *reflection relations*, in Sect. 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 (Sect. 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 the dihedral group $I_2(m)$ 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 [7]. In fact, the starting point of our research was to give an alternative proof for the correctness of Goemans' extended formulation that we would be able to generalize to other constructions. Additionally, we provide an extended formulation of the cardinality indicating polytope with $\mathrm{O}(n \log n)$ inequalities, where the cardinality indicating polytope has as its vertices the $2n+1$-dimensional vectors whose first n coordinates are the characteristic vectors of all sets $S \subseteq [n]$ and the last $n+1$ coordinates form the standard unit vector $\mathbb{e}_{|S|+1}$ (a polytope that has been investigated in [12]).

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 (Sect. 4.2). This *Huffman-polytope* $\mathrm{P}^{n}_{\mathrm{huff}} \subseteq \mathbb{R}^n$ is the convex hull of all vectors $(v_1, \dots, v_n)$ for which there is a rooted binary tree with n leaves labelled in arbitrary order by $1, \dots, n$ 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 [15] 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's [2], see Sect. 4.1.1) 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 the result of Carr and Konjevod [3], see Sect. 4.1.4).

We conclude by briefly discussing (Sect. 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 $G = I_2(m)$).

The present paper is an extension of the extended abstract [10] that has been presented at IPCO XV (2011), Yorktown Heights. The main parts that have been added concern additional material investigating more closely the concept of an *affinely generated* reflection relation (Propositions 2 and 3) and a detailed proof of the correctness of the extended formulation of the Huffman-polytopes of size $\mathrm{O}(n \log n)$ (Theorem 7).

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

$$R(X) = \left\{ y \in \mathbb{R}^m : (x, y) \in R \text{ for some } x \in X \right\}.$$

Clearly, we have the monotonicity relation $R(X) \subseteq R(\tilde{X})$ 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* $(k_0, \dots, k_r)$ is a sequence $R_1, \dots, R_r$, 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 $(z^1, \dots, z^{r-1})$ 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 $R_r \circ \cdots \circ R_1(X) = R_r(\dots R_1(X) \dots)$ 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 $R_1, \dots, R_r$. 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

$$z^0 \in P \quad \text{and} \quad \left(z^{i-1}, z^i\right) \in R_i \quad \text{for all } i \in [r] \tag{3}$$

satisfies $\pi(Q) = \mathcal{R}(P)$, where π is the projection defined via $\pi(z^0, \dots, z^r) = z^r$. 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, $R_1, \dots, R_r$ with $f_0, f_1, \dots, f_r$ 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 $R_1, \dots, R_r$, 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 aff(R) to the first and second factor of $\mathbb{R}^n \times \mathbb{R}^m$, respectively. If 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

$$\dim(Q) \le \min\left\{k_0 + \sum_{i=1}^{r} \delta_1(R_i), k_r + \sum_{i=1}^{r} \delta_2(R_i)\right\}.$$

Remark 1 For a sequential polyhedral relation $R_1, \ldots, R_r$ of type $(k_0, \ldots, k_r)$ with induced polyhedral relation $\mathcal{R} = R_r \circ \cdots \circ R_1$, 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

$$R = \left\{(x, y) \in \mathbb{R}^n \times \mathbb{R}^m : y = \varrho(x)\right\}$$

for some affine map $\varrho : \mathbb{R}^n \to \mathbb{R}^m$. 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 the projection ϱ.

The *domain* of a polyhedral relation $R \subseteq \mathbb{R}^n \times \mathbb{R}^m$ is the polyhedron

$$\mathrm{dom}(R) = \left\{x \in \mathbb{R}^n : (x, y) \in R \text{ for some } y \in \mathbb{R}^m\right\}.$$

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

$$R(x) = \mathrm{conv} \bigcup_{f \in F} \varrho^f(x)$$

holds for all $x \in \operatorname{dom}(R)$. The maps ϱ^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

$$R(P) = \bigcup_{x \in P \cap \operatorname{dom}(R)} R(x) = \bigcup_{x \in P \cap \operatorname{dom}(R)} \operatorname{conv} \bigcup_{f \in F} \varrho^f(x)$$
$$\subseteq \operatorname{conv} \bigcup_{x \in P \cap \operatorname{dom}(R)} \bigcup_{f \in F} \varrho^f(x) = \operatorname{conv} \bigcup_{v \in V} \bigcup_{f \in F} \varrho^f(v) \subseteq \operatorname{conv} \bigcup_{v \in V} R(v),$$

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*

$$R(P) = \operatorname{conv} \bigcup_{f \in F} \varrho^f\bigl(P \cap \operatorname{dom}(R)\bigr). \tag{5}$$

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*

$$\mathcal{R}(P) \subseteq \operatorname{conv} \bigcup_{f \in F} \varrho^f\bigl(P \cap \operatorname{dom}(\mathcal{R})\bigr)$$

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 $f = (f_1, \ldots, f_r) \in F$.

Proof If $\mathcal{R}(P)$ is empty then the statement holds trivially. Otherwise, for every $z^r \in \mathcal{R}(P)$ there is a sequence $(z^0, z^1, \ldots, z^r)$ such that $z^0 \in P \cap \operatorname{dom}(R)$ and $(z^{i-1}, z^i) \in R_i$ for all $i \in [r]$. Since every relation R_i is generated by the affine maps $(\varrho^{f_i})_{f_i \in F_i}$, we conclude that for every $i \in [r]$ we have

$$z^i = \sum_{f_i \in F_i} \mu_i^{f_i} \varrho^{f_i}\bigl(z^{i-1}\bigr)$$

with some $\mu_i^{f_i} \geq 0$ for all $f_i \in F_i$ satisfying $\sum_{f_i \in F_i} \mu_i^{f_i} = 1$. Applying this iteratively, we are able to represent z^r as

$$z^r = \sum_{(f_1,\ldots,f_r)\in F} \mu_1^{f_1} \cdots \mu_r^{f_r} \varrho^{(f_1,\ldots,f_r)}\big(z^0\big),$$

where all products $\mu_1^{f_1} \cdots \mu_r^{f_r}$ are non-negative, satisfying

$$\sum_{(f_1,\ldots,f_r)\in F} \mu_1^{f_1} \cdots \mu_r^{f_r} = \Bigg(\sum_{f_1 \in F_1} \mu_1^{f_1}\Bigg) \cdots \Bigg(\sum_{f_r \in F_r} \mu_r^{f_r}\Bigg) = 1.$$

This shows that z^r belongs to $\operatorname{conv} \bigcup_{f \in F} \varrho^f(z^0)$. □

We next provide a geometric characterization of affinely generated polyhedral relations. It in particular shows why the polyhedral relation R from the example following (4) is not affinely generated (which it cannot be due to Proposition 1): The domain of that R is the interval $[0, 2]$, but there are two facets of R which project to the subintervals $[0, 1]$ and $[1, 2]$, respectively.

Proposition 2 *A polyhedral relation $R \subseteq \mathbb{R}^n \times \mathbb{R}^m$ is affinely generated if and only if $R(x)$ is a polytope (i.e., bounded) for every $x \in \operatorname{dom}(R)$ and the following holds for the projection $p : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ onto the first factor of $\mathbb{R}^n \times \mathbb{R}^m$: For every face G of R with $\dim(p(G)) = \dim(\operatorname{dom}(R))$ we have $p(G) = \operatorname{dom}(R)$.*

Proof Applying a suitable projection onto a coordinate subspace of $\mathbb{R}^n$, we find that it suffices to prove the statement for the case $\dim(\operatorname{dom}(R)) = n$.

Let us first assume that the polyhedral relation $R \subseteq \mathbb{R}^n \times \mathbb{R}^m$ is affinely generated by some finite family $(\varrho^f)_{f \in F}$. Clearly, $R(x) = \operatorname{conv}\{\varrho^f(x) : f \in F\}$ is a polytope for every $x \in \operatorname{dom}(R)$. Let G be a face of R with $\dim(p(G)) = n$. We choose a hyperplane $H \subseteq \mathbb{R}^n \times \mathbb{R}^m$ with $G = R \cap H$. It suffices to show that there is some $f^\star \in F$ with

$$\big\{x \in \mathbb{R}^n : \big(x, \varrho^{f^\star}(x)\big) \in H\big\} = \mathbb{R}^n, \tag{6}$$

because then we have $(x, \varrho^{f^\star}(x)) \in G$ for all $x \in \operatorname{dom}(R)$, which implies $p(G) = \operatorname{dom}(R)$.

Therefore, suppose that no $f^\star$ satisfying (6) exists, i.e., the affine subspace

$$\big\{x \in \mathbb{R}^n : \big(x, \varrho^f(x)\big) \in H\big\}$$

of $\mathbb{R}^n$ has dimension less than n for every $f \in F$. As we have $\dim(p(G)) = n$, this implies the existence of some $x^\star \in p(G)$ with $(x^\star, \varrho^f(x^\star)) \notin G$ for all $f \in F$, contradicting the fact that the face G of R intersects the subsets

$$\big\{x^\star\big\} \times R\big(x^\star\big) = \operatorname{conv}\big\{\big(x^\star, \varrho^f\big(x^\star\big)\big) : f \in F\big\}$$

of R.

In order to prove the reverse direction of the statement, let us assume that $R(x)$ is a polytope for all $x \in \operatorname{dom}(R)$ and that for every face G of R with $\dim(p(G)) = n$ we have $p(G) = \operatorname{dom}(R)$. We have to exhibit a finite set of affine maps generating R.

If $x^\star$ is a point in $\operatorname{dom}(R)$ and $y^\star$ is a vertex of the polytope $R(x^\star)$, then we denote by $G^\star$ the smallest face of R containing $(x^\star, y^\star)$. We claim that, for every $x \in p(G^\star)$, there is a *unique* y with $(x, y) \in G^\star$. In order to see this, suppose that there are two different $y' \neq y''$ with $(x, y'), (x, y'') \in G^\star$. Thus, with $\tilde{y} = y' - y'' \neq \mathbb{O}$ the vector $(\mathbb{O}, \tilde{y})$ lies in the linear subspace parallel to $\operatorname{aff}(G^\star)$. Since $(x^\star, y^\star)$ is contained in the relative interior of $G^\star$ (due to its minimality), there is some $\varepsilon > 0$ with

$$\left(x^\star, y^\star\right) \pm \varepsilon(\mathbb{O}, \tilde{y}) \in G \subseteq R,$$

thus $y^\star \pm \varepsilon\tilde{y} \in R(x^\star)$, contradicting the fact that $y^\star$ is a *vertex* of $R(x^\star)$.

Hence, for every face $G^\star$ of R as defined above, the projection $p : G^\star \to p(G^\star)$ has an inverse, which is a restriction of an affine map $\varrho^{G^\star}$ to $p(G^\star)$. Of course, we can use such a map $\varrho^{G^\star}$ only as one of the maps in the family to be constructed if we have $p(G^\star) = \operatorname{dom}(R)$, which, by our assumption, is guaranteed to hold in case of $\dim(p(G^\star)) = n$. Therefore, let us denote by $\mathcal{G}$ the set of all faces G of R such that

- G is the face $G^\star$ defined for some pair $(x^\star, y^\star)$ as above and
- $\dim(p(G)) = n$.

The second condition here implies that, for every $G \in \mathcal{G}$, the affine map ϱ^G is defined on the entire polyhedron $\operatorname{dom}(R)$ with $(x, \varrho^G(x)) \in G \subseteq R$. Thus we have

$$\varrho^G(x) \in R(x) \quad \text{for all } x \in \operatorname{dom}(R), G \in \mathcal{G}. \tag{7}$$

Furthermore, for every $x^\star \in \operatorname{dom}(R)$ such that for every vertex $y^\star$ of $R(x^\star)$ the face $G^\star$ satisfies $\dim(p(G^\star)) = n$, we have

$$R\left(x^\star\right) = \operatorname{conv}\left\{\varrho^G\left(x^\star\right) : G \in \mathcal{G}\right\}.$$

As $\dim(p(G^\star)) = n$ can only fail to hold if $x^\star$ is contained in one of the finitely many less than n-dimensional images of faces of R under the projection p, we find that

$$R(x) = \operatorname{conv}\left\{\varrho^G(x) : G \in \mathcal{G}\right\} \tag{8}$$

holds for all $x \in D$ with some subset D of the n-dimensional polyhedron $\operatorname{dom}(R)$ whose topological closure equals $\operatorname{dom}(R)$.

It follows readily from (7) that in (8) the relation $\supseteq$ holds for all $x \in \operatorname{dom}(R)$. In order to establish the same for the reverse inclusion, let $(x, y) \in R$ and fix some $(\bar{x}, \bar{y}) \in R$ with $\bar{x} \in D$. For every $k \in \{1, 2, \dots\}$, we choose some

$$\left(x^k, y^k\right) \in \operatorname{conv}\left\{(x, y), (\bar{x}, \bar{y})\right\} \quad \text{with } \left\|\left(x^k, y^k\right) - (x, y)\right\| \le \frac{1}{k} \text{ and } x^k \in D.$$

Clearly, we have $\lim_{k\to\infty} x^k = x$ and $\lim_{k\to\infty} y^k = y$. For every $k \in \{1, 2, \dots\}$, due to $x^k \in D$, there is some multiplier vector $\lambda^k \in \mathbb{R}_+^{\mathcal{G}}$ with

$$\sum_{G\in\mathcal{G}} \lambda_G^k = 1 \quad \text{and} \quad y^k = \sum_{G\in\mathcal{G}} \lambda_G^k \varrho^G\bigl(x^k\bigr).$$

As the λ^k form a bounded sequence, there is some subset $K \subseteq \{1, 2, \dots\}$ such that the limit

$$\lim_{\substack{k\to\infty\\ k\in K}} \lambda^k =: \lambda \in \mathbb{R}_+^{\mathcal{G}}$$

exists with $\sum_{G\in\mathcal{G}} \lambda_G = 1$. Due to the continuity of the affine maps ϱ^G we hence obtain

$$y = \lim_{\substack{k\to\infty\\ k\in K}} y^k = \lim_{\substack{k\to\infty\\ k\in K}} \sum_{G\in\mathcal{G}} \lambda_G^k \varrho^G\bigl(x^k\bigr) = \sum_{G\in\mathcal{G}} \lambda_G \varrho^G(x),$$

showing that y indeed is contained in the set on the right-hand-side of (8). □

Proposition 2 provides a characterization of those polyhedral relations that are affinely generated. In particular, it shows that being affinely generated really is a special feature of a polyhedral relation.

For the purpose of designing an extended formulation for some polytope P, one often would like to find a polyhedral relation R that is affinely generated by some specific affine maps such that P is the convex hull of the images of some well-described polytope in $\operatorname{dom}(R)$ under these maps. The following result deals with the special case of $m = n$ and two affine maps, one of which being the identity map. It shows that the other map, restricted to the domain of the polyhedral relation to be constructed, must be of a very special type. The following section will then demonstrate how for these special types of maps one can indeed construct polyhedral relations with useful domains.

Before we state the result, let us introduce some notation, which will be used throughout the paper. For $a \in \mathbb{R}^n \setminus \{\mathbb{O}\}$ and $\beta \in \mathbb{R}$, we denote by

$$\mathrm{H}^{=}(a, \beta) = \bigl\{x \in \mathbb{R}^n : \langle a, x\rangle = \beta\bigr\}$$

the hyperplane defined by the equation $\langle a, x\rangle = \beta$ and by

$$\mathrm{H}^{\leq}(a, \beta) = \bigl\{x \in \mathbb{R}^n : \langle a, x\rangle \leq \beta\bigr\}$$

the halfspace defined by the inequality $\langle a, x\rangle \leq \beta$.

Proposition 3 *If $\varrho : \mathbb{R}^n \to \mathbb{R}^n$ is an affine map and $R \subseteq \mathbb{R}^n \times \mathbb{R}^n$ is a polyhedral relation that is affinely generated by ϱ and the identity map, then ϱ restricted to* $\operatorname{dom}(R)$ *is a translation, or there exist $a, c \in \mathbb{R}^n \setminus \{\mathbb{O}\}$ and $\beta \in \mathbb{R}$ such that*

- $\operatorname{dom}(R) \subseteq \mathrm{H}^{\leq}(a, \beta)$ *holds and*

- *for every $x \in \mathrm{dom}(R)$ the vector $\varrho(x) - x$ is parallel to c with*

$$\varrho(x) - x = \big(\langle a, x\rangle - \beta\big)c. \tag{9}$$

Proof Let $R \subseteq \mathbb{R}^n \times \mathbb{R}^n$ is a polyhedral relation that is affinely generated by the identity map and an affine map ϱ which on $\mathrm{dom}(R)$ is not a translation. We have to exhibit $a, c \in \mathbb{R}^n \setminus \{\mathbb{O}\}$ and $\beta \in \mathbb{R}$ as described in the statement.

First we show that for all $x, y \in \mathrm{dom}(R)$ the vectors $\varrho(x) - x$ and $\varrho(y) - y$ are parallel. Indeed, since R is generated by ϱ and the identity map, we have $(x, x), (x, \varrho(x)), (y, y), (y, \varrho(y)) \in R$. Thus, the one-dimensional polytope $R(\frac{1}{2} \times (x + y))$ contains the points $\frac{1}{2}(x + y)$, $\frac{1}{2}(\varrho(x) + y)$, and $\frac{1}{2}(x + \varrho(y))$, implying that $\varrho(x) - x$ and $\varrho(y) - y$ are linearly dependent.

As the restriction of ϱ to $\mathrm{dom}(R)$ is not the identity, we can choose c to be some non-zero vector $\varrho(x) - x$ with $x \in \mathrm{dom}(R)$. For simplicity of representation, we may assume that the vector c is the vector $\mathbb{e}_n$. Thus, for every point $x \in \mathrm{dom}(R)$ we have $\varrho(x)_i = x_i$ for all $i \in [n-1]$ and $\varrho(x)_n = \sum_{i \in [n]} \alpha_i x_i - \beta$ for some numbers $\alpha_i \in \mathbb{R}$ ($i \in [n]$) and $\beta \in \mathbb{R}$.

Let $a \in \mathbb{R}^n$ be the vector defined via $a_i = \alpha_i$ for all $i \in [n-1]$ and $a_n = \alpha_n - 1$. Clearly, with this choice (9) is satisfied. Furthermore, we have $a \neq \mathbb{O}$, because otherwise ϱ would be a translation (by $-\beta\mathbb{e}_n$). Hence, it remains to show that $\mathrm{dom}(R)$ is contained in one of the two closed halfspaces into which $\mathbb{R}^n$ is divided by the hyperplane $\mathrm{H}^{=}(a, \beta)$ (if this halfspace is not $\mathrm{H}^{\leq}(a, \beta)$, we can just replace $c = \mathbb{e}_n$ by $c = -\mathbb{e}_n$ and (a, β) by $(-a, -\beta)$). Note that a point from $\mathrm{dom}(R)$ is contained in $\mathrm{fH}^{=}(a, \beta)$ if and only if it is mapped to itself by ϱ.

Therefore, suppose that there are $x, y \in \mathrm{dom}(R)$ with

$$\langle a, x\rangle < \beta < \langle a, y\rangle.$$

Let $0 < \lambda < 1$ be the scalar with

$$z = \lambda x + (1 - \lambda)y \in \mathrm{H}^{=}(a, \beta).$$

We have $\varrho(z) = z$, thus $R(z) = \{z\}$ (since R is generated by ϱ and the identity). Due to $(x, x), (y, \varrho(y)) \in R$ we obtain $(z, \lambda x + (1 - \lambda)\varrho(y)) \in R$, hence

$$\lambda x + (1 - \lambda)\varrho(y) = z = \lambda x + (1 - \lambda)y,$$

which (due to $\lambda \neq 1$) implies $\varrho(y) = y$, contradicting $y \notin \mathrm{H}^{=}(a, \beta)$. □

The range of the affine maps ϱ left as possibilities in Proposition 3 includes translations, shearing transformations, and reflections at hyperplanes. For the latter ones we will explicitly construct polyhedral relations in the next section that will become quite useful later. This construction could in fact easily be modified to also work for all other types of affine maps covered by Proposition 3.

For the relation between the concept of affinely generated reflection relations and a construction method described in [13] see [4].

3 Reflection Relations

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

$$\mathrm{R}_{a,\beta} = \left\{(x,y) \in \mathbb{R}^n \times \mathbb{R}^n : y - x \in \left(\mathrm{H}^{=}(a,\beta)\right)^{\perp}, \langle a, x\rangle \le \langle a, y\rangle \le 2\beta - \langle a, x\rangle\right\}$$

(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 $(n,\dots,n)$ 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 4 *For $a \in \mathbb{R}^n \setminus \{\mathbb{O}\}$, $\beta \in \mathbb{R}$ 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 ϱ^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 $\varrho^H(x)$. □

Note that for all other affine maps described in Proposition 3 one may construct a corresponding polyhedral relation in a similar way as done above for reflection relations.

From Proposition 1 and Proposition 4, one obtains the following result.

Corollary 1 *If $P \subseteq \mathbb{R}^n$ is a polytope, then we have, for $a \in \mathbb{R}^n \setminus \{\mathbb{O}\}$ and $\beta \in \mathbb{R}$ defining the hyperplane $H = \mathrm{H}^{=}(a,\beta)$ and the halfspace $H^{\le} = \mathrm{H}^{\le}(a,\beta)$,*

$$\mathrm{R}_{a,\beta}(P) = \mathrm{conv}\left(\left(P \cap H^{\le}\right) \cup \varrho^H\left(P \cap H^{\le}\right)\right).$$

While Corollary 1 describes images under single reflection relations, for analyses of the images under sequences of reflection relations we define, for each $a \in \mathbb{R}^n \setminus \{\mathbb{O}\}$, $\beta \in \mathbb{R}$, $H^{\leq} = \mathrm{H}^{\leq}(a, \beta)$, and $H = \mathrm{H}^{=}(a, \beta)$, the map $\varrho^{\star(H^{\leq})} : \mathbb{R}^n \to \mathbb{R}^n$ via

$$\varrho^{\star(H^{\leq})}(y) = \begin{cases} y & \text{if } y \in H^{\leq} \\ \varrho^{H}(y) & \text{otherwise} \end{cases}$$

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 $\mathrm{R}_{H_r^{\leq}} \circ \cdots \circ \mathrm{R}_{H_1^{\leq}}$, then we have

$$y \in \mathcal{R}\big(\varrho^{\star(H_1^{\leq})} \circ \cdots \circ \varrho^{\star(H_r^{\leq})}(y)\big) \tag{10}$$

for all $y \in \mathbb{R}^n$.

Theorem 1 *For $\mathcal{R} = \mathrm{R}_{H_r^{\leq}} \circ \cdots \circ \mathrm{R}_{H_1^{\leq}}$ 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 ϱ^{H_i} lies in Q. Thus, from Lemma 1 we have the inclusion $\mathcal{R}(P) \subseteq Q$. By the second condition and (10), 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

$$\mathrm{sign}(P) = \mathrm{conv} \bigcup_{\varepsilon \in \{-,+\}^n} \varepsilon.P,$$

where $\varepsilon.x$ 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 $\mathrm{R}_{-\mathbb{e}_k,0}$, 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 5 *If $\mathcal{R}$ is the polyhedral relation $\mathrm{S}_n \circ \cdots \circ \mathrm{S}_1$ and $P \subseteq \mathbb{R}^n$ is a polytope with $v^{\mathrm{abs}} \in P$ 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 $W = \{\varepsilon.v : \varepsilon \in \{-,+\}^n, v \text{ vertex of } P\}$. As, for every $w \in W$ with $w = \varepsilon.v$ for some vertex v of P and $\varepsilon \in \{-,+\}^n$, 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 5 and Remark 2 imply the following.

Theorem 2 *For each polytope $P \subseteq \mathbb{R}^n$ with $v^{\mathrm{abs}} \in P$ 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 $\mathbb{O} \in H_i \subseteq \mathbb{R}^n$ containing the origin. We refer to [6, 8] 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 $\mathbb{O} \in H$)—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 $P \subseteq \mathbb{R}^n$ the *G-permutahedron $\Pi_G(P)$* of P is the convex hull 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.

To see this, let us assume that the set of reflection hyperplanes $\mathcal{H}$ can 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 $\mathcal{H}_2$. Let $\mathcal{H}_1$, $\mathcal{H}_2$ induce two reflection groups G_1, G_2. Then, we can represent the G-permutahedron as

$$\Pi_G(P) = \Pi_{G_1}\big(\Pi_{G_2}(P)\big).$$

Moreover, for every reflection map ϱ^{H_2}, $H_2 \in \mathcal{H}_2$, and for $a \in \mathbb{R}^n$, $b \in \mathbb{R}$, such that $H_1 = \mathrm{H}^{=}(a,b)$, $H_1 \in \mathcal{H}_1$, we have $\langle a, x\rangle = \langle a, \varrho^{H_2}(x)\rangle$ for all $x \in \mathbb{R}^n$. Hence, we can apply Theorem 1 for the polytope $\Pi_{G_2}(P)$ and the group G_1, whenever the conditions of Theorem 1 hold for the polytope P and for both groups G_1 and G_2.

4.1.1 The Reflection Group $I_2(m)$

For $\varphi \in \mathbb{R}$, let us denote

$$H_\varphi = \mathrm{H}^{=}\big((-\sin\varphi, \cos\varphi), 0\big) \quad \text{and} \quad H_\varphi^{\leq} = \mathrm{H}^{\leq}\big((-\sin\varphi, \cos\varphi), 0\big).$$

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 $\Phi_{I_2(m)} = \{x \in \mathbb{R}^2 : x_2 \geq 0, x \in H^{\leq}_{\pi/m}\}$ as the fundamental domain.

Proposition 6 *If $\mathcal{R}$ is the polyhedral relation*

$$\mathrm{R}_{H^{\leq}_{2^r\pi/m}} \circ \cdots \circ \mathrm{R}_{H^{\leq}_{2\pi/m}} \circ \mathrm{R}_{H^{\leq}_{\pi/m}}$$

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 $Q = \mathrm{conv}(W)$ 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 6 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, this construction yields extended formulations of regular m-gons 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].

4.1.2 The Reflection Group A_{n-1}

The group A_{n-1} is generated by the reflections at the hyperplanes $\mathrm{H}^{=}(\mathbb{e}_k - \mathbb{e}_\ell, 0)$ in $\mathbb{R}^n$ 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 $\mathrm{conv}\{\mathbb{e}_1, \ldots, \mathbb{e}_n\} \subseteq \mathbb{R}^n$. 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

$$\Pi_{A_{n-1}}(P) = \mathrm{conv} \bigcup_{\gamma \in \mathfrak{S}(n)} \gamma.P,$$

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 $\mathrm{T}_{k,\ell} = \mathrm{R}_{\mathbb{e}_k - \mathbb{e}_\ell, 0} \subseteq \mathbb{R}^n \times \mathbb{R}^n$. The corresponding reflection $\tau_{k,\ell} = \varrho^{H_{k,\ell}} : \mathbb{R}^n \to \mathbb{R}^n$ with $H_{k,\ell} = \mathrm{H}^{=}(\mathbb{e}_k - \mathbb{e}_\ell, 0)$ 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 $(k_1, \ell_1), \dots, (k_r, \ell_r) \in [n] \times [n]$ 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 7 *If $\mathcal{R}$ is a polyhedral relation $\mathrm{T}_{k_r,\ell_r} \circ \cdots \circ \mathrm{T}_{k_1,\ell_1}$, where the sequence $(k_1, \ell_1), \dots, (k_r, \ell_r) \in [n] \times [n]$ is a sorting network, and $P \subseteq \mathbb{R}^n$ is a polytope with $v^{\text{sort}} \in P$ 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 $Q = \text{conv}(W)$ 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 7 and Remark 2 we can conclude the following theorem.

Theorem 4 *For each polytope $P \subseteq \mathbb{R}^n$ with $v^{\text{sort}} \in P$ 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 $P = \{(1, 2, \dots, n)\} \subseteq \mathbb{R}^n$, 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 [7] (see the remarks in the introduction).

Letting $G = A_{n-1}$ act on the first n components of $\mathbb{R}^n \times \mathbb{R}^{n+1}$ and choosing P as the convex hull of the vectors $(\mathbb{1}_i, \mathbb{O}_{n-i}, \mathbb{e}_{i+1})$ $(i \in \{0, 1, \dots, n\})$, the construction behind Theorem 4 yields an extended formulation with $\mathrm{O}(n \log n)$ variables and inequalities of the cardinality indicating polytope $\Pi_G(P)$.

4.1.3 The Reflection Group B_n

The group B_n is generated by the reflections in $\mathbb{R}^n$ at the hyperplanes $\mathrm{H}^{=}(\mathbb{e}_k + \mathbb{e}_\ell, 0)$, $\mathrm{H}^{=}(\mathbb{e}_k - \mathbb{e}_\ell, 0)$ and $\mathrm{H}^{=}(\mathbb{e}_k, 0)$ for all pairwise distinct $k, \ell \in [n]$. It is the symmetry group of both the n-dimensional cube $\operatorname{conv}\{-1,+1\}^n$ and the n-dimensional cross-polytope $\operatorname{conv}\{\pm\mathbb{e}_1, \dots, \pm\mathbb{e}_n\}$. We choose $\Phi_{B_n} = \{x \in \mathbb{R}^n : 0 \le x_1 \le \cdots \le x_n\}$ 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 $v' = y^{\text{abs}}$.

Proposition 8 *If $\mathcal{R}$ is a polyhedral relation $S_n \circ \cdots \circ S_1 \circ \mathrm{T}_{k_r,\ell_r} \circ \cdots \circ \mathrm{T}_{k_1,\ell_1}$, where $(k_1,\ell_1), \dots, (k_r,\ell_r) \in [n] \times [n]$ is a sorting network (and the S_i are defined as at the end of Sect. 3) and $P \subseteq \mathbb{R}^n$ is a polytope with $v^{\text{sort-abs}} \in P$ 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 $W = \{\gamma.\varepsilon.v : \gamma \in \mathfrak{S}(n), \varepsilon \in \{-,+\}^n, v \text{ vertex of } P\}$. As, for every $w \in W$ with $w = \gamma.\varepsilon.v$ for some vertex v of P and $\gamma \in \mathfrak{S}(n)$, $\varepsilon \in \{-,+\}^n$, 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 8 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.*

4.1.4 The Reflection Group D_n

The group D_n is generated by the reflections in $\mathbb{R}^n$ at the hyperplanes $\mathrm{H}^{=}(\mathbb{e}_k + \mathbb{e}_\ell, 0)$ and $\mathrm{H}^{=}(\mathbb{e}_k - \mathbb{e}_\ell, 0)$ 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 $k \ne \ell$, we denote the polyhedral relation $\mathrm{R}_{-\mathbb{e}_k-\mathbb{e}_\ell,0} \circ \mathrm{R}_{\mathbb{e}_k-\mathbb{e}_\ell,0}$ by $E_{k,\ell}$.

Proposition 9 *If $\mathcal{R}$ is a polyhedral relation $E_{n-1,n} \circ \cdots \circ E_{1,2} \circ \mathrm{T}_{k_r,\ell_r} \circ \cdots \circ \mathrm{T}_{k_1,\ell_1}$, 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 $\varepsilon \in \{-,+\}^n$ with an even number of components equal to minus. Then, we have $Q = \operatorname{conv}(W)$ with

$$W = \left\{\gamma.\varepsilon.v : \gamma \in \mathfrak{S}(n), \varepsilon \in \{-,+\}^n_{\text{even}}, v \text{ vertex of } P\right\}.$$

For $k,\ell \in [n]$ with $k \neq \ell$, we define $\eta^\star_{k,\ell} = \varrho^{\star(\mathrm{H}^\leq(\mathbb{e}_k-\mathbb{e}_\ell,0))} \circ \varrho^{\star(\mathrm{H}^\leq(-\mathbb{e}_k-\mathbb{e}_\ell,0))}$. 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 $|y'_k| \leq y'_\ell$, where $\rho_{k,\ell}(y)$ arises from y by multiplying both components k and ℓ by -1. As, for every $w \in W$ with $w = \gamma.\varepsilon.v$ for some vertex v of P and $\gamma \in \mathfrak{S}(n)$, $\varepsilon \in \{-,+\}^n_{\text{even}}$, 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 9 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 9. 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 $y \mapsto \frac{1}{2}(\mathbb{1} - y)$, we derive extended formulations with $2(n-1)$ variables and $4(n-1)$ inequalities for the *parity polytopes* $\operatorname{conv}\{v \in \{0,1\}^n : \sum_i v_i \text{ odd}\}$ and $\operatorname{conv}\{v \in \{0,1\}^n : \sum_i v_i \text{ even}\}$, respectively. This reproves a result due to Carr and Konjevod [3]. In fact, one can show that their's and our extension are affinely isomorphic to each other.

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 $1, \dots, n$ 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 [15] proved that $\mathrm{P}^n_{\mathrm{huff}}$ has $(\Omega(n))!$ facets.

We are going to construct small extended formulations for Huffman polytopes, where the constructions rely on the following obvious properties.

Observation 1

1. *For every $\gamma \in \mathfrak{S}(n)$*
$$\gamma.\mathrm{V}^n_{\mathrm{huff}} = \mathrm{V}^n_{\mathrm{huff}}.$$
2. *For every $v \in \mathrm{V}^n_{\mathrm{huff}}$, there are at least two components of v equal to*
$$\max_{k \in [n]} v_k.$$
3. *For every $v \in \mathrm{V}^n_{\mathrm{huff}}$ and*
$$v_i = v_j = \max_{k \in [n]} v_k$$
for some pair of distinct i, j, the point
$$(v_1, \dots, v_{i-1}, v_i - 1, v_{i+1}, \dots, v_{j-1}, v_{j+1}, \dots, v_n)$$
lies in $\mathrm{V}^{n-1}_{\mathrm{huff}}$.
4. *For every $x \in \mathrm{V}^{n-1}_{\mathrm{huff}}$, the point*
$$(x_1, \dots, x_{n-2}, x_{n-1} + 1, x_{n-1} + 1)$$
lies in $\mathrm{V}^n_{\mathrm{huff}}$.

To construct an extended formulation of the Huffman polytope, we need to define the embedding

$$P^{n-1} = \left\{(x_1, \dots, x_{n-2}, x_{n-1} + 1, x_{n-1} + 1) : (x_1, \dots, x_{n-1}) \in \mathrm{P}^{n-1}_{\mathrm{huff}}\right\}$$

of $\mathrm{P}^{n-1}_{\mathrm{huff}}$ into $\mathbb{R}^n$.

Proposition 10 *If $\mathcal{R} \subseteq \mathbb{R}^n \times \mathbb{R}^n$ (with $n \geq 3$) is the polyhedral relation*

$$\mathrm{T}_{1,2} \circ \mathrm{T}_{2,3} \circ \cdots \circ \mathrm{T}_{n-2,n-1} \circ \mathrm{T}_{n-1,n} \circ \mathrm{T}_{1,2} \circ \mathrm{T}_{2,3} \circ \cdots \circ \mathrm{T}_{n-3,n-2} \circ \mathrm{T}_{n-2,n-1}, \quad (11)$$

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, what is due to parts (1) and (4) of Observation 1. We have $Q = \mathrm{conv}(X)$ with $X = \mathrm{V}^n_{\mathrm{huff}}$. Furthermore, for every $x \in X$ and $y = \tau^\star(x)$ with

$$\tau^\star = \tau^\star_{n-2,n-1} \circ \tau^\star_{n-3,n-2} \circ \cdots \circ \tau^\star_{2,3} \circ \tau^\star_{1,2} \circ \tau^\star_{n-1,n} \circ \tau^\star_{n-2,n-1} \circ \cdots \circ \tau^\star_{2,3} \circ \tau^\star_{1,2}, \quad (12)$$

we have

$$y_n = y_{n-1} = \max_{i \in [n]} x_i ,$$

the part (3) of Observation 1 implies $\tau^\star(x) \in P^{n-1}$. Therefore, the claim follows by Theorem 1. □

Thus, from Remark 2, we get 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. Since the Huffman polytope $\mathrm{P}^2_{\mathrm{huff}}$ is a single point, inductive application of this approach leads to the following result.

Proposition 11 *For the Huffman polytope* $\mathrm{P}^n_{\mathrm{huff}}$*, there is an extended formulation of size* $O(n^2)$.

Actually, the Huffman polytope $\mathrm{P}^n_{\mathrm{huff}}$ even has an extended formulation of size $O(n \log n)$, but this requires another sorting approach. In order to indicate the necessary modifications, let us denote by Θ_k the sequence

$$(k-2, k-1), (k-3, k-2), \dots, (1,2), (k-1, k), (k-2, k-1), \dots, (1,2)$$

of index pairs, which are used in (11) and (12). For every sequence

$$\Theta = \big((i_1, j_1), \dots, (i_r, j_r)\big)$$

of pairs of distinct indices, we define

$$\tau^\star_\Theta = \tau^\star_{i_1, j_1} \circ \cdots \circ \tau^\star_{i_r, j_r} ,$$

thus $\tau^\star_{\Theta_n}$ is denoted by $\tau^\star$ in (12). Furthermore, let $\pi_k : \mathbb{R}^k \to \mathbb{R}^{k-1}$ be the linear map defined via

$$\pi_k(y) = (y_1, \dots, y_{k-2}, y_{k-1} - 1)$$

for all $y \in \mathbb{R}^k$. For the above construction, we need, that for every $v \in \mathrm{V}^n_{\mathrm{huff}}$ and every $k \geq 3$, the vector

$$x^k = \tau^\star_{\Theta_k} \circ \pi_{k+1} \circ \tau^\star_{\Theta_{k+1}} \circ \cdots \circ \pi_n \circ \tau^\star_{\Theta_n}(v) \quad (13)$$

satisfies

$$x^k_{k-1} = x^k_k = \max_{i \in [k]} x^k_i .$$

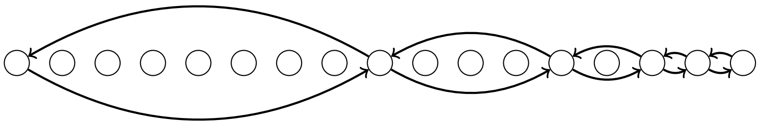


Fig. 1 Illustration of the sequence replacing Θ_k in order to reduce the size of the extension of $\mathrm{P}^n_{\mathrm{huff}}$ to $\mathrm{O}(n \log n)$

It turns out, that this property is preserved, when replacing the sequence Θ_n by an arbitrary sorting network, and for every $k \geq 3$, the sequence Θ_k by the sequence

$$\left(i^k_2, i^k_1\right), \left(i^k_3, i^k_2\right), \ldots, \left(i^k_{r_k}, i^k_{r_k-1}\right), \left(i^k_{r_k-1}, i^k_{r_k-2}\right), \ldots, \left(i^k_3, i^k_2\right), \left(i^k_2, i^k_1\right)$$

with

$$i^k_t = \begin{cases} k & \text{if } t = 1 \\ k-1 & \text{if } t = 2 \\ i^k_{t-1} - 2^{t-3} & \text{otherwise} \end{cases}$$

and where r_k is the maximal t, such that i^k_t is greater than zero. Denote by J_k the set of indices, involved in this sorting transformation Θ_k, i.e.

$$J_k = \left\{ i^k_t : t \in [r_k] \right\}$$

(see Fig. 1).

Proposition 12 *For every $2 \leq k \leq n$, the Huffman vector x^k, defined by (13), is sorted or the Huffman vector x^k has the following form*

$$\begin{aligned}
x^k_k &= \cdots = x^k_{k-p_k+1} &&= \max_{i\in[k]} x^k_i \\
x^k_{k-p_k} &= \cdots = x^k_{k-p_k-q_k+1} &&= \max_{i\in[k]} x^k_i - 1 \\
x^k_{k-p_k-q_k} &= \cdots = x^k_{k-p_k-q_k-\ell_k+1} &&= \max_{i\in[k]} x^k_i \\
x^k_1 &\leq \cdots \leq x^k_{k-p_k-q_k-\ell_k} &&\leq \max_{i\in[k]} x^k_i - 1,
\end{aligned}$$

where the index $k - p_k - q_k + 1$ belongs to J^k and p_k is strictly greater than ℓ_k.

Proof The proof is by induction on the number n, i.e. we assume, that if a vector $x^k \in \mathbb{R}^k$ satisfies

$$x^k = \tau^\star_{\Theta_k} \circ \pi_{k+1} \circ \tau^\star_{\Theta_{k+1}} \circ \cdots \circ \pi_m(x)$$

for a sorted Huffman vector $x \in \mathbb{R}^m$, where $m < n$, then the vector x^k satisfies the claim above.

If the Huffman vector

$$y^{n-1} = \pi_n \circ \tau^{\star}_{\Theta_n}(v)$$

is sorted, then we can apply the induction assumption for $m = n - 1$ and the Huffman vector

$$x = \tau^{\star}_{\Theta_{n-1}} \circ \pi_n \circ \tau^{\star}_{\Theta_n}(v).$$

Otherwise, for the Huffman vector y^{n-1}, we have

$$y^{n-1}_{n-1} = u - 1 \quad \text{and} \quad y^{n-1}_1 \le \cdots \le y^{n-1}_{n-2} = u,$$

where u is the maximum value among the coordinates of the Huffman vector x^n. After application of the sorting transformation Θ_{n-1} to y^{n-1}, we get the Huffman vector x^{n-1} with

$$\begin{aligned}
&x^{n-1}_{n-1} &&= \cdots = x^{n-1}_{(n-1)-p_{n-1}+1} &&= u\\
&x^{n-1}_{(n-1)-p_{n-1}} && &&= u - 1\\
&x^{n-1}_{(n-1)-p_{n-1}-1} &&= \cdots = x^{n-1}_{(n-1)-p_{n-1}-\ell_{n-1}} &&= u\\
&x^{n-1}_1 &&\le \cdots \le x^{n-1}_{(n-1)-p_{n-1}-\ell_{n-1}-1} &&\le u - 1,
\end{aligned}$$

where $p_{n-1} = 2^{i-1}$ and $\ell_{n-1} < 2^{i-1}$. If the Huffman vector

$$x^{n-1} = \tau^{\star}_{\Theta_{n-1}} \circ \pi_n \circ \tau^{\star}_{\Theta_n}(v)$$

is sorted, i.e. $l_{n-1} = 0$, then the induction assumption for $m = n - 1$ and $x = x^{n-1}$ finishes the proof. Otherwise, the index $(n - 1) - p_{n-1}$ belongs to J_{n-1}, thus the assumption of proposition holds for $k = n - 1$.

Let us assume, that for the Huffman vector

$$x^k = \tau^{\star}_{\Theta_k} \circ \cdots \circ \pi_n \circ \tau^{\star}_{\Theta_n}(v)$$

the claim holds. Then, the Huffman vector

$$y^{k-1} = \pi_k \circ \tau^{\star}_{\Theta_k} \circ \cdots \circ \pi_n \circ \tau^{\star}_{\Theta_n}(v)$$

has components

$$\begin{aligned}
&y^{k-1}_{k-1} && &&= u - 1\\
&y^{k-1}_{k-2} &&= \cdots = y^{k-1}_{k-p_k+1} &&= u\\
&y^{k-1}_{k-p_k} &&= \cdots = y^{k-1}_{k-p_k-q_k+1} &&= u - 1\\
&y^{k-1}_{k-p_k-q_k} &&= \cdots = y^{k-1}_{k-p_k-q_k-\ell_k+1} &&= u\\
&x^k_1 &&\le \cdots \le x^k_{k-p_k-q_k-\ell_k} &&\le u - 1.
\end{aligned}$$

Obviously, the set of indices J_{k-1} is obtained from the set of indices J_k, decreasing every element by one and excluding the index zero. Hence, the index $(k-1)-(p_k-1)-q_k$ belongs to the index set J_{k-1}.

Let us consider the coordinates of y^{k-1} with indices in J_{k-1}, i.e. the coordinates participating in Θ_{k-1}. Note that there exists just one u in this sequence before the $u-1$ block, since $(k-1)-(p_k-1)-q_k$ belongs to the indices set J_{k-1} and $p_k > l_k$. Clearly, the action of $\tau^\star_{\Theta_{k-1}}$ is equivalent to swapping of the first u-value with the last $(u-1)$-value in this sequence of coordinates. Thus after the sorting transformation $\tau^\star_{\Theta_{k-1}}$ the Huffman vector

$$x^{k-1} = \tau^\star_{\Theta_{k-1}} \circ \pi_k \circ \tau^\star_{\Theta_k} \circ \cdots \circ \pi_n \circ \tau^\star_{\Theta_n}(v)$$

has the desired form, and we have $l_{k-1} < p_{k-1}$ and $(k-1)-p_{k-1}-q_{k-1} \in J^{k-1}$. $\square$

To finish the construction, we have to verify that

$$x^k_{k-1} = x^k_k = \max_{i\in[k]} x^k_i$$

holds for the Huffman vector x^k (for $k \geq 3$). Obviously, this follows from Proposition 12, because the inequality $p_k > \ell_k$ implies $p_k \geq 2$, since every Huffman vector has an even number of maximal elements, i.e. $p_k + \ell_k$ has to be even. We obtain the following theorem, since the number r_k is bounded by $O(\log k)$ and since there are sorting networks of size $O(n \log n)$.

Theorem 7 *For the Huffman polytope $\mathrm{P}^n_{\mathrm{huff}}$, there is an extended formulation of size $O(n \log n)$.*

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 at all.

Acknowledgements Martin Grötschel is our academic grandfather (Volker Kaibel) and great-grandfather (Kanstantsin Pashkovich). Next to our deep appreciation of his scientific work and of his influence on the development of the field of Combinatorial Optimization, Volker Kaibel is in particular grateful for Martin Grötschel's great support and for the excellent conditions (except for occasionally having to prepare *PowerPoint*-slides) he enjoyed while working at Zuse-Institute

Berlin in 2005 and 2006. It is a true pleasure to contribute to this volume dedicated to Martin's 65th birthday.

With respect to the work presented in this paper, we thank Samuel Fiorini, Michel Goemans, Günter Rote and Dirk Oliver Theis for valuable hints and discussions as well as the referee for all her or his efforts.

Volker Kaibel acknowledges funding by *Deutsche Forschungsgemeinschaft* (KA 1616/4-1 *Extended Formulations in Combinatorial Optimization*) and Kanstantsin Pashkovich is grateful for support by the *International Max Planck Research School (IMPRS) for Analysis, Design and Optimization in Chemical and Biochemical Process Engineering Magdeburg* and by the *Progetto di Eccellenza 2008–2009* of the Fondazione Cassa Risparmio di Padova e Rovigo.

References

1. Ajtai, M., Komlós, J., Szemerédi, E.: Sorting in $c \log n$ parallel steps. Combinatorica **3**(1), 1–19 (1983). doi:10.1007/BF02579338
2. Ben-Tal, A., Nemirovski, A.: On polyhedral approximations of the second-order cone. Math. Oper. Res. **26**(2), 193–205 (2001). doi:10.1287/moor.26.2.193.10561
3. Carr, R.D., Konjevod, G.: Polyhedral combinatorics. In: Greenberg, H. (ed.) Tutorials on Emerging Methodologies and Applications in Operations Research, Chap. 2, pp. (2-1)–(2-48). Springer, Berlin (2004)
4. Conforti, M., Pashkovich, K.: The projected faces property and polyhedral relations. http://arxiv.org/abs/1305.3782 (2013)
5. Conforti, M., Cornuéjols, G., Zambelli, G.: Extended formulations in combinatorial optimization. 4OR **8**(1), 1–48 (2010). doi:10.1007/s10288-010-0122-z
6. Fomin, S., Reading, N.: Root systems and generalized associahedra. In: Geometric Combinatorics. IAS/Park City Math. Ser., vol. 13, pp. 63–131. Am. Math. Soc., Providence (2007)
7. Goemans, M.: Smallest compact formulation for the permutahedron. http://www-math.mit.edu/~goemans/publ.html
8. Humphreys, J.E.: Reflection Groups and Coxeter Groups. Cambridge Studies in Advanced Mathematics, vol. 29. Cambridge University Press, Cambridge (1990)
9. Kaibel, V., Loos, A.: Branched polyhedral systems. In: Eisenbrand, F., Shepherd, B. (eds.) Integer Programming and Combinatorial Optimization (Proc. IPCO XIV). LNCS, vol. 6080, pp. 177–190. Springer, Berlin (2010)
10. Kaibel, V., Pashkovich, K.: Constructing extended formulations from reflection relations. In: Günlük, O., Woeginger, G. (eds.) Integer Programming and Combinatorial Optimization (Proc. IPCO XV). LNCS, vol. 6655, pp. 287–300. Springer, Berlin (2011)
11. Kaibel, V., Pashkovich, K., Theis, D.O.: Symmetry matters for the sizes of extended formulations. In: Eisenbrand, F., Shepherd, B. (eds.) Integer Programming and Combinatorial Optimization (Proc. IPCO XIV). LNCS, vol. 6080, pp. 135–148. Springer, Berlin (2010)
12. Köppe, M., Louveaux, Q., Weismantel, R.: Intermediate integer programming representations using value disjunctions. Discrete Optim. **5**(2), 293–313 (2008)
13. Margot, F.: Composition de polytopes combinatoires: une approche par projection. Ph.D. thesis, École polytechnique Fédérale de Lausanne (1994)
14. Martin, R.K., Rardin, R.L., Campbell, B.A.: Polyhedral characterization of discrete dynamic programming. Oper. Res. **38**(1), 127–138 (1990). doi:10.1287/opre.38.1.127
15. Nguyen, V.H., Nguyen, T.H., Maurras, J.F.: On the convex hull of Huffman trees. Electron. Notes Discrete Math. **36**, 1009–1016 (2010)
16. Queyranne, M.: Structure of a simple scheduling polyhedron. Math. Program., Ser. A **58**(2), 263–285 (1993). doi:10.1007/BF01581271
17. Wolsey, L.A.: Personal communication
18. Yannakakis, M.: Expressing combinatorial optimization problems by linear programs. J. Comput. Syst. Sci. **43**(3), 441–466 (1991)

Mirror-Descent Methods in Mixed-Integer Convex Optimization

Michel Baes, Timm Oertel, Christian Wagner, and Robert Weismantel

Abstract In this paper, we address the problem of minimizing a convex function f over a convex set, with the extra constraint that some variables must be integer. This problem, even when f is a piecewise linear function, is NP-hard. We study an algorithmic approach to this problem, postponing its hardness to the realization of an oracle. If this oracle can be realized in polynomial time, then the problem can be solved in polynomial time as well. For problems with two integer variables, we show with a novel geometric construction how to implement the oracle efficiently, that is, in $\mathcal{O}(\ln(B))$ approximate minimizations of f over the continuous variables, where B is a known bound on the absolute value of the integer variables. Our algorithm can be adapted to find the second best point of a purely integer convex optimization problem in two dimensions, and more generally its k-th best point. This observation allows us to formulate a finite-time algorithm for mixed-integer convex optimization.

1 Introduction

One of the highlights in the list of publications of Martin Grötschel is his joint book with László Lovász and Alexander Schrijver on Geometric Algorithms and Combinatorial Optimization [8]. This book develops a beautiful and general theory of optimization over (integer) points in convex sets. The generality comes from the fact that the convex sets under consideration are presented by oracles (membership,

M. Baes · T. Oertel · C. Wagner · R. Weismantel (✉)
Department of Mathematics, Institut für Operations Research, ETH Zürich, Rämistrasse 101, 8092 Zürich, Switzerland
e-mail: robert.weismantel@ifor.math.ethz.ch

M. Baes
e-mail: michel.baes@ifor.math.ethz.ch

T. Oertel
e-mail: timm.oertel@ifor.math.ethz.ch

C. Wagner
e-mail: christian.wagner@ifor.math.ethz.ch

separation in different variations, optimization). The algorithms and their efficiency typically depend on the oracle presentation of the underlying convex set. This is precisely the theme of this paper as well: we present an algorithmic framework for solving mixed-integer convex optimization problems that is based on an oracle. Whenever the oracle can be realized efficiently, then the overall running time of the optimization algorithm is efficient as well.

One of the results from the book [8] that is perhaps closest to our results is the following. By $B(p, r)$ we denote a ball of radius r with center p.

Theorem 1 (Theorem 6.7.10 in [8]) *Let n be a fixed integer and $K \subseteq \mathbb{R}^n$ be any convex set given by a weak separation oracle and for which there exist $r, R > 0$ and $p \in K$ with $B(p, r) \subseteq K \subseteq B(0, R)$. There exists an oracle-polynomial algorithm that, for every fixed $\varepsilon > 0$, either finds an integral point in $K + B(0, \varepsilon)$ or concludes that $K \cap \mathbb{Z}^n = \emptyset$.*

The main distinction between results presented here and results from [8] of such flavor as Theorem 1 is the way in which the statements are proven. Proofs of similar results in [8] basically use a combination of the ellipsoid algorithm [10] and a Lenstra-type algorithm [11]. Our proof techniques rather rely on methods from convex optimization.

Let us now make precise our assumptions. We study a general mixed-integer convex optimization problem of the kind

$$\min\left\{f(\hat{x}, y) : (\hat{x}, y) \in S \cap \left(\mathbb{Z}^n \times \mathbb{R}^d\right)\right\}, \tag{1}$$

where the function $f : \mathbb{R}^{n+d} \to \mathbb{R}_+ \cup \{+\infty\}$ is a nonnegative proper convex function, i.e., there is a point $z \in \mathbb{R}^{n+d}$ with $f(z) < +\infty$. Moreover, $S \subseteq \mathbb{R}^{n+d}$ is a convex set that is defined by a finite number of convex functional constraints, i.e., $S := \{(x, y) \in \mathbb{R}^{n+d} : g_i(x, y) \leq 0 \text{ for } 1 \leq i \leq m\}$. We denote by $\langle \cdot, \cdot \rangle$ a scalar product. The functions $g_i : \mathbb{R}^{n+d} \to \mathbb{R}$ are differentiable convex functions and encoded by a so-called *first-order oracle*. Given any point $(x_0, y_0) \in \mathbb{R}^{n+d}$, this oracle returns, for every $i \in \{1, \ldots, m\}$, the function value $g_i(x_0, y_0)$ together with a subgradient $g_i'(x_0, y_0)$, that is, a vector satisfying:

$$g_i(x, y) - g_i(x_0, y_0) \geq \left\langle g_i'(x_0, y_0), (x - x_0, y - y_0)\right\rangle$$

for all $(x, y) \in \mathbb{R}^{n+d}$.

In this general setting, very few algorithmic frameworks exist. The most commonly used one is “outer approximation”, originally proposed in [4] and later on refined in [2, 6, 17]. This scheme is known to be finitely converging, yet there is no analysis regarding the number of iterations it takes to solve problem (1) up to a certain given accuracy.

In this paper we present oracle-polynomial algorithmic schemes that are (i) amenable to an analysis and (ii) finite for any mixed-integer convex optimization problem. Our schemes also give rise to the fastest algorithm so far for solving mixed-integer convex optimization problems in variable dimension with at most two integer variables.

2 An Algorithm Based on an "Improvement Oracle"

We study in this paper an algorithmic approach to solve (1), postponing its hardness to the realization of an improvement oracle defined below. If this oracle can be realized in polynomial time, then the problem can be solved in polynomial time as well. An oracle of this type has already been used in a number of algorithms in other contexts, such as in [1] for semidefinite problems.

Definition 1 (Improvement Oracle) Let $\alpha, \delta \geq 0$. For every $z \in S$, the oracle

a. returns $\hat{z} \in S \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ such that $f(\hat{z}) \leq (1+\alpha)f(z) + \delta$, and/or
b. asserts correctly that there is no point $\hat{z} \in S \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ for which $f(\hat{z}) \leq f(z)$.

We denote the query to this oracle at z by $\mathrm{O}_{\alpha,\delta}(z)$.

As stressed in the above definition, the oracle might content itself with a feasible point $\hat{z}$ satisfying the inequality in **a** without addressing the problem in **b**. However, we do not exclude the possibility of having an oracle that can occasionally report both facts. In that case, the point $\hat{z}$ that it outputs for the input point $z \in S$ must satisfy:

$$f(\hat{z}) - \hat{f}^* \leq \alpha f(z) + \delta + \big(f(z) - \hat{f}^*\big) \leq \alpha f(z) + \delta \leq \alpha \hat{f}^* + \delta,$$

where $\hat{f}^*$ is the optimal objective value of (1). Thus $f(\hat{z}) \leq (1+\alpha)f(z) + \delta$, and it is not possible to hope for a better point of S from the oracle. We can therefore interrupt the computations and output $\hat{z}$ as the final result of our method.

In the case where $\hat{f}^* > 0$ and $\delta = 0$, the improvement oracle might be realized by a relaxation of the problem of finding a suitable $\hat{z}$: in numerous cases, these relaxations come with a guaranteed value of α. In general, the realization of this oracle might need to solve a problem as difficult as the original mixed-integer convex instance, especially when $\alpha = \delta = 0$. Nevertheless, we will point out several situations where this oracle can actually be realized quite efficiently, even with $\alpha = 0$.

The domain of f, denoted by $\mathrm{dom}\, f$, is the set of all the points $z \in \mathbb{R}^{n+d}$ with $f(z) < +\infty$. For all $z \in \mathrm{dom}\, f$, we denote by $f'(z)$ an element of the subdifferential $\partial f(z)$ of f. We represent by $\hat{z}^* = (\hat{x}^*, y^*)$ a minimizer of (1), and set $\hat{f}^* := f(\hat{z}^*)$; more generally, we use a hat $(\hat{\cdot})$ to designate vectors that have their n first components integral by definition or by construction.

Let us describe an elementary method for solving *Lipschitz continuous* convex problems on S approximately. Lipschitz continuity of f on S, an assumption we make from now on, entails that, given a norm $\|\cdot\|$ on $\mathbb{R}^{n+d}$, there exists a constant $L > 0$ for which:

$$\big|f(z_1) - f(z_2)\big| \leq L\|z_1 - z_2\|$$

for every $z_1, z_2 \in S$. Equivalently, if $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$, we have $\|f'(z)\|_* \leq L$ for every $f'(z) \in \partial f(z)$ and every $z \in \mathrm{dom}\, f$.

Our first algorithm is a variant of the well-known Mirror-Descent Method (see Chap. 3 of [13]). It requires a termination procedure, which used alone constitutes our second algorithm as a minimization algorithm on its own. However, the second

Algorithm 1: Mirror-Descent Method

```
Data: $z_0 \in S$.
Set $\hat{z}_0 := z_0$, $w_0 := z_0$, $s_0 := 0$, and $\hat{f}_0 := f(\hat{z}_0)$.
Select sequences $\{h_k\}_{k\geq 0}$, $\{\alpha_k\}_{k\geq 0}$, $\{\delta_k\}_{k\geq 0}$.
for $k = 0, \ldots, N$ do
    Compute $f'(z_k) \in \partial f(z_k)$ and $\rho'(w_k) \in \partial\rho(w_k)$.
    Set $s_{k+1} := s_k - h_k f'(z_k) - h_k \|f'(z_k)\|_* \rho'(w_k)$.
    Set $w_{k+1} := \arg\max\{\langle s_{k+1}, z\rangle - V(z) : z \in \mathbb{R}^{n+d}\}$.
    Set $z_{k+1} := \arg\min\{\|w_{k+1} - z\| : z \in S\}$.
    Compute $f(z_{k+1})$.
    if $f(z_{k+1}) \geq \hat{f}_k$ then $\hat{z}_{k+1} := \hat{z}_k$, $\hat{f}_{k+1} := \hat{f}_k$.
    else
        Run $\mathrm{O}_{\alpha_{k+1},\delta_{k+1}}(z_{k+1})$.
        if the oracle reports a and b then
            Terminate the algorithm and return the oracle output from a.
        else if the oracle reports a but not b then
            Set $\hat{z}_{k+1}$ as the oracle output and $\hat{f}_{k+1} := \min\{f(\hat{z}_{k+1}), \hat{f}_k\}$.
        else
            Run the termination procedure with $z_0 := z_{k+1}$, $\hat{z}_0 := \hat{z}_{k+1}$,
            return its output, and terminate the algorithm.
        end
    end
end
```

algorithm requires as input an information that is a priori not obvious to get: a point $z \in S$ for which $f(z)$ is a (strictly) positive lower bound of $\hat{f}^*$.

Let $V : \mathbb{R}^{n+d} \to \mathbb{R}_+$ be a differentiable σ-strongly convex function with respect to the norm $\|\cdot\|$, i.e., there exists a $\sigma > 0$ for which, for every $z_1, z_2 \in \mathbb{R}^{n+d}$, we have:

$$V(z_2) - V(z_1) - \langle V'(z_1), z_2 - z_1\rangle \geq \frac{\sigma}{2}\|z_2 - z_1\|^2.$$

We also use the conjugate V_* of V defined by $V_*(s) := \sup\{\langle s, z\rangle - V(z) : z \in \mathbb{R}^{n+d}\}$ for every $s \in \mathbb{R}^{n+d}$. We fix $z_0 \in S$ as the starting point of our algorithm and denote by M an upper bound of $V(\hat{z}^*)$. We assume that the solution of the problem $\sup\{\langle s, z\rangle - V(z) : z \in \mathbb{R}^{n+d}\}$ exists and can be computed easily, as well as the function $\rho(w) := \min\{\|w - z\| : z \in S\}$ for every $w \in \mathbb{R}^{n+d}$, its subgradient, and the minimizer $\pi(w)$. In an alternative version of the algorithm we are about to describe, we can merely assume that the problem $\max\{\langle s, z\rangle - V(z) : z \in S\}$ can be solved efficiently.

A possible building block for constructing an algorithm to solve (1) is the continuous optimum of the problem, that is, the minimizer of (1) without the integrality constraints. Algorithm 1 is essentially a standard procedure meant to compute an

Algorithm 2: Termination procedure

```
Data: $z_0 \in S$ with $f(z_0) \le \hat{f}^*$, $\hat{z}_0 \in S \cap (\mathbb{Z}^n \times \mathbb{R}^d)$.
Set $l_0 := f(z_0)$, $u_0 := f(\hat{z}_0)$.
Choose $\alpha, \delta \ge 0$. Choose a subproblem accuracy $\varepsilon' > 0$.
for $k \ge 0$ do
    Compute using a bisection method a point $z_{k+1} = \lambda z_k + (1-\lambda)\hat{z}_k$
    for $0 \le \lambda \le 1$, for which $f(z_{k+1}) - (l_k(\alpha+1) + u_k)/(\alpha+2) \in [-\varepsilon', \varepsilon']$.
    Run $O_{\alpha,\delta}(z_{k+1})$.
    if the oracle reports a and b then
        Terminate the algorithm and return the oracle output from a.
    else if the oracle reports a but not b then
        Set $\hat{z}_{k+1}$ as the oracle output, $l_{k+1} := l_k$, $u_{k+1} := \min\{f(\hat{z}_{k+1}), u_k\}$.
    else
        Set $\hat{z}_{k+1} := \hat{z}_k$, $l_{k+1} := f(z_{k+1})$, $u_{k+1} := u_k$.
    end
end
```

approximation of this continuous minimizer, lined with our oracle that constructs simultaneously a sequence of mixed-integer feasible points following the decrease of f. Except in the rare case when we produce a provably suitable solution to our problem, this algorithm provides a point $z \in S$ such that $f(z)$ is a lower bound of $\hat{f}^*$. Would this lower bound be readily available, we can jump immediately to the termination procedure (see Algorithm 2).

The following proposition is an extension of the standard proof of convergence of Mirror-Descent Methods. We include it here for the sake of completeness.

Proposition 1 *Suppose that the oracle reports* **a** *for $k = 0, \ldots, N$ in Algorithm 1, that is, it delivers an output $\hat{z}_k$ for every iteration $k = 0, \ldots, N$. Then:*

$$\frac{1}{\sum_{k=0}^N h_k} \sum_{k=0}^N \frac{h_k f(\hat{z}_k)}{1+\alpha_k} - f\left(\hat{z}^*\right)$$
$$\le \frac{M}{\sum_{k=0}^N h_k} + \frac{2L^2}{\sigma} \cdot \frac{\sum_{k=0}^N h_k^2}{\sum_{k=0}^N h_k} + \frac{1}{\sum_{k=0}^N h_k} \sum_{k=0}^N \frac{h_k \delta_k}{1+\alpha_k}.$$

Proof Since V is σ-strongly convex with respect to the norm $\|\cdot\|$, its conjugate V_* is differentiable and has a Lipschitz continuous gradient of constant $1/\sigma$ for the norm $\|\cdot\|_*$, i.e., $V_*(y) - V_*(x) \le \langle V_*'(x), y - x\rangle + \frac{1}{2\sigma}\|y-x\|_*^2$ (see Chap. X in [9]). Also $w_k = V_*'(s_k)$, in view of Theorem 23.5 in [16]. Finally, for every $z \in S$, we can write $\rho(w_k) + \langle \rho'(w_k), z - w_k\rangle \le \rho(z) = 0$. Thus:

$$\left\langle \rho'(w_k), w_k - \hat{z}^* \right\rangle \ge \rho(w_k) = \left\| \pi(w_k) - w_k \right\| = \|z_k - w_k\|. \qquad (2)$$

Also, $\|\rho'(w_k)\|_* \leq 1$, because for every $z \in \mathbb{R}^{n+d}$:

$$\begin{aligned}\big\langle \rho'(w_k), z - w_k \big\rangle &\leq \rho(z) - \rho(w_k) = \big\|z - \pi(z)\big\| - \big\|w_k - \pi(w_k)\big\| \\ &\leq \big\|z - \pi(w_k)\big\| - \big\|w_k - \pi(w_k)\big\| \leq \|z - w_k\|. \end{aligned} \tag{3}$$

By setting $\phi_k := V_*(s_k) - \langle s_k, \hat{z}^* \rangle$, we can write successively for all $k \geq 0$:

$$\begin{aligned}\phi_{k+1} &= V_*(s_{k+1}) - \big\langle s_{k+1}, \hat{z}^* \big\rangle \\ &\leq V_*(s_k) + \big\langle V_*'(s_k), s_{k+1} - s_k \big\rangle + \frac{1}{2\sigma}\|s_{k+1} - s_k\|_*^2 - \big\langle s_{k+1}, \hat{z}^* \big\rangle. \\ &= \big(V_*(s_k) - \big\langle s_k, \hat{z}^* \big\rangle\big) + \big\langle V_*'(s_k) - \hat{z}^*, s_{k+1} - s_k \big\rangle + \frac{1}{2\sigma}\|s_{k+1} - s_k\|_*^2 \\ &= \phi_k - h_k\big\langle w_k - z_k, f'(z_k)\big\rangle + h_k\big\langle \hat{z}^* - z_k, f'(z_k)\big\rangle \\ &\quad - h_k\big\|f'(z_k)\big\|_*\big\langle w_k - \hat{z}^*, \rho'(w_k)\big\rangle + \frac{h_k^2\|f'(z_k)\|_*^2}{2\sigma}\left\|\frac{f'(z_k)}{\|f'(z_k)\|_*} + \rho'(w_k)\right\|_*^2, \end{aligned}$$

where the inequality follows from the Lipschitz continuity of the gradient of V_*, and the last equality from the identities $V_*'(s_k) = w_k$, $s_{k+1} - s_k = -h_k f'(z_k) - h_k\|f'(z_k)\|_*\rho'(w_k)$, and $V_*(s_k) - \langle s_k, \hat{z}^* \rangle = \phi_k$. By the definition of the dual norm, it holds $-h_k\langle w_k - z_k, f'(z_k)\rangle \leq h_k\|f'(z_k)\|_*\|w_k - z_k\|$. Moreover, convexity of f implies $h_k\langle \hat{z}^* - z_k, f'(z_k)\rangle \leq f(\hat{z}^*) - f(z_k)$. Using this in the above expression we get:

$$\begin{aligned}\phi_{k+1} &\leq \phi_k + h_k\big\|f'(z_k)\big\|_*\big(\|w_k - z_k\| - \big\langle w_k - \hat{z}^*, \rho'(w_k)\big\rangle\big) \\ &\quad + h_k\big(f\big(\hat{z}^*\big) - f(z_k)\big) + \frac{h_k^2\|f'(z_k)\|_*^2}{2\sigma}\left(\left\|\frac{f'(z_k)}{\|f'(z_k)\|_*}\right\|_* + \big\|\rho'(w_k)\big\|_*\right)^2 \\ &\leq \phi_k + h_k\big(f\big(\hat{z}^*\big) - f(z_k)\big) + \frac{2h_k^2\|f'(z_k)\|_*^2}{\sigma} \\ &\leq \phi_k + h_k\left(f\big(\hat{z}^*\big) - \frac{f(\hat{z}_k) - \delta_k}{1 + \alpha_k}\right) + \frac{2h_k^2\|f'(z_k)\|_*^2}{\sigma}, \end{aligned}$$

where the second inequality follows from (2) and $\|\rho'(w_k)\|_* \leq 1$, and the third inequality from the fact that the oracle reports **a**. Summing up the above inequalities from $k := 0$ to $k := N$ and rearranging, it follows:

$$\frac{1}{\sum_{k=0}^N h_k}\sum_{k=0}^N \frac{h_k(f(\hat{z}_k) - \delta_k)}{1 + \alpha_k} - f\big(\hat{z}^*\big) \leq \frac{\phi_0 - \phi_{N+1}}{\sum_{k=0}^N h_k} + \frac{2\sum_{k=0}^N h_k^2\|f'(z_k)\|_*^2}{\sigma\sum_{k=0}^N h_k}.$$

Note that $\|f'(z_k)\|_* \leq L$, $\phi_0 = \sup\{-V(z) : z \in \mathbb{R}^{n+d}\} \leq 0$, and $\phi_{N+1} \geq -V(\hat{z}^*) \geq -M$, yielding the desired result. □

In the special case when $\alpha_k = \alpha$ and $\delta_k = \delta$ for every $k \geq 0$, we can significantly simplify the above results. According to the previous proposition, we know that:

$$\begin{aligned}\left(\sum_{k=0}^{N} h_k\right)\left(\frac{\hat{f}_N - \delta}{1+\alpha} - \hat{f}^*\right) &= \left(\sum_{k=0}^{N} h_k\right)\left(\frac{\min_{1\leq i\leq N} f(\hat{z}_i) - \delta}{1+\alpha} - \hat{f}^*\right) \\ &\leq \sum_{k=0}^{N} \frac{h_k(f(\hat{z}_k) - \delta)}{1+\alpha} - \left(\sum_{k=0}^{N} h_k\right)\hat{f}^* \\ &\leq M + \frac{2L^2}{\sigma}\sum_{k=0}^{N} h_k^2. \end{aligned} \tag{4}$$

We can divide both sides of the above inequality by $\sum_{k=0}^{N} h_k$, then determine the step-sizes $\{h_k : 0 \leq k \leq N\}$ for which the right-hand side is minimized. However, with this strategy, h_0 would depend on N, which is a priori unknown at the first iteration. Instead, as in [14], we use a step-size of the form $h_k = c/\sqrt{k+1}$ for an appropriate constant $c > 0$, independent of N. Note that:

$$\sum_{k=0}^{N} \frac{1}{k+1} = \sum_{k=1}^{N+1} \frac{1}{k} \leq \int_1^{N+1} \frac{dt}{t} + 1 = \ln(N+1) + 1.$$

If we choose $c := \sqrt{\frac{\sigma M}{2L^2}}$, the right-hand side of (4) can be upper-bounded by $M\ln(N+1) + 2M$. Finally, since

$$\frac{1}{c}\sum_{k=0}^{N} h_k = \sum_{k=0}^{N} \frac{1}{\sqrt{k+1}} = \sum_{k=1}^{N+1} \frac{1}{\sqrt{k}} \geq \int_1^{N+2} \frac{dt}{\sqrt{t}} = 2\sqrt{N+2} - 2,$$

we can thereby conclude that:

$$\frac{\hat{f}_N - (1+\alpha)\hat{f}^* - \delta}{1+\alpha} \leq L\sqrt{\frac{M}{2\sigma}} \cdot \frac{\ln(N+1) + 2}{\sqrt{N+2} - 1}. \tag{5}$$

As the right-hand side converges to 0 when N goes to infinity, Algorithm 1 converges to an acceptable approximate solution or calls the termination procedure.

Let us now turn our attention to the termination procedure. We assume here that the oracle achieves a constant quality, that is, that there exists $\alpha, \delta \geq 0$ for which $\alpha_k = \alpha$ and $\delta_k = \delta$ for every $k \geq 0$.

Proposition 2 *Assume that* $f(\hat{z}_0) \geq f(z_0) > 0$, *and that there is no point* $\hat{z} \in S \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ *for which* $f(z_0) > f(\hat{z})$.

(a) *The termination procedure cannot guarantee an accuracy better than*:

$$f(\hat{z}) \leq \hat{f}^* + (2+\alpha)\big(\alpha\hat{f}^* + (1+\alpha)\varepsilon' + \delta\big). \tag{6}$$

(b) *For every* $\varepsilon > 0$, *the termination procedure finds a point* $\hat{z} \in S \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ *satisfying*:

$$f(\hat{z}) - \hat{f}^* \leq \varepsilon \hat{f}^* + (2+\alpha)\big(\alpha \hat{f}^* + (1+\alpha)\varepsilon' + \delta\big)$$

within

$$\max\left\{\left\lceil \ln\left(\frac{f(\hat{z}_0) - f(z_0)}{f(z_0)\varepsilon}\right) \Big/ \ln\left(\frac{2+\alpha}{1+\alpha}\right)\right\rceil, 0\right\}$$

iterations.

Proof Part (a). At every iteration k, there is by construction no $\hat{z} \in S \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ for which $l_k > f(\hat{z})$. Also, $f(\hat{z}_k) \geq u_k \geq \hat{f}^*$. For convenience, we denote $(1+\alpha)/(2+\alpha)$ by λ in this proof, and we set $\Delta_k := u_k - l_k$ for every $k \geq 0$.

Suppose first that the oracle finds a new point $\hat{z}_{k+1} \in S \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ at iteration k. Then:

$$f(\hat{z}_{k+1}) \leq (1+\alpha) f(z_{k+1}) + \delta \leq (1+\alpha)\big(\lambda l_k + (1-\lambda)u_k + \varepsilon'\big) + \delta,$$

where the first inequality is due to the definition of our oracle and the second one comes from the accuracy by which our bisection procedure computes z_{k+1}. Observe that the oracle might return a point $\hat{z}_k$ such that $f(\hat{z}_k)$ is smaller than the above right-hand side. In this case, no progress is done. As $u_k \leq f(\hat{z}_k)$, this implies:

$$(\lambda + \lambda\alpha) l_k + (1+\alpha)\varepsilon' + \delta \geq (\lambda + \lambda\alpha - \alpha) f(\hat{z}_k). \tag{7}$$

Using that $\hat{f}^* \geq l_k$ we get an upper bound of the left-hand side. Rearranging the terms and replacing λ by its value, we get:

$$\hat{f}^* + (2+\alpha)\big(\alpha \hat{f}^* + (1+\alpha)\varepsilon' + \delta\big) \geq f(\hat{z}_k).$$

Since all the inequalities in the above derivation can be tight, a better accuracy cannot be guaranteed with our strategy. Thus, we can output $\hat{z}_k$.

Part (b). Note that we can assume $f(\hat{z}_0) - f(z_0) > f(z_0)\varepsilon$, for otherwise the point $\hat{z}_0$ already satisfies our stopping criterion.

In order to assess the progress of the algorithm, we can assume that the stopping criterion (7) is not satisfied. As $l_{k+1} = l_k$ in our case where the oracle gives an output, we get:

$$\begin{aligned}
\Delta_{k+1} &= u_{k+1} - l_k \\
&\leq f(\hat{z}_{k+1}) - l_k \\
&\leq (1+\alpha)\big(\lambda l_k + (1-\lambda)u_k + \varepsilon'\big) + \delta - l_k \\
&= \frac{\alpha^2 + \alpha - 1}{2+\alpha} l_k + \frac{1+\alpha}{2+\alpha} u_k + (1+\alpha)\varepsilon' + \delta
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1+\alpha}{2+\alpha}(u_k - l_k) + \alpha l_k + (1+\alpha)\varepsilon' + \delta \\
&\leq \frac{1+\alpha}{2+\alpha}\Delta_k + \alpha \hat{f}^* + (1+\alpha)\varepsilon' + \delta.
\end{aligned}
$$

Suppose now that the oracle informs us that there is no mixed-integral point with a value smaller than $f(z_{k+1}) \geq \lambda l_k + (1-\lambda)u_k - \varepsilon'$. Then $\hat{z}_{k+1} = \hat{z}_k$ and $u_{k+1} = u_k$. We have:

$$
\begin{aligned}
\Delta_{k+1} = u_{k+1} - l_{k+1} &= f(\hat{z}_k) - f(z_{k+1}) \\
&\leq u_k - \big(\lambda l_k + (1-\lambda)u_k - \varepsilon'\big) = \lambda \Delta_k + \varepsilon' \\
&\leq \frac{1+\alpha}{2+\alpha}\Delta_k + \alpha \hat{f}^* + (1+\alpha)\varepsilon' + \delta.
\end{aligned}
$$

The above inequality is valid for every k that does not comply with the stopping criterion, whatever the oracle detects. Therefore, we get:

$$
\Delta_N \leq \left(\frac{1+\alpha}{2+\alpha}\right)^N \Delta_0 + (2+\alpha)\big(\alpha \hat{f}^* + (1+\alpha)\varepsilon' + \delta\big),
$$

and the proposition is proved because $f(\hat{z}_N) - \hat{f}^* \leq \Delta_N$. □

In the remainder of this paper, we elaborate on possible realizations of our hard oracle. We proceed as follows. In Sect. 3, we focus on the special case when $n = 2$ and $d = 0$. We present a geometric construction that enables us to implement the improvement oracle in polynomial time. With the help of this oracle we then solve the problem (1) with $n = 2$ and $d = 0$ and obtain a "best point", i.e., an optimal point. An adaptation of this construction can also be used to determine a second and, more generally, a "k-th best point". These results will be extended in Sect. 4 to the mixed-integer case with two integer variables and d continuous variables. The latter extensions are then used as a subroutine to solve the general problem (1) with arbitrary n and d in finite time.

3 Two-Dimensional Integer Convex Optimization

If $n = 1$ and $d = 0$, an improvement oracle can be trivially realized for $\alpha = \delta = 0$. Queried on a point $z \in \mathbb{R}$ the oracle returns $\hat{z} := \arg\min\{f(\lfloor z \rfloor), f(\lceil z \rceil)\}$ if one of these numbers is smaller or equal to $f(z)$, or returns $\mathbf{b}$ otherwise. The first non-trivial case arises when $n = 2$ and $d = 0$. This is the topic of this section.

3.1 Minimizing a Convex Function in Two Integer Variables

In this section we discuss a new geometric construction that enables us to implement efficiently the oracle $O_{\alpha,\delta}$ with $\alpha = \delta = 0$, provided that the feasible set is contained in a known finite box $[-B, B]^2$.

Theorem 2 *Let $f : \mathbb{R}^2 \to \mathbb{R}$ and $g_i : \mathbb{R}^2 \to \mathbb{R}$ with $i = 1, \ldots, m$ be convex functions. Let $B \in \mathbb{N}$ and let $x \in [-B, B]^2$ such that $g_i(x) \le 0$ for all $i = 1, \ldots, m$. Then, in a number of evaluations of f and $g_1, \ldots, g_m$ that is polynomial in* $\ln(B)$, *one can either*

(a) *find an $\hat{x} \in [-B, B]^2 \cap \mathbb{Z}^2$ with $f(\hat{x}) \le f(x)$ and $g_i(\hat{x}) \le 0$ for all $i = 1, \ldots, m$ or*
(b) *show that there is no such point.*

Note that we do not allow for the function f to take infinite values, in order to ensure that we can minimize f over the integers of any segment of $[-B, B]^2$ in $\mathcal{O}(\ln(B))$ evaluations of f using a bisection method. Indeed, if a convex function takes infinite values, it can cost up to $\mathcal{O}(B)$ evaluations of f to minimize it on a segment containing $\mathcal{O}(B)$ integer points, as there could be only one of those points on its domain.

The algorithm that achieves the performance claimed in Theorem 2 is described in the proof of the theorem. That proof requires two lemmata. We use the following notation. Let $Q \subset \mathbb{R}^2$. We denote by $\mathrm{vol}(Q)$ the volume of Q, i.e., its Lebesgue measure. By $\mathrm{aff}\{Q\}$ we denote the smallest affine space containing Q and by $\mathrm{conv}\{Q\}$ the convex hull of Q. The dimension $\dim(Q)$ of Q is the dimension of $\mathrm{aff}\{Q\}$. The scalar product we use in this section is exclusively the standard dot product.

Lemma 1 *Let $K \subset \mathbb{R}^2$ be a polytope with $\mathrm{vol}(K) < \frac{1}{2}$. Then* $\dim(\mathrm{conv}(K \cap \mathbb{Z}^2)) \le 1$.

Proof For the purpose of deriving a contradiction, assume that there exist three affinely independent points $\hat{x}, \hat{y}, \hat{z} \in K \cap \mathbb{Z}^2$. Then $\mathrm{vol}(K) \ge \mathrm{vol}(\mathrm{conv}(\{\hat{x}, \hat{y}, \hat{z}\})) = \frac{1}{2}|\det(\hat{x} - \hat{z}, \hat{y} - \hat{z})| \ge \frac{1}{2}$. □

Lemma 2 *Let $u, v, w \in \mathbb{R}^2$ be affinely independent. If*

$$\big(\mathrm{conv}\{u, u+v, u+v+w\} \setminus \big(\mathrm{conv}\{u+v, u+v+w\} \cup \{u\}\big)\big) \cap \mathbb{Z}^2 = \emptyset,$$

then the lattice points $\mathrm{conv}\{u, u+v, u+v-w\} \cap \mathbb{Z}^2$ *lie on at most three lines.*

Proof We partition $\mathrm{conv}\{u, u+v, u+v-w\}$ into three regions. Then we show that in each region the integer points must lie on a single line using a lattice covering argument.

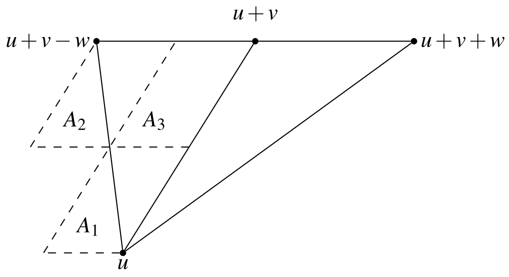


Fig. 1 Partitioning the *triangle* in regions

We define the parallelogram $P := \mathrm{conv}\{0, \frac{1}{2}v, \frac{1}{2}w, \frac{1}{2}v + \frac{1}{2}w\}$. Further, we set

$$A_1 := u - \frac{1}{2}w + P, \qquad A_2 := u + \frac{1}{2}v - w + P, \quad \text{and} \quad A_3 := u + \frac{1}{2}v - \frac{1}{2}w + P.$$

Note that $\mathrm{conv}\{u, u+v, u+v-w\} \subset A_1 \cup A_2 \cup A_3$ (see Fig. 1). Our assumption implies that the set $u + \frac{1}{2}v + P$ does not contain any integer point except possibly on the segment $u + v + \mathrm{conv}\{0, w\}$. Therefore, for a sufficiently small $\varepsilon > 0$, the set $(u + \frac{1}{2}v - \varepsilon(v + w) + P) \cap \mathbb{Z}^2$ is empty.

Assume now that one of the three regions, say A_1, contains three affinely independent integer points $\hat{x}, \hat{y}, \hat{z}$. We show below that $A_1 + \mathbb{Z}^2 = \mathbb{R}^2$, i.e., that P defines a lattice covering, or equivalently that the set $t + P$ contains at least one integer point for every $t \in \mathbb{R}^2$. This fact will contradict that $(u + \frac{1}{2}v - \varepsilon(v + w) + P) \cap \mathbb{Z}^2 = \emptyset$ and thereby prove the lemma.

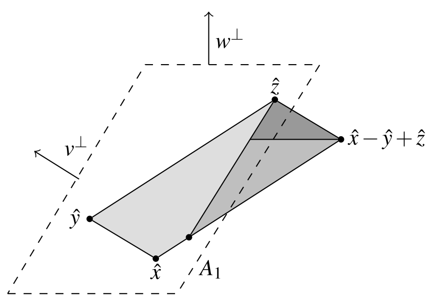
Clearly, the parallelogram $Q := \mathrm{conv}\{\hat{x}, \hat{y}, \hat{z}, \hat{x} - \hat{y} + \hat{z}\}$ defines a lattice covering, as it is full-dimensional and its vertices are integral. We transform Q into a set $Q' \subseteq A_1$ for which $a \in Q'$ iff there exists $b \in Q$ such that $a - b \in \mathbb{Z}^2$. Specifically, we define a mapping T such that $Q' = T(Q) \subset A_1$ and $T(Q) + \mathbb{Z}^2 = \mathbb{R}^2$. Let $v^\perp := (-v_2, v_1)^\top$ and $w^\perp := (-w_2, w_1)^\top$, i.e., vectors orthogonal to v and w. Without loss of generality (up to a permutation of the names $\hat{x}, \hat{y}, \hat{z}$), we can assume that $\langle \hat{x}, w^\perp \rangle \leq \langle \hat{y}, w^\perp \rangle \leq \langle \hat{z}, w^\perp \rangle$. If $\hat{x} - \hat{y} + \hat{z} \in A_1$ there is nothing to show, so we suppose that $\hat{x} - \hat{y} + \hat{z} \notin A_1$.

Note that $\langle \hat{x}, w^\perp \rangle \leq \langle \hat{x} - \hat{y} + \hat{z}, w^\perp \rangle \leq \langle \hat{z}, w^\perp \rangle$. Assume first that $\langle \hat{x} - \hat{y} + \hat{z}, v^\perp \rangle < \langle \hat{z}, v^\perp \rangle \leq \langle \hat{x}, v^\perp \rangle, \langle \hat{y}, v^\perp \rangle$—the strict inequality resulting from the fact $\hat{x} - \hat{y} + \hat{z} \notin A_1$. We define the mapping $T : Q \to A_1$ as follows,

$$T(l) = \begin{cases} l + \hat{y} - \hat{z}, & \text{if } \langle l, v^\perp \rangle < \langle \hat{z}, v^\perp \rangle \text{ and } \langle l, w^\perp \rangle > \langle \hat{x} - \hat{y} + \hat{z}, w^\perp \rangle, \\ l - \hat{x} + \hat{y}, & \text{if } \langle l, v^\perp \rangle < \langle \hat{z}, v^\perp \rangle \text{ and } \langle l, w^\perp \rangle \leq \langle \hat{x} - \hat{y} + \hat{z}, w^\perp \rangle, \\ l, & \text{otherwise} \end{cases}$$

(see Fig. 2). It is straightforward to show that $T(Q) \subset A_1$ and $T(Q) + \mathbb{Z}^2 = \mathbb{R}^2$. A similar construction can easily be defined for any possible ordering of $\langle \hat{x} - \hat{y} + \hat{z}, v^\perp \rangle$, $\langle \hat{z}, v^\perp \rangle$, $\langle \hat{x}, v^\perp \rangle$, and $\langle \hat{y}, v^\perp \rangle$. □

Remark 1 In each region A_i, the line containing $A_i \cap \mathbb{Z}^2$, if it exists, can be computed by the minimization of an arbitrary linear function $x \mapsto \langle c, x \rangle$ over $A_i \cap \mathbb{Z}^2$,



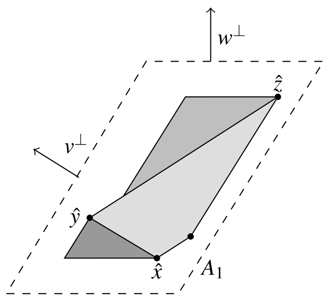


Fig. 2 Mapping T

with $c \neq 0$, and the maximization of the same function with the fast algorithm described in [5]. If these problems are feasible and yield two distinct solutions, the line we are looking for is the one joining these two solutions. If the two solutions coincide, that line is the one orthogonal to c passing through that point.

The algorithm in [5] is applicable to integer linear programs with two variables and m constraints. The data of the problem should be integral. This algorithm runs in $\mathcal{O}(m + \phi)$, where ϕ is the binary encoding length of the data.

Proof of Theorem 2 As described at the beginning of this section, a one-dimensional integer minimization problem can be solved polynomially with respect to the logarithm of the length of the segment that the function is optimized over. In the following we explain how to reduce the implementation of the two-dimensional oracle to the task of solving one-dimensional integer minimization problems. For notational convenience, we define $g(y) := \max_{i=1,\dots,m} g_i(y)$ for $y \in \mathbb{R}^2$ which is again a convex function.

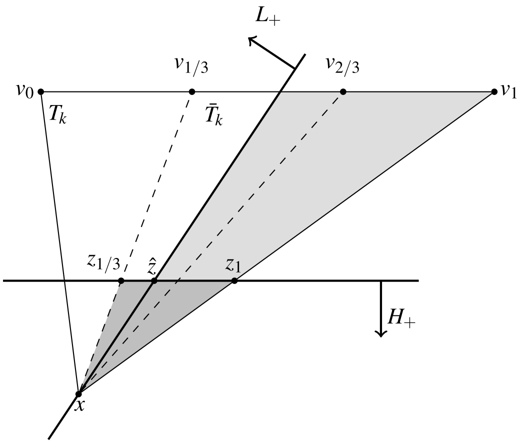
Let $F_1, \dots, F_4$ be the facets of $[-B, B]^2$. Then $[-B, B]^2 = \bigcup_{j=1}^{4} \operatorname{conv}\{x, F_j\}$. The procedure we are about to describe has to be applied to every facet $F_1, \dots, F_4$ successively, until a suitable point $\hat{x}$ is found. Let us only consider one facet F. We define the triangle $T_0 := \operatorname{conv}\{x, F\}$, whose area is smaller than $2B^2$.

To find an improving point within T_0, we construct a sequence $T_0 \supset T_1 \supset T_2 \supset \dots$ of triangles that all have x as vertex, with $\operatorname{vol}(T_{k+1}) \le \frac{2}{3} \operatorname{vol}(T_k)$, and such that $f(\hat{y}) > f(x)$ or $g(\hat{y}) > 0$ for all $\hat{y} \in (T_0 \setminus T_k) \cap \mathbb{Z}^2$. We stop our search if we have found an $\hat{x} \in [-B, B]^2 \cap \mathbb{Z}^2$ such that $f(\hat{x}) \le f(x)$ and $g(\hat{x}) \le 0$, or if the volume of one of the triangles T_k is smaller than $\frac{1}{2}$. The latter happens after at most $k = \lceil \ln(4B^2)/\ln(\frac{3}{2}) \rceil$ steps. Then, Lemma 1 ensures that the integral points of T_k are on a line, and we need at most $\mathcal{O}(\ln(B))$ iterations to solve the resulting one-dimensional problem.

The iterative construction is as follows. Let $T_k = \operatorname{conv}\{x, v_0, v_1\}$ be given. We write $v_\lambda := (1-\lambda)v_0 + \lambda v_1$ for $\lambda \in \mathbb{R}$ and we define the auxiliary triangle $\bar{T}_k := \operatorname{conv}\{x, v_{1/3}, v_{2/3}\}$. Consider the integer linear program

$$\min\left\{\langle h, \hat{y}\rangle : \hat{y} \in \bar{T}_k \cap \mathbb{Z}^2\right\} \tag{8}$$

Fig. 3 Illustration of Case 2



where h is the normal vector to $\operatorname{conv}\{v_0, v_1\}$ such that $\langle h, x\rangle < \langle h, y\rangle$ for every $y \in F$. We distinguish two cases.

Case 1. The integer linear program (8) is infeasible. Then $\bar{T}_k \cap \mathbb{Z}^2 = \emptyset$. It remains to check for an improving point within $(T_k \setminus \bar{T}_k) \cap \mathbb{Z}^2$. By construction, we can apply Lemma 2 twice (with $(u, u+v-w, u+v+w)$ equal to $(x, v_0, v_{2/3})$ and $(x, v_{1/3}, v_1)$, respectively) to determine whether there exists an $\hat{x} \in (T_k \setminus \bar{T}_k) \cap \mathbb{Z}^2$ such that $f(\hat{x}) \le f(x)$ and $g(\hat{x}) \le 0$. This requires to solve at most six one-dimensional subproblems.

Case 2. The integer linear program (8) has an optimal solution $\hat{z}$. If $f(\hat{z}) \le f(x)$ and $g(\hat{z}) \le 0$, we are done. So we assume that $f(\hat{z}) > f(x)$ or $g(\hat{z}) > 0$. Define $H := \{y \in \mathbb{R}^2 | \langle h, y\rangle = \langle h, \hat{z}\rangle\}$, that is, the line containing $\hat{z}$ that is parallel to $\operatorname{conv}\{v_0, v_1\}$, and denote by H_+ the closed half-space with boundary H that contains x. By definition of $\hat{z}$, there is no integer point in $\bar{T}_k \cap \operatorname{int} H_+$. Further, let $L := \operatorname{aff}\{x, \hat{z}\}$.

Due to the convexity of the set $\{y \in \mathbb{R}^2 | f(y) \le f(x), g(y) \le 0\}$ and the fact that $f(\hat{z}) > f(x)$ or $g(\hat{z}) > 0$, there exists a half-space L_+ with boundary L such that the possibly empty segment $\{y \in H | f(y) \le f(x), g(y) \le 0\}$ lies in L_+ (see Fig. 3). By convexity of f and g, the set $((T_k \setminus H_+) \setminus L_+)$ (the lightgray region in Fig. 3) contains no point y for which $f(y) \le f(x)$ and $g(y) \le 0$. It remains to check for an improving point within $((T_k \cap H_+) \setminus L_+) \cap \mathbb{Z}^2$. For that we apply again Lemma 2 on the triangle $\operatorname{conv}\{z_{1/3}, z_1, x\}$ (the darkgray region in Fig. 3), with $z_{1/3} = H \cap \operatorname{aff}\{x, v_{1/3}\}$ and $z_1 = H \cap \operatorname{aff}\{x, v_1\}$. If none of the corresponding subproblems returns a suitable point $\hat{x} \in \mathbb{Z}^2$, we know that $T_k \setminus L_+$ contains no improving integer point. Defining $T_{k+1} := T_k \cap L_+$, we have by construction $f(\hat{y}) > f(x)$ or $g(\hat{y}) > 0$ for all $\hat{y} \in (T_k \setminus T_{k+1}) \cap \mathbb{Z}^2$ and $\operatorname{vol}(T_{k+1}) \le \frac{2}{3} \operatorname{vol}(T_k)$.

It remains to determine the half-space L_+. If $g(\hat{z}) > 0$ we just need to find a point $y \in H$ such that $g(y) < g(\hat{z})$, or if $f(\hat{z}) > f(x)$, it suffices to find a point $y \in H$ such that $f(y) < f(\hat{z})$. Finally, if we cannot find such a point y in either case, convexity implies that there is no suitable point in $T_k \setminus H_+$; another application of Lemma 2 then suffices to determine whether there is a suitable $\hat{x}$ in $T_k \cap H_+ \cap \mathbb{Z}^2$. $\square$

The algorithm presented in the proof of Theorem 2 can be adapted to output a minimizer $\hat{x}^*$ of f over $S \cap [-B, B]^2 \cap \mathbb{Z}^2$, provided that we know in advance that the input point x satisfies $f(x) \leq \hat{f}^*$ (see Algorithm 3). It suffices to store and update the best value of f on integer points found so far. In this case the termination procedure is not necessary.

Corollary 1 *Let $f : \mathbb{R}^2 \to \mathbb{R}$ and $g_i : \mathbb{R}^2 \to \mathbb{R}$ with $i = 1, \ldots, m$ be convex functions. Let $B \in \mathbb{N}$ and let $x \in [-B, B]^2$ such that $g_i(x) \leq 0$ for all $i = 1, \ldots, m$. If $f(x) \leq \hat{f}^*$, then, in a number of evaluations of f and $g_1, \ldots, g_m$ that is polynomial in $\ln(B)$, one can either*

(a) *find an $\hat{x} \in [-B, B]^2 \cap \mathbb{Z}^2$ with $f(\hat{x}) = \hat{f}^*$ and $g_i(\hat{x}) \leq 0$ for all $i = 1, \ldots, m$ or*

(b) *show that there is no such point.*

Note that line 30 in Algorithm 3 requires the application of Lemma 1. Lines 11, 20 and 24 require the application of Lemma 2.

Remark 2 (Complexity) The following subroutines are used in Algorithm 3.

Line 9 and applications of Lemma 2. A two-dimensional integer linear program solver for problems having at most four constraints, such as the one described in [5]. The size of the data describing each of these constraints is in the order of the representation of the vector x as a rational number, which, in its standard truncated decimal representation, is in $\mathcal{O}(\ln(B))$.

Line 30 and applications of Lemma 2. A solver for one-dimensional integer convex optimization problems. At every iteration, we need to perform at most seven of them, for a cost of $\mathcal{O}(\ln(B))$ at each time.

Lines 18 and 19. Given a segment $[a, b]$ and one of its points z, we need a device to determine which of the two regions $[a, z]$ or $[z, b]$ intersects a level set defined by f and g that does not contain z. This procedure has a complexity of $\mathcal{O}(\ln(B))$ and only occurs in Case 2 above.

3.2 *Finding the k-th Best Point*

In this section we want to show how to find the k-th best point, provided that the $k - 1$ best points are known. A slight variant of this problem will be used in Sect. 4.3 as a subroutine for the general mixed-integer convex problem. In the following, we describe the necessary extensions of the previous Algorithm 3. Let $\hat{x}_1^* := \hat{x}^*$ and define for $k \geq 2$:

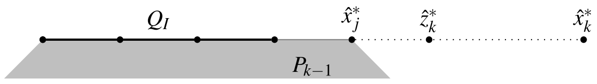
$$\hat{x}_k^* := \arg\min\left\{f(\hat{x}) | \hat{x} \in \left(S \cap [-B, B]^2 \cap \mathbb{Z}^2\right) \setminus \left\{\hat{x}_1^*, \ldots, \hat{x}_{k-1}^*\right\}\right\}$$

to be the k-th best point. Observe that, due to the convexity of f and $g_1, \ldots, g_m$, we can always assume that $\text{conv}\{\hat{x}_1^*, \ldots, \hat{x}_{k-1}^*\} \cap \mathbb{Z}^2 = \{\hat{x}_1^*, \ldots, \hat{x}_{k-1}^*\}$ for all $k \geq 2$.

Algorithm 3: Minimization algorithm for 2D problems

```
**Data**: $x \in [-B, B]^2$ with $f(x) \leq \hat{f}^*$ and $g_i(x) \leq 0$ for all $i = 1, \dots, m$.
1  Let $F_1, \dots, F_4$ be the facets of $[-B, B]^2$.
2  Set $\hat{x}^* := 0$ and $\hat{f}^* := +\infty$.
3  for $t = 1, \dots, 4$ do
4      Set $F := F_t$ and define $v_0, v_1 \in \mathbb{R}^n$ such that $F := \mathrm{conv}\{v_0, v_1\}$.
5      Write $h$ for the vector normal to $F$ pointing outward $[-B, B]^2$.
6      Set $T_0 := \mathrm{conv}\{x, F\}$ and $k := 0$.
7      while $\mathrm{vol}(T_k) \geq \frac{1}{2}$ do
8          Set $\bar{T}_k := \mathrm{conv}\{x, v_{1/3}, v_{2/3}\}$, with $v_\lambda := (1 - \lambda)v_0 + \lambda v_1$.
9          Solve $(\mathcal{P}) : \min\{\langle h, \hat{y}\rangle : \hat{y} \in \bar{T}_k \cap \mathbb{Z}^2\}$.
10         if (Case 1) $(\mathcal{P})$ is infeasible, then
11             Determine $\hat{x} := \arg\min\{f(\hat{y}) \mid \hat{y} \in (T_k \setminus \bar{T}_k) \cap \mathbb{Z}^2$ with $g(\hat{y}) \leq 0\}$.
12             if $\hat{x}$ exists and $f(\hat{x}) < \hat{f}^*$ then Set $\hat{x}^* := \hat{x}$ and $\hat{f}^* := f(\hat{x})$.
13         else
14             (Case 2) Let $\hat{z}$ be an optimal solution of $(\mathcal{P})$.
15             Set $H_+ := \{y \in \mathbb{R}^2 : \langle h, y\rangle \leq \langle h, \hat{z}\rangle\}$ and $H := \partial H_+$.
16             Define the points $v := \mathrm{aff}\{x, \hat{z}\} \cap F$ and $z_i = H \cap \mathrm{conv}\{x, v_i\}$ for
               $i = 0, 1$.
17             Denote $z_\lambda := (1 - \lambda)z_0 + \lambda z_1$ for $\lambda \in (0, 1)$.
18             if $g(\hat{z}) \leq 0$ and there is a $y \in \mathrm{conv}\{z_0, \hat{z}\}$ for which $f(y) < f(\hat{z})$ or
19               $g(\hat{z}) > 0$ and there is a $y \in \mathrm{conv}\{z_0, \hat{z}\}$ for which $g(y) < g(\hat{z})$
               then
20                 Determine $\hat{x} := \arg\min\{f(\hat{y}) \mid \hat{y} \in \mathrm{conv}\{x, z_{1/3}, z_1\} \cap \mathbb{Z}^2$
                   with $g(\hat{y}) \leq 0\}$.
21                 if $\hat{x}$ exists and $f(\hat{x}) < \hat{f}^*$ then Set $\hat{x}^* := \hat{x}$ and $\hat{f}^* := f(\hat{x})$.
22                 Set $v_1 := v$, $T_{k+1} := \mathrm{conv}\{x, v_0, v\}$, and $k := k + 1$.
23             else
24                 Determine $\hat{x} := \arg\min\{f(\hat{y}) \mid \hat{y} \in \mathrm{conv}\{x, z_0, z_{2/3}\} \cap \mathbb{Z}^2$
                   with $g(\hat{y}) \leq 0\}$.
25                 if $\hat{x}$ exists and $f(\hat{x}) < \hat{f}^*$ then Set $\hat{x}^* := \hat{x}$ and $\hat{f}^* := f(\hat{x})$.
26                 Set $v_0 := v$, $T_{k+1} := \mathrm{conv}\{x, v, v_1\}$, and $k := k + 1$.
27             end
28         end
29     end
30     Determine $\hat{x} := \arg\min\{f(\hat{y}) \mid \hat{y} \in T_k \cap \mathbb{Z}^2$ with $g(\hat{y}) \leq 0\}$.
31     if $\hat{x}$ exists and $f(\hat{x}) < \hat{f}^*$ then Set $\hat{x}^* := \hat{x}$ and $\hat{f}^* := f(\hat{x})$.
32 end
33 if $\hat{f}^* < +\infty$ then Return $\hat{x}^*$.
34 else Return "the problem is infeasible".
```

Fig. 4 Illustration of Part (a)



Although this observation appears plausible it is not completely trivial to achieve this algorithmically.

Lemma 3 *Let $\Pi_j := \{\hat{x}_1^*, \dots, \hat{x}_j^*\}$ be the ordered j best points of our problem and P_j be the convex hull of Π_j. Suppose that, for a given $k \geq 2$, we have $P_{k-1} \cap \mathbb{Z}^2 = \Pi_{k-1}$. Let $\hat{x}_k^*$ be a k-th best point.*

(a) *If $f(\hat{x}_k^*) > \hat{f}^*$, we can replace the point $\hat{x}_k^*$ by a feasible k-th best point $\hat{z}_k^*$ such that* $\mathrm{conv}\{\Pi_{k-1}, \hat{z}_k^*\} \cap \mathbb{Z}^2 = \Pi_{k-1} \cup \{\hat{z}_k^*\}$ *in $\mathcal{O}(1)$ operations.*
(b) *If $f(\hat{x}_k^*) = \hat{f}^*$, and if we have at our disposal the ν vertices of P_{k-1} ordered counterclockwise, we can construct such a point $\hat{z}_k^*$ in $\mathcal{O}(\nu \ln(B))$ operations.*

Proof Part (a). Suppose first that $f(\hat{x}_k^*) > \hat{f}^*$, and assume that we cannot set $\hat{z}_k^* := \hat{x}_k^*$, that is, that there exists $\hat{x} \in (P_k \cap \mathbb{Z}^2) \setminus \Pi_k$. Then $\hat{x} = \sum_{i=1}^k \lambda_i \hat{x}_i^*$ for some $\lambda_i \geq 0$ that sum up to 1. Note that $0 < \lambda_k < 1$, because $\hat{x} \notin P_{k-1} \cup \{\hat{x}_k^*\}$ by assumption, and that $f(\hat{x}) \geq f(\hat{x}_k^*)$. We deduce:

$$0 \leq f(\hat{x}) - f\left(\hat{x}_k^*\right) \leq \sum_{i=1}^{k} \lambda_i \left(f\left(\hat{x}_i^*\right) - f\left(\hat{x}_k^*\right)\right) \leq 0.$$

Thus $f(\hat{x}) = f(\hat{x}_k^*)$. Let $I := \{i : \lambda_i > 0\}$ and $Q_I := \mathrm{conv}\{\hat{x}_i^* : i \in I\}$, so that $\hat{x} \in \mathrm{relint}\, Q_I$. Observe that $|I| \geq 2$ and that f is constant on Q_I. Necessarily, Q_I is a segment. Indeed, if it were a two-dimensional set, we could consider the restriction of f on the line $\ell := \mathrm{aff}\{\hat{x}_1^*, \hat{x}\}$: it is constant on the open interval $\ell \cap \mathrm{int}\, Q_I$, but does not attain its minimum on it, contradicting the convexity of f. Let us now construct the point $\hat{z}_k^*$: it suffices to consider the closest point to $\hat{x}_k^*$ in $\mathrm{aff}\{Q_I\} \cap P_{k-1}$, say $\hat{x}_j^*$, and to take the integer point $\hat{z}_k^* \neq \hat{x}_j^*$ of $\mathrm{conv}\{\hat{x}_j^*, \hat{x}_k^*\}$ that is the closest to $\hat{x}_j^*$ (see Fig. 4).

Part (b). Suppose now that $f(\hat{x}_i^*) = \hat{f}^*$ for every $1 \leq i \leq k$, and define

$$\left\{\hat{y}_0^* \equiv \hat{y}_\nu^*, \hat{y}_1^*, \dots, \hat{y}_{\nu-1}^*\right\} \subseteq \Pi_{k-1}$$

as the vertices of P_{k-1}, labeled counterclockwise. It is well-known that determining the convex hull of $P_{k-1} \cup \{\hat{x}_k^*\}$ costs $\mathcal{O}(\ln(\nu))$ operations. From these vertices, we deduce the set $\{\hat{y}_i^* : i \in J\}$ of those points that are in the relative interior of that convex hull. Up to a renumbering of the $\hat{y}_l^*$'s, we have $J = \{1, 2, \dots, j-1\}$. We show below that Algorithm 4 constructs a satisfactory point $\hat{z}_k^*$.

We follow here the notation used in Algorithm 4. At every iteration i, the algorithm constructs from an integer point $\hat{x}_k^*(i)$ an integer point $\hat{x}_k^*(i+1)$, possibly identical to $\hat{x}_k^*(i)$. When the algorithm stops, after at most $j \leq \nu$ iterations, the point $\hat{z}_k^*$ we are looking for is, as we prove it below, the last $\hat{x}_k^*(i)$ we have constructed.

Algorithm 4: Constructing a point $\hat{z}_k^*$ with $\text{conv}\{\Pi_{k-1}, \hat{z}_k^*\} \cap \mathbb{Z}^2 = \Pi_{k-1} \cup \{\hat{z}_k^*\}$

```
**Data**: $\hat{x}_k^*, \hat{y}_0^*, \hat{y}_1^*, \ldots, \hat{y}_j^*$.
Set $i := 0$ and $x_k^*(0) := x_k^*$.
**while** $\det(\hat{x}_k^*(i) - \hat{y}_i^*, \hat{y}_{i+1}^* - \hat{y}_i^*) \geq 0$ **do**
    Set $\Delta_i := \text{conv}\{\hat{x}_k^*(i), \hat{y}_i^*, \hat{y}_{i+1}^*\} \setminus \text{aff}\{\hat{y}_i^*, \hat{y}_{i+1}^*\}$.
    Set $h_i$ a vector orthogonal to $\text{aff}\{\hat{y}_i^*, \hat{y}_{i+1}^*\}$ such that $\langle h_i, \hat{x}_k^*(i) - \hat{y}_i^*\rangle > 0$.
    Set $\hat{x}_k^*(i+1) := \arg\min\{\langle h_i, \hat{x}\rangle : \hat{x} \in \Delta_i \cap \mathbb{Z}^2\}$.
    Set $i := i + 1$.
**end**
Set $\hat{z}_k^* := \hat{x}_k^*(i)$.
```

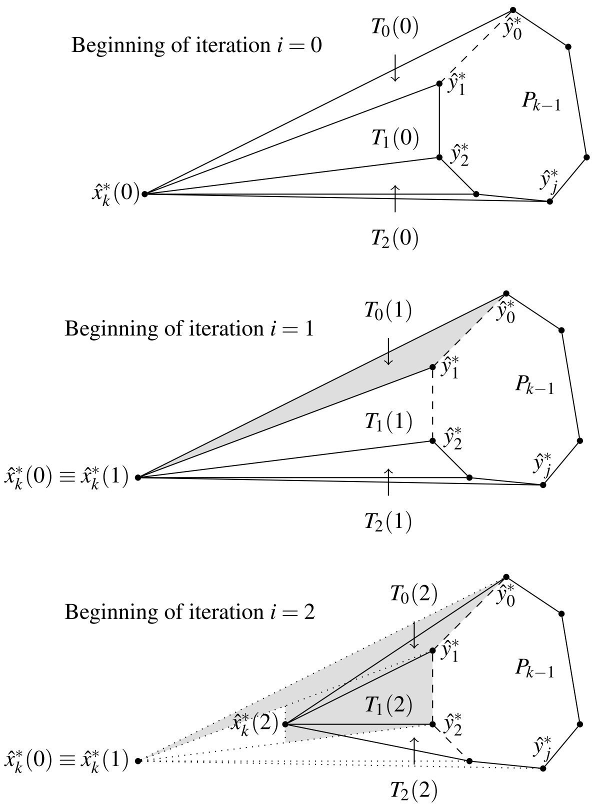


Fig. 5 Constructing P_k from P_{k-1}: first iterations of Algorithm 4. The point $\hat{x}_k^*(1)$ is the same as $\hat{x}_k^*(0)$ because $T_0(0)$ has no other integer point than $\hat{x}_k^*(0)$. The *gray areas* are, as the algorithm progresses, regions where we have established that they do not contain any integer point

Define $T_l(i) := \text{conv}\{\hat{x}_k^*(i), \hat{y}_l^*, \hat{y}_{l+1}^*\} \setminus P_{k-1}$ for $0 \leq l < j$ (see Fig. 5); the triangle Δ_i in Algorithm 4 corresponds to $T_i(i)$. Also, the vector h_i is orthogonal to the side $\text{aff}\{\hat{y}_i^*, \hat{y}_{i+1}^*\}$ of the triangle $T_i(i)$.

At iteration i, the algorithm considers the triangle $T_i(i)$ if its signed area

$$\frac{1}{2}\det\bigl(\hat{x}_k^*(i) - \hat{y}_i^*, \hat{y}_{i+1}^* - \hat{y}_i^*\bigr)$$

is nonnegative, and finds a point $\hat{x}_k^*(i+1) \in T_i(i)$ such that $T_i(i+1)$ has only $\hat{x}_k^*(i+1)$ as integer point.

We prove by induction on $i \geq 1$ that $T_l(i)$ contains only $\hat{x}_k^*(i)$ as integer point whenever $l < i$. Consider the base case $i = 1$. By construction, the triangle $T_0(1)$ contains only $x_k^*(1)$ as integer point, for otherwise $x_k^*(1)$ would not minimize $\langle h_0, \hat{x}\rangle$ over $T_0(0) \cap \mathbb{Z}^2$.

Suppose now that the statement is true for i and let $l \leq i$. Let us verify that $\hat{x}_k^*(i+1)$ is the only integer in $T_l(i+1)$. We have:

$$\hat{x}_k^*(i+1) \in T_i(i) \subseteq \operatorname{conv}\bigl\{\hat{x}_k^*(i), \hat{y}_0^*, \ldots, \hat{y}_{i+1}^*\bigr\} \setminus P_{k-1} = T_i(i) \cup \bigcup_{l=0}^{i-1} T_l(i).$$

This last equality represents a triangulation of the possibly non-convex polygon $\operatorname{conv}\{\hat{x}_k^*(i), \hat{y}_0^*, \ldots, \hat{y}_{i+1}^*\} \setminus P_{k-1}$. From the above inclusion, we deduce:

$$K := \operatorname{conv}\bigl\{\hat{x}_k^*(i+1), \hat{y}_0^*, \ldots, \hat{y}_{i+1}^*\bigr\} \setminus P_{k-1} \subseteq \operatorname{conv}\bigl\{\hat{x}_k^*(i), \hat{y}_0^*, \ldots, \hat{y}_{i+1}^*\bigr\} \setminus P_{k-1}.$$

As $T_l(i+1) \subseteq K$ for all $l \leq i$, the integers of $T_l(i+1)$ are either in $\bigcup_{l=0}^{i-1} T_l(i) \cap \mathbb{Z}^2$, which reduces to $\{\hat{x}_k^*(i)\}$ by induction hypothesis, or in $T_i(i)$. Since $\hat{x}_k^*(i) \in T_i(i)$, all the integers in $T_l(i+1)$ must be in $T_i(i)$. But $T_l(i+1) \cap T_i(i) \cap \mathbb{Z}^2 = \{\hat{x}_k^*(i+1)\}$ by construction of $\hat{x}_k^*(i+1)$. The induction step is proved.

It remains to take the largest value that i attains in the course of Algorithm 4 to finish the proof. We need to solve at most $\nu - 1$ two-dimensional integer linear problems over triangles to compute $\hat{x}_k^*$. As the data of these problems are integers bounded by B, the complexity of the minimization solver used to compute $x_k^*(i+1)$ at every step is bounded by $\mathcal{O}(\ln(B))$. The overall complexity of Algorithm 4 is thus bounded by $\mathcal{O}(\nu \ln(B))$. □

By Lemma 3, the k-th best point $\hat{x}_k^*$ can be assumed to be contained within $[-B, B]^2 \setminus \operatorname{conv}\{\hat{x}_1^*, \ldots, \hat{x}_{k-1}^*\}$. This property allows us to design a straightforward algorithm to compute this point. We first construct an inequality description of $\operatorname{conv}\{\hat{x}_1^*, \ldots, \hat{x}_{k-1}^*\}$, say $\langle a_i, x\rangle \leq b_i$ for $i \in I$ with $|I| < +\infty$. Then

$$[-B, B]^2 \setminus \operatorname{conv}\bigl\{\hat{x}_1^*, \ldots, \hat{x}_{k-1}^*\bigr\} = \bigcup_{i \in I} \bigl\{x \in [-B, B]^2 | \langle a_i, x\rangle > b_i\bigr\}.$$

As the feasible set is described as a union of simple convex sets, we could apply Algorithm 1 once for each of them. However, instead of choosing this straightforward approach one can do better: one can avoid treating each element of this disjunction separately by modifying Algorithm 3 appropriately.

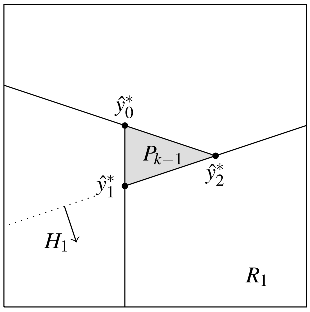


Fig. 6 Triangulation step 1

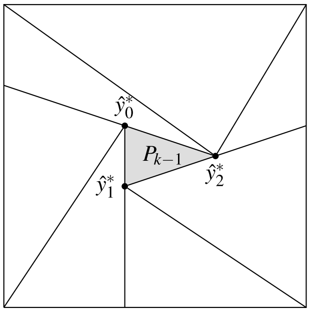


Fig. 7 Triangulation step 2

Suppose first that $k = 2$. To find the second best point, we apply Algorithm 3 to the point $\hat{x}_1^*$ with the following minor modification: in Line 9, we replace $(\mathcal{P})$ with the integer linear problem $(\mathcal{P}') : \min\{\langle h, \hat{y}\rangle : \hat{y} \in \bar{T}_k \cap \mathbb{Z}^2, \langle h, \hat{y}\rangle \geq \langle h, \hat{x}_1\rangle + 1\}$, where $h \in \mathbb{Z}^2$ such that $\gcd(h_1, h_2) = 1$. This prevents the algorithm from returning $\hat{x}_1^*$ again.

Let $k \geq 3$. Let $\hat{y}_0^*, \ldots, \hat{y}_{\nu-1}^*, \hat{y}_\nu^* \equiv \hat{y}_0^*$ denote the vertices of P_{k-1}, ordered counterclockwise (they can be determined in $\mathcal{O}(k \ln(k))$ operations using the Graham Scan [7]). Recall that the point we are looking for is not in P_{k-1}.

Let us call a triangle with a point $\hat{y}_i^*$ as vertex and with a segment of the boundary of $[-B, B]^2$ as opposite side a *search triangle* (see Fig. 7: every white triangle is a search triangle). The idea is to decompose $[-B, B]^2 \setminus P_{k-1}$ into search triangles, then to apply Algorithm 3 to these triangles instead of $(\text{conv}\{x, F_t\})_{t=1}^4$.

For each $0 \leq i < \nu$, we define $H_i := \{y \in \mathbb{R}^2 : \det(y - \hat{y}_i^*, \hat{y}_{i+1}^* - \hat{y}_i^*) \geq 0\}$, so that $H_i \cap P_{k-1} = \text{conv}\{\hat{y}_i^*, \hat{y}_{i+1}^*\}$. Consider the regions $R_i := ([-B, B]^2 \cap H_i) \setminus \text{int}\, H_{i-1}$ (see Fig. 6). Note that R_i contains only $\hat{y}_i^*$ and $\hat{y}_{i+1}^*$ as vertices of P_{k-1}. Also, at most four of the R_i's are no search triangles. If R_i is such, we triangulate it into (at least two) search triangles by inserting chords from $\hat{y}_i^*$ to the appropriate vertices of $[-B, B]^2$ (see Fig. 7).

Note that a search triangle can contain two or more integer points of P_{k-1}. In order to prevent us from outputting one of those, we need to perturb the search triangles slightly before using them in Algorithm 3. Let $T = \text{conv}\{\hat{y}_i^*, b_1, b_2\}$ be one of the search triangles, with b_1, b_2 being points of the boundary of $[-B, B]^2$.

The triangle T might contain $\hat{y}^*_{i+1}$, say $\hat{y}^*_{i+1} \in \operatorname{conv}\{\hat{y}^*_i, b_1\}$, a point we need to exclude from T. We modify b_1 slightly by replacing it with $(1-\varepsilon)b_1 + \varepsilon b_2$ for an appropriate positive $\varepsilon > 0$ whose encoding length is $\mathcal{O}(\ln(B))$.

So, we apply Algorithm 3 with all these modified search triangles instead of $\operatorname{conv}\{x, F_1\}, \ldots, \operatorname{conv}\{x, F_4\}$. A simple modification of Line 9 allows us to avoid the point $\hat{y}^*_i$ for $\hat{z}$: we just need to replace the linear integer problem $(\mathcal{P})$ with $\min\{\langle h, \hat{y}\rangle : \hat{y} \in \bar{T}_k \cap \mathbb{Z}^2, \langle h, \hat{y}\rangle \geq \langle h, \hat{y}^*_i\rangle + 1\}$, where $h \in \mathbb{Z}^2$ such that $\gcd(h_1, h_2) = 1$. Then, among the feasible integer points found, we return the point with smallest objective value.

Corollary 2 *Let $f : \mathbb{R}^2 \to \mathbb{R}$ and $g_i : \mathbb{R}^2 \to \mathbb{R}$ with $i = 1, \ldots, m$ be convex functions. Let $\hat{x}^*_1, \ldots, \hat{x}^*_{k-1}$ be the $k-1$ best points for* $\min\{f(\hat{x}) : \hat{x} \in S \cap [-B, B]^2 \cap \mathbb{Z}^2\}$. *Then, in a number of evaluations of f and $g_1, \ldots, g_m$ that is polynomial in* $\ln(B)$ *and in k, one can either find*

(a) *a k-th best point, $\hat{x}^*_k$, or*
(b) *show that there is no such point.*

4 Extensions and Applications to the General Setting

In this section, we extend our algorithm for solving two-dimensional integer convex optimization problems in order to solve more general mixed-integer convex problems. The first extension concerns mixed-integer convex problems with two integer variables and d continuous variables. For those, we first need results about problems with only one integer variable. We derive these results in Sect. 4.1 where we propose a variant of the well-known golden search method that deals with convex functions whose value is only known approximately. To the best of our knowledge, this variant is new.

In Sect. 4.2, we build an efficient method for solving mixed-integer convex problems with two integer and d continuous variables and propose an extension of Corollary 2. This result itself will be used as a subroutine to design a finite-time algorithm for mixed-integer convex problems in n integer and d continuous variables in Sect. 4.3.

In this section, the problem of interest is (1):

$$\min\left\{f(\hat{x}, y) : g_i(\hat{x}, y) \leq 0 \text{ for } 1 \leq i \leq m, (\hat{x}, y) \in \mathbb{Z}^n \times \mathbb{R}^d\right\}$$

with a few mild simplifying assumptions. We define the function

$$g : \mathbb{R}^n \to \mathbb{R}, \quad x \mapsto g(x) := \min_{y \in \mathbb{R}^d} \max_{1 \leq i \leq m} g_i(x, y).$$

We assume that this minimization in y has a solution for every $x \in \mathbb{R}^n$, making the function g convex. Let $S := \{(x, y) \in \mathbb{R}^{n+d} : g_i(x, y) \leq 0 \text{ for } 1 \leq i \leq m\}$. We assume that the function f has a finite spread

$$\max\{f(x,y) - f(x',y') : (x,y), (x',y') \in S\}$$

on S and that we know an upper bound V_f on that spread. Observe that, by Lipschitz continuity of f and the assumption that we optimize over $[-B, B]^n$, it follows $V_f \leq 2\sqrt{n}BL$. Finally, we assume that the partial minimization function:

$$\phi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}, \quad x \mapsto \phi(x) := \min\{f(x,y) : (x,y) \in S\}$$

is convex. As for the function g, this property can be achieved e.g. if for every $x \in \mathbb{R}^n$ for which $g(x) \leq 0$ there exists a point y such that $(x,y) \in S$ and $\phi(x) = f(x,y)$.

Our approach is based on the following well-known identity:

$$\min\{f(\hat{x},y) : (\hat{x},y) \in S \cap (\mathbb{Z}^n \times \mathbb{R}^d)\} = \min\{\phi(\hat{x}) : g(\hat{x}) \leq 0, \hat{x} \in \mathbb{Z}^n\}.$$

For instance, when $n = 2$, we can use the techniques developed in the previous section on ϕ to implement the improvement oracle for f. However, we cannot presume to know exactly the value of ϕ, as it results from a minimization problem. We merely assume that, for a known accuracy $\gamma > 0$ and for every $x \in \operatorname{dom}\phi$ we can determine a point y_x such that $(x, y_x) \in S$ and $f(x, y_x) - \gamma \leq \phi(x) \leq f(x, y_x)$. Determining y_x can be, on its own, a non-trivial optimization problem. Nevertheless, it is a convex problem for which we can use the whole machinery of standard Convex Programming (see e.g. [3, 14, 15] and references therein).

Since we do not have access to exact values of ϕ, we cannot hope for an exact oracle for the function ϕ, let alone for f. The impact of the accuracy γ on the accuracy of the oracle is analyzed in the next sections.

4.1 Mixed-Integer Convex Problems with One Integer Variable

The Algorithm 3 uses as indispensable tools the bisection method for solving two types of problems: minimizing a convex function over the integers of an interval, and finding, in a given interval, a point that belongs to a level set of a convex function. In this section, we show how to adapt the bisection methods for mixed-integer problems. It is well-known that the bisection method is the fastest for minimizing univariate convex functions over a finite segment (Chap. 1 in [12]).

Let $a, b \in \mathbb{R}$, $a < b$, and $\varphi : [a,b] \to \mathbb{R}$ be a convex function to minimize on $[a,b]$ and/or on the integers of $[a,b]$, such as the function ϕ in the preamble of this Sect. 4 when $n = 1$. Assume that, for every $t \in [a,b]$, we know a number $\tilde{\varphi}(t) \in [\varphi(t), \varphi(t) + \gamma]$. In order to simplify the notation, we scale the problem so that $[a,b] \equiv [0,1]$. The integers of $\operatorname{aff}\{a,b\}$ are scaled to a set of points of the form $t_0 + \tau\mathbb{Z}$ for a $\tau > 0$. Of course, the spread of the function φ does not change, but its Lipschitz constant does, and achieving the accuracy γ in its evaluation must be reinterpreted accordingly.

In the sequel of this section, we fix $0 \leq \lambda_0 < \lambda_1 \leq 1$.

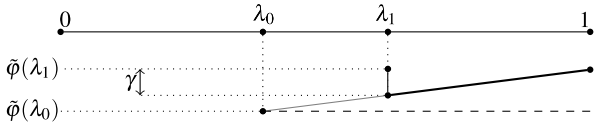


Fig. 8 Lemma 4: the *bold line* represents a lower bound on φ in Part (a)

Lemma 4 *Under our assumptions, the following statements hold.*

(a) *If* $\tilde{\varphi}(\lambda_0) \le \tilde{\varphi}(\lambda_1) - \gamma$, *then* $\varphi(\lambda) \ge \tilde{\varphi}(\lambda_0)$ *for all* $\lambda \in [\lambda_1, 1]$.
(b) *If* $\tilde{\varphi}(\lambda_0) \ge \tilde{\varphi}(\lambda_1) + \gamma$, *then* $\varphi(\lambda) \ge \tilde{\varphi}(\lambda_1)$ *for all* $\lambda \in [0, \lambda_0]$.

Proof We only prove Part (a) as the proof of Part (b) is symmetric. Thus, let us assume that $\tilde{\varphi}(\lambda_0) \le \tilde{\varphi}(\lambda_1) - \gamma$. Then there exists $0 < \mu \le 1$ for which $\lambda_1 = \mu\lambda + (1-\mu)\lambda_0$. Convexity of φ allows us to write:

$$\tilde{\varphi}(\lambda_0) \le \tilde{\varphi}(\lambda_1) - \gamma \le \varphi(\lambda_1) \le \mu\varphi(\lambda) + (1-\mu)\varphi(\lambda_0) \le \mu\varphi(\lambda) + (1-\mu)\tilde{\varphi}(\lambda_0),$$

implying $\tilde{\varphi}(\lambda_0) \le \varphi(\lambda)$ as $\mu > 0$. Figure 8 illustrates the proof graphically. □

If one of the conditions in Lemma 4 is satisfied, we can remove from the interval $[0, 1]$ either $[0, \lambda_0[$ or $]\lambda_1, 1]$. To have a symmetric effect of the algorithm in either case, we set $\lambda_1 := 1 - \lambda_0$, forcing λ_0 to be smaller than $\frac{1}{2}$. In order to recycle our work from iteration to iteration, we choose $\lambda_1 := \frac{1}{2}(\sqrt{5} - 1)$, as in the golden search method: if we can eliminate, say, the interval $]\lambda_1, 1]$ from $[0, 1]$, we will have to compute in the next iteration step an approximate value of the objective function at $\lambda_0\lambda_1$ and λ_1^2. The latter happens to equal λ_0 when $\lambda_1 = \frac{1}{2}(\sqrt{5} - 1)$.

It remains to define a strategy when neither of the conditions in Lemma 4 is satisfied. In the lemma below, we use the values for λ_0, λ_1 chosen above.

Lemma 5 *Assume that* $\tilde{\varphi}(\lambda_1) - \gamma < \tilde{\varphi}(\lambda_0) < \tilde{\varphi}(\lambda_1) + \gamma$. *We define*:

$$\lambda_{0+} := (1-\lambda_0)\cdot\lambda_0 + \lambda_0\cdot\lambda_1 = 2\lambda_0\lambda_1,$$
$$\lambda_{1+} := (1-\lambda_1)\cdot\lambda_0 + \lambda_1\cdot\lambda_1 = 1 - 2\lambda_0\lambda_1.$$

If $\min\{\tilde{\varphi}(\lambda_{0+}), \tilde{\varphi}(\lambda_{1+})\} \le \min\{\tilde{\varphi}(\lambda_0) - \gamma, \tilde{\varphi}(\lambda_1) - \gamma\}$, *then* $\varphi(t) \ge \min\{\tilde{\varphi}(\lambda_{0+}), \tilde{\varphi}(\lambda_{1+})\}$ *for all* $t \in [0, 1] \setminus [\lambda_0, \lambda_1]$. *Otherwise, it holds that* $\min\{\tilde{\varphi}(\lambda_0), \tilde{\varphi}(\lambda_1)\} \le \min\{\varphi(t) : t \in [0, 1]\} + (\kappa - 1)\gamma$, *where* $\kappa := \frac{2}{\lambda_0} \approx 5.236$.

Proof The first conclusion follows immediately from Lemma 4. The second situation involves a tedious enumeration, summarized in Fig. 9. We assume, without loss of generality, that $\tilde{\varphi}(\lambda_0) \le \tilde{\varphi}(\lambda_1)$. The bold lines in Fig. 9 represent a lower bound on the value of the function φ. We show below how this lower bound is constructed and determine its lowest point. In fact, this lower bound results from six applications of a simple generic inequality (9) that we establish below, before showing how we can particularize it to different segments of the interval $[0, 1]$.

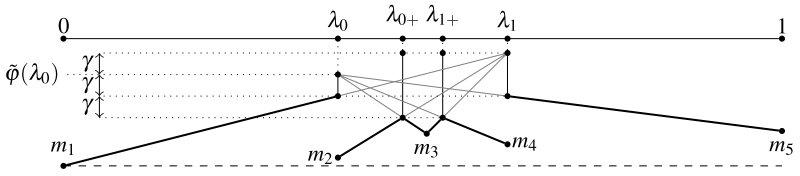


Fig. 9 Approximate bisection: *bold lines* represent a lower bound on φ in the termination case

Let $0 < t < 1$ and let $u, v \in \{\lambda_0, \lambda_{0+}, \lambda_{1+}, \lambda_1\}$. Suppose that we can write $v = \mu t + (1-\mu)u$ for a $\mu \in]\mu_0, 1]$ with $\mu_0 > 0$. If we can find constants $\gamma_-, \gamma_+ \geq 0$ that satisfy

$$\varphi(v) + \gamma_+ \geq \tilde{\varphi}(\lambda_0) \geq \varphi(u) - \gamma_-$$

then we can infer:

$$\mu\varphi(t) + (1-\mu)\big(\tilde{\varphi}(\lambda_0) + \gamma_-\big) \geq \mu\varphi(t) + (1-\mu)\varphi(u) \geq \varphi(v) \geq \tilde{\varphi}(\lambda_0) - \gamma_+,$$

and thus:

$$\varphi(t) - \tilde{\varphi}(\lambda_0) \geq \gamma_- - \frac{\gamma_+ + \gamma_-}{\mu} \geq \gamma_- - \frac{\gamma_+ + \gamma_-}{\mu_0}. \tag{9}$$

1. If $t \in]0, \lambda_0]$, we can take $u := \lambda_1$ and $v := \lambda_0$, giving $\mu_0 = 1 - \frac{\lambda_0}{\lambda_1} = \lambda_0$. Then $\gamma_- = \gamma_+ = \gamma$, and $\varphi(t) - \tilde{\varphi}(\lambda_0) \geq -\gamma(\frac{2}{\lambda_0} - 1)$.
2. If $t \in]\lambda_1, 1[$, we choose $u := \lambda_0$ and $v := \lambda_1$, and by symmetry with the previous case we obtain $\mu_0 = \lambda_0$. Now, $\gamma_- = 0$ and $\gamma_+ = \gamma$, yielding a higher bound than in the previous case.
3. Suppose $t \in]\lambda_0, \lambda_{0+}]$. Then with $u := \lambda_1$ and $v := \lambda_{0+}$, we get $\mu_0 = \frac{\lambda_1 - \lambda_{0+}}{\lambda_1 - \lambda_0} = \lambda_1$, $\gamma_- = \gamma$, $\gamma_+ = 2\gamma$, giving as lower bound $-\gamma(\frac{3}{\lambda_1} - 1)$, which is higher than the first one we have obtained.
4. Symmetrically, let us consider $t \in]\lambda_{1+}, \lambda_1]$. With $u := \lambda_0$ and $v := \lambda_{1+}$, we obtain also $\mu_0 = \lambda_1$. As $\gamma_- = 0$ and $\gamma_+ = 2\gamma$, the lower bound we get is larger than the one in the previous item.
5. Set $\lambda' := \frac{1}{5}(2\lambda_{0+} + 3\lambda_{1+})$. If $t \in]\lambda_{0+}, \lambda']$, we can use $u := \lambda_0$ and $v := \lambda_{0+}$, so that $\mu_0 = \frac{\lambda_{0+} - \lambda_0}{\lambda' - \lambda_0} = 5\lambda_0^2$, $\gamma_- = 0$, and $\gamma_+ = 2\gamma$. Thus, the lower bound is evaluated as $-\frac{2\gamma}{5\lambda_0^2}$, which is higher than any of the bounds we have obtained so far.
6. Finally, if $t \in]\lambda', \lambda_{1+}]$, we take $u := \lambda_1$ and $v := \lambda_{1+}$, so that $\gamma_- = \gamma$, $\gamma_+ = 2\gamma$, and $\mu_0 = \frac{\lambda_1 - \lambda_{1+}}{\lambda_1 - \lambda'} = \frac{5\lambda_0}{2+\lambda_0}$. Hence, we get $-\gamma(\frac{3(2+\lambda_0)}{5\lambda_0} - 1) = -\frac{2\gamma}{5\lambda_0^2}$ for the lower bound, just as in the previous item.

So, the lower bound for $\varphi(t) - \tilde{\varphi}(\lambda_0)$ on $[0, 1]$ can be estimated as $-\gamma(\frac{2}{\lambda_0} - 1) \approx -4.236\gamma$. $\square$

In the proof of the following proposition, we present an algorithm that returns a point $x \in [0,1]$ whose function value $\varphi(x)$ is close to $\min\{\varphi(t) : t \in [0,1]\}$.

Proposition 3 *There exists an algorithm that finds a point* $x \in [0,1]$ *for which* $\tilde{\varphi}(x) - (\kappa - 1)\gamma \leq \min\{\varphi(t) : t \in [0,1]\} \leq \varphi(x)$ *in at most* $2 + \lceil \ln(\frac{(\kappa-1)\gamma}{V_\varphi}) / \ln(\lambda_1) \rceil$ *evaluations of* $\tilde{\varphi}$, *where* V_φ *is the spread of* φ *on* $[0,1]$.

Proof We start with the interval $[0,1]$ and by evaluating $\tilde{\varphi}$ at λ_0 and λ_1. If one of the two conditions in Lemma 4 is satisfied, we can shrink the interval by a factor of $\lambda_0 \approx 38\ \%$ since it suffices to continue either with the interval $[0, \lambda_1]$ or with $[\lambda_0, 1]$. If not, then Lemma 5 applies: if the first condition stated in Lemma 5 is met, then it suffices to continue with the interval $[\lambda_0, \lambda_1]$ so as to shrink the starting interval by a factor of $2\lambda_0 \approx 76\ \%$. Otherwise, any $x \in [\lambda_0, \lambda_1]$ satisfies the requirement of the lemma and we can stop the algorithm. Therefore, either the algorithm stops or we shrink the starting interval by a factor of at least λ_0. Iterating this procedure, it follows that—if the algorithm does not stop—at every step the length of the remaining interval is at most λ_1 times the length of the previous interval. Moreover, by the choice of λ_0, the function $\tilde{\varphi}$ is evaluated in two points at the first step, and in only one point as from the second step in the algorithm. So, at iteration k, we have performed at most $2 + k$ evaluations of $\tilde{\varphi}$.

By construction, the minimum t^* of φ lies in the remaining interval I_k of iteration k. Also, the value of φ outside I_k is higher than the best value found so far, say $\tilde{\varphi}(\bar{t}_k)$. Finally, the size of I_k is bounded from above by λ_1^k. Consider now the segment $I(\lambda) := (1-\lambda)t^* + \lambda[0,1]$, of size λ. Observe that for every λ such that $1 \geq \lambda > \lambda_1^k$, the interval $I(\lambda)$ contains a point that is not in I_k. Therefore,

$$\begin{aligned}\tilde{\varphi}(\bar{t}_k) &\leq \max\bigl\{\varphi(t) : t \in I(\lambda)\bigr\} \leq (1-\lambda)\varphi\bigl(t^*\bigr) + \lambda \max\bigl\{\varphi(t) : t \in [0,1]\bigr\} \\ &\leq (1-\lambda)\varphi\bigl(t^*\bigr) + \lambda\bigl(V_\varphi + \varphi\bigl(t^*\bigr)\bigr).\end{aligned}$$

Hence $\tilde{\varphi}(\bar{t}_k) - \varphi(t^*) \leq \lambda V_\varphi$, and, by taking λ arbitrarily close to λ_1^k, we get $\tilde{\varphi}(\bar{t}_k) - \varphi(t^*) \leq \lambda_1^k V_\varphi$. If the algorithm does not end prematurely, we need at most $\lceil \ln(\frac{(\kappa-1)\gamma}{V_\varphi}) / \ln(\lambda_1) \rceil$ iterations to make $\lambda_1^k V_\varphi$ smaller than $(\kappa - 1)\gamma$. □

Remark 3 If we content ourselves with a coarser precision $\eta \geq (\kappa - 1)\gamma$, we merely need $\mathcal{O}(\ln(V_\varphi/\eta))$ evaluations of $\tilde{\varphi}$.

It is now easy to extend this procedure to minimize a convex function approximately over the integers of an interval $[a,b]$, or, using our simplifying scaling, over $(t_0 + \tau\mathbb{Z}) \cap [0,1]$ for given $t_0 \in \mathbb{R}$ and $\tau > 0$.

Proposition 4 *There exists an algorithm that finds a point* $\hat{x} \in (t_0 + \tau\mathbb{Z}) \cap [0,1]$ *for which*:

$$\tilde{\varphi}(\hat{x}) - \kappa\gamma \leq \min\bigl\{\varphi(\hat{t}) : \hat{t} \in (t_0 + \tau\mathbb{Z}) \cap [0,1]\bigr\} \leq \varphi(\hat{x})$$

in less than

$$\min\left\{4+\left\lceil\frac{\ln((\kappa-1)\gamma/V_\varphi)}{\ln(\lambda_1)}\right\rceil, 5+\left\lceil\frac{\ln(\tau)}{\ln(\lambda_1)}\right\rceil\right\}$$

evaluations of $\tilde{\varphi}$, *where* V_φ *is the spread of* φ *on* $[0, 1]$.

Proof We denote in this proof the points in $(t_0 + \tau\mathbb{Z})$ as *scaled integers*. To avoid a trivial situation, we assume that $[0, 1]$ contains at least two such scaled integers.

Let us use the approximate bisection method described in the proof of Proposition 3 until the remaining interval has a size smaller than τ, so that it contains at most one scaled integer. Two possibilities arise: either the algorithm indeed finds such a small interval I_k, or it finishes prematurely, with a remaining interval I_k larger than τ.

In the first case, which requires at most $2 + \lceil\ln(\tau)/\ln(\lambda_1)\rceil$ evaluations of $\tilde{\varphi}$, we know that I_k contains the continuous minimizer of φ. Hence, the actual minimizer of φ over $(t_0 + \tau\mathbb{Z}) \cap [0, 1]$ is among at most three scaled integers, namely the possible scaled integer in I_k, and, at each side of I_k, the possible scaled integers that are the closest to I_k. By convexity of φ, the best of these three points, say $\hat{x}$, satisfies $\tilde{\varphi}(\hat{x}) - \gamma \leq \varphi(\hat{x}) = \min\{\varphi(\hat{t}) : \hat{t} \in (t_0 + \tau\mathbb{Z}) \cap [0, 1]\}$.

In the second case, we have an interval $I_k \subseteq [0, 1]$ and a point $\bar{t}_k$ that fulfill $\tilde{\varphi}(\bar{t}_k) \leq \min\{\varphi(t) : t \in [0, 1]\} + (\kappa - 1)\gamma$, which was determined within at most $2 + \lceil\frac{\ln((\kappa-1)\gamma/V_\varphi)}{\ln(\lambda_1)}\rceil$ evaluations of $\tilde{\varphi}$. Consider the two scaled integers $\hat{t}_-$ and $\hat{t}_+$ that are the closest from $\bar{t}_k$. One of these two points constitutes an acceptable output for our algorithm. Indeed, suppose first that $\min\{\tilde{\varphi}(\hat{t}_-), \tilde{\varphi}(\hat{t}_+)\} \leq \tilde{\varphi}(\bar{t}_k) + \gamma$. Then:

$$\min\bigl\{\tilde{\varphi}(\hat{t}_-), \tilde{\varphi}(\hat{t}_+)\bigr\} \leq \tilde{\varphi}(\bar{t}_k) + \gamma \leq \min\bigl\{\varphi(t) : t \in [0, 1]\bigr\} + \kappa\gamma,$$

and we are done. Suppose that $\min\{\tilde{\varphi}(\hat{t}_-), \tilde{\varphi}(\hat{t}_+)\} > \tilde{\varphi}(\bar{t}_k) + \gamma$ and that there exists a scaled integer $\hat{t}$ with $\varphi(\hat{t}) < \min\{\varphi(\hat{t}_-), \varphi(\hat{t}_+)\}$. Without loss of generality, let $\hat{t}_- \in \operatorname{conv}\{\hat{t}, \bar{t}_k\}$, that is $\hat{t}_- = \lambda\hat{t} + (1-\lambda)\bar{t}_k$, with $0 \leq \lambda < 1$. We have by convexity of φ:

$$\varphi(\hat{t}_-) \leq \lambda\varphi(\hat{t}) + (1-\lambda)\varphi(\bar{t}_k) < \lambda\varphi(\hat{t}_-) + (1-\lambda)\bigl(\tilde{\varphi}(\hat{t}_-) - \gamma\bigr),$$

which is a contradiction because $\lambda < 1$ and $\tilde{\varphi}(\hat{t}_-) - \gamma \leq \varphi(\hat{t}_-)$. So, it follows that $\varphi(\hat{t}) \geq \min\{\varphi(\hat{t}_-), \varphi(\hat{t}_+)\}$ for every $\hat{t} \in (t_0 + \tau\mathbb{Z}) \cap [0, 1]$, proving the statement. □

In the following we extend the above results to the problem $\min\{\varphi(t) : t \in [0, 1], g(t) \leq 0\}$, where $g : [0, 1] \to \mathbb{R}$ is a convex function with a known spread V_g. In the case that we have access to exact values of g, an approach for attacking the problem would be the following: we first determine whether there exists an element $\bar{t} \in [0, 1]$ with $g(\bar{t}) \leq 0$. If $\bar{t}$ exists, we determine the exact bounds t_- and t_+ of the interval $\{t \in [0, 1], g(t) \leq 0\}$. Then we minimize the function f over $[t_-, t_+]$.

The situation where we do not have access to exact values of g or where we cannot determine the feasible interval $[t_-, t_+]$ induces some technical complications.

We shall not investigate them in this paper, except in the remaining of this subsection in order to appreciate the modification our method needs in that situation: let us assume, that we have only access to a value $\tilde{g}(t) \in [g(t), g(t) + \gamma]$. In order to ensure that the constraint g is well-posed we make an additional assumption: either $\{t \in [0,1] : |g(t)| \leq \gamma\}$ is empty, or the quantity $\min\{|g'(t)| : g'(t) \in \partial g(t), |g(t)| \leq \gamma\}$ is non-zero, and even reasonably large. This ensures that the (possibly empty) 0-level set of g is known with enough accuracy. We denote by $\theta > 0$ a lower bound on this minimum, and for simplicity assume that $\theta = 2^N\gamma$ for a suitable $N \in \mathbb{N}$.

Our strategy proceeds as follows. First we determine whether there exists a point $\bar{t} \in [0,1]$ for which $g(\bar{t}) < 0$ by applying the minimization procedure described in Proposition 3. If this procedure only returns nonnegative values, we can conclude after at most $2 + \lceil \ln((\kappa-1)\gamma/V_g)/\ln(\lambda_1)\rceil$ evaluations of $\tilde{g}$ that $g(t) \geq -(\kappa-1)\gamma$, in which case we declare that we could not locate any feasible point in [0, 1].

Otherwise, if we find a point $\bar{t} \in [0,1]$ with $\tilde{g}(\bar{t}) < 0$, we continue and compute approximate bounds t_- and t_+ of the interval $\{t \in [0,1], g(t) \leq 0\}$. For that, we assume $\tilde{g}(0), \tilde{g}(1) \geq 0$. By symmetry, we only describe how to construct t_- such that $\tilde{g}(t_-) \leq 0$ and $g(t_- - \eta) \geq 0$ for an $\eta > 0$ reasonably small. Note that $g(t) \leq 0$ on $[t_-, \bar{t}]$ by convexity of g.

In order to compute t_-, we adapt the standard bisection method for finding a root of a function. Note that the function $\tilde{g}$ might not have any root as it might not be continuous. Our adapted method constructs a decreasing sequence of intervals $[a_k, b_k]$ such that $\tilde{g}(a_k) > 0$, $\tilde{g}(b_k) \leq 0$, and $b_{k+1} - a_{k+1} = \frac{1}{2}(b_k - a_k)$. If $\tilde{g}(a_k) > \gamma$, we know that g is positive on $[0, a_k]$, and we know that there is a root of g on $[a_k, b_k]$. Otherwise, if $0 < \tilde{g}(a_k) \leq \gamma$ and that the interval $[a_k, b_k]$ has a length larger or equal to $\frac{\gamma}{\theta}$. Given the form of θ, we know that $k \leq N$. We claim that for every $0 \leq t \leq \min\{0, a_k - \frac{\gamma}{\theta}\}$ we have $g(t) \geq 0$, so that we can take $\eta := 2\frac{\gamma}{\theta}$ and $t_- := b_N$. Indeed, assume that $g'(a_k) \geq \theta$, then

$$\tilde{g}(b_k) \geq g(b_k) \geq g(a_k) + g'(a_k)(b_k - a_k) > -\gamma + \theta \cdot \frac{\gamma}{\theta} \geq 0$$

giving a contradiction, so we must have $g'(a_k) \leq -\theta$. We can exclude the case where t can only be 0. As claimed, we have

$$g(t) \geq g(a_k) + g'(a_k)(t - a_k) \geq -\gamma + \theta(a_k - t) \geq 0$$

as $\frac{\gamma}{\theta} \leq a_k - t$. This takes $\lceil \ln(\frac{\gamma}{\theta})/\ln(\frac{1}{2})\rceil$ evaluations of $\tilde{g}$.

Summarizing this, we just sketched the proof of the following corollary.

Corollary 3 *There exists an algorithm that solves* $\min\{\varphi(t) : t \in [0,1], g(t) \leq 0\}$ *approximately, in the sense that it finds, if they exist, three points* $0 \leq t_- \leq x \leq t_+ \leq 1$ *with*:

(a) $g(t) \leq \tilde{g}(t) \leq 0$ *for every* $t \in [t_-, t_+]$,
(b) *if* $t_- \geq 2\frac{\gamma}{\theta}$, *then* $g(t) \geq 0$ *for every* $t \in [0, t_- - 2\frac{\gamma}{\theta}]$,
(c) *if* $t_+ \leq 1 - 2\frac{\gamma}{\theta}$, *then* $g(t) \geq 0$ *for every* $t \in [t_+ + 2\frac{\gamma}{\theta}, 1]$,
(d) $\tilde{\varphi}(x) \leq \min\{\varphi(t) : t \in [t_-, t_+], g(t) \leq 0\} + (\kappa - 1)\gamma$

within at most $3 + \lceil\frac{\ln((\kappa-1)\gamma/V_g)}{\ln(\lambda_1)}\rceil + 2\lceil\frac{\ln(\gamma/\theta)}{\ln(1/2)}\rceil$ *evaluations of* $\tilde{g}$ *and at most* $2 + \lceil\frac{\ln((\kappa-1)\gamma/V_\varphi)}{\ln(\lambda_1)}\rceil$ *evaluations of* $\tilde{\varphi}$.

As stressed before above, we assume from now on that we can compute exactly the roots of the function g on a given interval, so that the segment $[t_-, t_+]$ in Corollary 3 is precisely our feasible set. This situation occurs e.g. in mixed-integer convex optimization with one integer variable when the feasible set $S \subset \mathbb{R} \times \mathbb{R}^d$ is a polytope.

Remark 4 In order to solve problem (1) with one integer variable, we can extend Proposition 4 to implement the improvement oracle $\mathrm{O}_{0,\kappa\gamma}$. We need three assumptions: first, $S \subseteq [a,b] \times \mathbb{R}^d$ with $a < b$; second, f has a finite spread on the feasible set; and third we can minimize $f(x,y)$ with $(x,y) \in S$ and x fixed up to an accuracy γ. That is, we have access to a value $\tilde{\varphi}(x) \in [\varphi(x), \varphi(x) + \gamma]$ with $\varphi(x) := \min\{f(x,y) : (x,y) \in S\}$ being convex.

Given a feasible query point $(x,y) \in [a,b] \times \mathbb{R}^d$, we can determine correctly that there is no point $(\hat{x}, \bar{y}) \in ((t_0 + \tau\mathbb{Z}) \cap [0,1]) \times \mathbb{R}^d$ for which $f(\hat{x}, \bar{y}) \leq f(x,y)$, provided that the output $\hat{x}$ of our approximate bisection method for integers given in Proposition 4 satisfies $\tilde{\varphi}(\hat{x}) - \kappa\gamma > f(x,y)$. Otherwise, we can determine a point $(\hat{x}, \bar{y})$ for which $f(\hat{x}, \bar{y}) \leq f(x,y) + \kappa\gamma$. Note that this oracle cannot report **a** and **b** simultaneously.

4.2 *Mixed-Integer Convex Problems with Two Integer Variables*

We could use the Mirror-Descent Method in Algorithm 1 to solve the generic problem (1) when $n = 2$ with $z \mapsto \frac{1}{2}\|z\|_2^2$ as function V, so that $\sigma = 1$ and $M = \frac{1}{2}\operatorname{diam}(S)^2$, where $\operatorname{diam}(S) = \max\{\|z - z'\|_2 : z, z' \in S\}$. According to (5), the worst-case number of iterations is bounded by a multiple of $L\sqrt{M/\sigma} = \mathcal{O}(L\operatorname{diam}(S))$, where L is the Lipschitz constant of f. As $V_f \leq L\operatorname{diam}(S)$, the resulting algorithm would have a worst-case complexity of $\Omega(V_f)$.

We improve this straightforward approach with a variant of Algorithm 3, whose complexity is polynomial in $\ln(V_f)$. This variant takes into account the fact that we do not have access to exact values of the partial minimization function ϕ defined in the preamble of this section.

Proposition 5 *Suppose that we can determine, for every* $x \in \mathbb{R}^n$ *with* $g(x) \leq 0$, *a point* $y_x \in \mathbb{R}^d$ *satisfying* $f(x, y_x) - \gamma \leq \min\{f(x,y) : (x,y) \in S\}$. *Then we can implement the oracle* $\mathrm{O}_{0,\kappa\gamma}$ *such that for every* $(x,y) \in S$ *it takes a number of evaluations of* f *that is polynomial in* $\ln(V_f/\gamma)$.

Proof We adapt the algorithm described in the proof of Theorem 2 for the function $\phi(x) := \min\{f(x,y) : (x,y) \in S\}$, which we only know approximately. Its available approximation is denoted by $\tilde{\phi}(x) := f(x, y_x) \in [\phi(x), \phi(x) + \gamma]$.

Let $(x, y) \in S$ be the query point and let us describe the changes that the algorithm in Theorem 2 requires. We borrow the notation from the proof of Theorem 2.

The one-dimensional integer minimization problems which arise in the course of the algorithm require the use of our approximate bisection method for integers in Proposition 4. This bisection procedure detects, if it exists, a point $\hat{x}$ on the line of interest for which $\tilde{\phi}(\hat{x}) = f(\hat{x}, y_{\hat{x}}) \leq f(x, y) + \kappa\gamma$ and we are done. Or it reports correctly that there is no integer $\hat{x}$ on the line of interest with $\phi(\hat{x}) \leq f(x, y)$.

In *Case 2*, we would need to check whether $\phi(\hat{z}) \leq f(x, y)$. In view of our accuracy requirement, we only need to check $\tilde{\phi}(\hat{z}) \leq f(x, y) + \kappa\gamma$.

We also need to verify whether the line H intersects the level set $\{x \in \mathbb{R}^2 | \phi(x) \leq f(x, y)\}$. We use the following approximate version:

"*check whether* there is a $v \in \operatorname{conv}\{z_0, \hat{z}\}$ for which $\tilde{\phi}(v) < f(x, y) + (\kappa - 1)\gamma$",

which can be verified using Proposition 3. If such a point v exists, the convexity of ϕ forbids any $w \in \operatorname{conv}\{\hat{z}, z_1\}$ to satisfy $\phi(w) \leq f(x, y)$, for otherwise:

$$\tilde{\phi}(\hat{z}) \leq \phi(\hat{z}) + \gamma \leq \max\{\phi(v), \phi(w)\} + \gamma \leq \max\{\tilde{\phi}(v), \tilde{\phi}(w)\} + \gamma < f(x, y) + \kappa\gamma,$$

a contradiction. Now, if such a point v does not exist, we perform the same test on $\operatorname{conv}\{\hat{z}, z_1\}$. We can thereby determine correctly which side of $\hat{z}$ on H has an empty intersection with the level set. $\square$

Similarly as in Corollary 1, we can extend this oracle into an approximate minimization procedure, which solves our optimization problem up to an accuracy of $\kappa\gamma$, provided that we have at our disposal a point $(x, y) \in S$ such that $f(x, y)$ is a lower bound on the mixed-integer optimal value.

Let us now modify our method for finding the k-th best point for two-dimensional problems to problems with two integer and d continuous variables. Here, we aim at finding—at least approximately—the k-th best fiber $\hat{x}_k^* \in [-B, B]^2$, so that:

$$\left(\hat{x}_k^*, y_k^*\right) \in \arg\min\left\{f(x, y) : (x, y) \in S \cap \left(\left(\mathbb{Z}^2 \setminus \left\{\hat{x}_1^*, \ldots, \hat{x}_{k-1}^*\right\}\right) \times \mathbb{R}^d\right)\right\}$$

for a $y_k^* \in \mathbb{R}^d$. We set $\hat{f}_{[k]}^* := f(\hat{x}_k^*, y_k^*)$. The following proposition summarizes the necessary extensions of Sect. 3.2.

Proposition 6 *Let $k \geq 2$ and let $\Pi_{k-1} := \{\hat{z}_1^*, \ldots, \hat{z}_{k-1}^*\} \subseteq [-B, B]^2 \cap \mathbb{Z}^2$ be points for which $\phi(\hat{z}_i^*) \leq \hat{f}_i^* + i\kappa\gamma$, $g(\hat{z}_i^*) \leq 0$ when $1 \leq i < k$ and such that* $\operatorname{conv}\{\Pi_{k-1}\} \cap \mathbb{Z}^2 = \Pi_{k-1}$. *In a number of approximate evaluations of f and $g_1, \ldots, g_m$ that is polynomial in $\ln(V_f/\gamma)$ and k, one can either*

(a) *find an integral point $\hat{z}_k^* \in [-B, B]^2$ for which $\phi(\hat{z}_k^*) \leq \hat{f}_{[k]}^* + k\kappa\gamma$, $g(\hat{z}_k^*) \leq 0$ and* $\operatorname{conv}\{\Pi_{k-1}, \hat{z}_k^*\} \cap \mathbb{Z}^2 = \Pi_{k-1} \cup \{\hat{z}_k^*\}$, *or*

(b) *show that there is no integral point $\hat{z}_k^* \in [-B, B]^2$ for which $g(\hat{z}_k^*) \leq 0$.*

Proof If $k = 2$, we run Algorithm 3 applied to $\hat{z}_1^*$ with Line 9 replaced by solving $\min\{\langle h, \hat{y}\rangle : \hat{y} \in \bar{T}_k \cap \mathbb{Z}^2, \langle h, \hat{y}\rangle \geq \langle h, \hat{z}_1^*\rangle + 1\}$, where $h \in \mathbb{Z}^2$ such that $\gcd(h_1, h_2) = 1$. We also need to use approximate bisection methods instead of exact ones. Following the proof of Proposition 5, the oracle finds, if it exists, a feasible point $\hat{z}_2^*$. Either $\tilde{\phi}(\hat{z}_2^*) \leq \tilde{\phi}(\hat{z}_1^*) + \kappa\gamma \leq \hat{f}_{[1]}^* + 2\kappa\gamma \leq \hat{f}_{[2]}^* + 2\kappa\gamma$, or $\tilde{\phi}(\hat{z}_2^*) > \tilde{\phi}(\hat{z}_1^*) + \kappa\gamma$, then $\phi(\hat{z}_2^*) \leq \tilde{\phi}(\hat{z}_2^*) \leq \hat{f}_{[2]}^* + \kappa\gamma$. Note that, if $\phi(\hat{z}_2^*) > \phi(\hat{z}_1^*) + \kappa\gamma$, we can conclude a posteriori that z_1^* corresponds precisely to $f_{[1]}^*$.

For $k \geq 3$, we can define the same triangulation as in Fig. 7. Replicating the observation sketched above, we generate indeed a feasible point $\hat{z}_k^*$ for which $\tilde{\phi}(\hat{z}_k^*) \leq \hat{f}_{[k]}^* + k\kappa\gamma$.

Lemma 3 is extended as follows. Suppose that there is an integer point $\hat{x}$ in $\operatorname{conv}\{\Pi_{k-1}, \hat{z}_k^*\} \setminus (\Pi_{k-1} \cup \{\hat{z}_k^*\})$. Since $\phi(x) \leq \tilde{\phi}(x) \leq \hat{f}_{[k]}^* + k\kappa\gamma$ and $g(x) \leq 0$ for every $x \in \Pi_{k-1} \cup \{\hat{z}_k^*\}$, we have $\phi(\hat{x}) \leq \hat{f}_{[k]}^* + k\kappa\gamma$ and $g(\hat{x}) \leq 0$ by convexity. Thus, we can apply Algorithm 4 to find a suitable point $\hat{z}_k^*$ in $\operatorname{conv}\{\Pi_{k-1}, \hat{z}_k^*\}$. □

4.3 A Finite-Time Algorithm for Mixed-Integer Convex Optimization

In this section, we explain how to use the results of the previous section in order to realize the oracle $\mathrm{O}_{\alpha,\delta}$ for $\alpha \geq 0$, $\delta > 0$ in the general case, i.e., with $n \geq 3$ integer and d continuous variables as in (1).

Let $z \in S \subseteq [-B, B]^n \times \mathbb{R}^d$ be the query point of the oracle. The oracle needs to find a point $\hat{z} \in S \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ for which $f(\hat{z}) \leq (1 + \alpha)f(z) + \delta$ (so as to report **a**), or to certify that $f(z) < f(\hat{z})$ for every $\hat{z} \in S \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ (so as to report **b**). To design such an oracle we have at our disposal a procedure to realize the oracle $\mathrm{O}_{\alpha,\delta}$ for any mixed-integer convex minimization problem of the kind (1) with $n = 2$. We propose a finite-time implementation of $\mathrm{O}_{\alpha,\delta}$ with $\alpha = 0$ and $\delta = \kappa\gamma$. The main idea is to solve the n-dimensional case iteratively through the fixing of integer variables. This works as follows. We start by solving approximately the relaxation:

$$\hat{f}_{12}^* := \min\left\{f(x, y) : (x, y) \in S \cap \left(\mathbb{Z}^2 \times \mathbb{R}^{(n-2)+d}\right)\right\}$$

with the techniques developed in the previous section. If we can solve the partial minimization problems up to an accuracy of $\gamma \leq \delta/\kappa$, we obtain a point $(\hat{u}_1^*, \hat{u}_2^*, x_3^*, \dots, x_n^*, y^*) \in S$ with $\hat{u}_1^*, \hat{u}_2^* \in \mathbb{Z}$ and for which:

$$\tilde{f}_{12}^* := f\left(\hat{u}_1^*, \hat{u}_2^*, x_3^*, \dots, x_n^*, y^*\right) \leq \hat{f}_{12}^* + \kappa\gamma.$$

As $\hat{f}_{12}^*$ is a lower bound on the mixed-integer optimal value $\hat{f}^*$, we can make our oracle output **b** if $\tilde{f}_{12}^* - \kappa\gamma > f(z)$. So, assume that $\tilde{f}_{12}^* - \kappa\gamma \leq f(z)$.

Then we fix $\hat{x}_i := \hat{u}_i^*$ for $i = 1, 2$ and solve (if $k \geq 4$; if $k = 3$, the necessary modifications are straightforward)

$$\hat{f}_{1234}^* := \min\left\{f(x, y) : (x, y) \in S \cap \left(\left(\hat{u}_1^*, \hat{u}_2^*\right) \times \mathbb{Z}^2 \times \mathbb{R}^{(n-4)+d}\right)\right\}.$$

We obtain a point $(\hat{u}_1^*, \ldots, \hat{u}_4^*, x_5^*, \ldots, x_n^*, y^*) \in S$ with $\hat{u}_i^* \in \mathbb{Z}$ for $1 \le i \le 4$ and for which:

$$\tilde{f}_{1234}^* := f\left(\hat{u}_1^*, \ldots, \hat{u}_4^*, x_5^*, \ldots, x_n^*, y^*\right) \le \hat{f}_{1234}^* + \kappa\gamma \le \hat{f}^* + \kappa\gamma.$$

Now, if $\tilde{f}_{1234}^* - \kappa\gamma > f(z)$, we can make our oracle output **b**. Thus, we assume that $\tilde{f}_{1234}^* - \kappa\gamma \le f(z)$ and fix $\hat{x}_i := \hat{u}_i^*$ for $1 \le i \le 4$. Iterating this procedure we arrive at the subproblem (again, the procedure can easily be modified if n is odd):

$$\min\left\{f(x, y) : (x, y) \in S \cap \left(\left(\hat{u}_1^*, \ldots, \hat{u}_{n-2}^*\right) \times \mathbb{Z}^2 \times \mathbb{R}^d\right)\right\}.$$

Let $(\hat{u}_1^*, \ldots, \hat{u}_n^*, y^*) \in \mathbb{Z}^n \times \mathbb{R}^d$ be an approximate optimal solution. If we cannot interrupt the algorithm, i.e., if $f(\hat{u}_1^*, \ldots, \hat{u}_n^*, y^*) \not\le (1 + \alpha) f(z) + \kappa\gamma$, we replace $(\hat{u}_{n-3}^*, \hat{u}_{n-2}^*)$ by the second best point for the corresponding mixed-integer convex minimization problem. In view of Proposition 6, the accuracy that we can guarantee on the solution is only $2\kappa\gamma$, so the criterion to output **b** must be adapted accordingly. Then we proceed with the computation of $(\hat{u}_{n-1}^*, \hat{u}_n^*)$ and so on.

It is straightforward to verify that this approach results in a finite-time algorithm for the general case. In the worst case the procedure forces us to visit all integral points in $[-B, B]^n$. However, in the course of this procedure we always have a feasible solution and a lower bound at our disposal. Once the lower bound exceeds the value of a feasible solution we can stop the procedure. It is precisely the availability of both, primal and dual information, that makes us believe that the entire algorithm is typically much faster than enumerating all the integer points in $[-B, B]^n$.

Acknowledgements We thank the anonymous referee for helpful comments. This work was partially supported by the German Science Foundation, SFB/Transregio 63 InPROMPT.

References

1. Arora, S., Kale, S.: A combinatorial, primal-dual approach to semidefinite programs [extended abstract]. In: STOC'07—Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, pp. 227–236. ACM, New York (2007). doi:10.1145/1250790.1250823
2. Bonami, P., Biegler, L., Conn, A., Cornuéjols, G., Grossmann, I., Laird, C., Lee, J., Lodi, A., Margot, F., Sawaya, N., Wächter, A.: An algorithmic framework for convex mixed integer nonlinear programs. Discrete Optim. **5**(2), 186–204 (2008). doi:10.1016/j.disopt.2006.10.011
3. Conn, A., Gould, N., Toint, P.: Trust-Region Methods. MPS/SIAM Series on Optimization. SIAM, Philadelphia (2000). doi:10.1137/1.9780898719857
4. Duran, M., Grossmann, I.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Program. **36**(3), 307–339 (1986). doi:10.1007/BF02592064
5. Eisenbrand, F., Laue, S.: A linear algorithm for integer programming in the plane. Math. Program., Ser. A **102**(2), 249–259 (2005). doi:10.1007/s10107-004-0520-0
6. Fletcher, R., Leyffer, S.: Solving mixed integer nonlinear programs by outer approximation. Math. Program., Ser. A **66**(3), 327–349 (1994). doi:10.1007/BF01581153
7. Graham, R.: An efficient algorithm for determining the convex hull of a finite planar set. Inf. Process. Lett. **1**, 132–133 (1972)

8. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Algorithms and Combinatorics: Study and Research Texts, vol. 2. Springer, Berlin (1988)
9. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms II: Advanced Theory and Bundle Methods. Grundlehren der Mathematischen Wissenschaften, vol. 306. Springer, Berlin (1993)
10. Khachiyan, L.: A polynomial algorithm in linear programming. Dokl. Akad. Nauk SSSR **244**, 1093–1096 (1979)
11. Lenstra, H. Jr.: Integer programming with a fixed number of variables. Math. Oper. Res. **8**(4), 538–548 (1983). doi:10.1287/moor.8.4.538
12. Nemirovski, A.: Efficient methods in convex programming. Lecture notes (1994). www2.isye.gatech.edu/~nemirovs/Lect_EMCO.pdf
13. Nemirovski, A., Yudin, D.: Problem Complexity and Method Efficiency in Optimization. Wiley, New York (1983)
14. Nesterov, Y.: Introductory Lectures on Convex Optimization. Applied Optimization, vol. 87. Kluwer Academic, Boston (2004)
15. Nesterov, Y., Nemirovski, A.: Interior-Point Polynomial Algorithms in Convex Programming. Studies in Applied Mathematics, vol. 13. SIAM, Philadelphia (1994). doi:10.1137/1.9781611970791
16. Rockafellar, R.: The Theory of Subgradients and Its Applications to Problems of Optimization: Convex and Non Convex Functions. R & E, vol. 1. Heldermann, Berlin (1981)
17. Viswanathan, J., Grossmann, I.: A combined penalty function and outer-approximation method for MINLP optimization. Comput. Chem. Eng. **14**(7), 769–782 (1990). doi:10.1016/0098-1354(90)87085-4

Beyond Perfection: Computational Results for Superclasses

Arnaud Pêcher and Annegret K. Wagler

"The favorite topics and results of a researcher change over time, of course. One area that I have always kept an eye on is that of perfect graphs. These graphs, introduced in the late 50s and early 60s by Claude Berge, link various mathematical disciplines in a truly unexpected way: graph theory, combinatorial optimization, semidefinite programming, polyhedral and convexity theory, and even information theory."
Martin Grötschel, Optima, June 1999

Abstract We arrived at the Zuse Institute Berlin, Annegret as doctoral student in 1995 and Arnaud as postdoc in 2001, both being already fascinated from the graph theoretical properties of perfect graphs. Through encouraging discussions with Martin, we learned about the manifold links of perfect graphs to other mathematical disciplines and the resulting algorithmic properties based on the theta number (where Martin got famous for, together with Laci Lovász and Lex Schrijver).

This made us wonder whether perfect graphs are indeed completely unique and exceptional, or whether some of the properties and concepts can be generalized to larger graph classes, with similar algorithmic consequences—the starting point of a fruitful collaboration. Here, we answer this question affirmatively and report on the recently achieved results for some superclasses of perfect graphs, all relying on the polynomial time computability of the theta number. Finally, we give some reasons that the theta number plays a unique role in this context.

A. Pêcher
Laboratoire Bordelais de Recherche Informatique (LaBRI)/INRIA Sud-Ouest, Université de Bordeaux, 351 cours de la Libération, 33405 Talence, France
e-mail: arnaud.pecher@labri.fr

A.K. Wagler (✉)
Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS)/CNRS, Université Blaise Pascal (Clermont-Ferrand II), BP 10125, 63173 Aubière Cedex, France
e-mail: annegret.wagler@univ-bpclermont.fr

1 Introduction

Berge introduced perfect graphs, motivated from Shannon's problem of finding the zero-error capacity of a discrete memoryless channel [36]. Shannon observed that this otherwise difficult problem becomes tractable for graphs $G = (V, E)$ with $\omega(G) = \chi(G)$, where $\omega(G)$ is the clique number (denoting the size of a maximum clique in G), $\chi(G)$ the chromatic number (referring to the minimal number of stable sets covering the node set V) and a clique (resp. stable set) is a set of mutually adjacent (resp. non-adjacent) nodes.

This motivated Berge [3] to say that G is a *perfect graph* if and only if

$$\omega(G') = \chi(G') \quad \text{for all induced subgraphs } G' \subseteq G. \tag{1}$$

Berge noticed that all examples of perfect graphs also have the property that

$$\alpha(G') = \overline{\chi}(G') \quad \text{for all induced subgraphs } G' \subseteq G, \tag{2}$$

where $\alpha(G)$ is the stability number (the size of a largest stable set in G) and $\overline{\chi}(G)$ the clique cover number (denoting the least number of cliques covering V).

Considering the complementary graph $\overline{G} = (V, V^2 \setminus E)$ where cliques of G turn into stable sets of $\overline{G}$ and, thus, colorings into clique covers, this means that every complement of a perfect graph G satisfies (2). This let Berge conjecture that a graph G is perfect if and only if

$$\overline{G} \text{ is a perfect graph.} \tag{3}$$

Developing the antiblocking theory of polyhedra, Fulkerson launched a massive attack to this conjecture, see [11, 12], before it was turned to the Perfect Graph Theorem by Lovász [23], who gave two short and elegant proofs. In addition, Lovász [22] characterized perfect graphs as those graphs G such that

$$\omega(G')\omega(\overline{G}') \geq |G'| \quad \text{holds for all induced subgraphs } G' \subseteq G. \tag{4}$$

Moreover, Berge observed that all chordless odd cycles C_{2k+1} with $k \geq 2$, called *odd holes*, and their complements, the *odd antiholes* $\overline{C}_{2k+1}$, satisfy $\omega(G) < \chi(G)$. This motivated Berge's famous Strong Perfect Graph Conjecture: G is perfect if and only if

$$G \text{ has no odd hole or odd antihole as induced subgraph.} \tag{5}$$

Many efforts to prove this conjecture stimulated the study of perfect graphs, but were not successful for over 40 years. Finally, Chudnovsky, Robertson, Seymour, and Thomas [6] turned this conjecture into the Strong Perfect Graph Theorem.

During the last decades, many fascinating structural properties of perfect graphs and interesting relationships to other fields of scientific inquiry have been discovered. In particular, the in general hard to compute parameters $\omega(G)$, $\chi(G)$, $\alpha(G)$ and $\overline{\chi}(G)$ can be determined in polynomial time if G is perfect by Grötschel, Lovász and Schrijver [15]. The latter result relies on characterizations of the stable set polytope of perfect graphs.

The *stable set polytope* $\mathrm{STAB}(G)$ of a graph G is defined as the convex hull of the incidence vectors of all stable sets of G. It can be alternatively represented by

$$\mathrm{STAB}(G) = \mathrm{conv}\left\{\mathbf{x} \in \{0,1\}^{|G|} : x(Q) = \sum_{i \in Q} x_i \leq 1,\ Q \subseteq G \text{ clique}\right\}$$

as a clique and a stable set have clearly at most one node in common and, thus, all clique constraints $x(Q) \leq 1$ are valid for $\mathrm{STAB}(G)$. A canonical relaxation of $\mathrm{STAB}(G)$ is, therefore, the *clique constraint stable set polytope*

$$\mathrm{QSTAB}(G) = \left\{\mathbf{x} \in \mathbb{R}_+^{|G|} : \sum_{i \in Q} x_i \leq 1,\ Q \subseteq G \text{ clique}\right\}.$$

We have $\mathrm{STAB}(G) \subseteq \mathrm{QSTAB}(G)$ for all graphs, but equality for perfect graphs only [7, 12, 26]: a graph G is perfect if and only if

$$\mathrm{STAB}(G) = \mathrm{QSTAB}(G). \tag{6}$$

Hence, one is tempted to look at the linear relaxation $\max \mathbf{w}^T\mathbf{x}$, $\mathbf{x} \in \mathrm{QSTAB}(G)$ for determining the weighted stability number $\alpha(G, \mathbf{w})$ for perfect graphs. The following chain of inequalities and equations is typical for integer/linear programming approaches to combinatorial problems:

$$\begin{aligned}
\alpha(G,\mathbf{w}) &= \max\left\{\sum_{i \in S} w_i : S \subseteq G \text{ stable}\right\}\\
&= \max\left\{\mathbf{w}^T\mathbf{x} : \mathbf{x} \in \mathrm{STAB}(G)\right\}\\
&= \max\left\{\mathbf{w}^T\mathbf{x} : x(Q) \leq 1\ \forall \text{cliques } Q \subseteq G, \mathbf{x} \geq \mathbf{0}, \mathbf{x} \in \{0,1\}^{|G|}\right\}\\
&\leq \max\left\{\mathbf{w}^T\mathbf{x} : x(Q) \leq 1\ \forall \text{cliques } Q \subseteq G, \mathbf{x} \geq \mathbf{0}\right\}\\
&= \min\left\{\sum_{Q \subseteq G} y_Q : \sum_{Q \ni i} y_Q \geq w_i\ \forall i \in G, y_Q \geq 0\ \forall \text{cliques } Q \subseteq G\right\}\\
&\leq \min\left\{\sum_{Q \subseteq G} y_Q : \sum_{Q \ni i} y_Q \geq w_i\ \forall i \in G, y_Q \geq 0, y_Q \in \mathbb{Z}_+\ \forall Q \subseteq G\right\}\\
&= \overline{\chi}(G,\mathbf{w}).
\end{aligned}$$

The inequalities come from dropping or adding integrality constraints, one of the equations is implied by linear programming duality. The last program can be interpreted as an integer programming formulation for determining the weighted clique cover number $\overline{\chi}(G, \mathbf{w})$.

It follows from the Perfect Graph Theorem that equality holds throughout the whole chain for all 0/1-vectors $\mathbf{w}$ if and only if G is perfect. This, in turn, is equivalent to saying that G is perfect if and only if

$$\text{the value } \max\left\{\mathbf{w}^T\mathbf{x} : \mathbf{x} \in \text{QSTAB}(G)\right\} \text{ is integral } \forall \mathbf{w} \in \{0, 1\}^{|G|} \tag{7}$$

and results of Fulkerson [11] and Lovász [23] imply that this is true for all $\mathbf{w} \in \mathbb{Z}^{|G|}$. This proves particularly that the constraint system defining QSTAB(G) is totally dual integral for perfect graphs G.

However, maximizing a linear objective function $\mathbf{w}^T\mathbf{x}$, $\mathbf{x} \in \text{QSTAB}(G)$ in polynomial time does not work in general [17].

For the class of perfect graphs, though, the optimization problem for QSTAB(G) (and, therefore, for STAB(G)) can be solved in polynomial time—albeit via a detour involving a geometric representation of graphs introduced by Lovász [24].

Let $G = (V, E)$ be a graph. An *orthonormal representation* of G is a sequence $(\mathbf{u}_i : i \in V)$ of $|V|$ vectors $\mathbf{u}_i \in \mathbb{R}^N$, where N is some positive integer, such that $\|\mathbf{u}_i\| = 1$ for all $i \in V$ and $\mathbf{u}_i^T\mathbf{u}_j = 0$ for all $ij \notin E$. Trivially, every graph has an orthonormal representation: just take all the vectors $\mathbf{u}_i$ mutually orthogonal in $\mathbb{R}^{|V|}$, but also less trivial orthonormal representations exist. For instance, taking an orthonormal basis $B = \{\mathbf{e}_1, \dots, \mathbf{e}_{|V|}\}$ of $\mathbb{R}^{|V|}$ and a clique Q of G, we obtain an orthonormal representation by setting $\mathbf{u}_i = \mathbf{e}_1$ for all $i \in Q$ and assigning different vectors of $B - \{\mathbf{e}_1\}$ to all the remaining nodes $j \in G - Q$ (where $\mathbf{e}_i$ denotes the i-th unit vector).

For any orthonormal representation $(\mathbf{u}_i : i \in V)$, $\mathbf{u}_i \in \mathbb{R}^N$ of G and any unit-length vector $\mathbf{c} \in \mathbb{R}^N$, the *orthonormal representation constraint (ONRC)*

$$\sum_{i \in V} \left(\mathbf{c}^T\mathbf{u}_i\right)^2 x_i \le 1$$

is valid for STAB(G) due to the following reason. For any stable set S of G, the vectors $\mathbf{u}_i$, $i \in S$ are mutually orthogonal by construction and, therefore, $\sum_{i \in S}(\mathbf{c}^T\mathbf{u}_i)^2 \le 1$ follows. We obtain

$$\sum_{i \in V} \left(\mathbf{c}^T\mathbf{u}_i\right)^2 x_i^S = \sum_{i \in S} \left(\mathbf{c}^T\mathbf{u}_i\right)^2$$

for the incidence vector $\mathbf{x}^S$ of a stable set S of G (with $x_i^S = 1$ if $i \in S$ and $x_i^S = 0$ otherwise) yielding the validity of the ONRCs for STAB(G).

Moreover, taking an orthonormal representation associated with a clique Q, then the corresponding orthonormal representation constraint for $\mathbf{c} = \mathbf{e}_1$ is just the clique constraint associated with Q (by $\mathbf{c}^T\mathbf{u}_i = 1$ for $i \in Q$ and $\mathbf{c}^T\mathbf{u}_j = 0$ otherwise). Hence, every clique constraint is a special orthonormal representation constraint.

For any graph $G = (V, E)$, the set

$$\text{TH}(G) = \left\{\mathbf{x} \in \mathbb{R}_+^V : \mathbf{x} \text{ satisfies all ONRCs}\right\}$$

is the intersection of infinitely many half-spaces (since G admits infinitely many orthonormal representations), so TH(G) is a convex set but no polytope in general. The above remarks imply

$$\text{STAB}(G) \subseteq \text{TH}(G) \subseteq \text{QSTAB}(G)$$

and all three convex sets coincide if and only if G is perfect. More precisely, Grötschel, Lovász and Schrijver [15] showed: G is perfect if and only if

$$\text{STAB}(G) = \text{TH}(G) \tag{8}$$

and if and only if

$$\text{TH}(G) = \text{QSTAB}(G). \tag{9}$$

This result is particularly remarkable since it states that a graph G is perfect if and only if

$$\text{the convex set } \text{TH}(G) \text{ is a polytope.} \tag{10}$$

The key property of TH(G) for linear programming was again established by Grötschel, Lovász, and Schrijver [15]: If $\mathbf{w} \in \mathbb{R}_+^V$ is a vector of node weights, the optimization problem (with infinitely many linear constraints) $\max \mathbf{w}^T\mathbf{x}$, $\mathbf{x} \in \text{TH}(G)$ can be solved in polynomial time for any graph G. This deep result rests on the fact that the value

$$\vartheta(G, \mathbf{w}) = \max\left\{\mathbf{w}^T\mathbf{x} : \mathbf{x} \in \text{TH}(G)\right\}$$

can be characterized in many equivalent ways, e.g., as the maximum

- eigenvalue of a certain set of symmetric matrices,
- value of some function involving orthonormal representations,
- value of a semidefinite program,

see [17] for details. As we have $\alpha(G, \mathbf{w}) = \vartheta(G, \mathbf{w})$ for all *perfect* graphs G by (8), this finally implies that the stable set problem can be solved in polynomial time for perfect graphs.

Therefore, the clique cover number $\overline{\chi}(G) = \alpha(G)$, the chromatic number $\chi(G) = \overline{\chi}(\overline{G})$, and the clique number $\omega(G) = \alpha(\overline{G})$ can be computed in polynomial time for perfect graphs G, even in the weighted versions.

To summarize the above presented results, perfect graphs truly deserve their name since they have interesting graph-theoretical properties and behave nicely from an algorithmic point of view. In addition, perfect graphs can be characterized in terms of many different concepts (see the above conditions (1)–(10) which form a part of what Martin Grötschel called his "favorite theorem" in [14]), thereby establishing links to

- polyhedral theory (G is perfect if and only if certain polyhedra are identical);
- integer programming (a graph G is perfect if and only if certain linear programs have integral objective values);
- semidefinite programming (a graph is perfect if and only if the feasible region of a certain semidefinite program is a polytope);

which indeed reflects the importance of perfect graphs in many different fields of scientific inquiry.

2 Beyond Perfection

The above considerations show that perfect graphs are a class with an extraordinarily rich structure. Unfortunately, most graphs are imperfect and do not admit such nice properties. Thus, it is natural to ask which imperfect graphs are close to perfection in some sense. Our aim is to generalize the underlying concepts of crucial properties of perfect graphs and to study possible algorithmic consequences for the resulting superclasses of perfect graphs.

We arrive at a natural superclass of perfect graphs if we require local perfection: we call a graph $G = (V, E)$ *neighborhood-perfect* if, for every node v of G, its closed neighborhood $N[v]$ induces a perfect subgraph of G (where $N[v]$ contains v together with all its neighbors $w \in N(v) = \{w \in V : vw \in E\}$).

Due to the Strong Perfect Graph Theorem [6], being neighborhood-perfect is equivalent to having neither *odd wheels* $C_{2k+1} * v$ nor *odd antiwheels* $\overline{C}_{2k+1} * v$, i.e., no odd (anti)holes completely joined to a node v.

Proposition 1 *The class of neighborhood-perfect graphs equals the class of odd wheel- and odd antiwheel-free graphs.*

Proof On the one hand, every neighborhood-perfect graph does not contain an odd wheel $C_{2k+1} * v$ or odd antiwheel $\overline{C}_{2k+1} * v$, as otherwise $N[v]$ would be not perfect. On the other hand, every odd wheel- and odd antiwheel-free graph G is neighborhood-perfect: for all nodes v, $N(v)$ cannot contain an odd hole or odd antihole and, therefore, induces a perfect subgraph of G by the Strong Perfect Graph Theorem [6]; that no odd (anti)hole contains a universal node implies that also $N(v) \cup \{v\} = N[v]$ induces a perfect subgraph of G. □

It turns out that neighborhood-perfect graphs contain two well-known superclasses of perfect graphs, circular-perfect graphs and rank-perfect graphs, obtained by generalizing coloring and polyhedral properties of perfect graphs, respectively.

Circular-perfect graphs are introduced by Zhu [46] as superclass of perfect graphs on the base of the following more general coloring concept due to Vince [40]. For positive integers $k \geq 2d$, a (k, d)*-coloring* of a graph $G = (V, E)$ is a mapping $f : V \to \{0, \ldots, k-1\}$ such that for each edge uv of G, $d \leq |f(u) - f(v)| \leq k - d$ holds. The *circular-chromatic number* is

$$\chi_c(G) = \min\left\{\frac{k}{d} : G \text{ has a } (k, d)\text{-coloring}\right\}$$

and satisfies $\chi_c(G) \leq \chi(G)$, as a $(k, 1)$-circular coloring is a usual coloring with k colors. In addition, it is known by [40] that $\chi(G) = \lceil \chi_c(G) \rceil$ for any graph G.

As generalizations of cliques, *circular-cliques* $K_{k/d}$ with $k \geq 2d$ nodes $0, \ldots, k-1$ and edges ij if and only if $d \leq |i - j| \leq k - d$ are considered. In fact, circular-cliques include all cliques $K_{k/1}$, antiholes $K_{k/2}$, and odd holes $K_{2t+1/t}$. Note that they are also known as *antiwebs* in the literature [37, 43].

The *circular-clique number* is

$$\omega_c(G) = \max\left\{\frac{k}{d} : K_{k/d} \text{ is an induced subgraph of } G, \gcd(k, d) = 1\right\}$$

and $\omega(G) \le \omega_c(G)$ holds as $K_{k/1}$ is a usual clique. It is also known from [46] that $\omega(G) = \lfloor \omega_c(G) \rfloor$ for any graph G.

In addition, $\omega_c(G)$ is always a lower bound for $\chi_c(G)$ by [4] which implies

$$\omega(G) = \lfloor \omega_c(G) \rfloor \le \omega_c(G) \le \chi_c(G) \le \lceil \chi_c(G) \rceil = \chi(G). \tag{11}$$

A graph G is *circular-perfect* [46] if each induced subgraph $G' \subseteq G$ satisfies $\chi_c(G') = \omega_c(G')$. By definition, every perfect graph is circular-perfect as we have everywhere equality in (11). All circular-cliques are circular-perfect as well by [46]. Two further classes of circular-perfect graphs are found by Bang, Jensen and Huang [2] and Pêcher and Wagler [28]. So circular-perfect graphs form a proper superclass of the class of perfect graphs. On the other hand, circular-perfect graphs are a subclass of neighborhood-perfect graphs by Zhu [46], since every circular-perfect graph G satisfies the property that $N[v]$ induces a perfect subgraph for every node v. This can also be inferred from the fact that odd wheels and odd antiwheels are minimally not circular-perfect, see [28, 45].

Rank-perfect graphs are introduced in [42] in order to obtain a superclass of perfect graphs by relaxing their polyhedral characterization (6). Accordingly, we have $\text{STAB}(G) \subset \text{QSTAB}(G)$ for all imperfect graphs G, and additional constraints are needed to obtain all facets of $\text{STAB}(G)$. As natural generalization of the clique constraints describing $\text{QSTAB}(G)$, we consider rank constraints

$$x(G') = \sum_{i \in G'} x_i \le \alpha(G')$$

associated with arbitrary induced subgraphs $G' \subseteq G$. By the choice of the right hand side $\alpha(G')$, rank constraints are obviously valid for $\text{STAB}(G)$ such that the *rank constraint stable set polytope*

$$\text{RSTAB}(G) = \left\{\mathbf{x} \in \mathbb{R}_+^{|G|} : \sum_{i \in G'} x_i \le \alpha(G'), G' \subseteq G\right\}$$

is a further linear relaxation of $\text{STAB}(G)$. As clique constraints are special rank constraints (namely exactly those with $\alpha(G') = 1$), we immediately obtain

$$\text{STAB}(G) \subseteq \text{RSTAB}(G) \subseteq \text{QSTAB}(G).$$

A graph G is *rank-perfect* by [42] if and only if $\text{STAB}(G) = \text{RSTAB}(G)$ holds.

By definition, rank-perfect graphs include all perfect graphs (where rank constraints associated with cliques suffice). In general, by restricting the facet set to rank constraints associated with certain subgraphs only, several well-known graph classes are defined, e.g., t-perfect [7] and h-perfect graphs [17] (where rank constraints associated with edges and odd cycles resp. cliques and odd holes are required). As further generalization, a graph G is called *a-perfect* if and only if rank constraints associated with circular-cliques suffice to describe $\text{STAB}(G)$. All circular-cliques are a-perfect by [43] and a further class of a-perfect graphs was found in [44]. Hence, rank-perfect graphs form a proper superclass of perfect graphs. On the other hand, rank-perfect graphs are indeed a subclass of neighborhood-perfect graphs. The proof is elementary and does not make use of the Strong Perfect Graph Theorem.

Proposition 2 *Rank-perfect graphs are neighborhood-perfect.*

Proof Consider a rank-perfect graph G and assume to the contrary that there is a node v in G such that $N[v]$ is imperfect. Then $N[v]$ contains a minimal imperfect subgraph G' (i.e., G' is imperfect, but all its proper induced subgraphs are perfect). As no minimal imperfect graph has a universal node, $v \notin V(G')$ follows and, therefore, v is completely joined to G'.

By Padberg [26], every minimal imperfect graph G' induces the full rank facet $x(G') \leq \alpha(G')$ of $\text{STAB}(G')$, and the complete join $G' * v$ gives rise to a facet

$$\alpha\big(G'\big)x_v + x\big(G'\big) \leq \alpha\big(G'\big)$$

of $\text{STAB}(G' * v)$. As any minimal imperfect graph G' has stability number $\alpha(G') \geq 2$ (only cliques have stability number 1), the above inequality is a non-rank constraint. Hence, G contains a non-rank facet producing subgraph $G' * v$, and any lifting of the above inequality to a facet of $\text{STAB}(G)$ remains non-rank. So, no node of a rank-perfect graph contains a minimal imperfect graph in its (closed) neighborhood. □

A further well-known subclass of neighborhood-perfect graphs, the *quasi-line graphs*, are defined as those graphs where the neighborhood of any node can be split into two cliques. Quasi-line graphs are neighborhood-perfect by definition, but not a superclass of perfect graphs (since no quasi-line graph contains a claw, that is a stable set of size three completely joined to a node, but the claw is perfect).

Remark 1 Note that all three studied subclasses of neighborhood-perfect graphs are pairwise incomparable.

Circular-perfect graphs contain a family of graphs (obtained from a K_4 by subdividing all edges incident to one of its nodes) which are not rank-perfect [8] and not quasi-line (since they contain a claw); see Fig. 1a for the smallest such graph.

Rank-perfect graphs contain a family of t-perfect graphs (obtained from certain cyclic concatenations of odd holes, called flowers) which are minimally not circular-perfect by [20, 33] and not quasi-line (since they contain claws); see Fig. 1b for the smallest such graph.

Quasi-line graphs contain all complements of circular-cliques (called webs) which are almost all not rank-perfect by [27] and contain a family $\overline{K}_{3p+1,4}$ of minimally not circular-perfect graphs by [28]; the smallest not rank-perfect web is $\overline{K}_{25,6}$

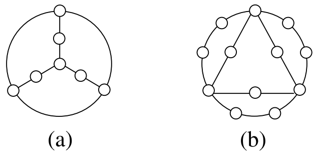


Fig. 1 **a** A circular-perfect graph which is neither rank-perfect nor quasi-line. **b** A t-perfect graph which is neither circular-perfect nor quasi-line

which contains $\overline{K}_{16,4}$ and $\overline{K}_{37,4}$ is the smallest web which is not rank-perfect and minimally not circular-perfect.

It is shown in [29] that the strong optimization problem over QSTAB(G) can be solved in polynomial time for G neighborhood-perfect. This implies for a circular-perfect graph G with neighborhood-perfect complement that $\omega(G)$, $\omega_c(G)$, $\chi_c(G)$ and $\chi(G)$ can be computed in polynomial time, see [29]. This applies to all circular-perfect graphs where the complement is circular-perfect, rank-perfect or quasi-line.

Moreover, in [34], it is shown that for claw-free circular-perfect graphs $\omega(G)$, $\omega_c(G)$, $\chi_c(G)$ and $\chi(G)$ can be computed in polynomial time.

We generalize these results by techniques relying on the polynomial time computability of the theta number.

We first address the problem of determining the clique number for neighborhood-perfect graphs G via computing $\vartheta(\overline{G})$. Next, we discuss how the chromatic number can be obtained for circular-perfect graphs G via computing $\vartheta(\overline{G})$. Finally, we present both a closed formula for computing the theta number of circular-cliques and consequences for computing circular-clique and -chromatic number for circular-perfect graphs.

2.1 On Computing the Clique Number

In this subsection, we extend the polynomial time computability of the weighted clique number for perfect graphs to the larger class of neighborhood-perfect graphs.

For that, we take advantage of the fact that every maximal clique Q of a graph G is contained in the closed neighborhood $N[v]$ of some node v. We denote by G_v the subgraph of G induced by $N[v]$. For every neighborhood-perfect graph G, all these subgraphs G_v are perfect by definition, so that their clique number equals the theta-value of their complement by [15]. This implies:

Corollary 1 *For any neighborhood-perfect graph $G = (V, E)$ and any weight vector $\mathbf{w} \in \mathbb{Q}_+^{|V|}$, the weighted clique number $\omega(G, \mathbf{w})$ is computable in polynomial time by*

$$\omega(G, \mathbf{w}) = \max\left\{\vartheta(\overline{G_v}, \mathbf{w}) : G_v = N[v], v \in V\right\}.$$

This result applies to all neighborhood-perfect graphs, including rank-perfect and circular-perfect graphs as two superclasses of perfect graphs, as well as quasi-line graphs. Note that for the latter, only the polynomial time computability of the weighted stability number has been widely studied so far, see e.g. [10, 25] and references therein.

Although $\omega(G, \mathbf{w})$ can be determined in polynomial time that way, it requires as many computations of $\vartheta(\overline{G_v}, \mathbf{w})$ as G has nodes. Next, we consider the unweighted clique number $\omega(G)$ and exhibit an easier way to obtain it for circular-perfect and a-perfect graphs by a single computation of $\vartheta(\overline{G})$ based on the following idea.

For all graphs G, we have $\omega(G) \le \vartheta(\overline{G})$ by $\mathrm{STAB}(\overline{G}) \subseteq \mathrm{TH}(\overline{G})$, but the gap between the two parameters can be large in general. If however, we can identify graphs G where the two parameters are sufficiently close such that

$$\omega(G) = \left\lfloor \vartheta(\overline{G}) \right\rfloor$$

holds, the polynomial time computability of their clique number would follow.

The circular-clique number $\omega_c(G)$ satisfies the desired property of being close to the clique number, since it is known from [46] that $\omega(G) = \lfloor \omega_c(G) \rfloor$ holds for any graph G. Determining $\omega_c(G)$ is clearly NP-hard in general and can be done by optimizing over a suitable polytope. For that, we introduced in [29] the *circular-clique polytope*

$$\mathrm{CLI}_c(G) = \mathrm{conv}\left\{1/\alpha(K)\mathbf{x}^K : K \subseteq G \text{ prime circular-clique}\right\}$$

where $\mathbf{x}^K$ denotes the incidence vector of $K \subseteq G$, and a circular-clique $K_{k/d}$ is prime if $\gcd(k, d) = 1$ holds. It is immediate to see that $\omega_c(G) = \max \mathbb{1}^T \mathbf{x}$, $\mathbf{x} \in \mathrm{CLI}_c(G)$. In [31, 33], we examined further properties of $\mathrm{CLI}_c(G)$ and showed that

$$\mathrm{STAB}(\overline{G}) \subseteq \mathrm{CLI}_c(G) \subseteq \mathrm{QSTAB}(\overline{G})$$

holds for any graph G which implies

$$\omega(G) \le \omega_c(G) \le \omega_f(G) \tag{12}$$

where $\omega_f(G) = \max \mathbb{1}^T \mathbf{x}$, $\mathbf{x} \in \mathrm{QSTAB}(\overline{G})$ is the fractional clique number.

Hence, both parameters $\vartheta(\overline{G})$ and $\omega_c(G)$ are sandwiched between $\omega(G)$ and $\omega_f(G)$, but are incomparable in general. For instance, consider the 5-hole $C_5 = K_{5/2}$ and the 5-wheel $W_5 = K_{5/2} * K_1$, then we have

$$\sqrt{5} = \vartheta(\overline{C}_5) < \omega_c(C_5) = \frac{5}{2} \quad \text{but } 3 = \omega_c(W_5) < \vartheta(\overline{W}_5) = \sqrt{5} + 1.$$

However, satisfying $\omega_c(G) = \omega_f(G)$ is a sufficient condition for a graph G to have

$$\omega(G) \le \vartheta(\overline{G}) \le \omega_c(G) = \omega_f(G) \tag{13}$$

which implies $\omega(G) = \lfloor \vartheta(\overline{G}) \rfloor$ by $\omega(G) = \lfloor \omega_c(G) \rfloor$.

In the next two paragraphs, we first provide the proof for the inequality chain (12) and, combining it with further facts, conclude that $\omega(G) = \lfloor \vartheta(\overline{G}) \rfloor$ holds for all circular-perfect graphs. Then we present the characterization of a-perfect graphs from [31, 33] as the graphs G with $\mathrm{CLI}_c(G) = \mathrm{QSTAB}(\overline{G})$ and conclude that $\omega(G) = \lfloor \vartheta(\overline{G}) \rfloor$ also holds for all a-perfect graphs.

2.1.1 About the Circular-Clique Polytope and the Clique Number of Circular-Perfect Graphs

In order to show that the circular-clique polytope $\mathrm{CLI}_c(G)$ is sandwiched between $\mathrm{STAB}(\overline{G})$ and $\mathrm{QSTAB}(\overline{G})$, we use the general relation of facets and extreme points of the latter two polytopes. This proof gives an understanding why the circular-clique polytope has to be defined with respect to *prime* circular-cliques only. Note that we consider all facets of the stable set polytope in a normalized form, scaled to have right hand side equal to one.

Theorem 1 *Consider a non-empty subgraph $G' \subseteq G$ and a weight vector $\mathbf{a} \in [0,1]^{|G|}$ with $0 < a_i \leq 1$ for $i \in G'$, $a_i = 0$ otherwise. The vector $\mathbf{a}$ is an extreme point of $\mathrm{QSTAB}(G)$ if and only if $\mathbf{a}^T\mathbf{x} \leq 1$ is a facet of $\mathrm{STAB}(\overline{G}')$.*

Proof If. Suppose that $\overline{G}$ contains a subgraph $\overline{G}'$ such that $\mathbf{a}^T\mathbf{x} \leq 1$ is a facet of $\mathrm{STAB}(\overline{G}')$ with $0 < a_i \leq 1$ for $i \in \overline{G}'$. Then there exist $n' = |\overline{G}'|$ stable sets $S'_1, \ldots, S'_{n'}$ of $\overline{G}'$ such that we have for their incidence vectors $\mathbf{a}^T\mathbf{x}^{S'_i} = 1$ for $1 \leq i \leq n'$ and $\mathbf{x}^{S'_1}, \ldots, \mathbf{x}^{S'_{n'}}$ are linearly independent.

These stable sets clearly correspond to n' cliques $Q'_1, \ldots, Q'_{n'}$ of G'. For any such clique Q'_i, choose a maximal clique $Q_i \subseteq G$ with $Q_i \supseteq Q'_i$. The vector $\mathbf{x}'$ with $x'_i = a_i$ for $i \in G'$ and $x'_i = 0$ otherwise is a point of $\mathrm{QSTAB}(G)$ as for every maximal clique Q of G, $Q \cap G'$ is a stable set of $\overline{G}'$ and so $\sum_{i \in Q} x'_i \leq \sum_{i \in Q \cap G'} x'_i \leq 1$. Furthermore $\mathbf{x}'$ satisfies the n' clique constraints associated with the maximal cliques $Q_1, \ldots, Q_{n'}$ at equality, as

$$\mathbf{x}'(Q_i) = \sum_{j \in Q'_i \subseteq Q_i} a_j = \mathbf{a}^T\mathbf{x}^{Q'_i} = \mathbf{a}^T\mathbf{x}^{S'_i} = 1$$

holds by the choice of G'. Furthermore, $\mathbf{x}'$ satisfies the $n - n' = |G \setminus G'|$ non-negativity constraints $-x'_j = 0\ \forall j \notin G'$ with equality. Hence, $\mathbf{x}'$ lies in the intersection of $n = |G|$ facets of $\mathrm{QSTAB}(G)$. In order to show that $\mathbf{x}'$ is an extreme point it remains to ensure that these facets are linearly independent. For that, construct an $(n \times n)$-matrix A as follows: Let the first n' columns of A correspond to nodes in G' and the last $n - n'$ columns to nodes in $G \setminus G'$. Choose further the incidence vectors of the cliques $Q_1, \ldots, Q_{n'}$ as first n' rows and the incidence vectors of the non-negativity constraints $-x'_j = 0\ \forall j \notin G'$ as last $n - n'$ rows, i.e.

$$A = \left(\begin{array}{c|c} A_1 & A_2 \\ \hline 0 & \mathrm{Id} \end{array}\right). \tag{14}$$

As the submatrix A_1 corresponds to the independent cliques $Q'_1, \ldots, Q'_{n'}$ of G', the whole matrix A is invertible due to its block structure. Thus, $\mathbf{x}'$ is indeed an extreme point of $\mathrm{QSTAB}(G)$.

Only if. Suppose conversely that $\mathbf{a}$ with $0 < a_i \le 1$ for $i \in \overline{G}'$ and $a_i = 0$ otherwise is an extreme point of QSTAB(G). Then $\mathbf{a}$ satisfies n linearly independent facets of QSTAB(G) with equality. Among them are clearly the $n - n'$ nonnegativity constraints $-a_j = 0\ \forall j \notin G'$. As QSTAB$(G)$ has only two types of facets, $\mathbf{a}$ satisfies also n' maximal clique facets with equality, say the clique constraints associated with the maximal cliques $Q_1, \dots, Q_{n'}$ of G. Let $Q_i' = Q_i \cap G'$, then

$$\mathbf{a}(Q_i) = \sum_{j \in Q_i} a_j = \sum_{j \in Q_i' \subseteq Q_i} a_j = \mathbf{a}^T \mathbf{x}^{Q_i'} = 1$$

follows by the choice of the vector $\mathbf{a}$. Clearly, the cliques $Q_1', \dots, Q_{n'}'$ of G' correspond to stable sets $S_1', \dots, S_{n'}'$ of $\overline{G}'$ and $\mathbf{a}^T \mathbf{x}^{S_i'} = 1$ holds for $1 \le i \le n'$.

By construction, $\mathbf{a}^T \mathbf{x} \le 1$ is valid for STAB$(\overline{G}')$. In order to show that it defines a facet, it remains to verify that $\mathbf{x}^{S_1'}, \dots, \mathbf{x}^{S_{n'}'}$ are linearly independent. For that, construct an $(n \times n)$-matrix A as above, choosing the nodes in G' and in $G \setminus G'$ as first n' and last $n - n'$ columns, respectively, the incidence vectors of the cliques $Q_1, \dots, Q_{n'}$ as first n' and the unit vectors corresponding to $-a_j = 0\ \forall j \notin G'$ as last $n - n'$ rows, see again (14).

As $\mathbf{a}$ is an extreme point, the matrix A is invertible. In order to show invertibility for the submatrix A_1, we subtract, for each 1-entry in A_2, the corresponding unit vector in (0, Id). That way, we turn A_2 into a matrix with 0-entries only but maintain all entries in A_1.

This shows that the rows of A_1 are linearly independent and, therefore, the incidence vectors of the cliques $Q_1', \dots, Q_{n'}'$ of G' respectively of the corresponding stable sets $S_1', \dots, S_{n'}'$ in $\overline{G}'$. Therefore, $\mathbf{a}^T \mathbf{x} \le 1$ is indeed a facet of STAB$(\overline{G}')$. □

Hence, Theorem 1 establishes a 1–1-correspondence between extreme points of QSTAB(G) and facet-inducing *sub*graphs of $\overline{G}$. On the other hand, the circular-clique polytope CLI$_c(G)$ is the convex hull of the normalized incidence vectors of all prime circular-cliques in G. Together, this shows that CLI$_c(G)$ is sandwiched between the stable set polytope and the clique constraint stable set polytope of the complement and we obtain the relations for the corresponding graph parameters:

Lemma 1 *For all graphs G, we have* STAB$(\overline{G}) \subseteq$ CLI$_c(G) \subseteq$ QSTAB$(\overline{G})$ *and, thus, $\omega(G) \le \omega_c(G) \le \omega_f(G)$ holds.*

Proof By definition, STAB$(\overline{G})$ has only (integral) extreme points corresponding to cliques of G and, by Theorem 1, QSTAB$(\overline{G})$ has extreme points corresponding to all facet-inducing subgraphs of G. As, by definition, the extreme points of CLI$_c(G)$ correspond to prime circular-cliques of G only, STAB$(\overline{G}) \subseteq$ CLI$_c(G) \subseteq$ QSTAB$(\overline{G})$ follows. Consequently, $\omega(G) \le \omega_c(G) \le \omega_f(G)$ holds. □

The relations of usual, circular and fractional chromatic number are studied by Deuber and Zhu [9] who established $\chi_f(G) \le \chi_c(G) \le \chi(G)$ for any G. Coupling

Lemma 1 with this result by LP-duality implies for all graphs G

$$\omega(G) \le \omega_c(G) \le \omega_f(G) = \chi_f(G) \le \chi_c(G) \le \chi(G). \tag{15}$$

Combining (15) with the definition of circular-perfect graphs verifies (13) for this class by $\omega_c(G) = \chi_c(G)$ and we conclude:

Corollary 2 *If G is circular-perfect, then $\omega(G) = \lfloor \vartheta(\overline{G}) \rfloor$.*

2.1.2 Computing the Clique Number for a-Perfect Graphs

The key to compute the clique number for a-perfect graphs is to characterize them as the graphs G with $\mathrm{CLI}_c(G) = \mathrm{QSTAB}(\overline{G})$.

The rank constraint $\mathbf{x}(K_{k/d}) \le \alpha(K_{k/d}) = d$ associated with a circular-clique $K_{k/d} \subseteq G$ is clearly valid for $\mathrm{STAB}(G)$ and defines a facet if and only if $K_{k/d}$ is prime [39]. For a graph $G = (V, E)$, let

$$\mathrm{ASTAB}(G) = \left\{ \mathbf{x} \in \mathbb{R}_+^{|G|} : \mathbf{x}(K_{k/d}) \le d,\, K_{k/d} \subseteq G, \gcd(k, d) = 1 \right\}.$$

By construction, we have $\mathrm{STAB}(G) \subseteq \mathrm{RSTAB}(G) \subseteq \mathrm{ASTAB}(G) \subseteq \mathrm{QSTAB}(G)$ in general, and a graph G is a-perfect if and only if $\mathrm{STAB}(G) = \mathrm{ASTAB}(G)$ holds.

To establish that the circular-clique polytope $\mathrm{CLI}_c(G)$ coincides with $\mathrm{QSTAB}(\overline{G})$ for a-perfect graphs, we show that $\mathrm{ASTAB}(G)$ and $\mathrm{CLI}_c(G)$ are antiblockers.

A polyhedron $P \subset \mathbb{R}_+^n$ is of *antiblocking type* if $\mathbf{x} \in P$ and $\mathbf{0} \le \mathbf{x}' \le \mathbf{x}$ implies $\mathbf{x}' \in P$. For any polyhedron P, its *antiblocker* $\mathrm{abl}(P)$ is defined by

$$\mathrm{abl}(P) := \left\{ \mathbf{y} \in \mathbb{R}_+^n : \mathbf{y}^T \mathbf{x} \le 1 \ \forall \mathbf{x} \in P \right\}.$$

Fulkerson [11, 12] showed that if P is of antiblocking type, then $\mathrm{abl}(P)$ is of antiblocking type as well and $\mathrm{abl}(\mathrm{abl}(P)) = P$.

Note that this concept applies to all convex sets, e.g. to the Theta-body $\mathrm{TH}(G)$. A famous result of Grötschel, Lovász and Schrijver [17] is that $\mathrm{TH}(G)$ is the antiblocker of $\mathrm{TH}(\overline{G})$. The most prominent pair of antiblocking polytopes is the stable set polytope $\mathrm{STAB}(G)$ of a graph G and the clique constraint stable set polytope $\mathrm{QSTAB}(\overline{G})$ of its complement $\overline{G}$ by [11, 12]. It turns out that also $\mathrm{ASTAB}(G)$ and $\mathrm{CLI}_c(G)$ form an antiblocking pair. Note that also the empty set is considered to be a circular-clique; thus $\mathrm{CLI}_c(G)$ contains the origin and is a polytope of antiblocking type. By construction, $\mathrm{ASTAB}(G)$ is clearly of antiblocking type.

Theorem 2 *For any graph G, we have* $\mathrm{CLI}_c(G) = \mathrm{abl}(\mathrm{ASTAB}(G))$.

Proof Let $\mathbf{x}$ be any point of $\mathrm{abl}(\mathrm{CLI}_c(G))$ and H any induced circular-clique of G. By definition of $\mathrm{CLI}_c(G)$, the point $1/\alpha(H)\mathbf{x}^H$ belongs to $\mathrm{CLI}_c(G)$. This shows that $\mathbf{x}^T(1/\alpha(H)\mathbf{x}^H) \le 1$ holds, thus $\sum_{i \in H} x_i \le \alpha(H)$ follows, and $\mathbf{x}$ is a point of $\mathrm{ASTAB}(G)$. This implies $\mathrm{abl}(\mathrm{CLI}_c(G)) \subseteq \mathrm{ASTAB}(G)$.

Conversely, let $\mathbf{x}$ be an arbitrary point of $\text{ASTAB}(G)$. If $\mathbf{x}$ does not belong to $\text{abl}(\text{CLI}_c(G))$, then there is a point $\mathbf{y}$ in $\text{CLI}_c(G)$ such that $\mathbf{x}^T\mathbf{y} > 1$. For some k, let $H_1, \ldots, H_k$ be induced circular-cliques of G, and $\lambda_1, \ldots, \lambda_k$ be k positive multipliers such that $\mathbf{y} = \sum_{i=1,\ldots,k} \lambda_i/\alpha(H_i)\mathbf{x}^{H_i}$ and $\sum_{i=1,\ldots,k} \lambda_i = 1$ holds. Then we obtain $\mathbf{x}^T\mathbf{y} = \sum_{i=1,\ldots,k} \lambda_i/\alpha(H_i)\mathbf{x}^T\mathbf{x}^{H_i}$. From $\mathbf{x}^T\mathbf{x}^{H_i} \leq \alpha(H_i)$ for every $i = 1, \ldots, k$ follows $\mathbf{x}^T\mathbf{y} \leq 1$, a contradiction. Hence, every point of $\text{ASTAB}(G)$ belongs to $\text{abl}(\text{CLI}_c(G))$ which shows $\text{ASTAB}(G) \subseteq \text{abl}(\text{CLI}_c(G))$.

Together, we have $\text{ASTAB}(G) = \text{abl}(\text{CLI}_c(G))$, and $\text{abl}(\text{ASTAB}(G)) = \text{CLI}_c(G)$ implies the assertion. □

The previous theorem enables us to show:

Theorem 3 *We have* $\text{CLI}_c(G) = \text{QSTAB}(\overline{G})$ *if and only if G is a-perfect.*

Proof By definition, a graph G is a-perfect if and only if $\text{STAB}(G) = \text{ASTAB}(G)$. As it is known from [12] that the antiblocker of $\text{STAB}(G)$ equals $\text{QSTAB}(\overline{G})$, we infer that $\text{CLI}_c(G) = \text{QSTAB}(\overline{G})$ holds if and only if G is a-perfect, since the antiblocker of $\text{ASTAB}(G)$ equals $\text{CLI}_c(G)$ for any graph by Theorem 2. □

Consequently, we have $\omega_c(G) = \omega_f(G)$ for a-perfect graphs and conclude:

Corollary 3 *If G is a-perfect, then* $\omega(G) = \lfloor \vartheta(\overline{G}) \rfloor$.

As $\vartheta(G)$ can be computed in polynomial time for any graph G by [15] with arbitrary precision $\varepsilon > 0$, provided ε has polynomial space encoding, we have:

Corollary 4 *For any a-perfect graph G, we can compute its clique number $\omega(G)$ in polynomial time.*

Proof Let n be the number of nodes of G. There is a circular-clique $K_{k/d} \subseteq G$ with $\omega_c(G) = k/d$ and $k, d \leq n$. From $\omega(G) \leq \omega_c(G) < \omega(G) + 1$ and $k, d \leq n$, we infer $\omega_c(G) \leq \omega(G) + 1 - \frac{1}{n}$. Hence, since G is a-perfect, we have

$$\omega(G) \leq \vartheta(\overline{G}) \leq \omega_f(G) \leq \omega(G) + 1 - \frac{1}{n}.$$

Let $\varepsilon = \frac{1}{2n}$ and $\overline{\vartheta}$ denote the output of a semi-definite program computing $\vartheta(\overline{G})$ with precision ε. Thus $\omega(G) - \frac{1}{2n} \leq \overline{\vartheta} \leq \omega(G) + 1 - \frac{1}{2n}$. Therefore $\omega(G)$ is either the closest integer to $\overline{\vartheta}$ if $\overline{\vartheta}$ is within $\frac{1}{2n}$ of an integer, or equal to $\lfloor \overline{\vartheta} \rfloor$. □

2.2 *On Computing the Chromatic Number*

Our aim is to address the question whether the polynomial time computability of the chromatic number can be extended to graphs with similarly nice coloring properties as perfect graphs. We consider graph classes $\mathcal{G}$ whose members G satisfy the best possible bound on the chromatic number for graph classes containing imperfect graphs, namely

$$\omega(G) \leq \chi(G) \leq \omega(G) + 1. \tag{16}$$

For such graphs $G \in \mathcal{G}$, $\omega(G) \leq \vartheta(\overline{G}) \leq \chi(G) \leq \omega(G) + 1$ holds, hence $\chi(G) = \lceil \vartheta(\overline{G}) \rceil$ follows for $\vartheta(\overline{G}) \notin \mathbb{Z}$. However, if $\vartheta(\overline{G}) \in \mathbb{Z}$ holds, three cases

$$\omega(G) < \lfloor \vartheta(\overline{G}) \rfloor = \chi(G)$$
$$\omega(G) = \lfloor \vartheta(\overline{G}) \rfloor < \chi(G)$$
$$\omega(G) = \lfloor \vartheta(\overline{G}) \rfloor = \chi(G)$$

are possible and the difficulty is to decide which of them indeed occurs in order to compute $\chi(G)$ in polynomial time for all graphs $G \in \mathcal{G}$.

For instance, the Kneser graph $G_{6,2}$ has one node for each 2-element subset of $\{1, \dots, 6\}$ and an edge between two nodes if the corresponding subsets are disjoint. The graph $G_{6,2}$ satisfies (16), but $\vartheta(\overline{G_{6,2}})$ is integral and we have

$$\omega(G_{6,2}) = 3 = \vartheta(\overline{G_{6,2}}) < \chi(G_{6,2}) = 4.$$

In the next paragraph, we present the results from [30, 32] where this method is applied to a superclass of perfect graphs satisfying (16), the circular-perfect graphs. Afterward, we briefly discuss whether similar techniques can be used to compute the chromatic number for other graph classes in polynomial time as well.

2.2.1 Computing the Chromatic Number of Circular-Perfect Graphs

For applying the aforementioned approach to compute $\chi(G)$ for circular-perfect graphs G, we note that such graphs indeed satisfy the bound (16) for their chromatic number. For that, recall that for any graph inequality chain (11) holds which implies for all circular-perfect graphs G

$$\omega(G) = \lfloor \omega_c(G) \rfloor \leq \omega_c(G) = \chi_c(G) \leq \lceil \chi_c(G) \rceil = \chi(G)$$

and, thus, clearly (16) follows. So circular-perfect graphs admit almost as nice coloring properties as perfect graphs.

Since a circular-perfect graph is $(\omega(G) + 1)$-colorable as G satisfies (16), computing $\chi(G)$ amounts to testing whether or not $\chi(G) = \omega(G)$. By Corollary 2, $\chi(G)$ can be approximated in polynomial time with a gap of at most 1. If $\vartheta(\overline{G}) \notin \mathbb{Z}$ holds, then $\chi(G) = \lceil \vartheta(\overline{G}) \rceil$ clearly follows, it remains to clarify which of the possible values $\chi(G) \in \{\vartheta(\overline{G}), \vartheta(\overline{G}) + 1\}$ is attained if $\vartheta(\overline{G}) \in \mathbb{Z}$.

For that, the following lemma is crucial. We say that a graph G is *homomorphic* to a graph H if there is a map f from the node set of G to the node set of H, preserving adjacency: if ij is an edge of G then $f(i)f(j)$ is an edge of H.

Lemma 2 *If G is homomorphic to H then $\vartheta(\overline{G}) \leq \vartheta(\overline{H})$.*

Proof Let $\mathcal{U}(G)$ denote the set of all orthonormal representations of G and let U be the set of vectors of unit length. We have by definition of $\vartheta(G)$ that

$$\vartheta(G) = \min_{(\mathbf{u_i}) \in \mathcal{U}(G)} \min_{\mathbf{c} \in U} \max_{i} \frac{1}{(\mathbf{c}^T \mathbf{u_i})^2}.$$

Let h be a homomorphism from G into H. Let $(\mathbf{v_i})$ be an orthonormal representation of $\overline{H}$ and $\mathbf{c} \in U$ such that $\vartheta(\overline{H}) = \max_i \frac{1}{(\mathbf{c}^T \mathbf{v_i})^2}$. For every $i \in V(G)$ let $\mathbf{u}_i = \mathbf{v}_{h(i)}$. If ij is an edge of G then $h(i)h(j)$ is an edge of H and so $\mathbf{u}_i \perp \mathbf{u}_j$. Thus $(\mathbf{u_i})$ is an orthonormal representation of $\overline{G}$. This implies

$$\vartheta(\overline{G}) \leq \max_{i \in V(G)} \frac{1}{(\mathbf{c}^T \mathbf{u_i})^2} \leq \max_{i \in V(H)} \frac{1}{(\mathbf{c}^T \mathbf{v_i})^2} \leq \vartheta(\overline{H}).$$ □

Lemma 2 implies the following key property of circular-perfect graphs:

Corollary 5 *If G is circular-perfect and $\chi_c(G) = p/q$, then $\vartheta(\overline{G}) = \vartheta(\overline{K_{p/q}})$.*

Proof In a circular-perfect graph G, there exists an induced circular-clique $K_{p/q} \subseteq G$ with $\omega_c(G) = p/q = \chi_c(G)$. Having $\chi_c(G) = p/q$ means that G is homomorphic to $K_{p/q}$, hence $\vartheta(\overline{G}) \leq \vartheta(\overline{K_{p/q}})$ holds by Lemma 2. On the other hand, $K_{p/q} \subseteq G$ clearly implies $\vartheta(\overline{K_{p/q}}) \leq \vartheta(\overline{G})$. □

Since every circular-clique $K_{p/q}$ is homomorphic to every circular-clique $K_{p'/q'}$ such that $p/q \leq p'/q'$ [4], we further obtain:

Corollary 6 *If $p/q \leq p'/q'$ then $\vartheta(\overline{K_{p/q}}) \leq \vartheta(\overline{K_{p'/q'}})$.*

In addition, Grötschel, Lovász and Schrijver established in [16] the bound

$$\vartheta(G) > \alpha(G) + 1/n^n \tag{17}$$

for the stability number of any minimal imperfect graph G. According to [22] and [26], every minimal imperfect graph G is partitionable, i.e., there are integers $p, q \geq 2$ such that $|V(G)| = pq + 1$ holds, and for each node v of G, the graph $G \setminus \{v\}$ admits a partition into p cliques of cardinality q as well as a partition into q stable sets of cardinality p. Since the proof in [16] to show inequality (17) for minimal imperfect graphs makes only use of properties satisfied by general partitionable graphs, we conclude:

Lemma 3 ([16]) *For any partitionable graph G with n nodes, we have that $\vartheta(G) > \alpha(G) + 1/n^n$ holds.*

Since $\overline{K_{p/q}}$ is partitionable if $p = 1 \pmod q$, the previous lemma implies that $\vartheta(\overline{K_{p/q}}) \geq \omega(\overline{K_{p/q}}) + 1/p^p$. Though this is enough for our purpose, we take advantage of the circulant structure of circular-cliques in the next lemma to provide a better distance to the clique number:

Lemma 4 *For a circular-clique $K_{p/q}$ with $\gcd(p,q) = 1$, we have $\vartheta(\overline{K_{p/q}}) > \omega + \frac{1}{2p^4}$ where $\omega = \lfloor p/q \rfloor$ denotes its clique number.*

Proof The proof relies on the following formula for $\vartheta(G)$ of any graph $G = (V, E)$, as a semidefinite program [24]:

$$\vartheta(G) = \max\left\{ \sum_{(x,y)\in V^2} B(x,y) : B \in \mathbb{R}^{V\times V}, B \succeq 0, \right.$$
$$\left. \sum_{x\in V} B(x,x) = 1\ \forall x \in V, B(x,y) = 0\ \forall xy \in E \right\} \tag{18}$$

where $B \succeq 0$ denotes that B is a symmetric, positive semidefinite matrix.

Let S be the circulant matrix whose top row is $(s_i)_{i=0,\dots,p-1}$ with $s_i = 1$ if $i = kq$ for some $0 \leq k < \omega$, and 0 otherwise. S is the incidence matrix of some maximal stable sets of $\overline{K_{p/q}}$. The eigenvalues of S are $\lambda_0, \dots, \lambda_{p-1}$ with

$$\lambda_i = \sum_{0\leq k<\omega} \zeta^{ikq}$$

for every $0 \leq i \leq p-1$ where $\zeta = e^{2i\pi/p}$.

Let $C = SS^T$. For every edge ij of $\overline{K_{p/q}}$, the entry $C(i,j)$ of C is equal to 0. Furthermore C is circulant and has therefore the same eigenvectors than S. It follows that the eigenvalues of C are ω^2 and $\mu_1, \dots, \mu_{p-1}$ with $\mu_i = \lambda_i\lambda_{p-i} = |\lambda_i|^2$ for every $1 \leq i \leq p-1$. Notice that for every $1 \leq i \leq p-1$,

$$|\lambda_i| = \left| \frac{1-\zeta^{i\omega q}}{1-\zeta^q} \right|$$

follows. Thus $|\lambda_i| \geq |1-\zeta^{i\omega q}|/2$ and so $|\lambda_i| \geq (1-\cos(2\pi/p))/2$. From $1 - \cos(x) \geq x^2/2 - x^4/24$ for every $x \geq 0$, we get $|\lambda_i| > 1/p^2$ for every $1 \leq i \leq p-1$. Hence $|\mu_i| > 1/p^4$ follows for every $0 \leq i \leq p-1$. The trace of C is equal to $p\omega$ and the sum of the entries of its first row is equal to $\omega + 2(1 + \cdots + (\omega-1))$, which is ω^2. Hence the sum of the entries of S is equal to $p\omega^2$.

Let $D = \frac{p^3}{p^4\omega-1}(C - \frac{1}{p^4}I)$, I being the identity $p \times p$-matrix. D is a semi-definite matrix, with trace 1, and such that $D(i,j) = 0$ for every edge ij of $\overline{K_{p/q}}$.

Hence, due to the formula (18), we indeed obtain

$$\vartheta(\overline{K_{p/q}}) \geq \sum_{i,j} D(i,j) \geq \frac{p^3}{p^4\omega - 1}\left(p\omega^2 - \frac{1}{p^3}\right) > \omega + \frac{1}{2p^4}.$$

□

We are now ready to prove the following:

Theorem 4 *For a circular-perfect graph G with n nodes, we have*

$$\chi(G) = \lceil \vartheta(\overline{G}) \rceil$$

and, if $\omega(G) < \chi(G)$, also $\omega(G) + \frac{1}{2n^4} < \vartheta(\overline{G}) < \omega(G) + (1 - \frac{1}{n})$.

Proof For a circular-perfect graph G, we have $\omega(G) \le \vartheta(\overline{G}) \le \chi(G) \le \omega(G) + 1$. Thus, if $\vartheta(\overline{G}) \notin \mathbb{Z}$ then clearly $\chi(G) = \lceil \vartheta(\overline{G}) \rceil$ follows. If $\omega(G) = \chi(G)$ then $\chi(G) = \lceil \vartheta(\overline{G}) \rceil$ is obvious. Hence we may assume that $\omega(G) \neq \chi(G)$. It is enough to establish that $\overline{\vartheta}$ is not integral to prove that $\chi(G) = \lceil \vartheta(\overline{G}) \rceil$. We are going to prove the stronger statement: $\omega(G) + \frac{1}{2n^4} < \vartheta(\overline{G}) < \omega + (1 - \frac{1}{n})$.

Since $\omega(G) < \chi(G)$, $\omega(G) < \omega_c(G)$ follows and there are integers p and $q \neq 0$, $\gcd(p, q) = 1$, $p, q \le n$, such that $K_{p/q} \subseteq G$ with $\omega_c(G) = p/q > \omega(G) = \omega$, i.e., $K_{p/q}$ is homomorphic to G. From Lemma 2 and Lemma 4, we get

$$\omega < \omega + \frac{1}{2n^4} < \vartheta(\overline{K_{p/q}}) \le \vartheta(\overline{G}).$$

We have $\chi_c(G) = \omega_c(G) = p/q \le \omega + (1 - 1/q)$ and $\vartheta(\overline{G}) \le \chi_f(G) \le \chi_c(G) \le \omega + (1 - 1/q)$ indeed follows. □

In order to use the fact that $\omega(G) = \lfloor \vartheta(\overline{G}) \rfloor$ and $\chi(G) = \lceil \vartheta(\overline{G}) \rceil$ holds to compute $\omega(G)$ and $\chi(G)$ for a circular-perfect graph in polynomial time, the difficulty is to decide algorithmically whether $\vartheta(\overline{G})$ is integral or not. This can be done by Algorithm 1 below which implies:

Corollary 7 *If G is circular-perfect then $\omega(G)$ and $\chi(G)$ are computable in polynomial time by Algorithm 1.*

Proof Let G be a circular-perfect graph with n nodes and $\varepsilon = 1/(2n^4)$. Computing $\overline{\vartheta} = \vartheta(\overline{G})$ with an error of at most $\varepsilon/2$ is polynomial in n. Denote this value by $\overline{\varphi}$. If $\overline{\varphi}$ is within distance $\varepsilon/2$ of an integer z, then due to Theorem 4, $\omega(G) = \chi(G) = z$ and so $\overline{\vartheta} = z$. Hence Algorithm 1 is a polynomial time algorithm with correct output, up to line 10. If $\overline{\varphi}$ is not within distance $\varepsilon/2$ of an integer then $\overline{\vartheta}$ is not an integer. Thus $\omega(G) < \overline{\vartheta} < \chi(G) \le \omega(G) + 1$ and so $\chi(G) = \omega(G) + 1$: the output given by the line 12 is correct. □

2.2.2 Why This Approach Does not Always Work for Other Classes

We briefly address the question whether the same approach to compute $\chi(G)$ in polynomial time can be applied to other graph classes $\mathcal{G}$ whose members satisfy (16). The difficulty is to handle the case if $\vartheta(\overline{G}) \in \mathbb{Z}$, and we notice that this situation

Algorithm 1: An algorithm to compute clique and chromatic number of circular perfect graphs

```
Input  : a circular perfect graph $G$ with $n$ nodes
Output: clique number $\omega$ and chromatic number $\chi$ of $G$
1  Let $\varepsilon = \frac{1}{2n^4}$
2  Compute $\vartheta(\overline{G})$ with precision $\varepsilon/2$ and denote this value by $\overline{\varphi}$
3  if $\overline{\varphi} - \lfloor\overline{\varphi}\rfloor < \varepsilon/2$ then
4      Let $\omega = \lfloor\overline{\varphi}\rfloor$ and $\chi = \omega$
5      return $\omega, \chi$
6  end
7  if $\lceil\overline{\varphi}\rceil - \overline{\varphi} < \varepsilon/2$ then
8      Let $\omega = \lceil\overline{\varphi}\rceil$ and $\chi = \omega$
9      return $\omega, \chi$
10 end
11 Let $\omega = \lfloor\overline{\varphi}\rfloor$ and $\chi = \omega + 1$
12 return $\omega, \chi$
```

cannot be resolved for two prominent classes with the studied property, line graphs and planar graphs.

A line graph is a graph $G = L(H)$ such that there is a graph H whose edges correspond to the nodes of G where two nodes of G are adjacent if the corresponding edges in H are. By construction, the clique number $\omega(G)$ equals the maximum degree $\Delta(H)$ (if H has a node of degree ≥ 3); hence the clique number of a line graph can be easily computed in polynomial time. The chromatic number $\chi(G)$ equals the edge chromatic number $\gamma(H)$. A famous result of Vizing [41] shows that $\gamma(H) \in \{\Delta(H), \Delta(H) + 1\}$ holds for all simple graphs H, but deciding which of the two values is attained is NP-complete [18]. Hence, line graphs indeed satisfy (16), but determining $\chi(G)$ is equivalent to edge color general graphs and thus NP-complete. With our approach, we can determine $\chi(G)$ for line graphs G with $\vartheta(\overline{G}) \notin \mathbb{Z}$, but the case with $\vartheta(\overline{G}) \in \mathbb{Z}$ remains open.

For a planar graph G, we have $\omega(G) \leq 4$ by Kuratowski [21] and thus the clique number of planar graphs can be clearly computed in polynomial time. If $\omega(G) = 2$, then $\chi(G) = 2$ holds if G is bipartite and $\chi(G) = 3$ otherwise. If $\omega(G) = 4$, then $\chi(G) = 4$ follows from the famous Four-Color-Theorem [35]. If, however, $\omega(G) = 3$, then $\chi(G) \in \{3, 4\}$ holds and it is NP-complete to decide which of the two values is attained [13]. With our approach, we can decide this for planar graphs G with $3 < \vartheta(\overline{G}) < 4$, but the case with $\vartheta(\overline{G}) = 3$ remains open.

To conclude, computing $\chi(G)$ in polynomial time with the help of the presented approach for circular-perfect graphs G strongly relies on some key properties of circular-perfect graphs (see Corollary 5), whereas the same procedure does not succeed for the two above graph classes $\mathcal{G}$, unless $P = NP$.

2.3 On Computing the Circular-Clique and Circular-Chromatic Number

In this subsection, we give an overview of the main arguments from [1] to extend the polynomial time computability of the chromatic number to the circular-chromatic number of circular-perfect graphs. In contrary to perfect graphs, $\vartheta(\overline{G})$ does not give directly the result as, in general, $\omega_c(G)$ and $\chi_c(G)$ are not integers, neither $\vartheta(\overline{G})$ is sandwiched between the two parameters. To bypass this difficulty, we make use of Corollary 5 that for every graph G with n nodes such that $\omega_c(G) = \chi_c(G) = k/d$, we have $\vartheta(\overline{G}) = \vartheta(\overline{K_{k/d}})$, where $k, d \le n$. Hence, to ensure the polynomial time computability of $\chi_c(G)$, it is sufficient to prove that the values $\vartheta(\overline{K_{k/d}})$ with $k, d \le n$ are all distinct and separated by at least ε for some ε with polynomial space encoding.

The proof is a joint work with Christine Bachoc and Alain Thiery [1]: we first established a closed formula for the theta number of the complements of circular-cliques $K_{k/d}$. Then we proved that the values are well-separated.

2.3.1 A Closed Formula

There are very few families of graphs for which an explicit formula for the theta number is known. As of complements of circular-cliques, Lovász solved the case of cycles ($d = 2$) in [24] and Brimkov et al. proved in [5] that, for $d = 3$ and k odd,

$$\vartheta(\overline{K_{k/3}}) = k\left(1 - \frac{1/2 - \cos((2\pi/k)\lfloor k/3\rfloor) - \cos((2\pi/k)(\lfloor k/3\rfloor + 1))}{(\cos((2\pi/k)\lfloor k/3\rfloor) - 1)(\cos((2\pi/k)(\lfloor k/3\rfloor + 1)) - 1)}\right).$$

The Chebyshev polynomials, denoted by $(T_\ell)_{\ell\ge 0}$, are defined by this property: $T_\ell(\cos(\theta)) = \cos(\ell\theta)$. They can be iteratively computed by the relation $T_{\ell+1}(x) = 2xT_\ell(x) - T_{\ell-1}(x)$ and the first terms are $T_0 = 1$, $T_1 = x$.

Since circular-cliques are circulant graphs (since their adjacency matrices are circulant matrices) and the set of optimal matrices for (18) is convex, there is an optimal circulant matrix for (18), and we may reformulate the semidefinite program (18) as the linear program given below:

Proposition 3 *Let* $k_0 := \lfloor k/2\rfloor$. *We have*:

$$\vartheta(\overline{K_{k/d}}) = \max\left\{kf_0 : f_j \ge 0, \sum_{j=0}^{k_0} f_j = 1, \right.$$
$$\left. \sum_{j=0}^{k_0} f_j T_j\left(\cos\left(\frac{2\ell\pi}{k}\right)\right) = 0, 1 \le \ell \le d-1\right\} \tag{19}$$

and by duality:

$$\vartheta(\overline{K_{k/d}}) = \min\Bigg\{ kg_0 : \sum_{\ell=0}^{d-1} g_\ell \geq 1, \\ \sum_{\ell=0}^{d-1} g_\ell T_l\left(\cos\left(\frac{2j\pi}{k}\right)\right) \geq 0,\, 1 \leq j \leq k_0 \Bigg\}. \tag{20}$$

To solve the linear program (19), we exhibit a candidate for an optimal solution which is feasible for this linear program and its dual. We give an interpretation of this element in terms of the coefficients of Lagrange interpolation polynomials on the basis of Chebyshev polynomials.

We choose as candidate the vector whose non-zero entries are the unique solution of the linear system:

$$\underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & T_1(a_1) & \cdots & T_{d-1}(a_1) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & T_1(a_{d-1}) & \cdots & T_{d-1}(a_{d-1}) \end{pmatrix}}_{M} \begin{pmatrix} f_0 \\ f_1^* \\ \vdots \\ f_{d-1}^* \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Solving this linear system amounts to inverting the matrix M: We assume for the rest of this subsection that $\gcd(k, d) = 1$. Then the real numbers a_n are pairwise distinct. Consider the Lagrange polynomials associated to $(a_0, \dots, a_{d-1})$:

$$L_n(y) := \prod_{\substack{s=0 \\ s \neq n}}^{d-1} \left(\frac{y - a_s}{a_n - a_s} \right).$$

Now we have two bases for the space of polynomials of degree at most equal to $d - 1$: the Chebyshev basis $\{T_0, \dots, T_{d-1}\}$ and the Lagrange basis $\{L_0, \dots, L_{d-1}\}$. We introduce the two $d \times d$ matrices $T = (\tau_{\ell,n})$ and $L = (\lambda_{n,\ell})$ such that

$$T_\ell(y) = \tau_{\ell,0} L_0(y) + \tau_{\ell,1} L_1(y) + \cdots + \tau_{\ell,d-1} L_{d-1}(y) \quad 0 \leq \ell \leq d-1$$

and

$$L_n(y) = \lambda_{n,0} T_0(y) + \lambda_{n,1} T_1(y) + \cdots + \lambda_{n,d-1} T_{d-1}(y) \quad 0 \leq n \leq d-1.$$

Obviously, $M = T$ as, for every $0 \leq n \leq q - 1$, $\tau_{\ell,n} = T_\ell(a_n)$ and $TL = LT = I_d$. In particular, the d-tuple $(\lambda_{0,0}, \lambda_{1,0}, \dots, \lambda_{d-1,0})$ satisfies the equations:

$$\sum_{n=0}^{d-1} \lambda_{n,0} T_\ell(a_n) = \delta_{\ell,0}, \quad 0 \leq \ell \leq d-1.$$

Summarizing the above results, our candidate for an optimal solution of (19) is therefore: $f^* = (f_0^*, \ldots, f_{k_0}^*)$ defined by

$$\begin{cases} f_j^* = \lambda_{n,0} & \text{for } j = \lfloor \frac{nk}{d} \rfloor, n = 0, \ldots, d-1 \\ f_j^* = 0 & \text{otherwise.} \end{cases}$$

The main difficulty of the proof is to verify that f^* is indeed optimal for (19) (see [1] for the details). The objective value is equal to $k\lambda_{0,0}$. So we have $\vartheta(\overline{K_{k/d}}) = k\lambda_{0,0}$. We recall that:

$$L_0(y) = \lambda_{0,0}T_0(y) + \lambda_{0,1}T_1(y) + \cdots + \lambda_{0,d-1}T_{d-1}(y). \tag{21}$$

If we plug in (21) the value $y = c_n$ and sum up for $n = 0, \ldots, d-1$, taking account of $T_0 = 1$ and $\sum_{n=0}^{d-1} T_j(c_n) = \sum_{n=0}^{d-1} \cos(2jn\pi/d) = 0$ for $1 \le j \le d-1$, we obtain the closed formula:

Theorem 5 *Let $d \geq 2$, $k \geq 2d$, with* $\gcd(k, d) = 1$. *Let, for $0 \leq n \leq d-1$,*

$$c_n := \cos\left(\frac{2n\pi}{d}\right), \qquad a_n := \cos\left(\left\lfloor \frac{nk}{d} \right\rfloor \frac{2\pi}{k}\right).$$

Then

$$\vartheta(\overline{K_{k/d}}) = \frac{k}{d} \sum_{n=0}^{d-1} \prod_{s=1}^{d-1} \left(\frac{c_n - a_s}{1 - a_s}\right). \tag{22}$$

2.3.2 Separating the Values

It remains to establish that the values $\vartheta(\overline{K_{k/d}})$ are well separated:

Theorem 6 *Let $p, p', q, q' \leq n$ such that $\frac{p}{q} \neq \frac{p'}{q'}$. Let $\Delta = |\vartheta(\overline{K_{p'/q'}}) - \vartheta(\overline{K_{p/q}})|$. Then there is a constant $c > 0$ such that*

$$\Delta \geq c^{-n^5}.$$

The proof is in two steps

(1) $\Delta \neq 0$
(2) if $\Delta \neq 0$ then $\Delta \geq c^{-n^5}$ for some $c > 0$

and involves some algebraic number theory: we now recall some basic terminology and results needed to sketch the proof.

For every k, let $\zeta_k = \exp^{2i\pi/k}$, and let Φ be the Euler function, $\mathbb{Q}(\zeta_k)$ denote the cyclotomic field (the smallest complex field containing ζ_k). For every $x \in \mathbb{Q}(\zeta_k)$, let polmin$(x) \in \mathbb{Q}[X]$ be the minimal polynomial of x: x is called an algebraic integer if polmin$(x) \in \mathbb{Z}[X]$. Then $\mathbb{Q}(\zeta_k)$ is a vector space over $\mathbb{Q}$ whose dimension is $\Phi(k)$ (hence at most k) and the set of algebraic integers is a ring. If K and L are two cyclotomic fields s.t. L is an extension of K, let $\mathrm{Gal}(L/K) = \mathrm{Aut}(L/K)$ be the Galois group of L over K, $\mathrm{Trace}_K^L(x) = \sum_{\sigma \in \mathrm{Gal}(L/K)} \sigma(x)$ be the trace of any element x of L, $\mathrm{Norm}_K^L(x) = \prod_{\sigma \in \mathrm{Gal}(L/K)} \sigma(x)$ be the norm of any element x of L. Let σ_i be the automorphism of $\mathbb{Q}(\zeta_k)$ s.t. $\sigma_i(\zeta_k) = \zeta_k^i$ (i coprime with k).

Then $\mathrm{Gal}(\mathbb{Q}(\zeta_k)) = \{\sigma_i, (i,k) = 1\}$, Trace_K^L is linear and for every element x of L, $\mathrm{Trace}_K^L(x) \in K$. For every $x \in \mathbb{Q}(\zeta_k)$, polmin$(x) = \prod_{\sigma \in \mathrm{Gal}(\mathbb{Q}(\zeta_k)/\mathbb{Q})}(X - \sigma(x))$.

We are now ready to outline the proof: assume by contradiction that $\Delta = 0$. Thus $\vartheta(\overline{K_{p/q}}) = \vartheta(\overline{K_{p'/q'}}) = \overline{\vartheta}$ for some $\frac{p}{q} < \frac{p'}{q'}$ and for every $\frac{a}{b} \in [\frac{p}{q}, \frac{p'}{q'}]$, $\vartheta(\overline{K_{a/b}}) = \overline{\vartheta}$.

Take $\frac{a}{b} \in [\frac{p}{q}, \frac{p'}{q'}]$ such that b is prime and b is coprime with a and $a+1$. We have $\mathrm{Gal}(\mathbb{Q}(\zeta_{ab})/\mathbb{Q}(\zeta_a)) = \{\sigma_i, 1 \le i \le b-1\}$ since a and b are coprime. For every i, $\sigma_i(\cos(2\pi/b)) = \cos(2i\pi/b)$ as $\cos(2i\pi/b) = \frac{1}{2}(\zeta_{ab}^{ia} + \zeta_{ab}^{ab-ia}) = \sigma_i(\frac{1}{2}(\zeta_b + \zeta_b^{b-1}))$. Hence (setting $A_0(x) = 2^{b-1}\prod_{i=1}^{b-1}(x - \cos(2i\pi/b))$ and $L_0(X) = A_0(X)/A_0(1)$):

$$\begin{aligned}
\vartheta(\overline{K_{a/b}}) &= \frac{a}{b}\Bigg(1 + \sum_{i=1,\dots,b-1} L_0\big(\sigma_i\big(\cos(2\pi/b)\big)\big)\Bigg) \\
&= \frac{a}{b}\Bigg(1 + \sum_{i=1,\dots,b-1} \sigma_i\big(L_0\big(\cos(2\pi/b)\big)\big)\Bigg) \\
&= \frac{a}{b}\Big(1 + \mathrm{Trace}_{\mathbb{Q}(\zeta_a)}^{\mathbb{Q}(\zeta_{ab})}\big(L_0\big(\cos(2\pi/b)\big)\big)\Big).
\end{aligned}$$

Thus $\overline{\vartheta} \in \mathbb{Q}(\zeta_a)$ and likewise $\overline{\vartheta} \in \mathbb{Q}(\zeta_{a+1})$. Hence $\overline{\vartheta} \in \mathbb{Q}(\zeta_a) \cap \mathbb{Q}(\zeta_{a+1}) = \mathbb{Q}$. Then, a short computation of the norm of $\overline{\vartheta}$ in a suitable cyclotomic field shows that $\overline{\vartheta}$ is integer. Theorem 4 implies that a/b is integer: a contradiction. Therefore $\Delta \neq 0$.

Recall that $\Delta = |\vartheta(\overline{K_{p'/q'}}) - \vartheta(\overline{K_{p/q}})|$ with $p, p', q, q' \le n$. Let $a_i = \cos(\lfloor \frac{ip}{q} \rfloor \frac{2\pi}{p})$, $a_i' = \cos(\lfloor \frac{ip'}{q'} \rfloor \frac{2\pi}{p'})$, and let $\alpha = qq' \prod_{i=1}^{q-1}(2 - 2a_i)\prod_{i=1}^{q'-1}(2 - 2a_i')\Delta$. Notice that α is an algebraic integer of $\mathbb{Q}(\zeta_{pp'qq'})$ and $\deg(\mathrm{polmin}(\alpha)) \le qq'pp' \le n^4$. Furthermore, for every $\sigma \in \mathrm{Gal}(\mathbb{Q}(\alpha))$, $|\sigma(\alpha)| \le 4^n n^3$. Since $\alpha \neq 0$ and α is an algebraic integer, we have $|\prod_{\sigma \in \mathrm{Aut}(\mathbb{Q}(\alpha))} \sigma(\alpha)| \ge 1$.

Hence, $|\alpha| \ge \prod_{\sigma \in \mathrm{Aut}(\mathbb{Q}(\alpha)), \sigma \neq \mathrm{Id}} \frac{1}{|\sigma(\alpha)|} \ge \left(\frac{1}{4^n n^3}\right)^{n^4}$ and, thus $\Delta \ge c^{-n^5}$ for some positive real c:

Theorem 7 *For every circular-perfect graph, circular-clique and circular-chromatic number are computable in polynomial time.*

3 Extending the Theta Function to Larger Convex Sets of Matrices

The proof that the chromatic number of perfect graphs is computable in polynomial time relies on three main properties of the theta function. Denote by f_ϑ the function defined by $f_\vartheta(G) = \vartheta(\overline{G})$ for every graph G, then f_ϑ satisfies the three assertions:

(P_1) f_ϑ is computable in polynomial time for every graph G;
(P_2) f_ϑ is monotonic with respect to homomorphism: if G is homomorphic to H then $f_\vartheta(G) \leq f_\vartheta(H)$;
(P_3) f_ϑ is strictly monotonic on cliques: for every integer $i \geq 1$, $f_\vartheta(K_i) < f_\vartheta(K_{i+1})$ and the difference has a polynomial space encoding.

In this section, we shall call f_θ, "the theta function", though it is actually the usual theta function applied to the complement of the input graph.

Notice that the key property to prove that the circular-chromatic number of circular-perfect graphs is computable in polynomial time was to extend property P_3 to the whole set of circular-cliques (Theorem 6):

(P_3') f_ϑ is strictly monotonic on circular-cliques: for every pair of rational p/q and p'/q' such that $p/q < p'/q'$, $f_\vartheta(K_{p/q}) < f_\vartheta(K_{p'/q'})$ and the difference has a polynomial space encoding.

The function f_ϑ is not the unique function satisfying properties P_1, P_2 and P_3, as for instance, some of its variants, such as the vectorial chromatic number [19] and the strong vectorial chromatic number [38], also satisfy these three properties.

The purpose of this subsection is to investigate "how unique" the theta function is, by considering a more general setting, based on some convex supersets of SDP matrices.

Let $\{0, 1\} \subseteq X \subseteq \mathbb{R}$. For every graph $G = (V, E)$ with at least one edge, denote by n its number of nodes and by $f_X(G)$ the value $1 - \frac{1}{s}$ where s is the optimum of the following program:

$$\begin{array}{ll} \min & s \\ \text{s.t.} & \exists M \in \mathcal{M}_X \\ & M \text{ is symmetric} \\ & M_{ii} = 1, \quad \forall i \in V \\ & M_{ij} = s, \quad \forall ij \in E \end{array}$$

where $\mathcal{M}_X$ is defined as the following set of matrices:

$$\mathcal{M}_X = \left\{ M \in \mathbb{R}^{V \times V},\ \text{s.t. } \mathbf{u}^T M \mathbf{u} \geq 0, \forall \mathbf{u} \in X^V \right\}.$$

If G does not have any edge, we let $f_X(G) = 1$. If M is a matrix of $\mathcal{M}_X$, we say that M is feasible. A feasible matrix which yields the value $f_X(G)$ is called optimal.

Here are some basic observations, for every graph G:

Table 1 Some numerical result for f_X, $X \in \{\{0,1\}, \{-1,0,1\}, \{-2,-1,0,1,2\}, \mathbb{R}\}$

	$\{0,1\}$	$\{-1,0,1\}$	$\{-2,-1,0,1,2\}$	$\mathbb{R}$
$C_9 = K_{9/4}$	2	2.061		2.064
$C_7 = K_{7/3}$	2	2.103	2.1096	2.1099
$C_5 = K_{5/2}$	2	2.200	2.231	2.236
$K_{8/3}$	2	2.333		2.343
$K_{11/4}$	2	2.3996		2.408
$K_{10/3}$	3	3.125		3.167
$\overline{C_7} = K_{7/2}$	3	3.222	3.294	3.318
$K_{11/3}$	3	3.400		3.452
$\overline{C_9} = K_{9/2}$	4	4.231		4.360
Petersen	2	2.5	2.5	2.5
$\overline{\text{Petersen}}$	4	4	4	4
$C_5 + 1$ multiplied node	2	2.210526		2.236

- $f_{\mathbb{R}}(G) = \vartheta(\overline{G})$ (Lovász's theta function), and thus $f_{\mathbb{R}}$ is computable in polynomial time with given accuracy;
- if $X \subseteq X'$ then $\mathcal{M}_{X'} \subseteq \mathcal{M}_X$ and thus $f_X(G) \leq f_{X'}(G)$;
- for every $\lambda \in \mathbb{R}^+$, $f_{\lambda X}(G) = f_X(G)$ as $\mathcal{M}_{\lambda X} = \mathcal{M}_X$.

Table 1 presents some numerical values $f_X(G)$ for some small graphs G and the sets X in $\{\{0,1\}, \{-1,0,1\}, \{-2,-1,0,1,2\}, \mathbb{R}\}$.

Lemma 5 *$\mathcal{M}_X$ is a convex cone and a superset of the set of semi-definite positive matrices of size $n \times n$.*

Proof $\mathcal{M}_X$ is a superset of SDP matrices as the set of SDP matrices of size $n \times n$ is equal to $\mathcal{M}_{\mathbb{R}}$. Let M and N be two elements of $\mathcal{M}_X$ and let λ, μ be two positive reals. For every $\mathbf{u} \in X^V$, we have $\mathbf{u}^T(\lambda M + \mu N)\mathbf{u} = \lambda \mathbf{u}^T M \mathbf{u} + \mu \mathbf{u}^T N \mathbf{u} \geq 0$, hence $\lambda M + \mu N$ is in $\mathcal{M}_X$. □

We first compute the value f_X for cliques:

Lemma 6 *$f_X(K_i) = i$ for every i.*

Proof Let M_s be the matrix with all main diagonal entries equal to 1 and all other entries equal to s. $M_s \in \mathcal{M}_{\{0,1\}}$ if and only if $j + j(j-1)s \geq 0$ for every $1 \leq j \leq i$ (j being the number of 1-entries in a vector $\mathbf{u} \in X^V$), and thus if and only if $s \geq \frac{-1}{i-1}$. Hence $f_{\{0,1\}}(K_i) = i$.

The result follows from $i = f_{\{0,1\}}(K_i) \leq f_X(K_i) \leq f_{\mathbb{R}}(K_i) = \vartheta(\overline{K_i}) = i$. □

Lemma 7 *If H is a subgraph of G then $f_X(H) \leq f_X(G)$.*

Proof If H is a spanning subgraph of G then $f_X(H) \leq f_X(G)$ as the program associated to H is formed by a subset of the constraints building the program associated to G (since the set of edges of H is a subset of the edges of G).

Hence we may assume without loss of generality that H is an induced subgraph of G. Let W denote the set of nodes of H. Let M^* be an optimal matrix for G. Let $\mathbf{u} \in X^W$. We extend $\mathbf{u}$ to the vector $\mathbf{u}^* \in X^V$ by adding 0-entries for indices in $V \setminus W$. Let M be the matrix of $\mathbb{R}^{W\times W}$ defined by $M_{i,j} = M^*_{i,j}$ for every $i, j \in W$. We have $\mathbf{u}^T M\mathbf{u} = \mathbf{u}^{*T} M^*\mathbf{u}^* \geq 0$. Thus M is a feasible matrix for H and therefore $f_X(H) \leq f_X(G)$. □

This implies the so-called sandwich-property:

Corollary 8 *For every graph G, $\omega(G) \leq f_X(G) \leq \vartheta(\overline{G}) \leq \chi(G)$.*

Proof Due to Lemma 6 and Lemma 7, we have $\omega(G) \leq f_X(G)$. Furthermore, $f_X(G) \leq \vartheta(\overline{G})$ by definition of $\mathcal{M}_X$. □

Lemma 8 *For every graph G, $f_{\mathbb{R}^+}(G) = \omega(G)$.*

Proof Let G be a graph with n nodes.

Due to the Lemmas 6 and 7, we have $\omega(G) \leq f_X(G)$ for every subset $X \in \mathbb{R}$.

We first prove that $f_{[0,1]}(G) = \omega(G)$.

Let $s = \frac{-1}{\omega(G)-1}$ and M be the $n \times n$ matrix, with all diagonal entries equal to 1, $M_{ij} = s$ whenever ij is an edge of G, and $M_{ij} = n^2$ otherwise.

Let $\mathbf{u} \in [0, 1]^V$ and let H be the set of nodes of G given by positive entries of $\mathbf{u}$. If $G[H]$ is a clique then $\mathbf{u}^T M\mathbf{u} \geq 0$ holds, as the submatrix of M with rows and columns corresponding to H is a semi-definite positive matrix.

If $G[H]$ is not a clique then $\mathbf{u}^T M\mathbf{u} = \sum_{i,j\in H} u_i u_j M_{ij} \geq n^2 s + n^2 \geq 0$. Hence $f_{[0,1]}(G) \leq \omega(G)$ and therefore $f_{[0,1]}(G) = \omega(G)$.

To complete the proof, we establish that $\mathcal{M}_{[0,1]} = \mathcal{M}_{\mathbb{R}^+}$. Obviously $\mathcal{M}_{\mathbb{R}^+} \subseteq \mathcal{M}_{[0,1]}$. If $\mathcal{M}_{[0,1]}$ is not contained by $\mathcal{M}_{\mathbb{R}^+}$ then there is a matrix M of $\mathcal{M}_{[0,1]}$ and a vector $\mathbf{u}$ of $\mathbb{R}^{+V}$ such that $\mathbf{u}^T M\mathbf{u} < 0$. Let m be an upper bound of the absolute entries of $\mathbf{u}$. Then $\frac{1}{m}\mathbf{u}^T M \frac{1}{m}\mathbf{u} < 0$, a contradiction as $\frac{1}{m}\mathbf{u}$ belongs to $[0, 1]^V$.

Thus $\mathcal{M}_{[0,1]} = \mathcal{M}_{\mathbb{R}^+}$ and so $f_{\mathbb{R}^+}(G) = f_{[0,1]}(G) = \omega(G)$. □

As an obvious consequence of Lemma 8, we get:

Corollary 9 *$f_{\{0,1\}}$ is NP-hard to compute.*

Multiplying a node v of a graph G means to replace v by a stable set S such that all nodes in S have the same neighbors in G as the original node v. Thus, multiplying a node of a graph G gives a homomorphically equivalent graph H. Hence if X is a set of reals such that f_X satisfies the monotonic property P_2, then $f_X(G) = f_X(H)$. Thus $f_{\{-1,0,1\}}$ does not satisfy P_2 as multiplying a node of a C_5 yields a different value (see Table 1). Therefore, additional constraints are needed for sets X in order to ensure that property P_2 is fulfilled. We next show that being closed with respect to addition is such a sufficient condition:

Lemma 9 *Assume that X is closed with respect to addition. If G is homomorphic to H then $f_X(G) \le f_X(H)$ (monotonic property).*

Proof Due to Lemma 7, it is sufficient to prove that $f_X(G) \le f_X(H)$ if H is a graph obtained from G by identifying two non-adjacent nodes i and j of G into one single node i. Let M be an optimal matrix for H.

Let N be the matrix obtained from M by duplicating the row of index i, and denote the index of the new row by j.

Let O be the matrix of $\mathbb{R}^{V(G)\times V(G)}$ obtained from N by duplicating the column of index i and denote by j the index of the new column.

Let $\mathbf{u} \in X^{V(G)}$ and define $\mathbf{u}^* \in X^{V(H)}$ by $\mathbf{u}^*_a = \mathbf{u}_a$ if $a \neq i$ and $\mathbf{u}^*_i = \mathbf{u}_i + \mathbf{u}_j$.

From $O\mathbf{u} = N\mathbf{u}^*$ and $\mathbf{u}^T N = \mathbf{u}^{*T} M$, we get $\mathbf{u}^T O \mathbf{u} = \mathbf{u}^{*T} M \mathbf{u}^* \ge 0$. Hence M is a feasible matrix for G: $f_X(G) \le f_X(H)$. □

Due to Lemma 8, the base set X has to have one negative element, say -1, in order to get a function f_X which is different from the clique number. If we apply the requirement of the previous lemma to get a function satisfying the monotonic property then X contains all integers. We next establish that this implies that f_X has to be the theta function:

Lemma 10 *If $\mathbb{Z} \subseteq X$ or $[-1,1] \subseteq X$ then $f_X(G) = \vartheta(\overline{G})$ for every graph G.*

Proof We are going to prove that

$$\mathcal{M}_{\mathbb{Z}} = \mathcal{M}_{[-1,1]} = \mathcal{M}_{\mathbb{R}}.$$

We obviously have $\mathcal{M}_{\mathbb{R}} \subseteq \mathcal{M}_{\mathbb{Z}}$. Conversely, let M be any matrix of $\mathcal{M}_{\mathbb{Z}}$. Let $\mathbf{u} \in \mathbb{Q}^V$ and m be an integer such that $m\mathbf{u} \in \mathbb{Z}^V$. Then we have $m\mathbf{u}^T M m\mathbf{u} \ge 0$, hence $\mathbf{u}^T M \mathbf{u} \ge 0$ and so $M \in \mathcal{M}_{\mathbb{Q}}$. Let $\mathbf{u} \in \mathbb{R}^V$ and pick a sequence of rational vectors $\mathbf{u_i}$ which converge to $\mathbf{u}$. Since for every i, $\mathbf{u_i}^T M \mathbf{u_i} \ge 0$, we get $\mathbf{u}^T M \mathbf{u} \ge 0$, by taking the limit. Hence $M \in \mathcal{M}_{\mathbb{R}}$.

We obviously have $\mathcal{M}_{\mathbb{R}} \subseteq \mathcal{M}_{[-1,1]}$. Conversely, let M be any matrix of $\mathcal{M}_{[-1,1]}$. Let $\mathbf{u} \in \mathbb{R}^V$ distinct of $\mathbf{0}$ and denote by l the maximum absolute value of an entry of $\mathbf{u}$. Hence $\frac{1}{l}\mathbf{u}^T M \frac{1}{l}\mathbf{u} \ge 0$, and thus $\mathbf{u}^T M \mathbf{u} \ge 0$. Hence $M \in \mathcal{M}_{\mathbb{R}}$. □

Our study seems to indicate that the clique number function and the theta function are the only functions in our setting that satisfy the monotonic requirement with respect to homomorphism (property P_2). Hence in this sense, the theta function is really unique, since it is also computable in polynomial time (property P_1).

As of the sandwich property, we point out that it holds even if the monotonic property is not satisfied (Corollary 8): there are many different functions f_X in between the clique and the chromatic numbers, all of them being a lower bound for the theta function.

For further works, it is worth to notice that the numerical values presented in Table 1 suggest that the function $f_{\{-1,0,1\}}$ gives already good lower bounds for the theta function. In particular, it would be interesting to use this to provide a better gap and a simpler proof for the theta number of circular-cliques (Lemma 4).

References

1. Bachoc, C., Pêcher, A., Thiery, A.: On the theta number of powers of cycle graphs. Combinatorica (to appear)
2. Bang-Jensen, J., Huang, J.: Convex-round graphs are circular-perfect. J. Graph Theory **40**, 182–184 (2002)
3. Berge, C.: Färbungen von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind. Wiss. Z., Martin-Luther-Univ. Halle-Wittenb., Math.-Nat.wiss. Reihe **10**, 114–115 (1961)
4. Bondy, J., Hell, P.: A note on the star chromatic number. J. Graph Theory **14**, 479–482 (1990)
5. Brimkov, V., Codenotti, B., Crespi, V., Leoncini, M.: On the Lovász number of certain circular graphs. Lect. Notes Comput. Sci. **1767**, 291–305 (2000)
6. Chudnovsky, M., Robertson, N., Seymour, P., Thomas, R.: The strong perfect graph theorem. Ann. Math. **164**, 51–229 (2006)
7. Chvátal, V.: On certain polytopes associated with graphs. J. Comb. Theory, Ser. B **18**, 138–154 (1975)
8. Coulonges, S., Pêcher, A., Wagler, A.: Characterizing and bounding the imperfection ratio for some classes of graphs. Math. Program., Ser. A **118**, 37–46 (2009)
9. Deuber, W., Zhu, X.: Circular coloring of weighted graphs. J. Graph Theory **23**, 365–376 (1996)
10. Faenza, Y., Oriolo, G., Stauffer, G.: An algorithmic decomposition of claw-free graphs leading to an $O(n^3)$-algorithm for the weighted stable set problem. In: Proceedings of SODA, pp. 630–646 (2011)
11. Fulkerson, D.: Blocking and antiblocking pairs of polyhedra. Math. Program. **1**, 168–194 (1971)
12. Fulkerson, D.: Anti-blocking polyhedra. J. Comb. Theory, Ser. B **12**, 50–71 (1972)
13. Garey, M., Johnson, D., Stockmeyer, L.: Some simplified NP-complete graph problems. Theor. Comput. Sci. **1**, 237–267 (1976)
14. Grötschel, M.: My favorite theorem: characterizations of perfect graphs. Optima **62**, 2–5 (1999)
15. Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica **1**(2), 169–197 (1981)
16. Grötschel, M., Lovász, L., Schrijver, A.: Polynomial algorithms for perfect graphs. In: Topics on Perfect Graphs. North-Holland Math. Stud., vol. 88, pp. 325–356 (1984)
17. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, Berlin (1988)
18. Holyer, I.: The NP-completeness of edge-colouring. SIAM J. Comput. **10**, 718–720 (1981)
19. Karger, D., Motwani, R., Sudan, M.: Approximate graph coloring by semidefinite programming. J. ACM **45**, 246–265 (1998)
20. Kuhpfahl, J., Wagler, A., Wagner, C.: Circular-imperfection of triangle free graphs. Electron. Notes Discrete Math. **29**, 163–167 (2007)
21. Kuratowski, K.: Sur le problème des courbes gauches en topologie. Fundam. Math. **15**, 271–283 (1930)
22. Lovász, L.: A characterization of perfect graphs. J. Comb. Theory, Ser. B **13**, 95–98 (1972)
23. Lovász, L.: Normal hypergraphs and the perfect graph conjecture. Discrete Math. **2**, 253–267 (1972)
24. Lovász, L.: On the Shannon capacity of a graph. IEEE Trans. Inf. Theory **25**(1), 1–7 (1979)

25. Oriolo, G., Stauffer, G., Ventura, P.: Stable sets in claw-free graphs: recent achievements and future challenges. Optima **86**, 2–5 (2011)
26. Padberg, M.: Perfect zero-one matrices. Math. Program. **6**, 180–196 (1974)
27. Pêcher, A., Wagler, A.: Almost all webs are not rank-perfect. Math. Program., Ser. B **105**(2–3), 311–328 (2006)
28. Pêcher, A., Wagler, A.: On classes of minimal circular-imperfect graphs. Discrete Appl. Math. **156**, 998–1010 (2008)
29. Pêcher, A., Wagler, A.: On the polynomial time computability of the circular-chromatic number for some superclasses of perfect graphs. Electron. Notes Discrete Math. **35**, 53–58 (2009)
30. Pêcher, A., Wagler, A.: Clique and chromatic number of circular-perfect graphs. Electron. Notes Discrete Math. **36**, 199–206 (2010)
31. Pêcher, A., Wagler, A.: Computing the clique number of a-perfect graphs in polynomial time. Electron. Notes Discrete Math. **38**, 705–710 (2011)
32. Pêcher, A., Wagler, A.: Computing clique and chromatic number of circular-perfect graphs in polynomial time. Math. Program., Ser. A (2012). doi:10.1007/s10107-012-0512-4
33. Pêcher, A., Wagler, A.: Computing the clique number of a-perfect graphs in polynomial time. Eur. J. Comb. (to appear)
34. Pêcher, A., Zhu, X.: Claw-free circular-perfect graphs. J. Graph Theory **65**, 163–172 (2010)
35. Robertson, N., Sanders, D.P., Seymour, P.D., Thomas, R.: The four colour theorem. J. Comb. Theory, Ser. B **70**, 2–44 (1997)
36. Shannon, C.E.: The zero-error capacity of a noisy channel. IRE Trans. Inf. Theory **2**, 8–19 (1956)
37. Shepherd, F.B.: Applying Lehman's theorem to packing problems. Math. Program. **71**, 353–367 (1995)
38. Szegedy, M.: A note on the number of Lovász and the generalized Delsarte bound. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science, pp. 36–39 (1994)
39. Trotter, L.E. Jr.: A class of facet producing graphs for vertex packing polyhedra. Discrete Math. **12**, 373–388 (1975)
40. Vince, A.: Star chromatic number. J. Graph Theory **12**, 551–559 (1988)
41. Vizing, V.: Vertex colorings with given colors. Diskretn. Anal. **29**, 3–10 (1976) (in Russian)
42. Wagler, A.: Rank-perfect and weakly rank-perfect graphs. Math. Methods Oper. Res. **95**, 127–149 (2002)
43. Wagler, A.: Antiwebs are rank-perfect. 4OR **2**, 149–152 (2004)
44. Wagler, A.: On rank-perfect subclasses of near-bipartite graphs. 4OR **3**, 329–336 (2005)
45. Xu, Y.: Minimally circular-imperfect graphs with a major vertex. Discrete Math. **301**, 239–242 (2005)
46. Zhu, X.: Circular perfect graphs. J. Graph Theory **48**(1–3), 186–209 (2005)

From Vertex-Telecenters to Subtree-Telecenters

Zaw Win and Cho Kyi Than

Abstract Let T be a tree and v a vertex in T. It is well-known that the branch-weight of v is defined as the maximum number of vertices in the components of $T - v$ and that a vertex of T with the minimum branch-weight is called a vertex-centroid of T. Mitchell (Discrete Math. 24:277–280, 1978) introduced a type of a central vertex called the telephone center or the vertex-telecenter of a tree and showed that v is a vertex-centroid of T if and only if it is a vertex-telecenter of T. In this paper we introduce the notions of the subtree-centroid and the subtree-telecenter of a tree which are natural extensions of the vertex-centroid and the vertex-telecenter, and generalize two theorems of Mitchell (Discrete Math. 24:277–280, 1978) in the extended framework of subtree-centroids and subtree-telecenters. As a consequence of these generalized results we also obtain an efficient solution method which computes a subtree-telecenter of a tree.

1 Introduction

On one day of the summer 1984, the first author of this paper (who was then studying the German language at the Goethe Institute in Göttingen and had to study at the University of Augsburg as a DAAD scholar under the supervision of Prof. Dr. Martin Grötschel) wrote the first letter to his supervisor introducing himself, informing of his arrival in Germany and also with a request of some mathematics literature to learn in advance in Göttingen before studying at the University of Augsburg. Just after a few days, the first author happily and thankfully received a packet of some research papers on Combinatorial Optimization sent by Prof. Dr. Martin Grötschel. It is the first one of so many generous supports, kind encouragements, warm helps and insightful academic enlightenments given by Prof. Dr. Martin Grötschel to the first author till now. After studying the research papers mentioned above, the subject of Combinatorial Optimization has become the most favorite research area of

Z. Win (✉) · C.K. Than
Department of Mathematics, University of Yangon, Yangon 11041, Myanmar
e-mail: zawwinbur@gmail.com

C.K. Than
e-mail: chokyi34@gmail.com

the first author under the wonderful guidance and supervision of Prof. Dr. Martin Grötschel. Now we, the authors would like to pay a great homage to Prof. Dr. Martin Grötschel, the doctoral father and grandfather of ours with the following work on a certain combinatorial optimization problem.

2 Vertex-Centroids and Vertex-Telecenters of a Tree

Many location problems in networks encountered in practice deal with the task of finding positions, for one or more servicing facilities, which will be the best with respect to some desired criteria. These problems can be usually modeled as those of finding subgraphs of a given graph which are most central with respect to a certain centrality measure. In the literature there are several measures of centrality of vertices in graphs and a well-known one is the branch-weight of a vertex of a tree whose origin dates back to Jordan [2].

Definition 1 Let T be a tree with the vertex set $V(T)$ and the edge set $E(T)$, and let $v \in V(T)$. Suppose that $T - v$ consists of k components $T_1, T_2, \dots, T_k$ which are subtrees of T and called *branches* of v. The *branch-weight* of v, denoted $b(v)$, is defined by

$$b(v) = \max_{1 \le i \le k} \left| V(T_i) \right|.$$

Thus $b(v)$ is the maximum number of vertices in the branches of v. A vertex $v \in V(T)$ is called a *vertex-centroid* of T if $b(v) \le b(w)$ for all w in $V(T)$, i.e., if v has the minimum branch-weight.

Example 1 Consider the tree T shown in Fig. 1. $T - v_1$ consists of five components T_i, $1 \le i \le 5$ as shown in Fig. 2. By definition the branch-weight of the vertex v_1 is

$$\begin{aligned} b(v_1) &= \max_{1 \le i \le 5} \left| V(T_i) \right| \\ &= \left| V(T_5) \right| \\ &= 5. \end{aligned}$$

Similarly we can find that

$$b(v_3) = b(v_4) = b(v_5) = b(v_7) = b(v_8) = b(v_{10}) = b(v_{13}) = b(v_{15}) = 14,$$
$$b(v_9) = b(v_{12}) = b(v_{14}) = 13, \qquad b(v_2) = b(v_6) = 12, \qquad b(v_{11}) = 10.$$

Thus the vertex-centroid of T, i.e., the vertex with the minimum branch-weight is v_1.

More information on the vertex-centroids of a tree can be found in Buckley and Harary [1], Jordan [2], Reid [4] and Zelinka [7].

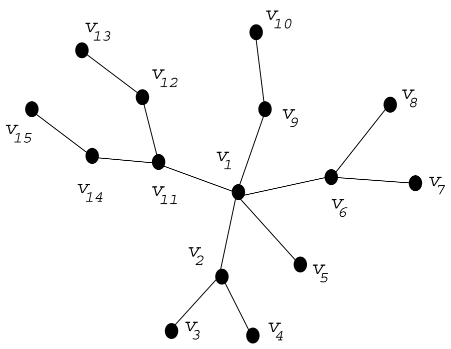


Fig. 1 A tree T

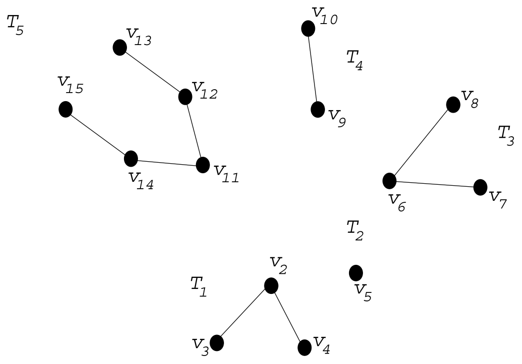


Fig. 2 $T - v_1$

Mitchell [3] introduced the following concept of the vertex-telecenter of a tree and proved the equivalence of the vertex-centroids and the vertex-telecenters.

Definition 2 Let T be a tree, v be a vertex in T and suppose that $T - v$ consists of k components $T_1, T_2, \dots, T_k$, i.e., v has k branches $T_1, T_2, \dots, T_k$.

A set $S = \{x, y\}$ of two vertices x and y in T such that $x \in V(T_i)$, $y \in V(T_j)$, $1 \leq i \leq k$, $1 \leq j \leq k$ and $i \neq j$ is called a *call* through v; two calls S_1 and S_2 through v are said to be *disjoint* if S_1 and S_2 have no common vertex, the largest possible number of disjoint calls through v is called the *switch-board number* of v and is denoted by $sb(v)$. A vertex z in T with the maximum switch-board number is called a *vertex-telecenter* of T.

Example 2 Consider the tree T shown in Fig. 1 and the graph $T - v_1$ of Fig. 2 comprised of five branches T_i, $1 \leq i \leq 5$. It can be checked that a largest possible collection of disjoint calls through the vertex v_1 is, for example,

$$\big\{\{v_2, v_5\}, \{v_3, v_6\}, \{v_4, v_{11}\}, \{v_7, v_{12}\}, \{v_8, v_{13}\}, \{v_9, v_{14}\}, \{v_{10}, v_{15}\}\big\}$$

consisting of 7 disjoint calls and so the switch-board number of v_1 is $sb(v_1) = 7$. Similarly we can calculate that

$$sb(v_3) = sb(v_4) = sb(v_5) = sb(v_7) = sb(v_8) = sb(v_{10}) = sb(v_{13}) = sb(v_{15}) = 0,$$

$$sb(v_9) = sb(v_{12}) = sb(v_{14}) = 1, \qquad sb(v_2) = sb(v_6) = 2, \qquad sb(v_{11}) = 4.$$

Thus the vertex-telecenter of T, i.e., the vertex with the maximum switch-board number is v_1.

For the vertex-centroids and vertex-telecenters, Mitchell [3] proved the following two theorems.

Theorem 1 *Let T be a tree, $v \in V(T)$, $T - v$ consist of k components $T_1, T_2, \ldots, T_k$ and suppose that $|V(T_1)| \le |V(T_2)| \le \cdots \le |V(T_k)|$. Then v is a vertex-telecenter of T if and only if $|V(T_k)| \le \sum_{i=1}^{k-1} |V(T_i)| + 1$.*

Theorem 2 *A vertex v in a tree T is a vertex-centroid of T if and only if it is a vertex-telecenter of T.*

3 Subtree-Centroids and Subtree-Telecenters of a Tree

In this section we introduce the notions of the subtree-centroid and the subtree-telecenter of a tree which are natural extensions of the vertex-centroid and the vertex-telecenter respectively.

Definition 3 Let T be a tree, n be a positive integer with $n < |V(T)|$, T' a subtree of T with n vertices and let $T - T'$ consist of k components $T_1, T_2, \ldots, T_k$. We denote the *branch-weight* of T' by $b(T')$ and define it by

$$b(T') = \max_{1 \le i \le k} |V(T_i)|.$$

We will call $T_1, T_2, \ldots, T_k$ the *branches* of T' and hence the branch-weight $b(T')$ of T' is the maximum number of vertices in the branches of T'.

Among the subtrees of T with n vertices, a subtree T^* with the minimum branch-weight is called a *subtree-centroid of T with n vertices*. By this definition, a vertex-centroid is a subtree-centroid with one vertex.

Definition 4 Let the tree T, the positive integer n, the subtree T' and the components $T_1, T_2, \ldots, T_k$ be as in Definition 3. A set $S = \{x, y\}$ of two vertices in T such that $x \in V(T_i)$, $y \in v(T_j)$, $1 \le i \le k$, $1 \le j \le k$ and $i \ne j$ is called a *call* through T'; two calls S_1 and S_2 through T' are *disjoint* if $S_1 \cap S_2 = \emptyset$; and the largest possible number of disjoint calls through T' is called the *switch-board number* of T' and denoted by $sb(T')$. Among the subtrees of T with n vertices, a subtree T^* with the maximum switch-board number is called a *subtree-telecenter of T with n vertices*. By this definition, a vertex-telecenter of T is a subtree-telecenter with one vertex.

Example 3 Consider the tree T shown in Fig. 3 and let T' be the subtree of T induced by the vertex set $\{d, e, f, k\}$.

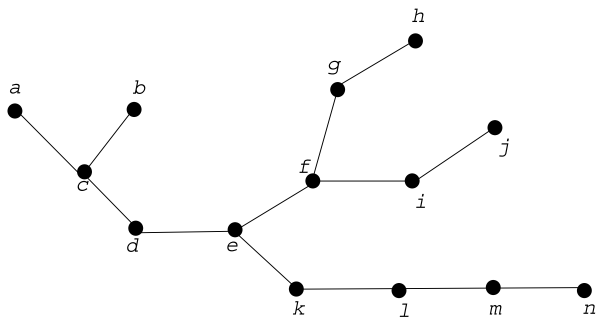


Fig. 3 A tree T

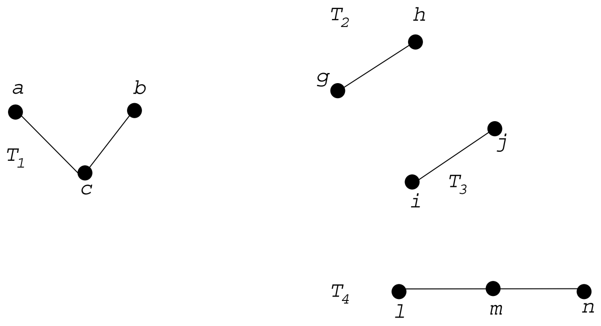


Fig. 4 $T - T'$

Figure 4 shows the graph $T - T'$ consisting of four components T_i, $1 \le i \le 4$. We can see that the branch-weight of T' is

$$\begin{aligned} b(T') &= \max_{1\le i\le 4} \left|V(T_i)\right| \\ &= \left|V(T_1)\right| \\ &= \left|V(T_4)\right| \\ &= 3. \end{aligned}$$

A largest possible collection of disjoint calls through T' is, for instance,

$$\left\{\{a,l\}, \{b,m\}, \{c,n\}, \{g,i\}, \{h,j\}\right\}$$

consisting of five calls and thus the switch-board number of T' is $sb(T') = 5$.

It can also be verified that T' is a subtree-centroid as well as a subtree-telecenter of T with four vertices.

Thwe [5] and Win and Thwe [6] studied the subtree-centroids of a tree from theoretical and algorithmic points of view.

4 A Characterization of Subtree-Telecenters

In this section we shall present a characterization of subtree-telecenters with a given number of vertices of a tree. For this purpose we first prove some lemmas.

Lemma 1 *Let T be a tree, n be a positive integer with $n < |V(T)|$ and T' be a subtree of T with n vertices. Then*

$$sb\big(T'\big) \leq \left\lfloor \frac{|V(T)| - n}{2} \right\rfloor.$$

Here $\lfloor x \rfloor$ denotes the largest integer not larger than x.

Proof To obtain a call through T' we have to pair up two vertices of $V(T - T')$ lying in distinct components. Therefore the largest possible number of disjoint calls through T' is at most $\lfloor \frac{|V(T-T')|}{2} \rfloor$, and hence

$$sb\big(T'\big) \leq \left\lfloor \frac{|V(T - T')|}{2} \right\rfloor = \left\lfloor \frac{|V(T)| - n}{2} \right\rfloor. \qquad \square$$

Lemma 2 *Let T be a tree, n be a positive integer with $n < |V(T)|$, T' be a subtree of T with n vertices, $T - T'$ consist of k components $T_1, T_2, \dots, T_k$ and suppose that $|V(T_1)| \leq |V(T_2)| \leq \cdots \leq |V(T_k)|$. If $|V(T_k)| \leq \sum_{i=1}^{k-1} |V(T_i)| + 1$, then*

$$sb\big(T'\big) = \left\lfloor \frac{|V(T)| - n}{2} \right\rfloor.$$

Proof Suppose that the vertices in $\bigcup_{i=1}^{k} V(T_i)$ have been paired up to obtain a largest possible collection C of disjoint calls through T' and let U be the set of vertices of $\bigcup_{i=1}^{k} V(T_i)$ which are left unpaired by C. Then all the vertices of U lie in the same subtree T_j with $1 \leq j \leq k$, otherwise, by pairing up two vertices of U lying in distinct subtrees T_i, we will have a collection of disjoint calls through T' that is larger than C, and this is a contradiction. We will next show that $|U| = 0$ or 1. To this end, suppose that $|U| \geq 2$ and x, y are two vertices in U. By the given inequality, we have

$$\big|V(T_j)\big| \leq \sum_{i=1, i \neq j}^{k} \big|V(T_i)\big| + 1$$

$$\big|V(T_j)\big| - 1 \leq \sum_{i=1, i \neq j}^{k} \big|V(T_i)\big|$$

$$\big|V(T_j)\big| - 2 < \sum_{i=1, i \neq j}^{k} \big|V(T_i)\big|.$$

From this inequality and the fact that all vertices in $\bigcup_{i=1,i\neq j}^{k} V(T_i)$ have been paired up by C, it follows that there must exist a call $\{p,q\}$ through T' in C such that $p \in V(T_\ell)$, $q \in V(T_m)$, $\ell \neq j$ and $m \neq j$. Now, by deleting the call $\{p,q\}$ from C and adding two new calls $\{x,p\}$ and $\{q,y\}$ to C, we have a collection of disjoint calls through T' which is larger than C and this is a contradiction. Thus we must have $|U| = 0$ or 1. If $|U| = 0$, then $|V(T)| - n$ is even and

$$sb(T') = \frac{|V(T)| - n}{2} = \left\lfloor \frac{|V(T)| - n}{2} \right\rfloor.$$

If $|U| = 1$, then $|V(T)| - n$ is an odd integer, say $2r+1$, and

$$\begin{aligned} sb(T') &= r \\ &= \left\lfloor \frac{2r+1}{2} \right\rfloor \\ &= \left\lfloor \frac{|V(T)| - n}{2} \right\rfloor. \end{aligned}$$

This completes our proof. □

Lemma 3 *Let T be a tree, n be a positive integer with $n < |V(T)|$, T' be a subtree of T with n vertices, $T - T'$ consist of k components $T_1, T_2, \ldots, T_k$ and suppose that $|V(T_1)| \le |V(T_2)| \le \cdots \le |V(T_k)|$. If $|V(T_k)| > \sum_{i=1}^{k-1} |V(T_i)| + 1$, then*

$$sb(T') = \sum_{i=1}^{k-1} |V(T_i)| < \left\lfloor \frac{|V(T)| - n}{2} \right\rfloor.$$

Proof Since every call through T' contains at least one vertex of $\bigcup_{i=1}^{k-1} V(T_i)$, we have

$$sb(T') \le \sum_{i=1}^{k-1} |V(T_i)|.$$

By the given inequality we can also construct $\sum_{i=1}^{k-1} |V(T_i)|$ disjoint calls through T' by pairing up each vertex of $\bigcup_{i=1}^{k-1} V(T_i)$ with a vertex of $V(T_k)$. So we have

$$sb(T') = \sum_{i=1}^{k-1} |V(T_i)|.$$

Next, the given inequality and the fact that

$$|V(T)| - n = \sum_{i=1}^{k-1} |V(T_i)| + |V(T_k)|$$

imply that

$$\begin{aligned}\big|V(T)\big| - n &\geq \sum_{i=1}^{k-1}\big|V(T_i)\big| + \left(\sum_{i=1}^{k-1}\big|V(T_i)\big| + 2\right)\\ &= 2\sum_{i=1}^{k-1}\big|V(T_i)\big| + 2\end{aligned}$$

and hence we have

$$\begin{aligned}\frac{|V(T)| - n}{2} &\geq \sum_{i=1}^{k-1}\big|V(T_i)\big| + 1\\ \left\lfloor \frac{|V(T)| - n}{2} \right\rfloor &\geq \sum_{i=1}^{k-1}\big|V(T_i)\big| + 1\\ &> \sum_{i=1}^{k-1}\big|V(T_i)\big|\\ &= sb\big(T'\big).\end{aligned}$$

So, the lemma is proved. □

Lemma 4 *Let T be a tree and n be a positive integer with $n < |V(T)|$. Then there exists a subtree T' of T with the following properties*:

(i) $|V(T')| = n$,
(ii) *$T - T'$ consists of k components $T_1, T_2, \ldots, T_k$ for some positive integer k with $|V(T_1)| \leq |V(T_2)| \leq \cdots \leq |V(T_k)|$ and*
(iii) $|V(T_k)| \leq \sum_{i=1}^{k-1} |V(T_i)| + 1$.

Proof Obviously there are subtrees T' satisfying conditions (i) and (ii). Among these subtrees choose a subtree T' which minimizes the difference

$$\big|V(T_k)\big| - \sum_{i=1}^{k-1}\big|V(T_i)\big|.$$

If possible, suppose that T' does not satisfy the condition (iii). Then we will have

$$\begin{aligned}\big|V(T_k)\big| &\geq \sum_{i=1}^{k-1}\big|V(T_i)\big| + 2\\ \big|V(T_k)\big| - 1 &\geq \sum_{i=1}^{k-1}\big|V(T_i)\big| + 1.\end{aligned} \tag{1}$$

Let x be a vertex in T_k which is nearest to T', y be a pendant vertex of T' which is not the nearest to T_k and let T^* be the subtree of T obtained from T' by adding the vertex x and deleting the vertex y. Evidently $|V(T^*)| = n$. Suppose that $T - T^*$ consists of ℓ components $T_1^*, T_2^*, \dots, T_\ell^*$ such that $|V(T_1^*)| \le |V(T_2^*)| \le \cdots \le |V(T_\ell^*)|$. It is not difficult to see from inequality (1) that

$$\bigl|V\bigl(T_\ell^*\bigr)\bigr| \le \bigl|V(T_k)\bigr| - 1.$$

This implies that

$$\begin{aligned}
\bigl|V\bigl(T_\ell^*\bigr)\bigr| - \sum_{i=1}^{\ell-1}\bigl|V\bigl(T_i^*\bigr)\bigr| &= \bigl|V\bigl(T_\ell^*\bigr)\bigr| - \bigl(\bigl|V(T)\bigr| - n - \bigl|V\bigl(T_\ell^*\bigr)\bigr|\bigr) \\
&\le \bigl|V(T_k)\bigr| - 1 - \bigl(\bigl|V(T)\bigr| - n - \bigl|V(T_k)\bigr| + 1\bigr) \\
&= \bigl|V(T_k)\bigr| - 1 - \bigl(\bigl|V(T)\bigr| - n - \bigl|V(T_k)\bigr|\bigr) - 1 \\
&= \bigl|V(T_k)\bigr| - \sum_{i=1}^{k-1}\bigl|V(T_i)\bigr| - 2 \\
&< \bigl|V(T_k)\bigr| - \sum_{i=1}^{k-1}\bigl|V(T_i)\bigr|.
\end{aligned}$$

This is a contradiction to our choice of T'. Therefore T' satisfies the condition (iii) and the lemma is proved. □

From these lemmas we immediately obtain the following characterization of subtree-telecenters of a tree which is an extension of Theorem 1.

Theorem 3 *Let T be a tree, n be a positive integer with $n < |V(T)|$, T' be a subtree of T with n vertices, $T - T'$ consist of k components $T_1, T_2, \dots, T_k$ and $|V(T_1)| \le |V(T_2)| \le \cdots \le |V(T_k)|$. Then T' is a subtree-telecenter of T with n vertices if and only if $|V(T_k)| \le \sum_{i=1}^{k-1} |V(T_i)| + 1$. Moreover the switch-board number of the subtree-telecenter T' is*

$$sb\bigl(T'\bigr) = \biggl\lfloor \frac{|V(T)| - n}{2} \biggr\rfloor.$$

5 Relation Between Subtree-Centroids and Subtree-Telecenters

In this section we will show that Theorem 2 can be only partially extended to the generalized framework of subtree-centroids and subtree-telecenters. First we have the following result.

Theorem 4 *Let T be a tree, n be a positive integer with $n < |V(T)|$ and T' a subtree of T. If T' is a subtree-centroid of T with n vertices, then T' is a subtree-telecenter of T with n vertices.*

Proof Let T' be a subtree-centroid of T with n vertices, $T - T'$ consist of k components $T_1, T_2, \dots, T_k$ and $|V(T_1)| \leq |V(T_2)| \leq \cdots \leq |V(T_k)|$. Obviously $b(T') = |V(T_k)|$. If possible, suppose that T' is not a subtree-telecenter of T. Then, by Theorem 3

$$\left|V(T_k)\right| \geq \sum_{i=1}^{k-1}\left|V(T_i)\right| + 2$$

$$\left|V(T_k)\right| - 1 \geq \sum_{i=1}^{k-1}\left|V(T_i)\right| + 1.$$

Now, by constructing a subtree T^* as in the proof of Lemma 4, we will have

$$\left|V\left(T^*\right)\right| = n$$

and

$$b\left(T^*\right) < b\left(T'\right).$$

Since this is a contradiction to the fact that T' is a subtree-centroid of T, we conclude that T' is a subtree-telecenter of T. □

The converse of Theorem 4 is not true. This is illustrated with the following example.

Example 4 Consider the tree T shown in Fig. 3. Let T' be the subtree of T induced by the vertex set $\{d, e, f, k\}$, and T'' be the subtree induced by the vertex set $\{d, e, k, \ell\}$. It is easy to check that T' is a subtree-centroid as well as a subtree-telecenter of T consisting of four vertices and having $b(T') = 3$ and $sb(T') = 5$. On the other hand T'' is a subtree-telecenter but not a subtree-centroid of T consisting of four vertices since it can be verified that $b(T'') = sb(T'') = 5$.

As a consequence of Theorem 4 and a known algorithm for finding a subtree-centroid with a given number of vertices of a tree, we obtain a simple and fast method of computing a subtree-telecenter with a given number of vertices of a tree.

5.1 A Solution Method for Finding a Subtree-Telecenter of a Given Tree

Thwe [5] suggested an efficient solution procedure which produces a subtree-centroid with a given number of vertices of a given tree. Since, by Theorem 4, every subtree-centroid with a given number of vertices is a subtree-telecenter with

Algorithm 1: An algorithm for finding a subtree-telecenter of a given tree

```
Input  : a tree $T$, and a positive integer $n$ with $n < |V(T)|$
Output: a subtree-telecenter $T'$ of $T$ with $n$ vertices

$T' := T$
$P(T') :=$ the set of pendant vertices of $T'$
$w(v) := 1$ for each $v \in V(T')$
$\deg(v) :=$ the degree of $v$ in $T'$ for each $v \in V(T')$
while $|V(T')| > n$ do
    Choose a vertex $v_i \in P(T')$ such that $w(v_i) = \min_{v \in P(T')} w(v)$
    $u_i :=$ the vertex in $T'$ adjacent to $v_i$
    $w(u_i) := w(u_i) + w(v_i)$
    $\deg(u_i) := \deg(u_i) - 1$
    if $\deg(u_i) = 1$ then
        $P(T') := (P(T') \setminus \{v_i\}) \cup \{u_i\}$
    else
        $P(T') := P(T') \setminus \{v_i\}$
    end
    $T' := T' \setminus \{v_i\}$
end
Output the subtree $T'$ which is a subtree-telecenter of $T$ with $n$ vertices.
```

the same number of vertices, we can apply the algorithm of Thwe [5] (Algorithm 1) to obtain a subtree-telecenter with a given number of vertices of a tree.

6 Conclusion

We have introduced the notions of the subtree-centroid and subtree-telecenter of a tree, gave a characterization of the subtree-telecenters with a given number of vertices and presented some relationships between the subtree-centroids and subtree-telecenters of a tree for the unweighted case, i.e., for the case of trees whose vertices have no weights. It may be interesting to carry out a similar study on the trees whose vertices are weighted.

Acknowledgements Our deepest gratitude goes to Prof. Dr. Martin Grötschel for his guided optimal traveling research-person tour through the world of combinatorial optimization. We are also very thankful to the referee and editors whose comments and suggestions have improved the readability of the paper.

References

1. Buckley, F., Harary, F.: Distance in Graphs. Addison-Wesley, Redwood City (1990)

2. Jordan, C.: Sur les assemblages de lignes. J. Reine Angew. Math. **70**, 185–190 (1869)
3. Mitchell, S.L.: Another characterization of the centroid of a tree. Discrete Math. **24**, 277–280 (1978)
4. Reid, K.B.: Centroids to centers in trees. Networks **21**, 11–17 (1991)
5. Thwe, A.M.: Medians and branch weight centroids in graphs. Dissertation, Department of Mathematics, University of Yangon (2007)
6. Win, Z., Thwe, A.M.: Some extensions of Zelinka's theorem on medians and centroids of trees. Research report 1, Department of Mathematics, University of Yangon (2010)
7. Zelinka, B.: Medians and peripherians of trees. Arch. Math. **4**, 87–95 (1968)

Algorithms for Junctions in Acyclic Digraphs

Carlos Eduardo Ferreira and Álvaro Junio Pereira Franco

Abstract Given targets u and v in a digraph D, we say that a vertex s is a junction of u and v if there are in D internally vertex-disjoint directed paths from s to u and from s to v. In this paper, we show how to characterize junctions in acyclic digraphs. We also consider the following problem and derive an efficient algorithm to solve it. Given an acyclic digraph D, a vertex s in D and k pairs of targets $\{u_1, v_1\}, \ldots, \{u_k, v_k\}$, determine the pairs of targets $\{u_i, v_i\}$ for which s is a junction. This problem arises in an application brought to our attention by an anthropologist. In this application the digraph represents the genealogy of an ethnic group in Brazilian Amazon region, and the pairs of targets are individuals that are married. We apply our algorithm to find all the junctions of k pairs of targets on those kinship networks. Experiments have shown that our algorithm had a good performance for the inputs considered. Some results are described in this paper.

1 Introduction

This work has been prepared for publication in a special issue in honor of Martin Grötschel. In some sense, many characteristics of Grötschel's works are presented here. First, it was motivated by a real world application brought to us by a friend from the Anthropology Department. Both working with interdisciplinary subjects and real applications are in the main stream of Martin Grötschel's work. Also, especially in the early years of his brilliant career, Grötschel was interested in graph theoretical problems for which he could provide good characterizations and propose efficient algorithms to solve them. An interesting example of such a problem is in [16]. In that work Grötschel provided an ear decomposition for strongly connected digraphs with no cut vertices (also known as strong blocks). This result is

C.E. Ferreira (✉) · Á.J.P. Franco
Instituto de Matemática e Estatística, Rua do Matão, 1010, Cidade Universitária, 05508-090, São Paulo, SP, Brazil
e-mail: cef@ime.usp.br

Á.J.P. Franco
e-mail: alvaro@ime.usp.br

useful to prove tight bounds on the number of arcs and on the in-degree and out-degree of at least two vertices of a minimal strong block.

We are interested in solving a simple path problem on digraphs. Consider an acyclic digraph D, a vertex s in D and k pairs of targets $\{u_1, v_1\}, \ldots, \{u_k, v_k\}$. The problem we want to solve is to determine the pairs of targets $\{u_i, v_i\}$ for which there exist in D paths from s to u_i and from s to v_i that are internally vertex-disjoint.

This problem has been brought to us by an anthropologist. In Anthropology terms, the acyclic digraph represents a kinship network and the pair of targets are marriages. The individuals for which the disjoint paths exist are in "alliance position", meaning that the marriage joins people that are, in some sense, related (see dal Poz and Silva [11]). In the application one is interested in finding all such individuals in alliance position for all the marriages. If we can solve the problem posed above, it is not difficult to list them all.

In this paper, we will focus on the path problem described above. Soon, we are going to see how it could be solved in polynomial time using a max-flow algorithm. Also other approaches can be used as the Suurballe and Tarjan's algorithm [20], or by means of dominator trees. Our contribution is a simple linear time algorithm to solve the problem which can be implemented easily. We use this algorithm to show some results.

2 Concepts and Notation

In this text a digraph is simple, weakly connected, and acyclic. The number of vertices is n, the number of arcs is m, and a path is always directed and simple. Given an acyclic digraph D and vertices s and t in D we say that s is a parent of t (or t is a child of s), when there exists an arc $s \to t$ in D. We denote the set of parents of a vertex u in D by $\delta_D^-(u)$ and the set of children of a vertex u in D by $\delta_D^+(u)$. We say that s is an ancestor of t or t is a descendant of s when there exists a path from s to t in D. A path in D is a sequence $P = v_1 v_2 \ldots v_k$ of vertices of D such that $v_i \to v_{i+1}$ is an arc in D for $i = 1, \ldots, k-1$. A subpath of P from vertex v_i to vertex v_j is denoted by $P[v_i, v_j]$, where $i \leq j$. A concatenation of two paths P and Q is denoted by PQ. In some cases we denote a simple vertex as a path.

A vertex s in a digraph D is a *junction* of a pair of targets u and v in D if there exist paths from s to u and from s to v that are internally vertex-disjoint. A vertex s in D is a *lowest common ancestor*, LCA for short, of a pair of targets u and v, if it is a junction and there is no path on D from s to s' for every other junction s' of the pair $\{u, v\}$. If D is a rooted tree, then junctions and LCAs are the same vertices and given a pair of targets u and v in D the LCA is unique. On the other hand, in digraphs, acyclic or not, we can have several different LCAs or junctions for a certain pair of targets. Note that LCAs are junctions, but the converse is not always true.

Let $J(u, v)$ denote the set of vertices in D that are junctions of the pair of targets u and v. The sets $J(u, v)$ and $J(v, u)$ are equal. We write $s \in J(u, v)$ if s is a

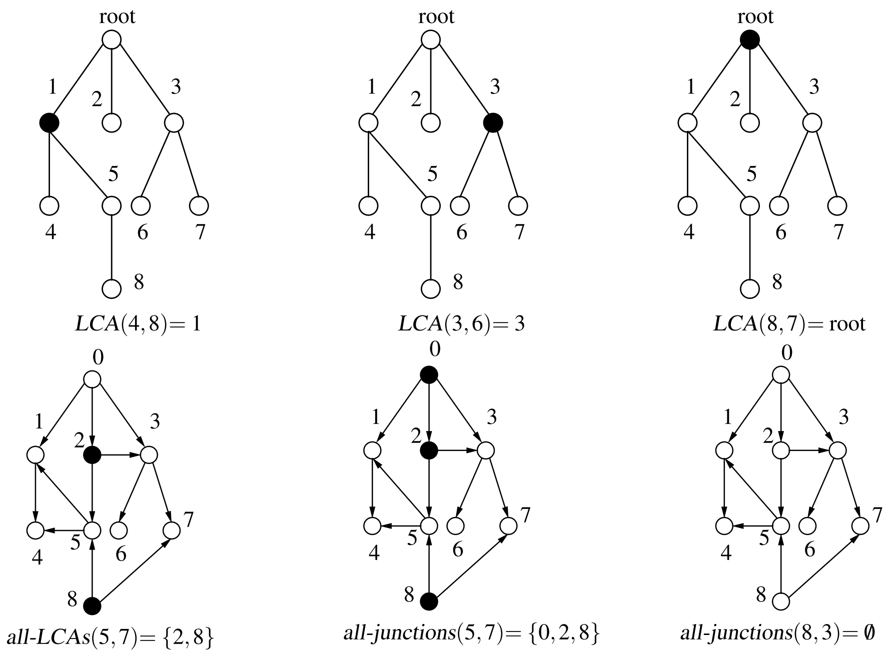


Fig. 1 The *vertices in bold* are LCAs and junctions of some pair of vertices

junction of the pair of targets u and v. We consider s to be a junction of pairs of targets s and u for every vertex u such that there is a path from s to u. Such pairs of targets are called degenerated. Analogously, $LCA(u, v)$ denotes the set of LCAs of the pair of targets u and v. If we want to specify the digraph D we are working on, then we write $LCA_D(u, v)$. When D is a rooted tree, we abuse the language writing $s = LCA_D(u, v)$. Let D^s denote the digraph induced by the descendants of s in D. A depth first spanning tree T^s of D^s, DFST for short, is a rooted spanning tree of D^s with root in s obtained by a depth-first search algorithm. Sometimes, we work on a subtree $T^{s'}$ of a DFST T^s, where s' is the root of $T^{s'}$ and s' is a descendant of s in T^s.

A *representative-junction*(u, v) query asks for a junction in D for given targets u and v. An *all-junctions*(u, v) query asks for all junctions in D for given targets u and v. Conversely, a *representative-LCA*(u, v) query asks for an LCA in D for given targets u and v. An *all-LCAs*(u, v) query asks for the all LCA in D for given targets u and v. All these previous queries return one or more vertices of D (LCAs or junctions). When the digraph D is a rooted tree, an $LCA(u, v)$ query returns the unique LCA of u and v in D. A special kind of query is the s-*junction*(u, v) query that asks whether a given vertex s is a junction of given targets u and v. Figure 1 illustrates a rooted tree, an acyclic digraph, and the lowest common ancestors and junctions of some pairs of targets.

3 Problem Definition, Literature Overview, and Main Results

For an acyclic digraph D, a vertex s in D and k pairs of targets $\{u_1, v_1\}, \ldots, \{u_k, v_k\}$, we want to determine for which pairs s is a junction. This problem is called *s-junction-k-pairs* problem. We solve it in two steps. First we preprocess D^s creating data structures able to answer whether or not s is a junction for any possible pair of targets in D^s. Then we answer the *s-junction*(u_i, v_i) query for $i = 1, \ldots, k$. Similar approaches are presented in many works in the literature on this problem or some related problems. For instance, Aho, Hopcroft and Ullman [2] have shown how to preprocess rooted trees in $O(n\alpha(n))$ time, where $\alpha(n)$ is the inverse Ackermann function. After preprocessing, an *LCA* query is answered in constant time. Harel and Tarjan [17] have shown how to preprocess rooted trees in $O(n)$ time and to answer an *LCA* query in constant time. Other similar works on rooted trees are Berkman and Vishkin [6], Nykänen and Ukkonen [18] and Wen [22].

Bender, Farach-Colton, Pemmasani, Skiena and Sumazin [5] have defined the *all-pairs-representative-lca* problem, i.e., given an acyclic digraph D, find for any pair of vertices u and v in D a representative LCA. They showed how to preprocess an acyclic digraph D in $\tilde{O}(n^{2.68635})$ time and then, answer a *representative-LCA* query in constant time. We say that $f(n) = \tilde{O}(g(n))$ if there is a constant c such that $f(n) = O(g(n) \log^c n)$. Later, Czumaj, Kowaluk and Lingas [10] have shown how to preprocess D in $O(n^{2.575})$ time, and Eckhardt, Mühling and Nowak [12] in $O(n^2 \log n)$ expected time.

The *all-pairs-all-lcas* problem asks how to preprocess efficiently a given acyclic digraph D such that an *all-LCAs* query spends $O(k')$ time for any given pair of targets u and v in D, where k' is the size of *LCA*(u, v). Baumgart, Eckhardt, Griebsch, Kosub and Nowak [4] have shown how to preprocess D in $O(\min\{n^2 m, n^{3.575}\})$ time. Eckhardt, Mühling and Nowak [12] have shown how to preprocess D in $O(n^3 \log\log n)$ expected time and $O(n^{3.3399})$ time (worst case).

Yuster [24] showed how to preprocess an acyclic digraph so that a *representative-junction*(u, v) query is done in constant time for any pair of targets u and v. This problem is known as *all-pairs-representative-junction* problem. Note that any algorithm that solves *all-pairs-representative-lca* could be used to solve *all-pairs-representative-junction*. Yuster [24] has shown how to preprocess the digraph in $\tilde{O}(n^{\omega})$ time, where $\omega < 2.3727$ (see Williams [23]) is the exponent of the fast Boolean matrix multiplication.

A problem involving junctions is also treated by Tholey [21]. In his paper the author considers the following problem. Given an acyclic digraph D, two vertices s_1 and s_2 and k pairs of targets $\{u_1, v_1\}, \ldots, \{u_k, v_k\}$ one wants to decide for each pair $\{u_i, v_i\}$ whether there exist vertex (or arc) disjoint paths from s_1 to u_i (or v_i) and from s_2 to v_i (resp. u_i). The author proposes a modification of the data structure due to Suurballe and Tarjan [20] that can be implemented in $O(n \log^2 n + (m + k) \log_{2+(m+k)/(n+k)} n)$ time and, with it, the queries can be answered in constant time.

Motivated by our application in Anthropology, we define the *k-pairs-all-junctions* problem as follows. Given an acyclic digraph D, and k pairs of targets $\{u_1, v_1\}, \ldots, \{u_k, v_k\}$, we want to preprocess efficiently D so that for any pair

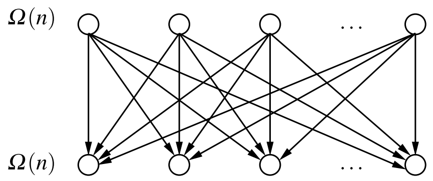


Fig. 2 Each *vertex in the first line* is a junction of every pair of *vertices in the second one*. Thus, to list all the junctions of all $\Omega(n^2)$ pairs would spend $\Omega(n^3)$ time

$\{u_i, v_i\}$, the *all-junctions*(u_i, v_i) query can be answered in $O(k')$ time where k' is the size of $J(u_i, v_i)$.

Consider an acyclic digraph D, a vertex s and k pairs of targets. Our main result is an $O(m+k)$ time algorithm to solve the *s-junction-k-pairs* problem. Using this algorithm one can solve the *k-pairs-all-junctions* problem by solving the *s-junction-k-pairs* problem for all s in D listing (or storing) s for a given pair of targets u_i and v_i, if s belongs to $J(u_i, v_i)$. This can be done in $O(n(m+k))$ time. An interesting use of our algorithm is its application to solve the *k-pairs-all-lcas* problem. Given an acyclic digraph D and k pairs of targets, we first find all junctions of the k given pairs of targets in D and then determine the transitive closure D' of D. The vertex s is an LCA for u and v if and only if for any junction s', $s' \neq s$, the arc $s \to s'$ is not in D'. Using the fact above we can solve the *k-pairs-all-lcas* problem in $O(n(m+k) + n^{\omega} + n^2k)$, where $O(n^{\omega})$ is the time for calculating transitive closure of D (ω is again the exponent of the fast Boolean matrix multiplication). If $k = o(n^{1.3399})$, then our simple approach for *k-pairs-all-lcas* is faster than the best algorithm (worst case) known for solving this problem due to Eckhardt, Mühling and Nowak [12].

To finish this section, we observe that any algorithm that aims at listing all junctions of all possible pairs of targets must spend $\Omega(n^3)$ time, in the worst case. To see that, consider the simple example in Fig. 2 and observe that all n^2 pairs of vertices in the second row have n junctions in the first row, which assures the lower bound.

4 Polynomial Time Algorithms for the *s-Junction-k-Pairs* Problem

In this section, we describe briefly three simple approaches to solve the *s-junction-k-pairs* problem in polynomial time. The reader that is not familiar with the algorithms and data structures in this section may refer to the following texts [1, 15, 20].

The first approach makes use of a well-known reduction from a vertex-disjoint to an arc-disjoint path problem and runs a max-flow algorithm. Given an acyclic digraph D we first construct a new acyclic digraph D'_i, for a fixed pair of targets u_i and v_i, as follows. Create two new vertices v' and v'' and one arc $v' \to v''$ in D'_i for each vertex v in D and one arc $u'' \to v'$ in D'_i for each arc $u \to v$ in D. Now, consider a new vertex t' and two new arcs $u''_i \to t'$ and $v''_i \to t'$. At last, set the capacity equal to 2 for arc $s' \to s''$ and capacity equal to 1 for all remaining arcs. We use a max-flow algorithm to determine the value of a maximum value flow from s' to t' in D'_i. If this value is 2, there exist vertex-disjoint paths in D from s to u_i and from s to v_i. This approach can be implemented by using, for example, Goldberg's and Tarjan's algorithm [15], in $O(knm\log(n^2/m))$ time.

The second approach also uses a reduction from a vertex-disjoint to an arc-disjoint path problem and afterward runs Suurballe and Tarjan's algorithm [20]. Given a digraph G with non-negative arc costs and a vertex s in G, Suurballe and Tarjan's algorithm consists of finding, for each target v, a pair of arc-disjoint paths from s to v where the sum of the cost of both paths is minimum. In this case, we construct an acyclic digraph D'. For each vertex v in D, we have in D' two vertices v' and v'' and one arc $v' \to v''$. For each arc $u \to v$ in D, create an arc $u'' \to v'$ in D'. Also, for each pair of targets $\{u_i, v_i\}$ create a vertex t'_i and two arcs $u''_i \to t'_i$ and $v''_i \to t'_i$. We set all costs of the arcs in D' to 1. Observe that if there exist two internally arc-disjoint paths from s'' to t'_i in D', then there exist vertex-disjoint paths from s to u_i and from s to v_i in the original digraph. Suurballe and Tarjan's algorithm solves this problem on D' in $O(m'\log_{1+m'/n'} n')$ time, where n' and m' are the number of vertices and arcs of D'.

The third approach uses a data structure named dominator trees (see Aho and Ullman, [1]) and a data structure for answering *LCA* queries in rooted trees. First we construct a dominator tree T rooted in s in linear time considering the digraph D^s. This can be done using, for example, Georgiadis and Tarjan's algorithm [14]. After that, we construct a data structure for *LCA* queries in linear time. Lastly, $LCA(u_i, v_i)$ queries are performed, for $i = 1, \ldots, k$. We know that s is a junction of the pair u_i, v_i in D if and only if $s = LCA_T(u_i, v_i)$. The structures used to construct a dominator tree (see Appendix C in [3]) and for answering dynamic *LCA* queries in rooted trees (see Cole and Hariharan, [8]) are rather intricate.

5 An $O(m+k)$ Time Algorithm for the *s-Junction-k-Pairs* Problem

In the next sections we present two algorithms to solve the *s-junction-k-pairs* problem. The first one is a dynamic programming algorithm that has cubic runtime. The second one is a recursive linear time algorithm. A time complexity analysis and correctness proofs for the algorithms are presented. To start we pose the following recurrence.

Lemma 1 *Let T^{s_1} and T^{s_2} be subtrees of a DFST T^s, where s_1 and s_2 are children of s in T^s and let $z = LCA_{T^s}(u, v)$. Then, s is a junction of u and v if and only if one of the following cases applies.*

1. *If u is equal to s or v is equal to s.*
2. *If u is in T^{s_1} and v is in T^{s_2}.*
3. *If u and v are in the same subtree (say T^{s_1}), z is not equal to u neither v and or s is a junction of z and u, or s is a junction of z and v.*
4. *If u and v are in the same subtree (say T^{s_1}), z is equal to u (or v) and s is a junction of z and t for some t in $\delta^-_D(v)$ (or $\delta^-_D(u)$) and t different from z.*

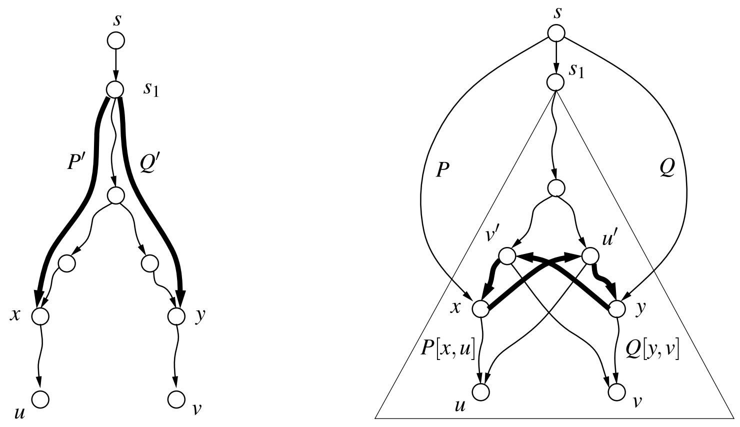


Fig. 3 A DFST and a cycle composed by the paths from y to v', from v' to x, from x to u' and from u' to y. This cannot occur in an acyclic digraph

Proof If the first case applies, then, by our definition of degenerated pair, s is a junction of the pair. For the second case observe that the paths in the DFST are a certificate for the fact that s is a junction. The third case is proven by Lemma 2 and Corollary 1. The last case is proven by Lemma 3.

Now, consider that no cases apply, i.e., $\{u, v\}$ is not a degenerated pair, u and v are in the same subtree and condition 3 and 4 are false. Then, s cannot be a junction of u and v. □

Before we prove Lemmas 2 and 3, and Corollary 1, note the following properties. Consider a DFST T^s of D, a child s_1 of s in T^s and a path P from s to u in D, where u is a descendant of s_1 in T^s. Suppose that x is the first vertex in P which is in the subtree T^{s_1}. Thus, all vertices of the path $P[x, u]$ belong to the subtree T^{s_1} of T^s. Additionally, note that if s is a junction of targets u and v and the targets u and v are in T^{s_1}, then there exists a pair of internally vertex-disjoint paths in D from s to u and from s to v, such that all internal vertices from one of them are in T^{s_1}. We can prove this by contradiction. Consider two internally vertex-disjoint paths in D, P from s to u and Q from s to v (see Fig. 3). Suppose that both P and Q do not have all vertices in T^{s_1}. Consider the first vertices x of P and y of Q that belong to T^{s_1}. We know that the paths $P[x, u]$ and $Q[y, v]$ are internal to T^{s_1}. Take the path P' from s_1 to x in T^{s_1}. If there is no common vertex in the paths P' and $Q[y, v]$, then the paths $sP'P[x, u]$ and Q are internally vertex-disjoint paths and all internal vertices of one of them are in T^{s_1}. So, P' and $Q[y, v]$ have at least a common vertex. Let v' be the common vertex in these paths nearest to v. Analogously, take the path Q' from s_1 to y in T^{s_1}. If it is disjoint to $P[x, u]$, then the paths $sQ'Q[y, v]$ and P are internally vertex-disjoint paths and all internal vertices of one of them are in T^{s_1}. Therefore, Q' and $P[x, u]$ have at least a common vertex. Let u' be the common vertex nearest to u. Thus, we find a cycle with the paths from y to v', from v' to x, from x to u' and from u' to y. This is a contradiction, since D

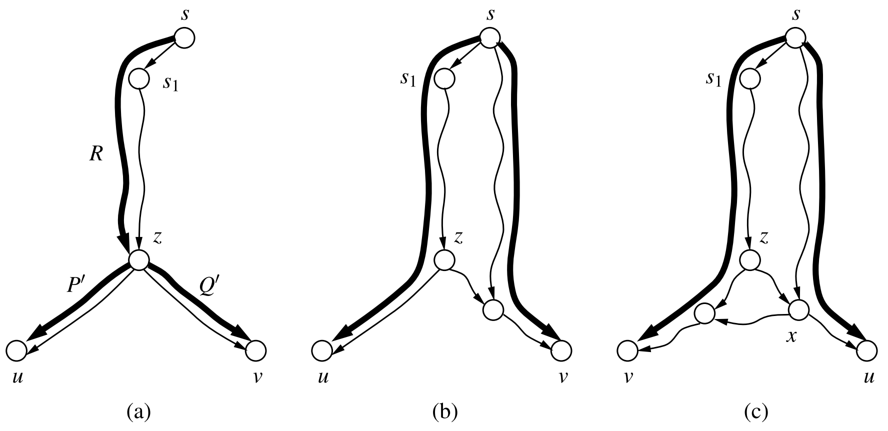


Fig. 4 A DFST and new internally vertex-disjoint paths—Lemma 2 Case 1. One of them includes vertex $z = LCA_{T^s}(u, v)$

is an acyclic digraph. Therefore, all internal vertices of P or Q are in T^{s_1}. And now we can prove the following lemma.

Lemma 2 *Let u and v be targets in the subtree T^{s_1} of T^s, where s_1 is a child of s in T^s. Call z the LCA of u and v in T^s and consider that z is not equal to u or v. Vertex s is a junction of the pair of targets u and v if and only if z belongs to a pair of internally vertex-disjoint paths from s to u or from s to v.*

Proof Since s is a junction of u and v, take a pair of internally vertex-disjoint paths P from s to u and Q from s to v. Suppose all internal vertices of P are in the subtree T^{s_1} and Q has an external arc entering the subtree T^{s_1}. Consider R to be the path from s to z in T^s. Let P' and Q' be the paths in T^s from z to u and from z to v, respectively (see Fig. 4(a)). Here it is really important to note that, by the construction of T^s, any path that passes through a vertex in Q' (P') and then passes through a vertex in P' (Q') cannot come back to Q' (P'). Let us divide the proof in two cases:

Case 1. There is no vertex of Q in R. If Q is also disjoint of P', then the paths RP' and Q are internally vertex-disjoint (see Fig. 4(b)). If there is some vertex of Q in P', then consider x to be the common vertex in Q and P' nearest to z. So, the paths $Q[s, x]P'[x, u]$ and RQ' are internally vertex-disjoint (see Fig. 4(c)).

Case 2. There are vertices of Q in R. In this case, R could intersect with P, Q or even both. From all vertices from R that are in P or Q, consider that y is the nearest vertex to z. Without loss of generality consider that y is in Q (the case when y is in P is symmetric).

Case 2.1. There is no vertex of P in Q'. Then, the following paths are internally vertex-disjoint: P and $Q[s, y]R[y, z]Q'$ (see Fig. 5(b)).

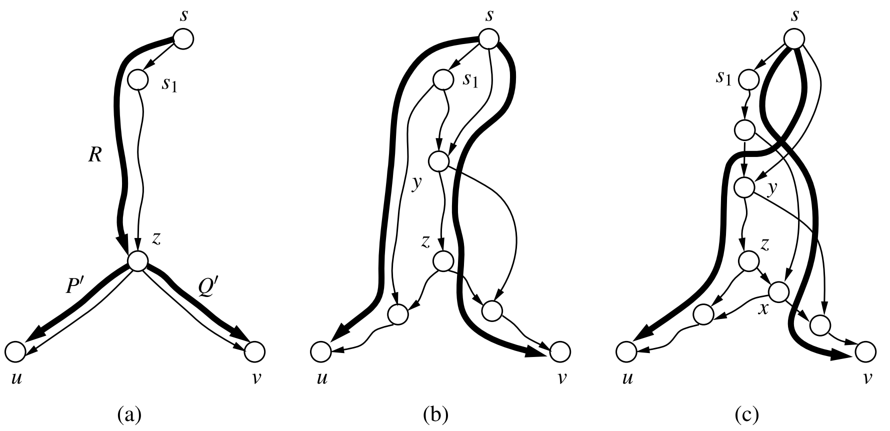


Fig. 5 A DFST and new internally vertex-disjoint paths—Lemma 2 Case 2. One of them includes vertex $z = LCA_{T^s}(u, v)$. The vertex $y \in Q$ is the nearest to z in $(P \cup Q) \cap R$

Case 2.2. There are some vertices of P in Q'. Consider x, the common vertex in P and Q' nearest to z. Then, the following paths are internally vertex-disjoint: $Q[s, y]R[y, z]P'$ and $P[s, x]Q'[x, v]$ (see Fig. 5(c)).

Therefore, if s is a junction of the pair of targets u and v, then there exists a pair of internally vertex-disjoint paths with z belonging to one of them. This ends the first part of the proof.

The converse is trivial. By definition, s is a junction of the pair of targets u and v if there is a pair of internally vertex-disjoint paths, regardless of z belonging to one of them. □

A consequence of Lemma 2 is the following corollary.

Corollary 1 *Let u and v be targets in the subtree T^{s_1} of T^s, where s_1 is a child of s in T^s. Call z the LCA of u and v in the subtree T^{s_1} and consider that z is not equal to u or v. Vertex s is a junction of the pair of targets u and v if and only if s is a junction of targets z and u or s is a junction of targets z and v.*

Proof Take a pair of vertex-disjoint paths P from s to u and Q from s to v such that z is in P or Q as given by Lemma 2. If z is in Q, then the paths $Q[s, z]$ and P are internally vertex-disjoint. So, s is a junction of targets z and u. If z is in P, then the paths $P[s, z]$ and Q are internally vertex-disjoint. So, s is a junction of targets z and v.

To prove the converse, take a pair of vertex-disjoint paths P from s to z and Q from s to v. Consider P' and Q' the paths on T^s from z to u and from z to v, respectively. Once more, we divide the proof into two cases:

Case 1. There is no vertex of Q in P'. So, the paths PP' and Q are internally vertex-disjoint (see Fig. 6(b)).

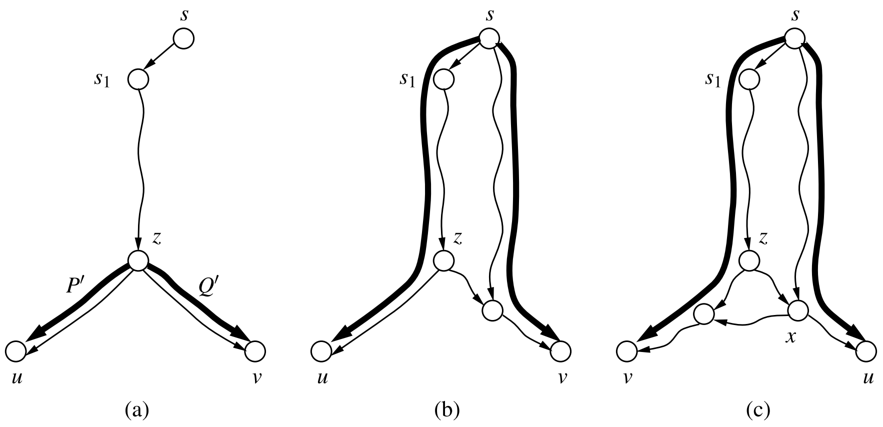


Fig. 6 A DFST and new internally vertex-disjoint paths—Corollary 1 Cases 1 and 2

Case 2. There are some vertices of Q in P'. As D is an acyclic digraph, there is no inner vertex of P in P'. Let x be the nearest common vertex of z in Q and P'. Thus, the paths $Q[s,x]P'[x,u]$ and PQ' are internally vertex-disjoint (see Fig. 6(c)).

With a similar proof we can show that it is possible to construct a pair of vertex-disjoint paths from s to u and from s to v when we take a pair of vertex-disjoint paths from s to u and from s to z. So, if s is a junction of targets z and u or s is a junction of targets z and v, then s is a junction of targets u and v. □

The previous corollary characterizes s as a junction of the pair of targets u, v when the LCA of u and v in T^s is not equal to them. In the next lemma we consider the case where the LCA is equal to one of the targets.

Lemma 3 *Let z and u be targets in a subtree T^{s_1} of T^s and let z be a proper ancestor of u in T^s, i.e., z is the LCA of z and u in T^s and z is not equal to u. Vertex s is a junction of targets z and u if and only if s is a junction of targets z and t, for some t in $\delta^-_D(u)$ and t different from z.*

Proof To prove the first part we take two internally vertex-disjoint paths Q from s to z and P from s to u. Consider the last vertex t before u in P. We know that t is in $\delta^-_D(u)$ since it is a parent of u in D and t is different from z as Q and P are vertex-disjoint paths. Therefore, the path $P[s,t]$ and Q are vertex-disjoint, and then s is a junction of targets z, t.

To prove the converse, we consider two internally vertex-disjoint paths P from s to t and Q from s to z. If u is in P, then the subpath $P[u,t]$ plus the arc $t \to u$ is a cycle. But D is an acyclic digraph. Thus, u is not in P. If u is in Q, then u is a proper ancestor of z. By assumption, z is a proper ancestor of u. Thus, we can again produce a cycle contradicting the fact that D is an acyclic digraph. Then, also u is not in Q. As t is different from z, we can use the arc $t \to u$ extending the path P

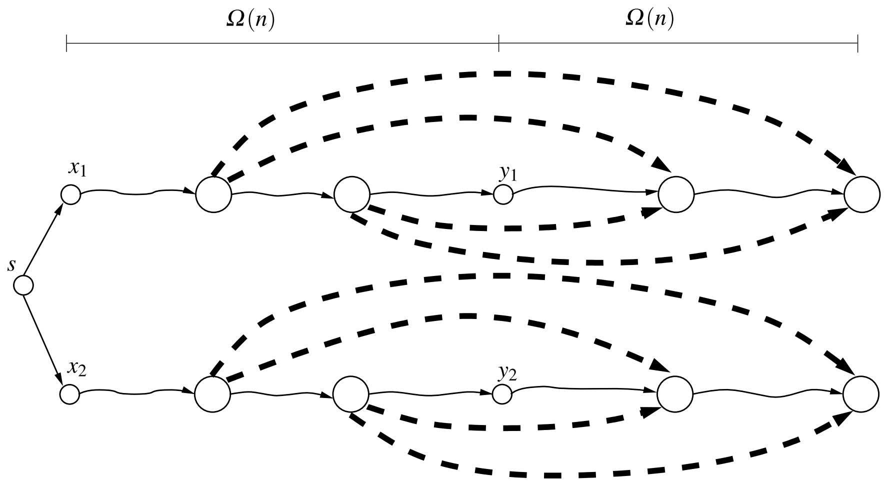


Fig. 7 There are $\Omega(n)$ vertices in the path from x_i to y_i and in T^{y_i}. There is an arc linking any vertex from this path to any proper descendant of y_i (represented by the *large dashed arcs linking large vertices*). For a quadratic number of pairs of proper descendant of y_i, the dynamic programming algorithm earlier described processes $\Omega(n)$ arcs. In this case the total time spent is $\Omega(n^3)$

and constructing two internally vertex-disjoint paths P plus the arc $t \rightarrow u$ and Q. So, s is a junction of targets z and u. □

6 Algorithms

A binary matrix $\mathbf{M^s_{n\times n}}$ is a natural way to represent all pairs of targets that have a vertex s as a junction. Vertex s is a junction of a pair u and v if and only if $M^s(u,v) = 1$. Using $\mathbf{M^s}$ and the recurrence presented in Lemma 1, a dynamic programming algorithm naturally arises. It uses a topological order of D to fill in matrix $\mathbf{M^s}$. First, it initializes $\mathbf{M^s} = \mathbf{0}$. After that, for each vertex u in D, $M^s(s,u) = M^s(u,s) = 1$. And finally, for each pair u, v in D such that u precedes v in the topological order, let $z = LCA_{T^s}(u,v)$. Do $M^s(u,v) = M^s(v,u) = 1$ if $z = s$ (this implies $u \in T^{s_1}$ and $v \in T^{s_2}$); or if $z \neq u$ and $(M^s(z,u) = 1$ or $M^s(z,v) = 1)$; or if $z = u$ and $M^s(z,t) = 1$ for some $t \in \delta^-_D(v)$. Unfortunately, this naive algorithm could spend $\Omega(n^3)$ time. As seen in Fig. 7, many arcs could be looked at by a quadratic number of pairs of vertices.

However, we can provide a better solution to the problem. We first show a compact representation of matrix $\mathbf{M^s}$ using an unidimensional array. The algorithm (SJKP) that solves the *s-junction-k-pairs* problem has two phases:

Phase 1. Construct in $O(m)$ time a partition B^s of the set of vertices of D^s that obeys the following property: *Two targets u and v are in different subset of B^s if and only if s is a junction of u and v.*

Phase 2. Solve in $O(1)$ time the *s-junction*(u_i, v_i) query for $i = 1, \ldots, k$.

The total time spent by SJKP is $O(m + k)$. We have seen that a matrix representation can be used to solve problems involving LCAs and junctions. If a matrix representation is required, then it could be obtained in $O(n^2)$ time.

Next we show how to solve Phase 1. The data structure used to represent the partition B^s as previously described is an array r, similar to a structure to represent disjoint subsets (see Cormen, Leiserson, Rivest and Stein [9]). Each set of B^s has a representative vertex, and for those vertices $r(v) = v$. For the remaining vertices $r(v)$ points to another vertex in the same set of the partition. These links lead to the representative vertex of each set. The algorithm starts constructing a DFST T^s of D^s. The array r is then initialized such that each vertex v points to its parent in T^s, and the root vertex s has $r(s) = s$. It represents a partition with a single set and s as representative. During the construction of T^s each vertex v becomes a label $post(v)$ with the index of the vertex in a post-order traversal of T^s. Also the values $minpost(v)$ are set with the minimum post-order value of a vertex in the subtree T^v of T^s. It is not difficult to construct a DFST and maintain those values. The interested reader can refer to Sedgewick [19]. The algorithm to solve Phase 1 is recursive. In each call it receives a representative vertex z and sets values of r for vertices in subtree T^z of T^s correctly. The initial calls to the recursive algorithm are done on each child s_j of s in T^s, $r(s_j) = s_j$ and $z = s_j$. In each recursive call the following cases are analyzed:

Case 1: T^z has only one vertex. Nothing must be set in array r.

Case 2: T^z has at least two vertices. Let w be the vertex such that $post(w) = post(z) - 1$. Observe that w must be initially a child of z. Consider all vertices t in $\delta^-_D(w)$. If, for some of those vertices, $r(z) \neq r(t)$ (i.e., s is a junction of the z and t) we can set w as a representative, solve the problem recursively for the subtree T^w and set w, after the end of the recursive call, to $minpost(w) - 1$. In the other case, $r(z) = r(t)$ for all t in $\delta^-_D(w)$, then (by Lemma 3) we can set $r(w)$ to z and update w to the vertex with post-order value equal to $post(w) - 1$. This is done until w is set outside T^z, i.e., $post(w) < minpost(z)$.

Note that the Phase 1 of the SJKP algorithm explores each arc in D^s at most twice. So, its time complexity is linear in m.

Now we prove that the algorithm is correct. Note that $r(u)$ is changed at most once for all u in D^s. The recursion considers a path in T^z from z to w. Let Z denote such a path. Note that, for all w' in Z, $r(w') = z$. So we have to prove that s is not a junction of targets w' and w'' for all w' and w'' in Z. It can be proven if we assume that s is not a junction of targets z and w' for all w' in Z.

Lemma 4 *If s is not a junction of targets z and w' for all w' in Z, then s is not a junction of targets w' and w'' for all w' and w'' in Z.*

Proof Proof by contradiction. Suppose that s is a junction of targets w' and w'' where w' and w'' are in Z. Consider P and Q, the internally vertex-disjoint paths from s to w' and from s to w'', respectively. Consider the path R in T^s from s to z.

Consider the vertices of R that are common to P or Q, and let z' be a such vertex nearest to z. Suppose that z' is in Q (the other case, when z' is in P, is analogous). Then, the paths $Q[s, z']R[z', z]$ and P are internally vertex-disjoint. Therefore, s is a junction of targets z and w', a contradiction. □

Consider again the path Z in T^z from z to w. Now, suppose that s is not a junction of targets z and w' for all w' in Z different from w however s is a junction of targets z and w. From the previous procedure, $r(u)$ is different from $r(v)$ for all targets u in $T^z \setminus T^w$ and v in T^w. It implies that s is a junction of targets u and v. The next lemma proves it in two steps.

Lemma 5 *If s is not a junction of targets z and w' for all w' in Z different from w and s is a junction of targets z and w, then*

a. *s is a junction of targets w' and w for all w' in Z; and*
b. *s is a junction of targets u and v for all targets u in $T^z \setminus T^w$ and v in T^w.*

Proof

a. Consider paths P from s to w and Q from s to z that are internally vertex-disjoint. We know that the only common vertex in Q and Z is z, since Z contains descendants of z and Q contains ancestors of z. Note also that the only common vertex in P and Z is w. If it is not the case, then s is a junction of targets z and w' for some w' different from w in Z, opposing our assumption. So we can extend the path Q using the path Z from z up to any w' in Z and then s is a junction of targets w' and w because the paths $QZ[z, w']$ and P are internally vertex-disjoint, for any w' in Z.
b. Take u in $T^z \setminus T^w$ and v in T^w. Let w'' be the LCA in T^s of targets u and v. By Lemma 5(a), we have, in particular, that s is a junction of w'' and w. Therefore, there exist internally vertex-disjoint paths P from s to w'' and Q from s to w. Note that we can extend the path Q adding the path on T^w from w to v and this new path, denoted by R, does not intersect P or $Q \setminus \{w\}$. Otherwise, there would be an ancestor and descendant vertex of w. Thus, the only common vertex in R and P is s. Therefore, we have that s is a junction of w'' and v. If u is equal to w'', then we are done. If u is different from w'', then by Corollary 1, we have that s is a junction of u and v. □

Still following the algorithm, we can easily prove some properties on vertex z and the vertices that are not in subtree T^z.

Proposition 1 *Consider a call to the recursive procedure where initially $z = s_i$ for some child s_i of s in T^s. In any recursive call (with z as parameter) derived from this initial call, the following assertions are true:*

- *Vertex z is a representative,*
- *For all vertices $x \in T^{s_i} \setminus T^z$, $r(z) \neq r(x)$.*

Proof The first assertion is true as in the first call $z = s_i$ is a representative and before any recursive call of the procedure passing w as parameter, we make $r(w) = w$.

The second assertion is true as for any vertex $v \in T^s$, $r(v)$ points to a representative that is its ancestor in T^s. By the first assertion, $r(z) = z$. But the vertex z is not an ancestor of x. Therefore, $r(z) \neq r(x)$. □

Now we are going to prove the correctness of the recursive procedure.

Lemma 6 *Consider a call to the recursive procedure where initially z is set to some child s_i of s in T^s. For all vertices u and v in T^{s_i} such that* $\min\{post(u), post(v)\} \geq post(w) + 1$ *the assertions are true*:

- *$r(u)$ and $r(v)$ point to a representative,*
- *s is a junction of targets u and v if and only if $r(u) \neq r(v)$.*

Proof The former assertion is true since $r(w)$ points either to z or to w (when it is a representative).

Finally, the last assertion is proven by induction on the number of comparisons "$post(w) \geq minpost(z)$" done. In the first comparison we have $z = s_i$ and w is the child of the greatest post-order value, $post(w) = post(s_i) - 1$. This means that s_i is the unique vertex in T^{s_i} with post-order value greater than $post(w)$. The vertex s is not a junction of the pair of targets s_i and s_i and $r(s_i) = s_i$. Now suppose that this assertion is true in the i-th comparison $post(w) \geq minpost(z)$, and let us prove that this assertion remains true in the next comparison. Consider the path Z in T^z from z to w in the i-th comparison. This path was considered in iterations before. By the order that the vertices are being processed and Lemma 3, we know that s is not a junction of targets z and w' for all w' in Z different from w. We know that, $\min\{post(z), post(t)\} \geq post(w) + 1$, for z and for all t in $\delta^-_D(w)$. Thus the assertion is true for the pair of targets z and t, and therefore s is a junction of z and t if and only if $r(z) \neq r(t)$. Applying Lemma 3 we have two possibilities:

Case 1. s is not a junction of targets z and t, for all t in $\delta^-_D(w)$. So, $r(z) = r(t)$, for all t in $\delta^-_D(w)$ and by Lemma 3, s is not a junction of targets z and w. In this case, the algorithm sets $r(w) = z$. Take x in T^{s_i} with $post(x) \geq post(w) + 1$. Consider y to be the LCA in T^s of targets w and x (possibly $x = y$). Let us show that exactly before we update w to the next vertex, s is a junction of targets x and w if and only if $r(x) \neq r(w)$. Suppose x in $T^{s_i} \setminus T^z$. Then, y is some vertex in the path from s_i to z in T^{s_i} and, by the second assertion in Proposition 1, $r(y) \neq r(z)$ and $r(x) \neq r(z) = z = r(w)$. Therefore, we have to show that s is a junction of targets x and w. We have that s is a junction of targets y and z since $r(y) \neq r(z)$. As D is an acyclic digraph, we know that s is a junction of targets y and w because we can use the path from z to w in T^{s_i}. If $y = x$, then we are done. Otherwise, by Corollary 1, we have that s is a junction of targets x and w. Now suppose x is in T^z. So vertex y is in Z, i.e., the LCA in T^s of w and x is in Z (possibly $x = y$).

Case 1.1. s is not a junction of targets x and z. In this case, $r(x) = r(z)$. Moreover, $r(w) = z = r(z) = r(x)$. So we have to show that s is not a junction of targets x and w. As mentioned, s is not a junction of targets z and w', where w' is in Z. By

Lemma 4, in particular s is not a junction of targets y and w and $r(y) = r(w)$. Thus, if $x = y$, then s is not a junction of targets x and w. If $x \neq y$, then by induction s is not a junction of targets y and x as $r(y) = r(w) = r(x)$. Finally, by Corollary 1, s is not a junction of targets x and w because s is not a junction of targets y and w, and y and x.

Case 1.2. s is a junction of targets x and z. Then, $r(x) \neq r(z)$. Moreover, $r(w) = z = r(z) \neq r(x)$. So we have to show that s is a junction of targets x and w. As y is in Z, we know that s is not a junction of targets z and y. Then our assertion says $r(z) = r(y)$. So, $r(y) = r(z) \neq r(x)$. It means that s is a junction of targets y and x. By Corollary 1, s is a junction of targets x and w.

After that, the algorithm updates w to be the vertex with post-order value equal to $post(w) - 1$ and then our assertion is restored.

Case 2. s is a junction of targets z and t, for some t in $\delta^-_D(w)$. So, $r(z) \neq r(t)$, for some t in $\delta^-_D(w)$ and by Lemma 3, s is a junction of targets z and w. In this case, we set $r(w) = w$. Take $x \in T^{s_i}$ with $post(x) \geq post(w) + 1$. We are going to show that exactly before we perform the recursive call, s is a junction of targets x and w if and only if $r(x) \neq r(w)$. As w is not an ancestor of x in T^s and $r(x)$ points to a representative that is its ancestor in T^s, we have $r(x) \neq r(w)$. Thus, we have to show that s is a junction of targets x and w. If x is in $T^z \setminus T^w$, then Lemma 5(b) ensures that s is a junction of targets x and w. If x is in $T^{s_i} \setminus T^z$, then y, the lowest common ancestor of targets x and w in T^s, is a vertex in the path from s_i to z in T^{s_i} (possibly $x = y$). By the second assertion in Proposition 1, $r(y) \neq r(z)$. Thus, s is a junction of targets y and z, and we can use the path from z to w in T^{s_i} to obtain that s is a junction of targets y and w. If $x = y$, then we are done. If $x \neq y$, then, by Corollary 1, s is a junction of targets x and w. After the recursive call, we have that s is a junction of targets u and v if and only if $r(u) \neq r(v)$ for all targets u and v in T^{s_i}, and the post-order values of vertices u and v are greater than or equal to the post-order value of the vertex $minpost(w)$. After that, we set w to be the vertex with post-order value equal to $minpost(w) - 1$ and then our assertion is restored. □

Now consider Phase 2, i.e., we want to solve in $O(1)$ time an s*-junction*(u_i, v_i) query for $i = 1, \ldots, k$. It is easily done by verifying the values of array r, and remembering that s is a junction of u_i and v_i if and only if $r(u_i) \neq r(v_i)$. Now we can state the following result.

Theorem 1 *Given an acyclic digraph D with n vertices and m arcs, and a vertex s in D, we can construct a data structure for disjoint sets in time $O(m)$, and answer in constant time for any pair of targets u and v in D, if s is a junction of targets u and v. Moreover, after such construction, all k pairs of vertices that have s as a junction can be printed out in $O(k)$ time.*

We have solved the s*-junction-k-pairs* problem in linear time. Observe that we can further solve the k*-pairs-all-junctions* problem in $O(n(m + k))$ time and $O(m + k)$ space if we use the results of Theorem 1 for each s in D and we print out all the junctions during the process.

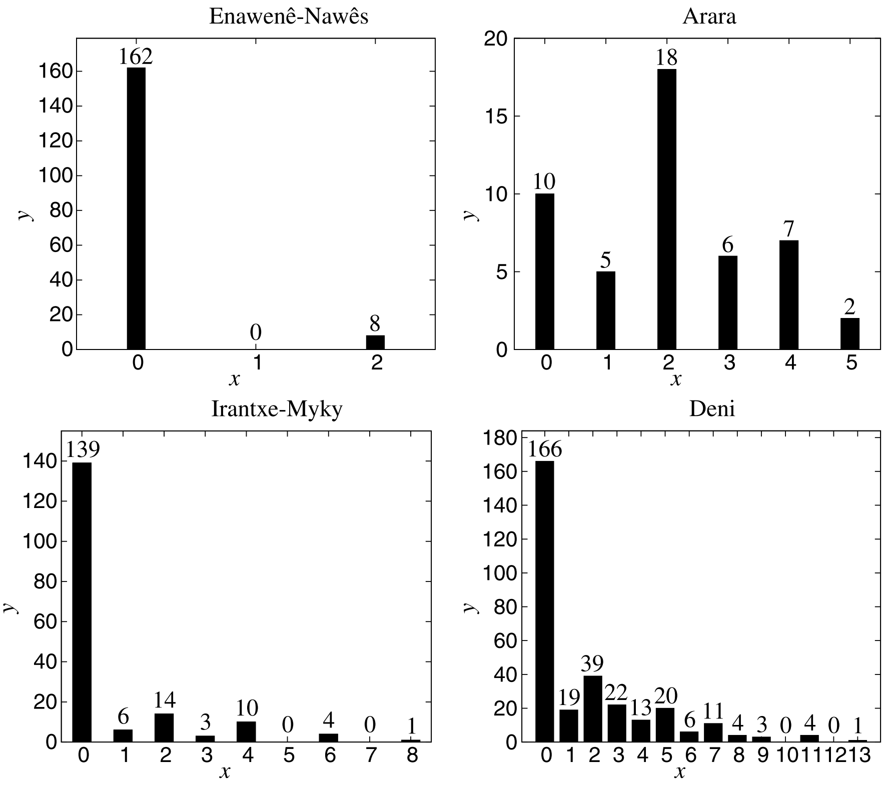


Fig. 8 Statistical information about the number of marriages with a fixed number of junctions for the Enawenê-Nawês, Arara, Irantxe-Myky and Deni kinship networks

7 Experiments

Let us consider again our application in Anthropology brought to our attention by some friends from that department. Given a kinship network D, and k marriages $\{u_1, v_1\}, \ldots, \{u_k, v_k\}$ between individuals in D, one wants to list for each marriage $\{u_i, v_i\}$ all the individuals that are junctions of u_i and v_i. This is the *k-pairs-all-junctions* problem. Four real kinship networks from Brazilian Amazon region were available for these experiments. The Enawenê-Nawês network has 789 vertices (individuals), 1368 arcs (parental relationship) and 170 marriages, the Arara network has 105 vertices, 197 arcs and 48 marriages, the Irantxe-Myky network has 618 vertices, 1003 arcs and 177 marriages and the Deni network has 788 vertices, 1380 arcs and 308 marriages. The anthropologists previously solved this problem initially by hand and, when it became impossible, they developed a "kinship machine" [11] using a database system to answer those queries. This machine could only solve the Enawenê-Nawês instance, and therefore they looked for help in the Computer Science Department. Our approach was able to find all junctions of the k given marriages for all the kinship networks available. We implemented it in C++

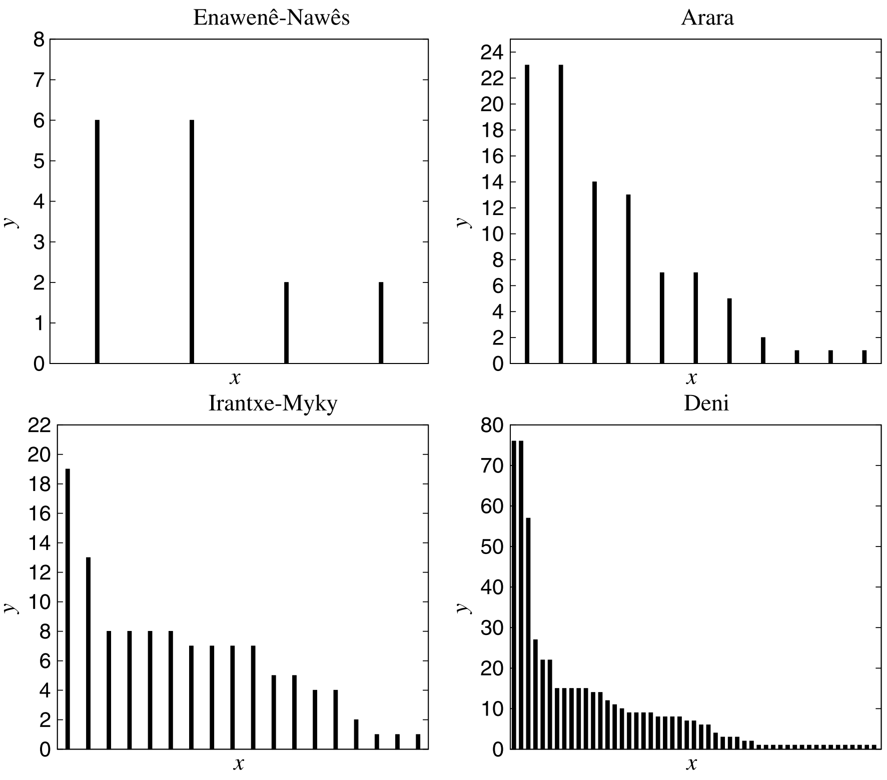


Fig. 9 Statistical information about individuals that participate most as a junction of marriages for the Enawenê-Nawês, Arara, Irantxe-Myky and Deni kinship networks

and the resulting code is available on request. We ran the experiments on a 32-bit Ubuntu 12.04 machine with 8 GB of RAM and a Intel Core i7-2640M CPU (2.80 GHz × 4). All the instances were solved instantaneously and the time and space spent were insignificant. In Fig. 8 there are four charts. A number in the x-axis means a fixed number of junctions ($0 \le x \le n$) and a number in the y-axis means a fixed number of pairs ($0 \le y \le k$). It gives us an idea of how many marriages a certain number of junctions has. For instance, in the Enawenê-Nawês chart 162 marriages do not have junctions and 8 marriages have 2 junctions. It implies that almost all Enawenê-Nawês marriages do not link relatives. This is also the case for Irantxe-Myky and, in some extent, for Arara. For Deni, however, some marriages have up to 13 junctions. It has some anthropological implication due to how are the taboos or marriage preferences in those societies.

In Fig. 9 we present another statistical information. A number in the x-axis means an individual in the network ($0 \le x \le n - 1$) and a number in the y-axis means a fixed number of pairs ($0 \le y \le k$). It gives us an idea of how many times an individual appears as a junction of marriages, i.e., how often some individual is in alliance position. We ordered the x-axis in decreasing order of number of marriages. We did

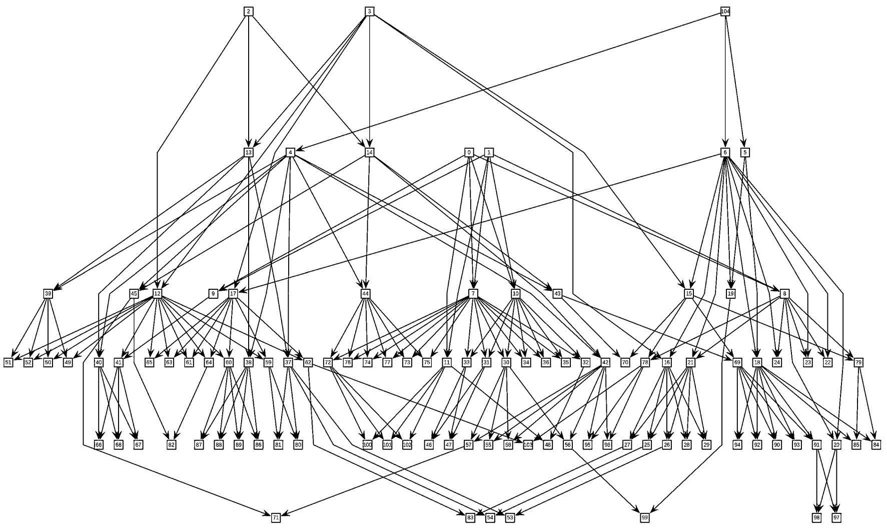


Fig. 10 Arara kinship network. Note a junction 4 (*second vertex in the second level*) for the marriage 45 (*second vertex in the third level*) and 57 (*in the middle of the fifth level*)

not consider individuals that are not a junction. For instance, in the Enawenê-Nawês chart only four individuals are junctions. Individuals 0 and 1 are junctions of 6 marriages, and individuals 2 and 3 are junctions of 2 marriages. In the Deni network, there are at least 50 individuals that are junctions of marriages and there are two individuals that are junctions of more than 70 marriages. Again, we observe that the Enawenê-Nawês presents a simpler pattern, while the Irantxe-Myky and the Deni are more complexes. It also corroborates some conjectures of the anthropologists on the behavior of those groups.

Figure 10 presents one of those networks. It has been done with support of the Open Graph Drawing Framework with optimization (OGDF with CPLEX 12.1.0) [7]. As an example, we can see in Fig. 10 a good drawing of the Arara kinship network.

8 Conclusion

We showed how to solve in linear time the following path problem on digraphs. Given an acyclic digraph D, a vertex s and k pairs of targets $\{u_1, v_1\}, \ldots, \{u_k, v_k\}$, determine the pairs of targets $\{u_i, v_i\}$ for which there exist paths from s to u_i and from s to v_i in D that are internally vertex-disjoint. If such paths exist for a pair of targets, vertex s is called a junction of such a pair. This problem (called *s-junction-k-pairs* problem) was used to solve a related problem that stems from some application in Anthropology (*k-pairs-all-junctions* problem). Given an acyclic digraph D, and k pairs of targets $\{u_1, v_1\}, \ldots, \{u_k, v_k\}$, preprocess D so that for any pair $\{u_i, v_i\}$, a query for all junctions of u_i and v_i can be answered in the time of the number of junctions of u_i and v_i. We applied our approach to solve the *k-pairs-all-junctions* problem for four real kinship networks: Enawenê-Nawês, Arara, Irantxe-Myky and Deni. Our approach worked very well on all four networks. At last, some statistical information concerning those kinship networks were presented. This is an ongoing collaboration. Other interesting problems arise on kinship networks such as searching for the so-called k-rings [13] or simulation of networks based on anthropological premises (incest taboo, crossing cousin preference, etc).

Acknowledgements C.E. Ferreira is supported by CNPq (Proc. 302736/10-7 and 477203/12-4). A.J.P. Franco is supported by CAPES (Proc. 33002010176P0). We want to thank Professor Marcio Ferreira da Silva (Anthropology Department, University of São Paulo) who brought this problem (and other related to kinship networks) to our attention and provided us the data used in these experiments. We also thank the anonymous referees for their constructive comments.

References

1. Aho, A.V., Ullman, J.D.: The Theory of Parsing, Translation, and Compiling. Vol. I: Parsing. Prentice-Hall Series in Automatic Computation. Prentice-Hall, Englewood Cliffs (1972)
2. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: On finding lowest common ancestors in trees. SIAM J. Comput. **5**(1), 115–132 (1976)
3. Alstrup, S., Harel, D., Lauridsen, P.W., Thorup, M.: Dominators in linear time. SIAM J. Comput. **28**(6), 2117–2132 (1999)
4. Baumgart, M., Eckhardt, S., Griebsch, J., Kosub, S., Nowak, J.: All-pairs ancestor problems in weighted dags. In: Chen, B., Paterson, M., Zhang, G. (eds.) ESCAPE. Lecture Notes in Computer Science, vol. 4614, pp. 282–293. Springer, Berlin (2007)
5. Bender, M.A., Farach-Colton, M., Pemmasani, G., Skiena, S., Sumazin, P.: Lowest common ancestors in trees and directed acyclic graphs. J. Algorithms **57**(2), 75–94 (2005). doi:10.1016/j.jalgor.2005.08.001
6. Berkman, O., Vishkin, U.: Finding level-ancestors in trees. J. Comput. Syst. Sci. **48**(2), 214–230 (1994). doi:10.1016/S0022-0000(05)80002-9
7. Chimani, M., Gutwenger, C., Jünger, M., Klau, G.W., Klein, K., Mutzel, P.: The open graph drawing framework (OGDF). In: Handbook of Graph Drawing and Visualization. CRC Press, Boca Raton (2013)
8. Cole, R., Hariharan, R.: Dynamic LCA queries on trees. SIAM J. Comput. **34**(4), 894–923 (2005). doi:10.1137/S0097539700370539
9. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press, Cambridge (2009)
10. Czumaj, A., Kowaluk, M., Lingas, A.: Faster algorithms for finding lowest common ancestors in directed acyclic graphs. Theor. Comput. Sci. **380**(1–2), 37–46 (2007). doi:10.1016/j.tcs.2007.02.053
11. dal Poz, J., da Silva, F.M.: Maqpar: a homemade tool for the study of kinship networks. Vibrant **6**(2), 29–51 (2009)
12. Eckhardt, S., Mühling, A.M., Nowak, J.: Fast lowest common ancestor computations in dags. In: Arge, L., Hoffmann, M., Welzl, E. (eds.) ESA. Lecture Notes in Computer Science, vol. 4698, pp. 705–716. Springer, Berlin (2007)
13. Ferreira, C.E., Franco, Á.J.P.: Finding rings in genealogies: computational complexity and algorithms. In preparation
14. Georgiadis, L., Tarjan, R.E.: Finding dominators revisited: extended abstract. In: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'04, pp. 869–878. SIAM, Philadelphia (2004)

15. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum flow problem. In: Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing, STOC'86, pp. 136–146. ACM, New York (1986). doi:10.1145/12130.12144
16. Grötschel, M.: On minimal strong blocks. J. Graph Theory **3**, 213–219 (1979)
17. Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestors. SIAM J. Comput. **13**(2), 338–355 (1984). doi:10.1137/0213024
18. Nykänen, M., Ukkonen, E.: Finding lowest common ancestors in arbitrarily directed trees. Inf. Process. Lett. **50**(6), 307–310 (1994). doi:10.1016/0020-0190(94)00050-6
19. Sedgewick, R.: Algorithms in C—Part 5: Graph Algorithms, 3rd edn. Addison-Wesley, Reading (2002)
20. Suurballe, J.W., Tarjan, R.E.: A quick method for finding shortest pairs of disjoint paths. Networks **14**(2), 325–336 (1984). doi:10.1002/net.3230140209
21. Tholey, T.: Finding disjoint paths on directed acyclic graphs. In: Kratsch, D. (ed.) WG. Lecture Notes in Computer Science, vol. 3787, pp. 319–330. Springer, Berlin (2005)
22. Wen, Z.: New algorithms for the LCA problem and the binary tree reconstruction problem. Inf. Process. Lett. **51**(1), 11–16 (1994). doi:10.1016/0020-0190(94)00058-1
23. Williams, V.V.: Multiplying matrices faster than Coppersmith-Winograd. In: Proceedings of the 44th Symposium on Theory of Computing, STOC'12, pp. 887–898. ACM, New York (2012). doi:10.1145/2213977.2214056
24. Yuster, R.: All-pairs disjoint paths from a common ancestor in $\tilde{O}(n^{\omega})$ time. Theor. Comput. Sci. **396**(1–3), 145–150 (2008). doi:10.1016/j.tcs.2008.01.032

Algorithms for Scheduling Sensors to Maximize Coverage Time

Rafael da Ponte Barbosa and Yoshiko Wakabayashi

Abstract We study a one-dimensional sensor cover problem, known as the Restricted Strip Cover (RSC) problem, defined as follows. We are given an interval U of the real line, and a set of n sensors, each of which covers some subinterval of U and is powered with a battery of limited duration. The RSC problem consists in finding a scheduling of the sensors (that is, an assignment of the activating times of the given sensors) so that the whole interval U is covered for as long as possible. We investigate two variants of this problem: one denoted simply as RSC, the non-preemptive variant; and the other, denoted as RSCP, the preemptive variant. In the first, each sensor can be activated at most once and it remains on through the duration of its battery. In the second variant, preemption is allowed, that is, each sensor can be activated and deactivated many times along the duration of its battery. Buchsbaum, Efrat, Jain, Venkatasubramanian and Yi showed that RSC is NP-hard and designed an $O(\log\log n)$-approximation algorithm. More recently, Gibson and Varadarajan presented a greedy-like algorithm which they proved to have approximation ratio at most 5. We prove that the approximation ratio of this algorithm is 4, and exhibit an instance showing that this ratio analysis is tight. We also show an integer programming formulation for this problem and present some computational results obtained with the implementation of this approach. For the same set of instances, we compute the quality of the solution found by the approximation algorithm. For the preemptive variant RSCP, we present an exact polynomial-time algorithm.

1 Introduction

Martin Grötschel has visited the University of São Paulo, Brazil, many times. His first visit happened in 1977, a year after he obtained his doctoral degree, when he

R. da Ponte Barbosa (✉) · Y. Wakabayashi
Instituto de Matemática e Estatística, Universidade de São Paulo, Rua do Matão, 1010, Cidade Universitária, 05508-090, São Paulo, SP, Brazil
e-mail: rafaelb@ime.usp.br

Y. Wakabayashi
e-mail: yw@ime.usp.br

was interested in the study of the polytope associated with the TSP problem. He conjectured that hypohamiltonian digraphs and also hypotraceable digraphs would induce facets of the asymmetric TSP polytope. To establish this, he was interested in finding such digraphs, whose existence was not known at that time. This is the first problem M. Grötschel proposed to Y. Wakabayashi, and this is how their collaboration started. He was right: after finding infinite families of such digraphs, it was possible to prove that many of them indeed induce facets of the asymmetric TSP polytope. Five years later, Y. Wakabayashi went to Germany for her doctoral studies under his supervision. That was in the end of 1982, when M. Grötschel was already leaving the University of Bonn for a *Lehrstuhl* at the University of Augsburg.

The years in Augsburg were very exciting: M. Jünger and G. Reinelt were already there, working on the acyclic subgraph and the linear ordering problems. The cutting plane methods were flourishing, and it was a privilege to have as advisor *the* expert in this topic. This was also when the book "Geometric Algorithms and Combinatorial Optimization" was in preparation, and therefore L. Lovász and A. Schrijver were visiting quite often, making those years very special. This unique environment in Augsburg and the research produced there shaped the careers of many. Y. Wakabayashi is grateful for having had the chance of being there at that time, and feels privileged for the continued friendship of more than 30 years.

The first author, R. da Ponte Barbosa, obtained his M.Sc. degree at the University of São Paulo, in 2011, under the supervision of the second author. Now, on the occasion of the 65th birthday of M. Grötschel, both congratulate him, most wholeheartedly, for all his achievements and profound contributions to the area of combinatorial optimization and much beyond.

2 The Problem

The problem we focus here was introduced by Buchsbaum, Efrat, Jain, Venkatasubramanian and Yi [3] as the *sensor cover problem*. We consider the one-dimensional case, known as the RESTRICTED STRIP COVER (RSC) problem, and study the non-preemptive and the preemptive variants. The first, we denote simply as RSC, and the second, as RSCP.

We start with the non-preemptive variant. An informal definition of the problem RSC is the following.

Suppose we have a fence to be monitored and a set of sensors placed at various fixed locations, each one with a battery of limited duration. Knowing the battery duration of each sensor and which is the section of the fence each sensor covers, in the RSC problem the objective is to schedule the time each sensor has to be turned on so that the fence is fully monitored for as long as possible. In this variant, once a sensor is turned on, it remains on through the duration of its battery. See an example in Fig. 1.

We present now a formal definition of the RSC problem, and introduce all concepts that are needed to present the main results of this paper. We adopt (most of)

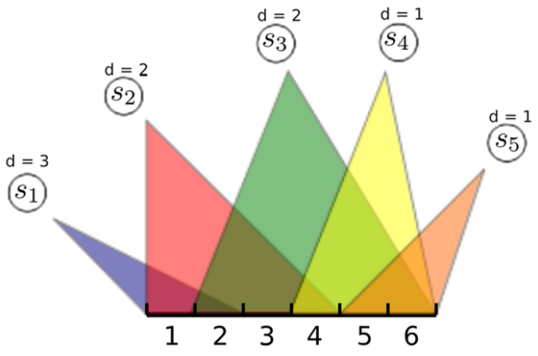


Fig. 1 An instance of the Restricted Strip Cover problem, in which sensor s_1 has duration 3, sensors s_2 and s_3 have duration 2, and sensors s_4 and s_5 have duration 1

the notation and terminology that were used by Buchsbaum et al. [3] and Gibson and Varadarajan [7].

An instance of the RSC problem consists of an interval $U = \{1, \dots, m\}$ of the real line, a set of sensors $S = \{s_1, s_2, \dots, s_n\}$, and for each sensor s in S, a positive integer $d(s)$ and a range $R(s) = \{l(s), l(s) + 1, \dots, r(s)\}$, which is a subinterval of U. The value $d(s)$, called the *duration* of sensor s, is the amount of time the sensor s remains on (once it is turned on). For simplicity, we assume that such an instance consists of a pair (U, S).

For each i in $R(s)$, we say that s is *live* at i. Note that we may assume w.l.o.g. that $m \le 2n$ because n sensors give rise to at most $2n$ distinct subinterval endpoints.

A *schedule* A of a set of sensors S is an assignment of a start time $t(s)$, which is a positive integer, to each sensor s in a subset of sensors in S. With respect to a schedule A, a sensor s is said to be *active* at times $\{t(s), t(s) + 1, \dots, t(s) + d(s) - 1\}$. A point $i \in U$ is said to be *covered* at time $t > 0$ if there is some sensor live at i and active at time t. Further, we say that an interval of U is covered at time t if all points in U are covered at time t.

The *duration of a schedule A at a point i* is defined as

$$M(A, i) = \max\bigl\{t : \text{for all } t' \le t, \exists s \in A \text{ that covers } i \text{ at time } t'\bigr\},$$

and the *duration of a schedule A* is defined as $M(A) = \min_i M(A, i)$.

The RSC problem can now be stated as the problem of finding a maximum duration schedule for a given instance (U, S).

The *load at a point $i \in U$*, denoted as $L(i)$, is the sum of the duration of every sensor that is live at i, that is, $L(i) = \sum_{s \in S, s \text{ live at } i} d(s)$. The *load of an instance* (U, S) is defined as $L(U, S) = \min_i L(i)$. We denote by $\mathrm{OPT}(U, S)$ the *duration of an optimal schedule* for the instance (U, S). When (U, S) is clear from the context, we write simply L and OPT, instead of $L(U, S)$ and $\mathrm{OPT}(U, S)$. Clearly, L is an upper bound for OPT, that is, $\mathrm{OPT} \le L$. This bound will be used in the analysis of the approximation algorithm to be presented in Sect. 3.

The RSC problem can be viewed as a problem of covering a largest possible region using rectangles that may be slided only vertically. This way of viewing the RSC problem helps in devising an algorithm for it. For that, consider that each sensor s is a rectangle of base $|R(s)|$ and height $d(s)$, and think that initially each

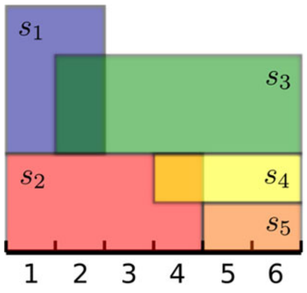


Fig. 2 A schedule in which sensors s_2 and s_5 are turned on at time 1; sensor s_4 at time 2; sensors s_1 and s_3 at time 3, covering the whole fence for four time units

rectangle s is placed (in the plane) in such a way that its base is parallel to a horizontal line segment that has length $|U|$, and occupies the subinterval $R(s)$ of U. We may think that initially all rectangles are placed in (the horizontal) level 1. In this way, an assignment of start time $t(s)$ to a sensor s corresponds to a placement of the rectangle representing s with its basis aligned with the horizontal level (height) t. Thus the problem consists in sliding vertically the given rectangles so as to obtain a rectangular region with basis U and maximum height, fully contained in their union. Figure 2 illustrates this concept for the instance in Fig. 1 and the schedule: $t(s_1) = 3$, $t(s_2) = 1$, $t(s_3) = 3$, $t(s_4) = 2$, $t(s_5) = 1$.

The RSC problem was introduced in 2007 by Buchsbaum et al. [3]. These authors showed that this problem is NP-hard by exploring a similarity with the DYNAMIC STORAGE ALLOCATION problem [2, 6], and presented a non-constant ratio approximation algorithm for it. In 2008, in an e-print [4] submitted to the ARXIV, these authors showed an $O(\log\log n)$-approximation algorithm for this problem. They also presented a $(2+\varepsilon)$-approximation algorithm for the case all sensors have ranges of the same length, and showed that when all sensors have the same duration there is a simple greedy algorithm that finds an optimal solution.

In 2009, Gibson and Varadarajan [7] presented a 5-approximation algorithm for the RSC problem. It is a rather simple algorithm that can be implemented to run in $O(n^2)$ time.

In Sect. 3, we describe the algorithm of Gibson and Varadarajan, and present a more detailed analysis of this algorithm, proving that its approximation ratio is, in fact, 4. We also exhibit instances showing that the ratio analysis we present is tight. Our analysis starts similar to that presented by Gibson and Varadarajan, but it has an additional part which proves that two certain situations cannot occur simultaneously. The proof of this additional part is crucial to obtain the better approximation ratio, and also to show that the ratio cannot be improved.

In Sect. 4 we present an integer programming formulation for the RSC problem and report on some computational results obtained with this approach.

In Sect. 5, we consider the RSCP problem, the *preemptive* version of the problem RSC. For this problem we present a rather simple exact polynomial-time algorithm.

A preliminary version of this paper appeared in a volume of LNCS dedicated to LATIN 2012 [5]. This paper presents complete proofs and an additional part on the preemptive variant, which has not been presented elsewhere.

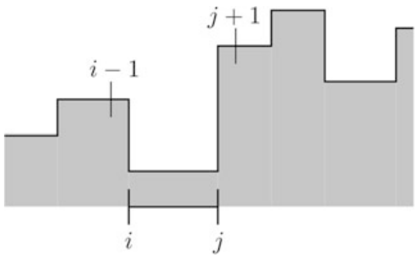


Fig. 3 Interval $[i, j]$ corresponding to the deepest uncovered "valley"

3 The Approximation Algorithm for RSC and Its Analysis

In this section we describe the algorithm designed by Gibson and Varadarajan [7] for the RSC problem and analyze its performance. Before, we introduce some definitions and conventions.

We denote by A the current schedule of the algorithm at any stage of its execution. In the construction of a schedule A, whenever a start time $t(s) > 0$ is assigned to a sensor s, we say that s is *scheduled* or *assigned to* A and write $A \leftarrow A \cup \{s\}$. Thus, a schedule A is seen as a subset of sensors (together with a start time of each sensor in A). A sensor not assigned to A is *unassigned.*

With respect to a schedule A, and a position $i \in U$, we say that an unassigned sensor s *dominates* i *to the right* if s is live at i and has the largest $r(s)$ among all unassigned sensors live at i. In case of a tie, we take the sensor that has the smallest $l(s)$. Further ties may be broken arbitrarily, but we will assume that the sensor with the smallest index is taken, so that we may refer to *the* sensor that dominates i to the right, and also to *the* schedule returned by the algorithm. The sensor that *dominates* i *to the left* is defined analogously (in a symmetric way). We also define $M(A, 0) = M(A, m+1) = \infty$.

The main idea behind the algorithm of Gibson and Varadarajan is the following. At each iteration, for a time t—corresponding to the lowest uncovered level—the algorithm considers the first uncovered point i and the largest point j such that $[i, j]$ is uncovered. The interval $[i, j]$ defines (geometrically) a "deepest valley" considering all the sensors which have been assigned so far, as illustrated in Fig. 3. For such $[i, j]$, an unassigned sensor s that dominates i to the right is considered. (If such a sensor does not exist, then the algorithm halts.) If s is live at j and the left side of the valley (position $i-1$) is lower than its right side (position $j+1$), as shown in Fig. 3, then a sensor s' that dominates j to the left is selected; otherwise, s is selected.

The method is more formally described in Algorithm 1.

3.1 Approximation Ratio of the Algorithm

We present now a proof that the algorithm RSC-GV is, in fact, a polynomial-time 4-approximation for the RSC problem. The first part of the analysis is similar to

Algorithm 1: RSC-GV

```
Input  : a pair $(U, S)$
Output: a schedule $A$ of $S$
1  $t \leftarrow 0$
2  $A \leftarrow \emptyset$; $M(A) = 0$
3  while TRUE do
4      $t \leftarrow M(A) + 1$
5      $i \leftarrow$ the leftmost uncovered point at time $t$
6      $j \leftarrow \max\{j' \in U : [i, j']$ is uncovered at time $t\}$
7      $s \leftarrow$ the sensor that dominates $i$ to the right        /* $s$ is right going */
8      if $s$ does not exist then
9          break
10     end
11     if $s$ is live at $j$ and $M(A, i-1) < M(A, j+1)$ then
12         $s' \leftarrow$ the sensor that dominates $j$ to the left        /* $s'$ is left going */
13         $A \leftarrow A \cup \{s'\}$; $t(s') \leftarrow t$
14     else
15         $A \leftarrow A \cup \{s\}$; $t(s) \leftarrow t$
16     end
17 end
18 return $A$
```

the one presented by the authors in [7]. For completeness and ease of reading, we reproduce two lemmas, stated in the sequel, both given in the aforementioned paper.

Lemma 1 (Gibson and Varadarajan, 2009) *Let A be the schedule returned by the algorithm* RSC-GV *applied to an instance (U, S). Let s' and s'' be two distinct sensors that were assigned to A. If $R(s'')$ is strictly contained in $R(s')$, then s'' is assigned to A after s', and furthermore, $t(s'') \geq t(s') + d(s')$.*

Proof At some point of the execution of the algorithm RSC-GV, let s' and s'' be two distinct unassigned sensors such that $R(s'')$ is strictly contained in $R(s')$. In each iteration, the algorithm chooses an unassigned sensor that dominates some position, say i, to the left or to the right.

Consider the iteration at which the algorithm assigns sensor s'' to A, and let $i \in R(s'')$ be the point that is considered in this moment (to be dominated to the right or to the left). Since both s' and s'' are live at i, if s' has not been assigned to A, then s' has to be chosen before s'' because of the definition of domination. This way, s' will be assigned to A before s'' and the algorithm will not consider another sensor to dominate any point in $R(s'')$ until after time $t(s') + d(s')$. □

For any point $i \in U$ and time $t > 0$, we define *coverage*(i, t) to be the number of sensors that cover i at time t in a schedule returned by the algorithm RSC-GV. We define the *MaxCoverage* of a schedule A, denoted MaxCoverage(A), as the value $\max\{\text{coverage}(i,t) : i \in U \text{ and } t > 0\}$. The duration of a schedule A is related to the MaxCoverage(A) as follows.

Lemma 2 (Gibson and Varadarajan, 2009) *Let A be a schedule returned by the algorithm* RSC-GV. *If* $\text{MaxCoverage}(A) \le c$, *then* $M(A) \ge \text{OPT}/c$.

Proof Let (U, S) be an instance of the RSC problem, and let A be the schedule returned by the algorithm RSC-GV when applied to (U, S). At the end of the execution, there is a point $i' \in U$ such that $M(A, i') = M(A)$, and furthermore, there are no unassigned sensors that are live at i'. Thus, $cM(A) = cM(A, i') \ge L(i') \ge L \ge \text{OPT}$, and therefore, $M(A) \ge \text{OPT}/c$. □

Gibson and Varadarajan proved that the algorithm RSC-GV returns a schedule A such that $\text{MaxCoverage}(A) \le 5$, showing this way the ratio 5 achieved by this algorithm. We will prove that $\text{MaxCoverage}(A) \le 4$, by doing a more careful analysis, based on the times and the intervals for which the sensors are scheduled. For that, we need a few more definitions.

At each iteration, for a time t, the algorithm considers an uncovered point i and a largest point j such that $[i, j]$ is uncovered (see steps 5 and 6). The interval $[i, j]$ defines (geometrically) a "deepest valley" considering all the sensors which have been assigned so far, as illustrated in Fig. 3. For such $[i, j]$, an unassigned sensor s that dominates i is chosen and assigned to A. In this case we say that $[i, j]$ is an *interval for which s is scheduled*, and we also say that s is *assigned to A because of* $[i, j]$. Such a sensor s is called *right going* if it was chosen to dominate i to the right (see steps 7 and 15). Analogously, we say that s is *left going* if it was chosen to dominate j to the left (see steps 12 and 13). If a point i was not covered at time $t(s)$ before some sensor s was scheduled, but is covered by s at time $t(s)$ (when s is assigned to A), we say that *s closes i* at time $t(s)$. We denote by A'_s the schedule constructed by the algorithm immediately before scheduling sensor s.

In what follows, we adopt the following convention: if a sensor is called s_p, then $[i_p, j_p]$ denotes the interval for which s_p is scheduled.

We call attention to the concept of the interval for which a sensor is scheduled, as it plays an important role in the proof of the next result. Although this concept is present in the algorithm, in the proof of Gibson and Varadarajan it is not used directly. We believe it helps clarifying the proofs. Claim 1 and Claim 2 stated in the proof of the next result were proved by Gibson and Varadarajan, in a different way. In fact, the inequalities derived in the end of these claims are not explicitly mentioned by these authors, but here they are needed in the proof of Claim 3.

In Fig. 4, we represent the structure of the proof of Theorem 1, which gives the approximation ratio of algorithm RSC-GV. We call attention to the fact that the assumptions and terminology used in the proof of Lemma 3 is shared by all claims indicated in the figure. An arrow (resp. dashed arrow) between two boxes indicates that the statement (resp. a subcase) mentioned in the source box is used in the proof of the destination box.

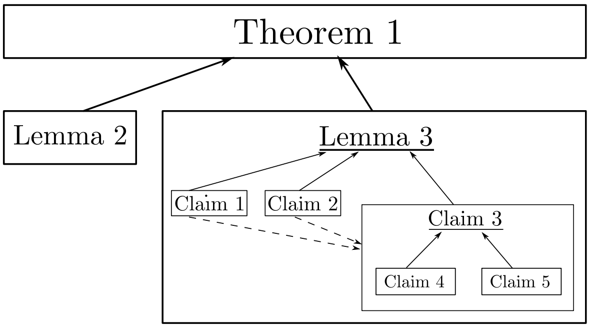


Fig. 4 Structure of the proof of Theorem 1

Lemma 3 *If A is a schedule returned by the algorithm* RSC-GV, *then we have* $\text{MaxCoverage}(A) \leq 4$.

Proof Let (U, S) be an instance for the RSC problem and let A be the schedule returned by the algorithm RSC-GV when applied to (U, S). Consider an arbitrary point $i \in U$ and an arbitrary time t, $0 < t \leq M(A)$. We shall prove that $\text{coverage}(i, t) \leq 4$.

Denote by s_0 the first sensor that covers i at time t. By convention, $[i_0, j_0]$ is the interval for which s_0 is scheduled. Now classify any other sensor s_p *that covers i at time t* into the following four types:

- Type LL: if $[i_p, j_p]$ is to the LEFT of i and s_p is LEFT going;
- Type LR: if $[i_p, j_p]$ is to the LEFT of i and s_p is RIGHT going;
- Type RL: if $[i_p, j_p]$ is to the RIGHT of i and s_p is LEFT going;
- Type RR: if $[i_p, j_p]$ is to the RIGHT of i and s_p is RIGHT going.

The main ingredients of the proof are the following three claims.

Claim 1 *At most two sensors of types* LL *or* LR *are assigned to A.*

Proof Let s_1 and s_2 be the two first sensors of type LL or LR that are scheduled after s_0. Suppose s_2 is scheduled after s_1. Consider the two possible cases for sensor s_1, illustrated in Fig. 5.

Case (a): Sensor s_1 is of type LL.

We recall that $[i_1, j_1]$ denotes the interval for which s_1 is scheduled. As s_1 is left going (dominates j_1 to the left), we have $l(s_1) \leq l(s_2)$. Thus, $i_2 \in [l(s_2), i] \subseteq [l(s_1), i]$. In this case, note that the interval $[i_2, j_2]$ can only be considered after time $t(s_1) + d(s_1) - 1$ (that is, only when sensor s_1 is not active anymore). Therefore,

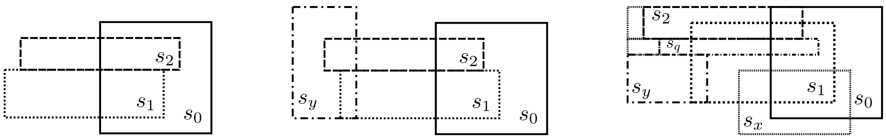


Fig. 5 Cases (a), (b1) and (b2), respectively

$t(s_2) > t$. But then, s_2 does not cover i at time t, a contradiction. This means that after a sensor of type LL, no sensor of type LL or LR is scheduled.

Case (b): Sensor s_1 is of type LR.

In this case, since s_1 is right going (dominates i_1 to the right), and is live at i, then it is live at j_1, and therefore (as it is scheduled in step 15) we have

$$M\big(A'_{s_1}, i_1 - 1\big) \geq M\big(A'_{s_1}, j_1 + 1\big).$$

If $i_1 = 1$, then clearly i_2 lies in the interval $[i_1, l(s_0)]$. In this case, after scheduling s_1, the interval $[i_1, i]$ is entirely covered at time t. So, the algorithm will only consider a sensor of type LL or LR after time t. Thus, $t(s_2) > t$. But then, s_2 does not cover i at time t, a contradiction.

Let us assume that $i_1 > 1$. In this case, the point $i_1 - 1$ is covered by some sensor, say s_y, at time $M(A'_{s_1}, i_1 - 1)$. Note that, $l(s_2) > l(s_y)$ (resp. $l(s_1) > l(s_y)$), otherwise we would have a contradiction to Lemma 1, as we would have $R(s_y) \subset R(s_2)$ (resp. $R(s_y) \subset R(s_1)$) and s_2 (resp. s_1) would have been assigned to A before s_y.

Given a scheduled sensor s, define $h(s) := t(s) + d(s) - 1$, that is, the time when the battery of s becomes discharged. Let us analyze two subcases.

Subcase (b1): $j_1 + 1 = l(s_0)$.

We know that $h(s_y) \geq h(s_0)$ (because s_1 is right going). In this case, after s_1 is assigned to A, all positions in the interval $[l(s_y), i]$ are covered at time t. As $l(s_2) > l(s_y)$, the algorithm can consider a position $i_2 \in [l(s_2), l(s_0)]$ only after sensor s_1 is not active anymore, and therefore after time t. In this case, s_2 does not cover i at time t, a contradiction.

Subcase (b2): $j_1 + 1 < l(s_0)$.

Let s_x be the sensor assigned to A before s_1, such that $j_1 + 1 = l(s_x)$ and $t(s_x) < t(s_1) \leq t(s_x) + d(s_x) - 1$ (that is, in the moment s_1 was assigned to A, sensor s_x was still active).

Now consider the assignment of s_2. If s_2 is of type LL, using an argument analogous to that of case (a) we can conclude that no other sensor of type LL or LR is assigned to A. So, assume now that s_2 is of type LR.

Note that $i_2 > l(s_y)$ (as we have seen that $l(s_2) > l(s_y)$). Likewise, note also that $l(s_1) > l(s_y)$.

If $h(s_1) \leq h(s_y)$, then $t \leq h(s_y)$. Thus, as we have seen that $i_2 > l(s_y)$, it follows that $t(s_2) > t$. But then, s_2 does not cover i at time t, a contradiction.

Assume now that $h(s_1) > h(s_y)$. Since s_2 covers i at time t, then $[i_2, j_2]$ lies in the interval $[l(s_y), l(s_1)]$.

We argue that $j_2 + 1 = l(s_1)$. Let s_q be the last sensor scheduled before s_2 such that $j_q + 1 = l(s_1)$. We claim (and prove in the sequel) that s_2 is only assigned after the end of the execution of s_q and, therefore, $j_2 + 1 = l(s_1)$. Note that if there is no such s_q, the statement is vacuously true.

If s_q was chosen to be left going, we must have $t(s_2) > h(s_q)$, by reasoning as in case (a). Otherwise, if s_q was chosen to be right going, we have $M(A'_{s_q}, i_q - 1) \geq M(A'_{s_q}, j_q + 1) = h(s_1) \geq t$. Note that, after the assignment of s_q, the interval $[i_q - 1, i]$ is completely covered until time $h(s_q)$. Hence, $t(s_2) > h(s_q)$.

It follows that $j_2 + 1 = l(s_1)$. In this case, $M(A'_{s_2}, i_2 - 1) \geq M(A'_{s_2}, j_2 + 1)$ (because s_2 is right going). Since s_2 is live at all points in $[i_1, j_1]$, because of the priority given to s_1, we conclude that $r(s_2) < r(s_1)$.

Now suppose there is a third sensor, say s_3, that is of type LL or LR and is the next of this type assigned after s_2. Suppose $i_2 > 1$ and let s_r be the sensor that covers the point $i_2 - 1$ at time $M(A'_{s_2}, i_2 - 1)$ (if $i_2 = 1$, it is immediate that s_3 does not exist). If $h(s_2) \leq h(s_r)$, using arguments similar to those used above, we can conclude that no sensor assigned after s_2 can cover i at time t. So, suppose $h(s_2) > h(s_r)$. Since $h(s_r) > h(s_1)$ and $t \leq h(s_1)$, we have that $h(s_r) > t$. Since $[i_3, j_3]$ has to lie in the interval $[l(s_r), l(s_2)]$, it is immediate that s_3 cannot cover i at time t.

Summarizing what we have proved in the subcase (b2), when there are sensors s_1 of type LR and s_2 of type LL or LR (covering i at time t), we have proved that there is a sensor s_x such that the following holds:

$$l(s_1) < l(s_x) \quad \text{and} \quad r(s_1) < r(s_x). \tag{1}$$

This completes the proof of Claim 1. □

Claim 2 *At most two sensors of types* RL *or* RR *are assigned to* A.

Proof (Sketch only). Let s_3 and s_4 be the first sensors of type RR or RL that are scheduled after s_0. Suppose s_3 is scheduled before s_4. Analogously to Claim 1, we analyze two cases:

Case (c): Sensor s_3 is of type RR.

In this case, we conclude that no sensor of type RR or RL can be assigned after s_3.

Case (d): Sensor s_3 is of type RL.

Subcase (d1): $i_3 - 1 = r(s_0)$.

In this subcase the conclusion is like in case (c).

Subcase (d2): $i_3 - 1 > r(s_0)$.

Analogously to subcase (b2), we can prove that after the assignment of sensor s_4, no other sensor of type RL or RR is assigned. Moreover, in this subcase we have that there is a sensor s_w such that, $t(s_1) \leq t(s_w) + d(s_w) - 1$, and the following holds:

$$l(s_w) < l(s_3) \quad \text{and} \quad r(s_w) < r(s_3). \tag{2}$$

This ends the sketch of the proof of Claim 2. □

Claim 3 *The subcase* (b2), *in the proof of Claim* 1, *and the subcase* (d2), *in the proof of Claim* 2, *cannot occur together.*

Proof Suppose, by contradiction, that both subcases occur. Consider the sensors s_1, s_2 and s_x (resp. s_3, s_4 and s_w) that we have mentioned in the analysis of subcase (b2) (resp. (d2)) which satisfy the inequalities (1) and (2). From these inequalities, and the hypothesis that $s_1, \ldots, s_4$ cover i at time t, we conclude that

$$r(s_x) > i > l(s_w). \tag{3}$$

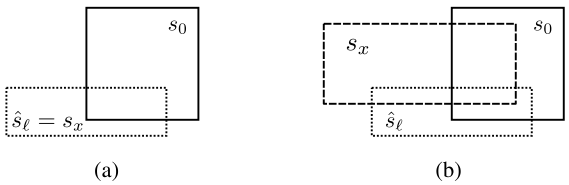


Fig. 6 Sensor $\hat{s}_\ell$ and its relation to s_x and s_0, as stated in Claim 4

On the other hand, the following claim (proved later) holds.

Claim 4 *There exists a sensor $\hat{s}_\ell$ such that*

(1) $l(s_x) \le l(\hat{s}_\ell) < l(s_0)$ *and* $r(s_x) \le r(\hat{s}_\ell) < r(s_0)$; *and*
(2) $t(\hat{s}_\ell) \le t(s_0) \le t(\hat{s}_\ell) + d(\hat{s}_\ell) - 1$.

Putting into words, Claim 4 states that there is a sensor $\hat{s}_\ell$ such that: (1) $l(\hat{s}_\ell)$ is between $l(s_x)$ and $l(s_0)$, and $r(\hat{s}_\ell)$ is between $r(s_x)$ and $r(s_0)$; and (2) sensor s_0 is activated at some time while $\hat{s}_\ell$ is active.

Note that it may happen that s_x is activated before s_0 and, in this situation (case illustrated in Fig. 6(a)), we can consider $\hat{s}_\ell$ to be s_x, a case for which Claim 4 holds. Otherwise, there is a sensor $\hat{s}_\ell$ as stated in Claim 4 (case illustrated in Fig. 6(b)).

Analogously to the existence of sensor $\hat{s}_\ell$ (related to s_x and s_0), as stated in Claim 4, we claim there is a sensor $\hat{s}_r$ related to s_w and s_0 in the following way.

Claim 5 *There exists a sensor $\hat{s}_r$ such that*

(1) $l(s_0) < l(\hat{s}_r) \le l(s_w)$ *and* $r(s_0) < r(\hat{s}_r) \le r(s_w)$; *and*
(2) $t(\hat{s}_r) < t(s_0) \le t(\hat{s}_r) + d(\hat{s}_r) - 1$.

Thus, s_0 must be scheduled after $\hat{s}_\ell$ and $\hat{s}_r$ have been scheduled, but before the end of their execution. From this fact and the conditions in Claim 4 and Claim 5, it follows that

$$r(s_x) \le r(\hat{s}_\ell) < i_0 \le j_0 < l(\hat{s}_r) \le l(s_w), \tag{4}$$

as the interval $[i_0, j_0]$ is not covered at time $t(s_0)$ in the schedule A'_{s_0} (because it is closed by s_0). We have, therefore, a contradiction to the inequality (3). This completes the proof of Claim 3. □

From Claims 1, 2 and 3, we conclude that coverage$(i, t) \le 4$. □

We prove now Claim 4. The proof of Claim 5 is analogous.

Proof of Claim 4 We know that $t(s_0) \le t(s_1) \le t(s_x) + d(s_x) - 1$. If $t(s_x) \le t(s_0)$, then s_x satisfies the conditions required for $\hat{s}_\ell$, and therefore the claim holds (taking $\hat{s}_\ell = s_x$). Let us then assume that $t(s_0) < t(s_x)$.

Let s' be the first sensor scheduled after s_0 such that $l(s_x) \le l(s') < l(s_0)$ and $t(s') < t(s_0) + d(s_0) - 1$, and furthermore, there exists a sensor s'' such that $j'' + 1 = l(s')$ and $t(s') < t(s'') \le t(s') + d(s') - 1$, where $[i'', j'']$ is the interval for which s''

was scheduled. We know that there exists at least one such s', as by the hypotheses, s_x and s_1 satisfy the conditions required for s' and s'', respectively.

Let $[i', j']$ be the interval for which s' was scheduled. (Note that $j' \leq l(s_0)$.) Since $l(s'') < l(s')$, and s' is scheduled before s'', we conclude that s' was scheduled because it is right going (as s'' is live at any position in the interval $[i', j']$). Also, note that $t > t(s') + d(s') - 1$, since s' is live at i but does not cover i at time t.

Suppose $j' + 1 = l(s_0)$. In this case, $M(A'_{s'}, i' - 1) \geq M(A'_{s'}, j' + 1) = M(A'_{s'}, l(s_0)) \geq t > t(s') + d(s') - 1$, where t is the time we have fixed in the beginning of the proof of Lemma 3. Under these conditions, we would have $t(s'') > t(s') + d(s') - 1$, a contradiction with our choice of s' and s''.

Thus, $j' + 1 < l(s_0)$. In this case, there is some sensor $\hat{s}_\ell$ such that $j' + 1 = l(\hat{s}_\ell)$ and $t(\hat{s}_\ell) < t(s') \leq t(\hat{s}_\ell) + d(\hat{s}_\ell) - 1$.

Note, however, that if $\hat{s}_\ell$ is scheduled after s_0, then it contradicts the choice of s'. Thus, we conclude that $\hat{s}_\ell$ is scheduled before s_0 and its execution ends after the schedule of s_0. Furthermore, we have that $l(s_x) \leq l(\hat{s}_\ell) < l(s_0)$. From this, we conclude that $r(s_x) \leq r(\hat{s}_\ell) < r(s_0)$, for otherwise we would have a contradiction to Lemma 1. □

Theorem 1 *The algorithm* RSC-GV *is a polynomial-time* 4*-approximation algorithm for the RSC problem. Furthermore, its ratio analysis is tight.*

Proof The approximation ratio follows immediately from Lemma 2 and Lemma 3. As mentioned by Gibson and Varadarajan, the algorithm has polynomial running time. Indeed, the loop starting at line 3 is executed at most n times, as one sensor is scheduled in each iteration. Each iteration can be implemented to run in time $O(n + m) = O(n)$, since we can assume that $m \leq 2n$. Therefore, the algorithm runs in time $O(n^2)$.

In order to see that the ratio analysis is tight, consider first the instance of the RSC problem shown in Fig. 7, together with the corresponding output A of the algorithm RSC-GV and an optimal schedule. Note that the gray rectangles are also sensors (s_{12} to s_{31}, except for s_{22}).

As we can see, $M(A) = 5$, but $OPT = L = 12$. Furthermore, for $i = 11$ we have coverage$(i, 5) = 4$ in the schedule A (see Fig. 7(top part)). This example suggests how to construct other instances for which the "maximally covered region" becomes much larger than the remaining regions in the solution output by the algorithm. For that, it suffices to scale appropriately the durations of the sensors. More specifically, consider the instance $I(k)$, parameterized by an integer $k \geq 1$, defined as follows (the instance shown in Fig. 7 corresponds to $k = 1$).

$U = [1, 20]$; $S = \{s_1, \ldots, s_{31}\}$;
$R(s_1) = [1, 16]$, $d(s_1) = 1$;
$R(s_2) = [17, 20]$, $d(s_2) = 4$;
$R(s_3) = [1, 6]$, $d(s_3) = 1$;
$R(s_4) = [7, 13]$, $d(s_4) = 2$;
$R(s_5) = [9, 16]$, $d(s_5) = k + 3$;
$R(s_6) = [1, 4]$, $d(s_6) = 2$;
$R(s_7) = [5, 12]$, $d(s_7) = k + 2$;

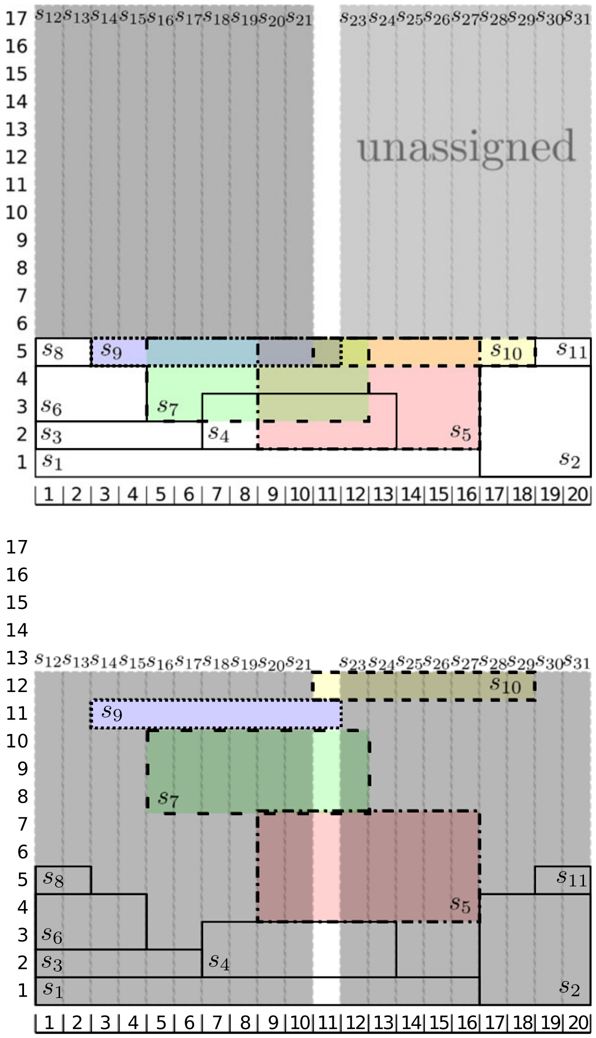


Fig. 7 *Top*: The schedule output by the algorithm RSC-GV. *Bottom*: An optimal schedule

$R(s_8) = [1, 2]$, $d(s_8) = k$;
$R(s_9) = [3, 11]$, $d(s_9) = k$;
$R(s_{10}) = [11, 18]$, $d(s_{10}) = k$;
$R(s_{11}) = [19, 20]$, $d(s_{11}) = k$; and
$R(s_{11+j}) = [j, j]$, $d(s_{11+j}) = 4k + 8$, for $1 \le j \le 20$ such that $j \ne 11$.

The algorithm RSC-GV schedules the sensors precisely in the increasing order of their indices, that is, from s_1 to s_{21} (the remaining sensors are not scheduled). Note that for the instance $I(k)$ the point $i = 11$ is covered by the sensors s_5, s_7, s_9 and s_{10} from time 5 to $5 + k - 1$. With respect to Lemma 3, sensor s_5 plays the role of s_0 in the proof; and the remaining sensors that cover point i are the sensors s_7, which is of type LR; s_9, which is of type LR; and s_{10}, which is of type RR.

Note that $M(A) = k + 4$ and $OPT = L = 4k + 8$. Thus, the approximation ratio is $(4k + 8)/(k + 4) = 4 - 8/(k + 4)$. Hence, for a sufficiently large value of k, this ratio can get arbitrarily close to 4, showing that the ratio 4 is tight. □

4 ILP Formulation for the RSC Problem and Computational Results

We present now an integer programming formulation for the RSC problem, an approach not yet treated in the literature.

First, consider the sets $I = \{1, 2, \ldots, m\}$, $J = \{1, 2, \ldots, L\}$ and $S = \{1, 2, \ldots, n\}$. For the variables, we have $y_{i,j} \in \{0, 1\}$, for all $i \in I$ and $j \in J$, which is 1 if point i is covered by some sensor in time j, and is 0 otherwise. We have $z_{s,j} \in \{0, 1\}$, for all $s \in S$ and $j \in J$, which is 1 if sensor s is turned on at time j, and is 0 otherwise. Further, we have the variable $M \in \mathbb{Z}$, which indicates the value of the solution. The proposed IP formulation is the following.

$$
\begin{aligned}
\max\ & M \\
\text{s.t.}\ & \sum_j z_{s,j} \le 1 \quad \forall s \in S && (5) \\
& y_{i,j} \le \sum_{s: i \in R(s)} \sum_{k=j-d(s)+1}^{j} z_{s,k} \quad \forall i \in I, \forall j \in J && (6) \\
& y_{i,j+1} \le y_{i,j} \quad \forall i \in I, \forall j \in J \setminus \{L\} && (7) \\
& M \le \sum_j y_{i,j} \quad \forall i \in I && (8) \\
& y_{i,j} \in \{0, 1\} \quad \forall i \in I, \forall j \in J \\
& z_{s,j} \in \{0, 1\} \quad \forall s \in S, \forall j \in J \\
& M \in \mathbb{Z}
\end{aligned}
$$

Constraints (5) express that each sensor can only be turned on once. Constraints (6) state that a point i is only covered in time j if there is some sensor on at that time. Constraints (7) assure point i is only covered in time $j + 1$ if that point is covered in time j. Finally, constraints (8) guarantee that M is the minimum, taken over all points i, of the total time position i is covered.

We now show some preliminary computational results obtained with the implementation of the model proposed.

The instances were generated from instances of the Strip Packing problem, a minimization problem that consists in packing rectangles into a strip of fixed width and unbounded height. These instances were obtained from the OR-LIBRARY [1].

The first 18 instances shown in Table 1 were obtained in the following way. Given an instance for the Strip Packing problem with strip width m', we translate

Table 1 Computational results obtained with the ILP formulation for the RSC problem

n	m	L	$L_{\max}$	OPT	RSC-GV	Ratio	Time (sec.)
24	10	27	52	27	22	1.23	0.26
23	10	20	60	20	17	1.18	0.10
19	10	27	59	27	24	1.13	0.24
35	7	47	118	47	32	1.47	1.35
34	7	61	109	61	48	1.28	2.54
30	7	64	110	64	43	1.49	0.96
34	15	56	191	56	36	1.56	15.04
35	15	59	185	59	58	1.02	1.61
35	15	83	180	83	58	1.44	31.89
49	30	50	181	50	45	1.12	13.41
64	30	62	164	62	52	1.20	94.22
65	30	68	179	68	61	1.12	159.70
85	45	73	166	73	69	1.06	493.30
96	45	57	309	57	47	1.22	120.59
81	45	62	179	62	57	1.09	192.48
115	60	74	252	0	61	–	TLE
113	60	85	239	85	63	1.35	1518.52
113	60	80	238	80	75	1.07	1381.26
47	10	67	92	67	43	1.56	15.50
40	10	49	110	49	48	1.03	1.06
46	10	58	110	58	50	1.16	4.80
65	7	127	204	127	90	1.42	134.31
64	7	143	199	143	112	1.28	223.52
62	7	153	206	153	103	1.49	354.96
77	15	152	316	152	96	1.59	1165.65
77	15	164	349	164	144	1.14	1290.11
73	15	149	336	149	144	1.04	782.95
179	30	126	879	–	86	–	TLE
383	30	206	6745	–	136	–	TLE
120	30	156	322	–	121	–	TLE
167	45	196	343	–	129	–	TLE
364	45	160	2121	–	92	–	TLE
173	45	173	350	–	120	–	TLE
234	60	189	432	–	151	–	TLE
226	60	226	423	–	176	–	TLE
225	60	222	405	–	170	–	TLE

each rectangle as a sensor s in our instance of the RSC. In order to get uniform load distribution, we use the following procedure to set $l(s)$. We choose $l(s)$ uniformly at random between 1 and $m = m'/2$. If $r(s) > m$, we "break" s into two sensors s_1 and s_2, both with duration $d(s)$, but one with $l(s_1) = l(s)$ and $r(s_1) = m$ and the other with $l(s_2) = 1$ and $r(s_2) = w - (m - l(s_1) + 1)$, where w is the width of the original sensor s. In order to derive larger instances, we duplicated each rectangle and used the same procedure. These instances correspond to the other 18 instances in Table 1.

The implementation was done using IBM ILOG CPLEX Optimizer. Table 1 shows the time (in seconds) our code took to find an optimal solution for the instances (generated as above mentioned) with the indicated values of n, m, L and $L_{\max}$, where $L_{\max} = \max_i L(i)$. We note that all instances for which an optimal solution was found have optimum value equal to L. Table 1 also shows the value of the solutions found by the algorithm RSC-GV and the approximation ratios for these instances.

For each execution, we have set a time limit of 1800 seconds. In the table, "TLE" stands for "Time Limit Exceeded", which means that the program was aborted after that amount of time and could not find a solution.

The computational experiments show that, for instances with n up to 100 (and nL up to 7000), optimal solutions could be found within 1800 seconds. As we can see, in general, the time spent increases as nL increases, what is natural, as the number of constraints is $O(nL)$. The largest time, 1518 seconds, occurred for $nL = 113 \cdot 85 = 9605$ and $m = 60$. For these randomly generated instances, we conclude that this model can be useful when nL is not too large.

Having found the optimal solutions for some of these instances, we could analyze experimentally the performance of the algorithm RSC-GV. We note that its performance ratios on these instances range from 1.02 to 1.59, being on the average 1.26: a performance that can be considered very good.

5 The RSCP Problem, the Preemptive Variant

In this section, we present an exact polynomial-time algorithm for the preemptive version of RSC, called RSC-PREEMPTIVE.

Buchsbaum et al. [3] stated that

> For RSC, a simple algorithm based on maximum flow yields an optimal preemptive schedule in polynomial time.

However, such algorithm has not been shown by these authors.

As we will see, the algorithm RSC-PREEMPTIVE is quite simple, and does not require the use of a maximum flow algorithm. We will prove that it produces a schedule of duration L. For a better understanding of the algorithm, the reader should see first some definitions and a brief description of the main idea behind it.

Let (U, S) be an instance of RSCP, and L the load of (U, S). We say that a set of sensors $C \subseteq S$ is a *cover* of U if for every $i \in U$ there is a sensor s in C such that

```
Algorithm 2: RSC-PREEMPTIVE
  Input  : a pair $(U, S)$, where $S$ is ordered (as mentioned on the bottom of this
           page)
  Output: a schedule of $S$

if $L = 0$ then
    return $\emptyset$        /* $L$ is the load of $(U, S)$ */
else
    $C \leftarrow \emptyset$
    $i \leftarrow 1$
    while $i \leq m$ do
        /* $C$ is not a minimal cover of $U$ */
        Let $s$ be the last sensor in $S$ which is live at $i$
        Let $C' := \{s' \in C : l(s') > l(s)\}$        /* set of redundant sensors */
        $C \leftarrow C \setminus C' \cup \{s\}$
        $i \leftarrow r(s) + 1$
    end
    $h \leftarrow \min_{s \in C}\{d(s)\}$        /* $h = h(C)$ is the height of the cover $C$ */
    for every sensor $s$ in $C$ do
        $d(s) \leftarrow d(s) - h$
        if $d(s) = 0$ then
            $S \leftarrow S \setminus \{s\}$
        end
    end
end
$L \leftarrow L - h$
return $(C, h) \cup$ RSC-PREEMPTIVE$(U, S)$
```

$i \in R(s)$. A sensor s in C is called *redundant* if for every i in $R(s)$, there is a sensor s' in C such that $i \in R(s')$. We say that a cover C is *minimal* if C does not contain redundant sensors. Further, we say that a subinterval U' of U is a *critical region* if $L(i) = L$, for every position i in U'. We say that two sensors s and s' *overlap* if $R(s) \cap R(s') \neq \emptyset$.

The main idea of the algorithm RSC-PREEMPTIVE consists in finding first a minimal cover C (that satisfies certain properties), then identifying the smallest among the durations of all sensors in C, say h, and scheduling (simultaneously) all sensors in C, letting them turned on for the period of time h. Next, it updates the duration of the sensors in C and repeats the process with the new instance, until there is no minimal cover.

In the Algorithm RSC-PREEMPTIVE (Algorithm 2), we consider that the given instance (U, S) is such that $U = \{1, 2, \ldots, m\}$ and the set S of sensors is sorted according to the following criterion: in non-decreasing order of $l(\cdot)$ and, in case of tie, in non-decreasing order of $r(\cdot)$.

We note that the union of pairs (C, h) returned by the algorithm indicate which sensors are activated and when they are deactivated, defining this way the schedule for the input (U, S). We prove in the following that the algorithm produces an optimal schedule.

Theorem 2 *The algorithm* RSC-PREEMPTIVE *returns a schedule of duration L.*

Proof Let (U, S) be the instance given to the algorithm. Note that the algorithm is recursive and halts only when it receives an input with load equal to zero.

Throughout this proof, L denotes the load of the original instance (U, S). Suppose $L > 0$. Note that, if $L > 0$ then, after executing steps 5–11, the set C constructed by the algorithm is a minimal cover of U. The minimality is assured by the removal of the sensors, done in the step 10 of the algorithm.

Denote by $C_1, \ldots, C_k$ the minimal covers found by the algorithm, in the order they were constructed. Denote by $h(C_i)$ the height h of the cover C_i, that is, $h(C_i) = \min_{s \in C_i} d(s)$ (see step 13).

Note that L is decreased (of $h(C_i)$) each time a new cover C_i is found. We shall prove that $\sum_{j=1}^{k} h(C_j) = L$, by induction on k.

If $k = 1$, then after finding C_1, the algorithm does not find any other cover. This means that the load is zero in the second recursive call of the algorithm. Thus, there is a position $i \in U$ at which only one sensor (of the original set S) of duration $h(C_1)$ is live. In this case, $L \leq L(i) = h(C_1)$, and therefore, $h(C_1) = L$, as $h(C_1) \leq L$.

Now suppose that $k > 1$. Consider the first iteration in which the cover C_1 is constructed. Let $h := h(C_1)$. We shall prove that the new instance (U, S'), obtained after updating S (steps 14 to 19 of the algorithm), has load L' equal to $L - h$. For that, it suffices to show the following two properties (of the cover C_1):

(1) There is no overlap of sensors in C_1 in critical regions; and
(2) If i is a position in which an overlap of sensors in C_1 occur, then $L(i) \geq L + h$.

As C_1 is a minimal cover, each position in U is covered by at most two sensors in C_1.

To show (1), suppose that there is an overlap of two sensors s_1 and s_2 of C_1 in a position i that belongs to a critical region. Suppose $l(s_1) < l(s_2)$. Let $i' \in R(s_2)$ be the first position (smallest integer) where there is no overlap of sensors in C_1, that is, s_2 is the only sensor in C_1 which covers i'. There must be at least one such position, otherwise C_1 would have redundant sensors.

By the choice of s_2, every sensor of S live at i' is also live at i. This way, $L(i) \geq L(i') + d(s_1) > L$, where the last inequality follows from the fact that $d(s_1)$ is nonzero. By the definition of critical region, however, $L(i) = L$. Thus, we have a contradiction, and this concludes the proof of property (1).

Now we shall prove (2), using a reasoning similar to the one presented to prove (1). Let $i \in U$ be a position in which there is an overlap of two sensors s_1 and s_2 of C_1. By (1), we know that i does not belong to a critical region. Suppose $l(s_1) < l(s_2)$; and let $i' \in R(s_2)$ be the first position where there is no overlap of sensors of C_1.

By the choice of s_2, every sensor in S live at i' is also live at i. Therefore, we have

$$\begin{aligned} L(i) &\geq L\big(i'\big) + d(s_1) \\ &\geq L\big(i'\big) + h \quad \text{as } h = h(C_1) = \min_{s \in C_1} d(s) \\ &\geq L + h \quad \big(\text{as } L\big(i'\big) \geq L\big). \end{aligned}$$

The properties (1) and (2) guarantee that, after the assignment of the sensors in the cover C_1, the new instance (U, S'), obtained after removing from S the sensors in the cover C_1 and updating the durations of the sensors in C_1, has total load $L' = L - h$. By the induction hypothesis, $\sum_{j=2}^{k} h(C_i) = L - h$. Thus, we have that $\sum_{j=1}^{k} h(C_i) = L$, and, therefore, the algorithm returns a schedule of duration L. □

From the previous theorem and the analysis of the algorithm RSC-PREEMPTIVE, we have the following result.

Corollary 1 *The algorithm* RSC-PREEMPTIVE *finds an exact optimal solution for the* RSCP *problem, the preemptive version of* RSC, *in quadratic time.*

Proof The sorting of the n sensors in S takes time $O(n \log n)$. In each iteration (in which a new cover is found), the algorithm removes at least one sensor (steps 14–19). Then, there are at most n calls to the algorithm.

Note that the construction of a cover takes time $O(n)$, as each sensor in the sorted set of (at most n) sensors is checked at most once and can be selected only once. Therefore, each iteration runs in time $O(n)$. Hence, the algorithm returns a schedule in time $O(n^2)$.

By Theorem 2, the algorithm finds a schedule with duration L. Since L is an upper bound for OPT, we conclude that the algorithm finds an optimal solution. □

6 Concluding Remarks

As we have shown, the algorithm RSC-GV is a 4-approximation for the RSC problem, and its ratio analysis cannot be improved. This is a rather simple algorithm, so it is a challenge to devise as simple algorithms with better bounds. It would also be interesting to prove a non-trivial inapproximability bound for this problem.

It can be shown that the RSC problem admits no FPTAS (fully polynomial time approximation scheme), or that it cannot be approximated within a factor better than $\frac{L}{L-1}$, unless P = NP. These inapproximability results can be derived by a reduction from the 3-PARTITION problem (known to be NP-complete in the strong sense). For that, one can use the reduction shown by L. Stockmeyer to prove that the DYNAMIC STORAGE ALLOCATION problem is NP-complete (see in [4] the reduction provided by Garey and Johnson [6]).

The possibility of a PTAS for the RSC problem is not discarded. It would be interesting to settle this.

As far as the integer programming formulation for the RSC problem, we note that it has $O(nL)$ constraints, a number that can be exponential in the input size. In view of this, we found it surprising that instances up to $n = 100$ sensors (and nL up to 7000) could be solved within 1800 seconds. Finding valid inequalities to use as cuts in the implementation can be of interest as they may accelerate the process. It would be interesting to investigate the possibility of finding an ILP formulation of size polynomial in the input.

As for the preemptive variant, we have shown an exact algorithm that runs in quadratic time. The algorithm is simple, does not require finding maximum flows, and can be easily implemented.

Acknowledgements The authors thank the referee for the valuable comments and suggestions. They also thank FAPESP (Proc. 2009/03589-0), CNPq (Proc. 475064-2010-0, 303987/2010-3, 477203/2012-4) and Project USP MaCLinC/NUMEC for the partial support obtained.

References

1. Beasley, J.:. OR library. people.brunel.ac.uk/~mastjjb/jeb/info.html
2. Buchsbaum, A., Karloff, H., Kenyon, C., Reingold, N., Thorup, M.: OPT versus LOAD in dynamic storage allocation. SIAM J. Comput. **33**(3), 632–646 (2004). doi:10.1137/S0097539703423941
3. Buchsbaum, A., Efrat, A., Jain, S., Venkatasubramanian, S., Yi, K.: Restricted strip covering and the sensor cover problem. In: Proceedings of the 18th Annual ACM Symposium on Discrete Algorithms, pp. 1056–1065. SIAM, Philadelphia (2007)
4. Buchsbaum, A., Efrat, A., Jain, S., Venkatasubramanian, S., Yi, K.: Restricted strip covering and the sensor cover problem. E-print (2008). arXiv:cs/0605102
5. da Ponte Barbosa, R., Wakabayashi, Y.: A better approximation ratio and an IP formulation for a sensor cover problem. In: LATIN 2012: Theoretical Informatics. Lecture Notes in Comput. Sci., vol. 7256, pp. 49–60. Springer, Berlin (2012)
6. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. A Series of Books in the Mathematical Sciences. Freeman, San Francisco (1979)
7. Gibson, M., Varadarajan, K.: Decomposing coverings and the planar sensor cover problem. In: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, pp. 159–168 (2009)

How Many Steiner Terminals Can You Connect in 20 Years?

Ralf Borndörfer, Nam-Dũng Hoang, Marika Karbstein, Thorsten Koch, and Alexander Martin

Abstract Steiner trees are constructed to connect a set of terminal nodes in a graph. This basic version of the Steiner tree problem is idealized, but it can effectively guide the search for successful approaches to many relevant variants, from both a theoretical and a computational point of view. This article illustrates the theoretical and algorithmic progress on Steiner tree type problems on two examples, the Steiner connectivity and the Steiner tree packing problem.

1 Introduction

The *Steiner tree problem* (STP) is one of the showcases of combinatorial optimization. It deals with finding a best connection of a number of vertices in a network and can formally be stated as follows:

> *Given a weighted graph $G = (V, E, c)$ and a non-empty set of vertices $T \subseteq V$ called* terminals*, find an edge set S^* such that $(V(S^*), S^*)$ is a tree of minimal weight that spans T.*

The STP is extensively covered in the literature, see [33, 35] for an introduction and [32] for a state-of-the-art survey on models and solution techniques. Many

R. Borndörfer (✉) · M. Karbstein · T. Koch
Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, 14195 Berlin, Germany
e-mail: borndoerfer@zib.de

M. Karbstein
e-mail: karbstein@zib.de

T. Koch
e-mail: koch@zib.de

N.-D. Hoang
Faculty of Mathematics, Mechanics, and Informatics, Vietnam National University, 334 Nguyen Trai, Hanoi, Vietnam
e-mail: hoangnamdung@hus.edu.vn

A. Martin
Department Mathematik, Friedrich-Alexander-Universität Erlangen-Nürnberg, Cauerstr. 11, 91058 Erlangen, Germany
e-mail: alexander.martin@math.uni-erlangen.de

papers claim real-world applications, especially in VLSI-*design* and *wire-routing*, but also in *telecommunication and traffic network design*. Such applications usually refer to generalizations of the STP, which still require to connect a number of terminals, but by more than one tree, by something that is not precisely a tree, a constrained tree, etc. Such stipulations have always made Steiner trees a lively and exciting research topic for theoreticians and practitioners alike. Theoretical progress on the STP will typically serve as a blueprint to approach a particular Steiner tree problem; this knowledge is supplemented by specific developments, which in turn can advance the general theory. This is the Martin Grötschel way of optimization.

We illustrate (t)his approach in this article using two examples, the Steiner connectivity problem (SCP) and the Steiner tree packing problem (STPP). The SCP, discussed in Sect. 2, generalizes the STP by connecting the terminals using a set of paths instead of edges; this is motivated by line planning problems in public and rail transit, see [3]. The Steiner connectivity problem serves as an example that often results can be transferred from the basic Steiner tree problem to a more general setting, however, not all results and not always in a completely straightforward way. The STPP, studied in Sect. 3, deals with connecting several sets of terminals by several trees that cannot share edges; this models the internal wiring in a chip. The Steiner tree packing problem is difficult because it integrates a connection and a packing aspect; it is particularly well suited to illustrate the algorithmic progress of the last 20 years. However, chips are much larger now and the challenge has also grown. Steiner trees will therefore remain an exciting topic, at least until Martin Grötschel's 100th birthday—can there be a better message?

2 The Steiner Connectivity Problem

Transportation and networks have always been two of Martin Grötschel's favorite topics and it was therefore inevitable that at some point he would get involved in a research project on designing the line system of a public transportation system. Identifying origin-destination demand points with terminals, and lines with hyperedges, this leads directly to a hypergraph version of the Steiner tree problem, which we denote the Steiner connectivity problem, see [4, 23]. How difficult is this? In such a case, Martin Grötschel will always advocate a thorough investigation with a careful look at details, which can or cannot make a big difference. In particular, when it comes to graphs, digraphs, and hypergraphs, he becomes furious about "sloppy notation" that doesn't distinguish between uv, (u, v), and $\{u, v\}$, because results do *not* automatically carry over between these cases. Martin Grötschel is correct, and the ability to seamlessly shift his attention from little details to the grand picture is without doubt one of his greatest strengths.

A formal description of the *Steiner connectivity problem* (SCP) is as follows. We are given an undirected graph $G = (V, E)$, a set of *terminal nodes* $T \subseteq V$, and a set of elementary *paths* $\mathcal{P}$ in G. The paths have nonnegative costs $c \in \mathbb{R}^{\mathcal{P}}_{\geq 0}$. The problem is to find a set of paths $\mathcal{P}' \subseteq \mathcal{P}$ of minimal cost $\sum_{p\in\mathcal{P}'} c_p$ that *connect the*

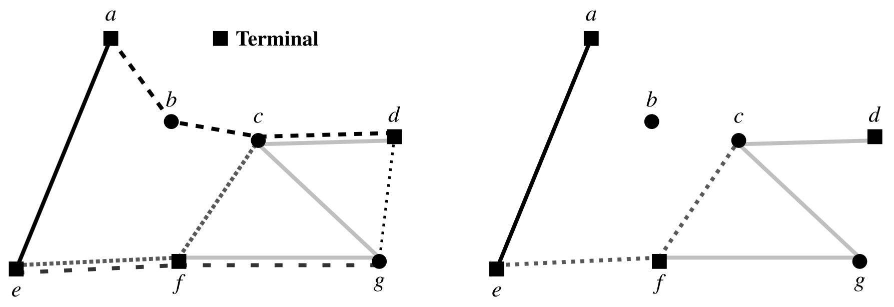


Fig. 1 Example of a Steiner connectivity problem. *Left*: A graph with four terminal nodes ($T = \{a, d, e, f\}$) and six paths ($\mathcal{P} = \{p_1 = (ab, bc, cd), p_2 = (ef, fg), p_3 = (ae), p_4 = (ef, fc), p_5 = (gd), p_6 = (fg, gc, cd)\}$). *Right*: A feasible solution with three paths ($\mathcal{P}' = \{p_3, p_4, p_6\}$)

terminals, i.e., such that for each pair of distinct terminal nodes $t_1, t_2 \in T$ there exists a path q from t_1 to t_2 in G such that each edge of q is covered by at least one path of $\mathcal{P}'$. We can assume w.l.o.g. that every edge is covered by a path, i.e., for every $e \in E$ there is a $p \in \mathcal{P}$ such that $e \in p$; in particular, G has no loops. Figure 1 gives an example of a Steiner connectivity problem and a feasible solution. Identifying the paths $\mathcal{P}$ with hyperedges in the hypergraph $(V, \mathcal{P})$ leads to an equivalent statement in terms of hypergraphs, in which the terminals have to be connected by a minimum cost set of hyperedges.

The Steiner connectivity problem is a good example for an extension study. Indeed, many results on Steiner trees can be generalized to the hypergraph setting, but not all, and not all directly. We illustrate this for two cases, namely, the 2-terminal case, and the "all-terminal case", where all nodes are terminals. The latter generalizes spanning trees to "spanning sets", but, in contrast to the graphical case, is $\mathcal{NP}$-hard. The first deals with finding a shortest hyperpath; we will see that structural properties of shortest paths carry over to this situation by proving the companion theorem to Menger's theorem for hypergraphs. A further polyhedral and computational investigation of the Steiner connectivity problem can be found in [4].

2.1 The All-Terminal Case and the Greedy Algorithm

The all-terminal case $T = V$ of the Steiner tree problem is a simple minimum spanning tree problem. In the Steiner connectivity setting, however, this case is hard. This is because of a strong relation between the SCP and the set covering problem that will be discussed now.

Proposition 1 *The Steiner connectivity problem is $\mathcal{NP}$-hard for $T = V$, even for unit costs.*

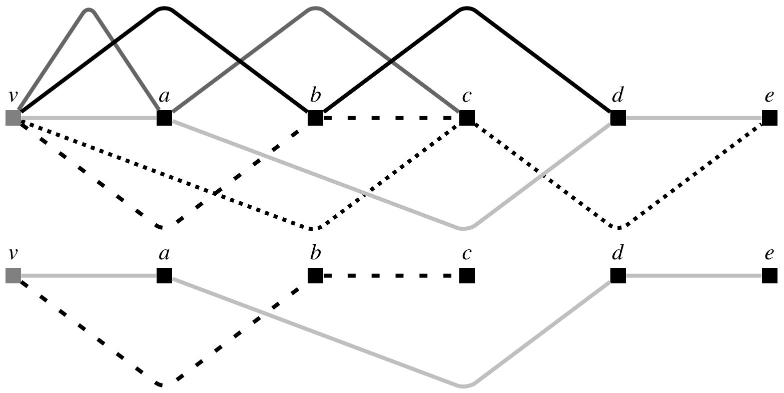


Fig. 2 *Top*: A Steiner connectivity instance corresponding to a set covering instance with $S = \{a, b, c, d, e\}$ and $\mathcal{M} = (\{a, c\}, \{b, d\}, \{b, c\}, \{c, e\}, \{a, d, e\})$. *Bottom*: A minimal solution for the Steiner connectivity problem corresponding to the minimal cover $\mathcal{M}' = (\{b, c\}, \{a, d, e\})$

Proof We reduce the set covering problem to the all-terminal Steiner connectivity problem. In a set covering problem we are given a finite set S and a set $\mathcal{M} \subseteq 2^S$. The problem is to find a subset $\mathcal{M}' \subseteq \mathcal{M}$ of minimal cardinality $|\mathcal{M}'|$ such that for all $s \in S$ there exists an $M \in \mathcal{M}'$ with $s \in M$.

Given a set covering instance, we define an all-terminal Steiner connectivity instance in a graph $G = (V, E)$ as follows: The nodes are $V = S \cup \{v\} = T$ with v being one extra node. Let us write $V = \{s_0, s_1, s_2, \ldots\}$, where $v = s_0$. All nodes are terminal nodes. We first assume that G is a complete graph and later remove all edges that are not covered by paths after their construction. For each set $M \in \mathcal{M}$ order the elements in M arbitrarily and construct a path beginning in node v and passing through all nodes of M in the given order, compare with Fig. 2. The cost of each such path is 1.

It is easy to see that a cover $\mathcal{M}'$ with at most k elements exists if and only if a set of paths exists that connects V with cost at most k, $k \geq 0$. □

Corollary 1 *SCP is strongly $\mathcal{NP}$-hard for $|T| = |V| - k$, k constant.*

Proof We add k isolated nodes to the graph G in the proof of Proposition 1. □

Proposition 2 *There is no polynomial time α-approximation algorithm for SCP with $\alpha = \gamma \cdot \log |V|$, $\gamma \leq 1$, unless $\mathcal{P} = \mathcal{NP}$.*

Proof The transformation in Proposition 1 is approximation preserving, since there exists a cost preserving bijection between the solutions of the set covering instance and its corresponding Steiner connectivity instance. It has been shown that the set covering problem is not approximable in the sense that there exists no polynomial time approximation algorithm with approximation factor smaller than logarithmic (in the number of nodes) unless $\mathcal{P} = \mathcal{NP}$, see Feige [10]. □

The proof of Proposition 1 shows that the set covering problem can be transformed to the all-terminal Steiner connectivity problem. On the other hand, the all-terminal Steiner connectivity problem can be interpreted as a submodular set covering problem. Recall that a function $z : 2^N \to \mathbb{R}$ from a set $N = \{1, \ldots, n\}$ to the reals is *submodular* if the following inequalities hold:

$$z(A) + z(B) \geq z(A \cup B) + z(A \cap B) \quad \forall A, B \subseteq N.$$

The problem

$$\min_{S \subseteq N} \left\{ \sum_{j \in S} c_j : z(S) = z(N) \right\}$$

is called the *submodular set covering problem* if z is a nondecreasing submodular function. We call this problem *integer-valued* if $z : 2^N \to \mathbb{Z}$. Let $N = \mathcal{P}$ and define for $\mathcal{P}' \subseteq \mathcal{P}$

$$z(\mathcal{P}') = |V| - \text{number of connected components in } \big(V, E(\mathcal{P}')\big),$$

where $E(\mathcal{P}')$ denotes the set of edges covered by the paths in $\mathcal{P}'$; $z(\mathcal{P}')$ can be interpreted as the maximum number of edges in $(V, E(\mathcal{P}'))$ containing no cycle. Note that this definition corresponds to the rank function for an edge set in a graphical matroid, i.e., $z(\mathcal{P}') = \text{rank}(E(\mathcal{P}'))$, see, e.g., Oxley [31]. The function z is, therefore, a nondecreasing, integer-valued, submodular set function; this follows since $E(\mathcal{P}') \subseteq E(\mathcal{P}'')$ for $\mathcal{P}' \subseteq \mathcal{P}''$. Note that $z(\mathcal{P}') = z(N) = z(\mathcal{P}) = |V| - 1$ means that $\mathcal{P}'$ connects V. Hence, the Steiner connectivity problem can be seen as an integer-valued submodular set covering problem. We have $z(p) = |p|$ for $p \in \mathcal{P}$ and $z(\emptyset) = 0$. For such problems, there exists a greedy algorithm that works fairly well:

Theorem 1 (Wolsey [37], 1982) *There is a greedy heuristic that gives an* $H(k) = \sum_{i=1}^{k} \frac{1}{i}$ *approximation guarantee for integer-valued submodular set covering problems, where* $k = \max_{j \in N} z(\{j\}) - z(\emptyset)$.

Such a greedy algorithm therefore also gives an approximation guarantee of $H(k) = \sum_{i=1}^{k} \frac{1}{i}$ for the all-terminal Steiner connectivity problem if all paths contain at most k edges. This bound is asymptotically optimal, see Feige [10] and compare with Proposition 2.

Wolsey's result generalizes an earlier one of Chvátal [8] who showed that a greedy algorithm gives an $H(k) = \sum_{i=1}^{k} \frac{1}{i}$ approximation guarantee for the set covering problem, where k is the largest column sum. Chvátal's reasoning can be extended to an elementary proof involving combinatorial counting arguments that are interesting in their own right. It goes as follows.

The proof analyzes the greedy heuristic in Algorithm 1. This procedure starts in an initial state in which each single node forms a smallest possible *(connected)*

Algorithm 1: Greedy heuristic for the SCP

```
Input  : A connected graph $G = (V, E)$, a set of paths $\mathscr{P}$ with costs $c \in \mathbb{R}^{\mathscr{P}}_{\geq 0}$.
Output: A set of paths $\mathscr{P}' \subseteq \mathscr{P}$ that connects all nodes.
1 $B^0 := \{\{v\}|v \in V\}$, $\mathscr{P}^0 := \emptyset$, $i := 1$
2 while $|B^{i-1}| > 1$ do
3     $p(i) := \arg\min_{p\in\mathscr{P}}\{\frac{c_p}{N(p,i)} : N(p,i) > 0\}$
4     $\mathscr{P}' := \mathscr{P}^i := \mathscr{P}^{i-1} \cup \{p(i)\}$
5     $B^i := (B^{i-1} \setminus \{b_1, \dots, b_j\}) \cup \{b_1 \cup \dots \cup b_j\}$ with
      $\{b_1, \dots, b_j\} := \{b \in B^{i-1} : p(i) \in \mathscr{P}, p(i) \cap b \neq \emptyset\}$
6     $i := i + 1$
7 end
```

component. The algorithm then chooses in each iteration a path that minimizes the ratio of cost over the number of components that are connected by the path minus one. These connected components are merged into a new connected component. The algorithm terminates when everything has been merged into a single connected component.

We use the following notation. Let B^i be the set of connected components and $\mathscr{P}^i$ the set of chosen paths after iteration i of Algorithm 1. Note that in each iteration at least two connected components are merged, i.e., $|B^i|$ decreases strictly with increasing i. Let us further denote by

$$\begin{aligned}N(p,i) &= z\big(\mathscr{P}^{i-1} \cup \{p\}\big) - z\big(\mathscr{P}^{i-1}\big)\\ &= \operatorname{rank}\big(E\big(\mathscr{P}^{i-1} \cup \{p\}\big)\big) - \operatorname{rank}\big(E\big(\mathscr{P}^{i-1}\big)\big)\\ &= \text{no. of conn. comp. in } \big(V, E\big(\mathscr{P}^{i-1}\big)\big)\\ &\quad - \text{no. of conn. comp. in } \big(V, E\big(\mathscr{P}^{i-1} \cup \{p\}\big)\big)\end{aligned}$$

the *component reduction number* of path p and iteration i, i.e., if p were chosen in iteration i, the total number of connected components would reduce by $N(p,i)$. Note that $N(p,i)$ is nonincreasing for increasing i, i.e., $N(p,1) \geq \dots \geq N(p,n)$ where n is the last iteration of Algorithm 1. Let $\mathscr{P}' = \{p(1), \dots, p(n)\}$. Algorithm 1 then computes a solution of cost $c(\mathscr{P}') = \sum_{i=1}^{n} c_{p(i)}$. Let, further, $\mathscr{P}_{\text{opt}} = \{o_1, \dots, o_m\}$ be an optimal V-connecting set. Finally, we denote by $H(k) = \sum_{i=1}^{k} \frac{1}{i}$ the sum of the first k terms of the harmonic series.

In order to analyze the greedy algorithm, we derive a lemma concerning the sum of the component reduction numbers of the optimal paths in iteration $i \in \{1, \dots, n\}$. This number is always greater than or equal to the sum of the component reduction numbers of the paths that are chosen by the greedy algorithm.

Lemma 1 *In Algorithm* 1 *holds*

$$\sum_{o \in \mathscr{P}_{\text{opt}}} N(o,i) \geq \sum_{j=i}^{n} N\bigl(p(j), j\bigr) \quad \forall i = 1, \dots, n. \tag{1}$$

Proof Consider the right hand side of inequality (1). We get

$$\begin{aligned}\sum_{j=i}^{n} N\bigl(p(j), j\bigr) &= z\bigl(\mathscr{P}^{i-1} \cup \bigl\{p(i)\bigr\}\bigr) - z\bigl(\mathscr{P}^{i-1}\bigr) + z\bigl(\mathscr{P}^{i} \cup \bigl\{p(i+1)\bigr\}\bigr) - z\bigl(\mathscr{P}^{i}\bigr) \\ &\quad + \dots + z\bigl(\mathscr{P}^{n-1} \cup \bigl\{p(n)\bigr\}\bigr) - z\bigl(\mathscr{P}^{n-1}\bigr) \\ &= z\bigl(\mathscr{P}^{n}\bigr) - z\bigl(\mathscr{P}^{i-1}\bigr) = |V| - 1 - z\bigl(\mathscr{P}^{i-1}\bigr) \\ &= \text{no. of conn. comp. in } \bigl(V, E\bigl(\mathscr{P}^{i-1}\bigr)\bigr) - 1.\end{aligned}$$

The claim then follows since each V-connecting set has to connect all connected components in $(V, E(\mathscr{P}^{i-1}))$. □

Proposition 3 *The greedy Algorithm* 1 *gives an $H(k)$ approximation guarantee for Steiner connectivity problems, where $k = \max_{p \in \mathscr{P}} |p|$ is the maximum path length, i.e.,*

$$c\bigl(\mathscr{P}'\bigr) \leq \sum_{p \in \mathscr{P}_{\text{opt}}} H\bigl(|p|\bigr) c_p \leq H(k) c(\mathscr{P}_{\text{opt}}).$$

Proof The idea of the proof is as follows. In a first step (assignment), we assign the path $p(i) \in \mathscr{P}'$ (added to $\mathscr{P}'$ in iteration $i = 1, \dots, n$ in Algorithm 1) to a subset of optimal paths $O(i) \subseteq \mathscr{P}_{\text{opt}}$. In a second step (bounding), we show that the cost of path $p(i)$ can be bounded from above by the cost of the paths in $O(i)$. In a third step (summation), we show that the cost of each path of the optimal solution $\mathscr{P}_{\text{opt}}$ is used at most $H(k)$ times in the bounding step.

1. *Step: Assignment*. Consider Algorithm 2. It assigns to each path $p(i)$, $i = 1, \dots, n$, of the greedy algorithm (passed in reverse order), a set $O(i) \subseteq \mathscr{P}_{\text{opt}}$ of optimal paths. The component reduction value $N(p(i), i)$ for each path $p(i)$, $i = 1, \dots, n$, is distributed to the paths $o \in O(i)$. To this purpose values $\upsilon(o,i)$ are computed such that $\upsilon(o,i) > 0 \Leftrightarrow o \in O(i)$. More precisely, in each iteration i a set $O(i) \subseteq \mathscr{P}_{\text{opt}}$ is chosen such that

$$\sum_{o \in O(i)} \upsilon(o,i) = N\bigl(p(i), i\bigr) \quad \forall i = 1, \dots, n. \tag{2}$$

Here, the values $\upsilon(o,i)$, $o \in O$, $i = 1, \dots, n$, satisfy the following condition

$$\sum_{j=i}^{n} \upsilon(o,j) \leq N(o,i) \quad \forall o \in O(i), i = 1, \dots, n. \tag{3}$$

Algorithm 2: Assigning optimal paths to the paths of the greedy algorithm

```
$\upsilon(o,i) := 0, \forall o \in \mathcal{P}_{\text{opt}}, \forall i = 1,\dots,n$
**for** $i = n$ to 1 **do**
    $O(i) := \emptyset, z := 0$
    **while** $z < N(p(i),i)$ **do**
        Choose $o \in \mathcal{P}_{\text{opt}} \setminus O(i)$ with $N(o,i) - \sum_{j=i}^{n} \upsilon(o,j) > 0$
        $\upsilon(o,i) := \min\{N(o,i) - \sum_{j=i}^{n} \upsilon(o,j), N(p(i),i) - z\}$
        $z := z + \upsilon(o,i)$
        $O(i) := O(i) \cup \{o\}$
    **end**
**end**
```

Lemma 1 ensures that these values $\upsilon(o,i)$, $i = 1,\dots,n$, exist.

2. *Step: Bounding*. Consider the path $p(i)$, $i \in \{1,\dots,n\}$, in iteration i of Algorithm 1 and the corresponding set $O(i) = \{o_1,\dots,o_h\}$ defined in Algorithm 2. Path $p(i)$ achieves the minimum in the ratio test in Step 3 of Algorithm 1. Using this fact and equation (2), the cost of $p(i)$ can be bounded as follows

$$\left.\begin{array}{c} \left.\begin{array}{c} \frac{c_{p(i)}}{N(p(i),i)} \le \frac{c_{o_1}}{N(o_1,i)} \\ \vdots \\ \frac{c_{p(i)}}{N(p(i),i)} \le \frac{c_{o_1}}{N(o_1,i)} \end{array}\right\} \upsilon(o_1,i) \text{ times} \\ \vdots \\ \left.\begin{array}{c} \frac{c_{p(i)}}{N(p(i),i)} \le \frac{c_{o_h}}{N(o_h,i)} \\ \vdots \\ \frac{c_{p(i)}}{N(p(i),i)} \le \frac{c_{o_h}}{N(o_h,i)} \end{array}\right\} \upsilon(o_h,i) \text{ times} \end{array}\right\} N\big(p(i),i\big) \text{ times.}$$

Hence, we have $c_{p(i)} \le \sum_{o \in O(i)} \frac{c_o}{N(o,i)} \upsilon(o,i)$.

3. *Step: Summation*. We finally consider how often the costs of a path $o \in \mathcal{P}_{\text{opt}}$ are used in the bounding step. We have $N(p,i) \in \{0,1,2,\dots,|p|\}$, $\forall p \in \mathcal{P}$, $i = 1,\dots,n$, hence, the total cost for a path $o \in O \subseteq \mathcal{P}$ in the bounding step can be rewritten as

$$\sum_{i=1}^{n} \frac{c_o}{N(o,i)} \upsilon(o,i) = \frac{c_o}{1} a_1 + \frac{c_o}{2} a_2 + \ldots + \frac{c_o}{|o|} a_{|o|}. \tag{4}$$

Here, the coefficients

$$a_k = \sum_{\substack{i=1 \\ N(o,i)=k}}^{n} \upsilon(o,i), \quad k = 1,\dots,|o|,$$

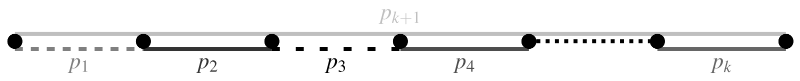


Fig. 3 Worst case Steiner connectivity example for the greedy algorithm

are sums of the values $\upsilon(o,i)$. Note that $N(o,i)$ is increasing for decreasing i. Let $s_k \in \{1,\dots,n\}$ be the smallest iteration index of Algorithm 1 such that $N(o,s_k)=k$, $k=1,\dots,|o|$ (for $k=|o|$ we have $s_k=1$), if such an index exists. Then equation (3) implies

$$a_k \le \sum_{j=1}^{k} a_j \le k, \quad k=1,\dots,|o|. \tag{5}$$

This follows immediately if $a_k=0$, i.e., if $N(o,i)\neq k$ for all $i=1,\dots,n$. Otherwise

$$\sum_{j=1}^{k} a_j = \sum_{j=1}^{k} \sum_{\substack{i=1\\ N(o,i)=j}}^{n} \upsilon(o,i) = \sum_{i=s_k}^{n} \upsilon(o,i) \le N(o,s_k) = k.$$

The term $\frac{c_o}{k}$ decreases with increasing k and is maximal for $k=1$, and (5) implies $a_1 \le 1$. This means that the sum (4) is maximal for $a_1=1$. Repeating this argument for a_2, etc., the sum (4) is maximal if all coefficients a_k, $k=1,\dots,|o|$, are 1. We then get

$$\sum_{i=1}^{n} \frac{c_o}{N(o,i)} \upsilon(o,i) \le \frac{c_o}{1} + \frac{c_o}{2} + \dots + \frac{c_o}{|o|} \le c_o H\big(|o|\big).$$

Putting everything together, we get

$$\begin{aligned} c\big(\mathcal{P}'\big) &= \sum_{i=1}^{n} c_{p(i)} \le \sum_{i=1}^{n} \sum_{o\in O(i)} \frac{c_o}{N(o,i)} \upsilon(o,i) \\ &\overset{\upsilon(o,i)=0 \text{ if } o\notin O(i)}{=} \sum_{o\in\mathcal{P}_{\text{opt}}} \sum_{i=1}^{n} \frac{c_o}{N(o,i)} \upsilon(o,i) \le \sum_{o\in\mathcal{P}_{\text{opt}}} H\big(|o|\big) c_o \\ &\le H(k)c(\mathcal{P}_{\text{opt}}). \end{aligned}$$

□

Figure 3 shows a worst-case example for the greedy heuristic. We have k paths p_i consisting of one edge with cost $c_{p_i} = \frac{1}{i}$, $i=1,\dots,k$, and one path consisting of k edges with cost $c_{p_{k+1}} = 1+\varepsilon$, $\varepsilon>0$. The greedy algorithm takes the paths $p_1,\dots,p_k$ in reverse order at a total cost of $H(k)$. The optimal solution contains only path p_{k+1} with a cost of $1+\varepsilon$.

2.2 The 2-Terminal Case and the Companion Theorem to Menger's Theorem

The 2-terminal case of the Steiner connectivity problem is to find a shortest set of paths connecting two given nodes. This is equivalent to finding a shortest hyperpath in a hypergraph that arises by interpreting paths as hyperedges. This problem can in turn be transformed into an ordinary shortest path problem in a graph replacing each path by a clique with edge weights equal to the path weight. In other words, shortest hyperpaths behave just like shortest paths. Turning to several st-paths, it is also known that Menger's theorem, stating that the maximum number of edge disjoint st-paths equals the minimum number of edges in an st-cut, generalizes to hypergraphs, see Frank [11]. In the graph case, a further result is known, the *companion theorem* to Menger's theorem, that is obtained by interchanging the roles of st-paths and st-cuts, see Robacker [36]. Both results together establish paths and cuts in graphs as a blocking pair, see Fulkerson [12]. We show in this section that the companion to Menger's theorem also holds for hypergraphs. More precisely, we prove the following theorem.

Theorem 2 *The minimum cardinality of an st-hyperpath is equal to the maximum number of hyperedge-disjoint st-hypercuts.*

This result does not follow from the above mentioned shortest path transformation. In fact, we show a stronger result, namely, that the inequality system of the cut formulation to find a shortest st-hyperpath (or 2-terminal Steiner connecting set) is totally dual integral. This extends the blocking property of paths and cuts to hypergraphs and establishes a complete structural similarity between paths and 2-terminal Steiner connecting sets.

We use the following notation. Let $H = (V, \mathcal{E})$ be a connected undirected hypergraph with costs $c_e \in \mathbb{R}$ for all hyperedges $e \in \mathcal{E}$, and s and t be two different nodes of H. Consider the following linear program

$$\begin{aligned} \text{(SH)} \quad & \min \sum_{e \in \mathcal{E}} c_e x_e \\ & \text{s.t.} \sum_{e \in \delta(W)} x_e \geq 1 \quad \forall s \in W \subseteq V \setminus \{t\} \qquad (6) \\ & \qquad x_e \geq 0 \quad \forall e \in \mathcal{E}. \end{aligned}$$

Here, we have a variable x_e for each hyperedge $e \in \mathcal{E}$. For $W \subseteq V \setminus \{t\}$, $s \in W$, an *st-hypercut* $\delta(W) = \{e \in \mathcal{E} | e \cap W \neq \emptyset, e \cap (V \setminus W) \neq \emptyset\}$ is the set of all hyperedges having at least one node in each shore. The inequality (6) guarantees for each st-hypercut that the sum of the x-values over all edges in this hypercut is at least 1. We will see that for nonnegative costs the program (SH) always has an optimal solution that is an *st-hyperpath*, i.e., a minimum cost set of hyperedges that connects s and t.

Algorithm 3: Primal-dual shortest hyperpath algorithm

```
Input  : A connected hypergraph $H = (V, \mathcal{E})$, costs $c \in \mathbb{R}^{\mathcal{E}}_{\geq 0}$, $s, t \in V$.
Output: A shortest $st$-hyperpath $x \in \{0, 1\}^{\mathcal{E}}$.
$d(s) := 0$, $d(v) := \infty\ \forall v \in V \backslash \{s\}$, $p(v) := s$, $e(v) := \emptyset\ \forall v \in V$
$i := 1$, $v_0 := s$, $W_0 := \emptyset$, $y_W := 0\ \forall W \subseteq V \setminus \{t\}, s \in W$, $x_e := 0\ \forall e \in \mathcal{E}$
while $t \notin W_{i-1}$ do
    $v := \arg\min\{d(w) | w \in V \setminus W_{i-1}\}$
    for all $f \in \mathcal{E} \setminus \delta(W_{i-1})$ with $v \in f$ do
        for all $w \in f \setminus W_{i-1}$ do
            if $d(w) > d(v) + c_f$ then
                $d(w) := d(v) + c_f$, $p(w) := v$, $e(w) := f$
            end
        end
    end
    $v_i := v$
    $y_{W_{i-1}} := d(v_i) - d(v_{i-1})$
    $W_i := W_{i-1} \cup \{v_i\}$
    $i := i + 1$
end
$k := 1$, $u_k := t$
while $u_k \neq s$ do
    $x_{e(u_k)} := 1$, $u_{k+1} := p(u_k)$, $k := k + 1$
end
```

For nonnegative costs, the primal-dual Algorithm 3 computes an optimal integral solution for program (SH) with $x_e \leq 1, \forall e \in \mathcal{E}$. It generalizes Dijkstra's algorithm to the hypergraph setting and has a better complexity than the "clique transformation method" mentioned at the beginning of the section. Its main purpose, however, is to show that the inequality system of program (SH) is totally dual integral (TDI).

Theorem 3 *The inequality system of program* (*SH*) *is TDI.*

Proof Program (SH) has the following dual

$$
\begin{aligned}
\max \quad & \sum_{W \in \mathcal{W}} y_W \\
\text{s.t.} \quad & \sum_{W \in \mathcal{W} : e \in \delta(W)} y_W \leq c_e \quad \forall e \in \mathcal{E} \\
& y_W \geq 0 \quad \forall W \in \mathcal{W},
\end{aligned}
\tag{7}
$$

where $\mathcal{W} = \{W \subseteq V \backslash \{t\} | s \in W\}$. If $c_e < 0$ for an $e \in \mathcal{E}$ then (SH) has no finite solution since x is not bounded from above; with $x_e \to \infty$ we can improve the objective arbitrarily. This means that we can assume a nonnegative integer cost vector in the following. We prove the claim for this case by showing that the primal-dual shortest hyperpath Algorithm 3 constructs optimal integral solutions x for (SH) and y for (7) with the same objective value.

The algorithm adds nodes v_i to sets $W_{i-1} = \{v_1, \dots, v_{i-1}\}$ in the order of increasing distance $d(v_i)$ from $s = v_0$ $(= v_1)$, i.e., $d(v_{i-1}) \le d(v_i) < \infty$, $i = 1, \dots, h$, with h being the last iteration of the while loop 3. This produces a sequence of nested st-hypercuts $\delta(W_i)$, $i = 1, \dots, h-1$.

We first show that y is a solution of program (7). Lines 2 and 13 imply $y \ge 0$. In fact, the variables y_W can take positive values only for $W \in \{W_1, \dots, W_{h-1}\}$.

It remains to show that

$$\sum_{W \in \mathcal{W}: e \in \delta(W)} y_W \le c_e \quad \forall e \in \mathcal{E}. \tag{8}$$

Let $e \in \mathcal{E}$. If $v_i \notin e$ for all $i = 1, \dots, h-1$, then $e \notin \delta(W_i)$, $i = 1, \dots, h-1$, i.e., $\sum_{W \in \mathcal{W}: e \in \delta(W)} y_W = 0 \le c_e$. Otherwise, let $1 \le i < h$ be the minimal index smaller than h such that $v_i \in e$, i.e., $e \notin \delta(W_j)$ for $1 \le j < i < h$ but $e \in \delta(W_i)$, and let $i \le \ell \le h-1$ be the maximal index such that $e \in \delta(W_j)$ for $i \le j \le \ell$. Then inequality (8) becomes

$$\begin{aligned}\sum_{W \in \mathcal{W}: e \in \delta(W)} y_W &= \sum_{j=i}^{\ell} y_{W_j} = \sum_{j=i}^{\ell} d(v_{j+1}) - d(v_j) \\ &= d(v_{\ell+1}) - d(v_i) \le c_e.\end{aligned}$$

For the last inequality we distinguish the cases $v_{\ell+1} \in e$ and $v_{\ell+1} \notin e$. The first case follows since $v_i \in e$. In the second case, $\ell + 1 = h$, i.e., $v_{\ell+1} = v_h = t$ and there exists a node $w \in e$ with $w \notin W_{h-1}$. Since $d(v_{\ell+1}) = d(t) \le d(w)$ and $w, v_i \in e$, the second case follows analogously.

Now we show that x is a solution of program (SH). Due to the definition of x we have $x \ge 0$. We have to show that

$$\sum_{e \in \delta(W)} x_e \ge 1 \quad \forall s \in W \subseteq V \backslash \{t\}. \tag{9}$$

Consider the nodes $t = u_1, \dots, u_k = s$ computed in the while loop starting in line 18 and an st-hypercut $\delta(W)$. Let i be the largest index with $u_i \notin W$ and $u_{i+1} \in W$. This index exists since $u_1 = t \notin W$ and $u_k = s \in W$. Then we have $x_{e(u_i)} = 1$, $e(u_i) \in \delta(W)$, and inequality (9) is satisfied.

The objective value of program (7) is

$$\sum_{i=1}^{h-1} y_{W_i} = \sum_{i=1}^{h-1} d(v_{i+1}) - d(v_i) = d(v_h) - d(v_1) = d(t) - d(s) = d(t).$$

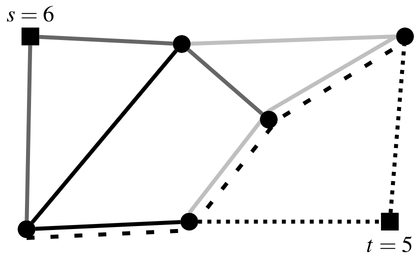


Fig. 4 Example for a Grötschel 65-index of 3: The minimum cardinality of a 65-hyperpath is $|\{p_{\text{gray}}, p_{\text{lightgray}}, p_{\text{dotted}}\}| = 3$. This equals the maximum number of hyperedge-disjoint 65-hypercuts $|\{\{p_{\text{gray}}\}, \{p_{\text{black}}, p_{\text{dashed}}, p_{\text{lightgray}}\}, \{p_{\text{dotted}}\}\}| = 3$

We, finally, get for the objective value of program (SH)

$$\begin{aligned} d(t) &= d(u_1) = d(u_2) + c_{e(u_1)} = d(u_3) + c_{e(u_2)} + c_{e(u_1)} = \dots \\ &= d(u_k) + \sum_{i=1}^{k} c_{e(u_i)} = 0 + \sum_{e \in \mathcal{E}} c_e x_e, \end{aligned}$$

i.e., x and y have the same objective value and are therefore optimal for (SH) and (7). Since c_e is integral, it follows that $d(v_i)$ is integral for $i = 0, \dots, h$. Therefore, y_{W_i}, $i = 1, \dots, h-1$, is also integral (line 13). This shows the claim for $c_e \geq 0$, $e \in \mathcal{E}$. □

Setting $c \equiv 1$ yields Theorem 2, a combinatorial result on hypergraph connectivity which has, to the best of our knowledge, not been considered before. Denote the maximum number of hyperedge-disjoint st-hypercuts the *Grötschel st-index* of a hypergraph. We can then restate Theorem 2 as follows:

Theorem 4 (Grötschel st-index Theorem) *The minimum cardinality of an st-hyperpath is equal to the Grötschel st-index.*

Figure 4 gives an illustration. As this result is derived from a careful analysis of the hypergraph vs. the graph case, we feel that it fits very well to dedicate this Theorem to Martin Grötschel on the occasion of his 65th birthday.

Interchanging the roles of st-hyperpaths and st-hypercuts yields Menger's theorem for hypergraphs, see [11].

Theorem 5 *The minimum cardinality of an st-hypercut is equal to the maximum number of hyperedge-disjoint st-hyperpaths.*

Grötschel's and Menger's Theorems 4 and 5 are therefore companion theorems indeed.

3 The Steiner Tree Packing Problem

When Martin Grötschel came in 1989 to Alexander Martin and suggested the topic of packing Steiner trees in graphs as a Ph.D. thesis, he became immediately very curious about it. Martin Grötschel further supported it by saying that this is a topic for the next decades. He claimed that very few is known when it comes to packing problems in general, one reason among others is that it is open on how to integrate and exploit dual information. And he seems to be right until today. We all know that some progress has been made when it comes to classical packing problems such as the set packing or the bin packing problem. But for the STPP, when the objects to be packed have no fixed shape and are flexible in size and structure in dependence on the other objects to be packed, the problem seems to be harder by orders of magnitudes. None of the successful techniques like preprocessing or heuristics for the single Steiner tree problem work anymore, and new ideas must be developed. With very few exceptions, cf. [14–19], the STPP is still open for wonderful discoveries, supporting once more the great ability of Martin Grötschel to identify future trends and challenging problems.

The Steiner tree packing problem (STPP) looks at the following situation. Instead of having one set of terminals, we have N non-empty disjoint sets $T_1, \dots, T_N$, called *Nets*, that have to be "packed" into the graph simultaneously, i.e., the resulting edge sets $S_1, \dots, S_N$ have to be pairwise disjoint. In these applications, G is usually some sort of 3D grid graph. [19, 20, 27] give detailed explanations of the modeling requirements in VLSI-design. From a theoretical point of view, much less is known, see [7, 26].

Three routing models for 2D or 3D grid graphs are of particular interest:

Channel routing: Here, a complete rectangular grid graph is used. The terminals of the nets are exclusively located on two opposing borders. The size of the routing area is not fixed in advance. All nets have only two terminals, i.e., $|T_i| = 2$.

Switchbox routing: We are given a complete rectangular grid graph. The terminals may be located on all four sides of the graph. Thus, the size of the routing area is fixed.

General routing: In this case the grid graph may contain holes or have a non-rectangular shape. The size of the routing area is fixed and the terminals may be located arbitrarily.

The intersection of the nets is an important issue in Steiner tree packing. Again three different models are possible:

Manhattan (Fig. 5(a)) Consider some (planar) grid graph. The nets must be routed in an edge disjoint fashion with the additional restriction that nets that meet at some node are not allowed to bend at this node, i.e., so-called *Knock-knees* are not allowed. This restriction guarantees that the resulting routing can be laid out on two layers at the possible expense of causing long detours.

Knock-knee (Fig. 5(b)) Again, some (planar) grid graph is given and the task is to find an edge disjoint routing of the nets. In this model Knock-knees are possible.

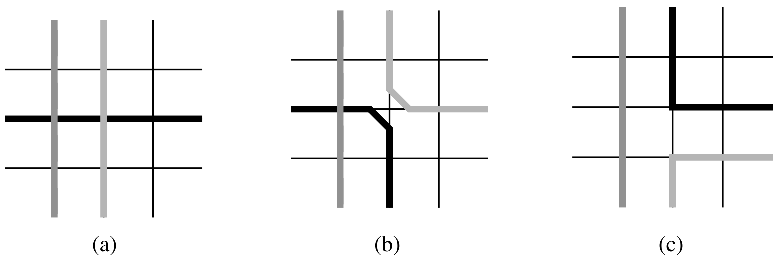


Fig. 5 STPP intersection models. **a** Manhattan model. **b** Knock-knee model. **c** Node disjoint model

Very frequently, the wiring length of a solution is smaller than in the Manhattan model. The main drawback is that the assignment to layers is neglected.

Node disjoint (Fig. 5(c)) The nets have to be routed in a node disjoint fashion. Since no crossing of nets is possible in a planar grid graph, this requires a multi-layer model, i.e., a 3D grid graph.

While channel routing usually involves only a single layer, switchbox and general routing problems are typically multi-layer problems. Using the Manhattan and Knock-knee intersection is a way to reduce the problems to a single layer. Accordingly, the multi-layer models typically use the node disjoint intersection. While the multi-layer model is well suited to reflect reality, the resulting graphs become quite large. We consider two possibilities to model multiple layers; a third possibility is to use a single-layer model with edge capacities greater than one:

k-crossed layers (Fig. 6(a)) A k-dimensional grid graph (i.e., k copies of a grid graph are stacked on top of each other and corresponding nodes are connected by perpendicular lines, so-called *vias*) is given, where k denotes the number of layers. This is called the k-layer model in [27].

k-aligned layers (Fig. 6(b)) This model is similar to the crossed-layer model, but in each layer there are only connections in one direction, either east-to-west or north-to-south. [27] calls this the *directional* multi-layer model. [26] indicate that for $k = 2$ this model resembles the technology used in VLSI-wiring best. It is mentioned in [2] that current technology can use a much higher number of layers (20 and more).

Note that for switchbox routing there is a one-to-one mapping between feasible solutions for the Manhattan one-layer model (MOL) and the node disjoint two-aligned-layer model (TAL), assuming that there are never two terminals on top of each other, i.e., connected by a via.

For the general routing model, this mapping might not be possible. If a terminal is within the grid, there is no easy way to decide the correct layer for the terminal in the two-layer model.

Unfortunately, in the seven "classic" instances given by [6, 9, 29] two terminals are connected to a single corner in several cases. This stems from the use of *connec-*

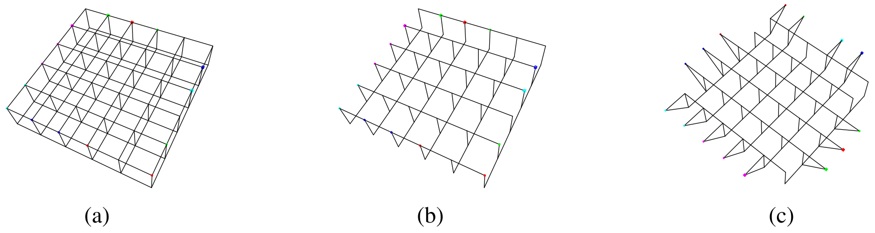


Fig. 6 STPP modeling taxonomy. **a** Multi-crossed layers. **b** Multi-aligned layers. **c** With connectors

tors, i.e., the terminal is outside the grid and connected to it by a dedicated edge. In the multi-layer models there has to be an edge from the terminal to all permissible layers (Fig. 6(c)).

The Knock-knee one-layer model can also be seen as an attempt to approximate the node disjoint two-crossed-layer model. But mapping between these two models is not as easy. [5] have designed an algorithm that guarantees that any solution in the Knock-knee one-layer model can be routed in a node disjoint four-crossed-layer model, but deciding whether three layers are enough has been shown to be $\mathcal{NP}$-complete by [28].

For our computational investigations we will use a multicommodity flow formulation [25] that was proposed by [38] for the STP. Given a weighted bidirectional grid digraph $G = (V, A, c)$ and sets $T_1, \dots, T_N$, $N > 0$, $|T_n| > 0$ of terminals, we arbitrarily choose a root $r_n \in T_n$ for each $n \in \mathcal{N} := \{1, \dots, N\}$. Let $R = \{r_n | n \in \mathcal{N}\}$ be the set of all roots and $T = \bigcup_{n \in \mathcal{N}} T_n$ be the union of all terminals. We introduce binary variables x_{ij}^n for all $n \in \mathcal{N}$ and $(i, j) \in A$, where $x_{ij}^n = 1$ if and only if arc $(i, j) \in S_n$. Additionally, we introduce binary variables y_{ij}^t, for all $t \in T \setminus R$. For all $i \in V$, we define $\delta_i^+ := \{(i, j) \in A\}$ and $\delta_i^- := \{(j, i) \in A\}$. For all $t \in T_n$, $n \in \mathcal{N}$, we define $\sigma(t) := n$. The following formulation models all routing choices for any number of layers, crossed and aligned, with Knock-knee intersection:

$$\min \sum_{n \in \mathcal{N}} \sum_{(i,j) \in A} c_{ij}^n x_{ij}^n \tag{10}$$

$$\sum_{(i,j) \in \delta_j^-} y_{ij}^t - \sum_{(j,k) \in \delta_j^+} y_{jk}^t = \begin{Bmatrix} 1 & \text{if } j = t \\ -1 & \text{if } j = r_{\sigma(t)} \\ 0 & \text{otherwise} \end{Bmatrix} \quad \text{for all } j \in V, t \in T \setminus R \tag{11}$$

$$0 \le y_{ij}^t \le x_{ij}^{\sigma(t)} \quad \text{for all } (i, j) \in A, t \in T \setminus R \tag{12}$$

$$\sum_{n \in \mathcal{N}} (x_{ij}^n + x_{ji}^n) \le 1 \quad \text{for all } (i, j) \in A \tag{13}$$

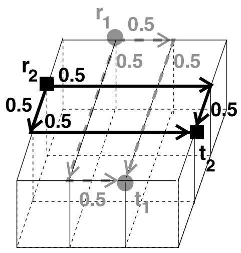


Fig. 7 LP relaxation solution violates (17)

$$x_{ij}^n \in \{0, 1\} \quad \text{for all } n \in \mathcal{N}, (i, j) \in A \tag{14}$$

$$y_{ij}^t \in \{0, 1\} \quad \text{for all } t \in T \setminus R, (i, j) \in A \tag{15}$$

To use node disjoint intersection we have to add:

$$\sum_{n \in \mathcal{N}} \sum_{(i,j) \in \delta_j^-} x_{ij}^n \leq \begin{cases} 0 & \text{if } j \in R \\ 1 & \text{otherwise} \end{cases} \quad \text{for all } j \in V. \tag{16}$$

3.1 Valid Inequalities

For the node disjoint crossed-layer model there is a class of simple but very effective cuts, which are introduced in [21]. Consider a STPP problem of two nets, each has two terminals as in Fig. 7. The flows shown in the picture correspond to an optimal solution of the LP relaxation. However, it can be seen that if none of the two flows (r_1, t_1) and (r_2, t_2) leaves the upper layer, they have to cross each other, i.e., the node disjoint intersection condition is violated. In the following we construct cuts, which cut off the fractional solution in Fig. 7.

A pair $(s_1, t_1), (s_2, t_2) \in T \times (T \setminus R)$ are called crossed if these four terminals lie on the boundary and in the same layer, $\sigma(s_1) = \sigma(t_1)$, $\sigma(s_2) = \sigma(t_2)$, $\sigma(s_1) \neq \sigma(s_2)$, and, moreover, the line segments (s_1, t_1) and (s_2, t_2) cross each other. Let $\mathcal{C}$ be the set of all crossing pairs and for each node v we denote by v_z the layer number of node v. Then the following inequality is valid for (10):

$$\sum_{\substack{ij \in A \\ i_z = (r_1)_z, j_z \neq i_z}} y_{ij}^{t_1} + y_{ij}^{t_2} \geq 1, \quad \forall\big((r_1, t_1), (r_2, t_2)\big) \in \mathcal{C}(R), \tag{17}$$

where $\mathcal{C}(R) := \{((s_1, t_1), (s_2, t_2)) \in \mathcal{C} | s_1, s_2 \in R\}$. Equation (17) means that at least one of the two flows (r_1, t_1) and (r_2, t_2) has to leave the layer containing these terminals.

A triple $(r_1, t_1), (r_2, t_2), (r_3, t_3) \in R \times (T \setminus R)$ is called a crossing triple if each two of them are a crossing pair in $\mathcal{C}(R)$. Let $\mathcal{CT}$ be the set of all crossing triples then the following inequality is valid for (10):

$$\sum_{\substack{ij\in A\\ i_z=(r_1)_z,\, j_z\neq i_z}} y_{ij}^{t_1}+y_{ij}^{t_2}+y_{ij}^{t_3}\geq 2,\quad \forall\big((r_1,t_1),(r_2,t_2),(r_3,t_3)\big)\in\mathcal{CT}. \tag{18}$$

This is a direct implication from summing up the three corresponding inequalities of type (17) taking into account that the variables have to be integral.

Based on the same principle, we can derive more valid cuts involving both variables x and y. Again, we assume that all terminals lie on the boundary. Let u and v be two terminals in one layer of a net n and the root of this net does not belong to the layer containing u and v. If there exist a root r and a terminal t of another net such that (u, v) and (r, t) cross each other, i.e., $((u, v), (r, t)) \in \mathcal{C}$, then the following inequality is valid for (10)

$$\sum_{\substack{ij\in A\\ j_z=t_z,\, j_z\neq i_z}} y_{ij}^{t}+x_{ij}^{n}\geq 2. \tag{19}$$

The following valid cut is similar to (19) but two terminal pairs and three terminals of a third net are involved. We consider an arbitrary net n with at least three terminals and four terminals r_1, t_1, r_2 and t_2 of two other nets n_1 and n_2, $\sigma(r_1)=\sigma(t_1)=n_1\neq n$, $\sigma(r_2)=\sigma(t_2)=n_2\neq n$, $n_1\neq n_2$, with $r_1, r_2 \in R$. If there exist three terminals u, v and w of net n lying in the same layer such that

$$\forall\{s,t\}\subset\{u,v,w\},\, s\neq t:\big((s,t),(r_1,t_1)\big)\in\mathcal{C}\text{ or }\big((s,t),(r_2,t_2)\big)\in\mathcal{C}, \tag{20}$$

then the following inequality is valid:

$$\sum_{\substack{ij\in A\\ j_z=(t_1)_z,\, j_z\neq i_z}} y_{ij}^{t_1}+y_{ij}^{t_2}+x_{ij}^{n}\geq 2. \tag{21}$$

Since all terminals lie on the boundary, each line segment between two terminals, which crosses an edge of a "triangle" of three terminals (including the case that they lie in a line), crosses exactly two edges of this triangle. Therefore, condition (20) just ensures that the line segments (r_1, t_1) and (r_2, t_2) cross the triangle (u, v, w) in two different pairs of edges. Inequality (21) means that the total number of vias used by the flows (r_1, t_1) and (r_2, t_2) and the net n to enter the layer containing these terminals is at least two. The proof can be found in [21]. Moreover, one can prove that, if u, v and w satisfy the above condition and u, v and w do not lie in the same layer as the root of net n then the following inequality is valid:

$$\sum_{\substack{ij\in A\\ j_z=(t_1)_z,\, j_z\neq i_z}} y_{ij}^{t_1}+y_{ij}^{t_2}+x_{ij}^{n}\geq 3. \tag{22}$$

The proof can also be found in [21].

The next type of valid inequality does not require any of the terminals to be the root of the net they belong to. For arbitrary three terminals u_1, v_1 and w_1 of a net n_1, and arbitrary three terminals u_2, v_2 and w_2 of a net $n_2 \neq n_1$, if

$$\forall \{s_1, t_1\} \subset \{u_1, v_1, w_1\}, \exists \{s_2, t_2\} \subset \{u_2, v_2, w_2\} : \big((s_1, t_1), (s_2, t_2)\big) \in \mathcal{C}, \tag{23}$$

and

$$\forall \{s_2, t_2\} \subset \{u_2, v_2, w_2\}, \exists \{s_1, t_1\} \subset \{u_1, v_1, w_1\} : \big((s_1, t_1), (s_2, t_2)\big) \in \mathcal{C}, \tag{24}$$

i.e., the two triangles (u_1, v_1, w_1) and (u_2, v_2, w_2) cross each other, then the following inequality is valid for (10):

$$\sum_{\substack{ij \in A \\ j_z = (u_1)_z, j_z \neq i_z}} x_{ij}^{n_1} + x_{ij}^{n_2} \geq 2 + \delta_1 + \delta_2, \tag{25}$$

where δ_i is 1 if the root of net n_i does not lie in the same layer as u_i, v_i and w_i, and 0 otherwise.

3.2 *Heuristics*

As we will see in the following the above described model provides very strong lower bounds in practice. Nevertheless, current IP solvers often not only fail to solve the IP model to optimality, it is even very time-consuming to find a feasible solution.

However, based on the special structure of the grid graph we can find optimal solutions for all of our considered instances in reasonable time. The following is an extension to [21] and describes the heuristics we developed to improve the solution of the STPP. The heuristics presented in this section are based on solving the IP of relaxed problems. There are two kinds of relaxed problems. The first one is also the STPP problem using the multicommodity flow formulation (10) but on a subgraph of the original grid graph. The second one is the original multicommodity flow IP where some variables are fixed based on a given feasible solution of the original problem. In the following we call these two kinds of heuristics phase 1 and phase 2, respectively. The solving process of the STPP starts with phase 1 and then executes phase 2. The two phases of the heuristics are presented in detail below.

3.2.1 Heuristics Phase 1

Computational results show that feasible solutions of the multi-aligned layers can be found easily if they exist. This motivates us to consider a heuristic process, where instead of starting solving the original multi-crossed layers model, we solve several STP problems corresponding to some sparser underlying grids, e.g., the multi-aligned layers grid. After each step we obtain a feasible solution. Then we add some

Algorithm 4: Heuristics phase 1

```
Step 1: Solve the STPP on the multi-aligned layers grid.
Step 2: Add all missing edges in the set $S(l_2, r_2, b_2, t_2)$ to the grid graph in
    Step 1. Find a good solution of the STPP on this new grid graph using the
    solution of Step 1 as starting solution.
Step 3: Add all missing edges in the set $S(l_3, r_3, b_3, t_3)$ to the grid graph in
    Step 2. Find a good solution of the STPP on this new grid graph using the
    solution of Step 2 as starting solution.
```

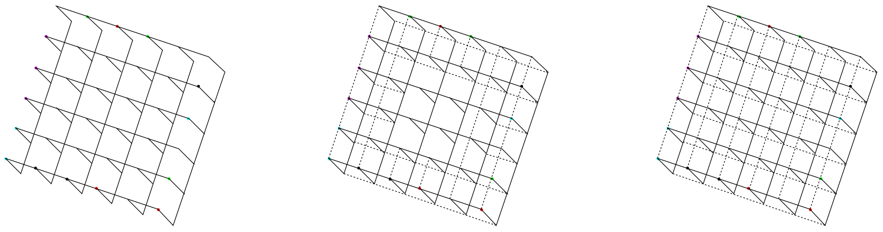


Fig. 8 STPP heuristics phase 1—underlying grid graphs. **a** Step 1. **b** Step 2. **c** Step 3

edges to the grid and resolve the STP problem which corresponds to the new grid, using the solution from the previous step as a start value for the optimization problem. The question is which edges should be added in each step. There is no optimal strategy at the moment. Algorithm 4 describes our implemented algorithm, where $S(l, r, b, t)$ is the set of all edges lying outside $[l, K - r] \times [b, L - t]$, i.e.,

$$S(k, l) := \big\{(i, j) \in E \;\big|\; (i_x, i_y), (j_x, j_y) \notin [l, K - r] \times [b, L - t]\big\},$$

with (i_x, i_y) is the coordinate of the vertex i in the layer which it belongs to, $[0, K] \times [0, L]$ is the given grid, and E is the set of edges in the multi-crossed layer model.

Figure 8 demonstrates the grid graphs in the three steps, where the added edges are dotted.

It is still unknown what is the best way to choose the parameters (l_2, r_2, b_2, t_2) and (l_3, r_3, b_3, t_3). For the instances in Sect. 3.3 we choose $l_2 = r_2 = b_2 = t_2 = 3$ for pedabox-2 and $l_2 = r_2 = b_2 = t_2 = 5$ for the other instances. In step 3, to obtain the parameters l_3, r_3, b_3, t_3, we increase the corresponding parameters l_2, r_2, b_2, t_2 from step 2 by at most 2. For example, we choose $l_3 = r_3 = 7$ and $b_3 = t_3 = 6$ for the instances difficult-2 and more-diff-2.

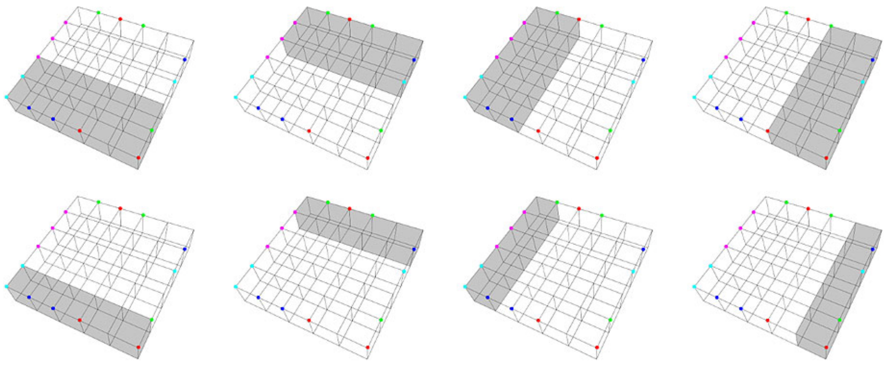


Fig. 9 STPP heuristics phase 2—fixing regions

3.2.2 Heuristics Phase 2

Having a feasible solution from heuristics phase 1 we can start the fixing heuristics. Figure 9 shows an example for the procedure of these heuristics. This figure reads from left to right and from top to bottom. Let us start with the picture on top left. We fix the variables corresponding to the edges in the gray region to the values of the given feasible solution for those variables. Then we solve the IP (10) with those fixings and obtain a possibly better feasible solution. With the newly obtained feasible solution we go to the second step described by the second picture in the top of Fig. 9, where again the fixing area is colored by gray, and so on. The given feasible solution from the beginning is improved step by step, where each step uses the best feasible solution obtained in the previous step for fixing. At the moment there is no common rule for the fixing region and the number of steps. Algorithm 5 presents the pseudo code of phase 2 in our implemented code. We execute four steps with the sequence of the fixing regions as the four pictures either on the left side or on the right side of Fig. 9, respectively, as follows. Let $K \times L$ be the size of the grid. Without loss of generality, we assume that $K \leq L$. Otherwise we have to modify the following definition accordingly by switching the role of K and L. The four fixing regions are defined as

$$F_l := \left\{(i,j) \in E \;\middle|\; i_y, j_y \leq \left\lfloor \frac{K}{p_l} \right\rfloor \right\}, \quad l = 1, 3$$

$$F_l := \left\{(i,j) \in E \;\middle|\; i_y, j_y \geq K - \left\lfloor \frac{K}{p_l} \right\rfloor \right\}, \quad l = 2, 4,$$

where $3 \leq p_1, p_2 \leq 4$ and $5 \leq p_3, p_4 \leq 6$ are chosen depending on instances. For our considered instances, this procedure gives us already optimal solutions. However, for larger instances, we may need to execute more steps and/or use more types of fixing regions, e.g., all eight steps in Fig. 9.

Algorithm 5: Heuristics phase 2

```
for $i = 1$ to 4 do
    Choose $F_i$ as the fixing region and the solution obtained from phase 1 in
    the case $i = 1$ and from the previous loop $i - 1$ otherwise as the solution
    used for fixing. Find a good solution of the corresponding relaxed problem.
end
```

Table 1 Results for the Knock-knee-one-layer model

Name	Size	N	$\lvert T\rvert$	B&B Nodes	Time [s]	LP relaxation	Arcs
aug-dense-1	16×18	19	59	63	1,649	466.5	469
dense-1	15×17	19	59	150	1,199	438.0	441
difficult-1	23×15	24	66	1	17	464.0	464
mod-dense-1	16×17	19	59	1	29	452.0	452
more-diff-1	22×15	24	65	1	12	452.0	452
pedabox-1	15×16	22	56	1	7	331.0	331
termintens-1	23×16	24	77	1	96	535.0	536

3.3 *Computational Results*

In this section, we present computational results obtained by generating the integer program resulting from the directed multicommodity flow formulation with ZIMPL [24] and then solving it with CPLEX 12.3 for the STPP instances taken from [30]. All computations are done on a 48 GB RAM dual quad-core Intel Xeon X5672 at 3.20 GHz with TurboBoost active and Hyperthreading deactivated. Since the crossed-layer model proved to be much harder to solve, we used all eight cores, while just one core was utilized for the other models. Still, for the crossed-layer models we will use minutes as the unit for reporting time, in contrast to seconds for the other models. As we had expected from earlier experiments, the MCF-Cuts [1, 34] introduced by CPLEX 12 had no impact on solving the instances. The reason is that the models used in this paper are not capacitated. If not noted otherwise, CPLEX was used in default mode with integer optimality gap tolerance set to 0.0.

3.3.1 Results for the Knock-Knee One-Layer Model

Table 1 shows the results for the Knock-knee one-layer model. *B&B Nodes* denotes the number of Branch-and-Bound nodes including the root node evaluated by CPLEX. The column labeled *Time* shows the consumed CPU time in seconds. LP *relaxation* lists the objective function value of the initial LP relaxation of the root node before any cuts applied by CPLEX. Finally, *arcs* is the total number of arcs used in the optimal solution which for one-layer models is equivalent to the optimal objective value.

As we can see from the table, the LP relaxation of the flow model is rather strong. This is in line with other reported results including [22, 30]. Since for *difficult-1*, *mod-dense-1*, *more-diff-1*, and *pedabox-1* the relaxation already provides the optimal value, it is possible to solve these instances without any branching. For *termintens-1* the relaxation is one below the optimum, but CPLEX is able to push the lower bound up by generating Gomory rounding and 0–1/2-Chvátal-Gomory cuts. The number of B&B nodes and therefore the computing time depends very much on the branching decisions taken. During our experiments the solutions were always found in the tree and not by heuristics. By using improved settings, like switching off the heuristics and just trying to move the best bound, the number of nodes needed for *aug-dense-1* and *dense-1* can be at least halved.

3.3.2 Results for the Node Disjoint Multi-aligned-layer Model

Table 2 shows results for the node disjoint multi-aligned-layer model. Since this is a multi-layer model we have to assign costs to the vias. These are given in the column labeled *Via-cost*. The column labeled *LP relaxation* gives the objective value of the initial LP relaxation. The next three columns list the numbers of vias, "regular" arcs, and vias+arcs in the optimal solution.

In case of unit via costs, the objective value of the LP relaxation is equal to the objective value of the optimal integer solution for all instances except for *more-diff-2*. The value of the LP relaxation for *more-diff-2* is 518.6 (optimal 522). This is weaker than the value reported in [19], which indicates that some of the strengthening cuts used by [19] to tighten the undirected partitioning formulation can also be used to tighten the directed flow formulation. On the other hand, for *pedabox-2* the relaxation is stronger than reported. The instances where the LP relaxation does not reach the optimum are different ones from the Knock-knee-one-layer model.

Via Minimization Traditionally via minimization is viewed as a separate problem after the routing has taken place [13]. Since we work with multi-layer models, via minimization is part of the routing. As can be seen in Table 2 we tried the "classical" instances with three different cost settings for the vias. First unit costs were used to minimize the total number of arcs, including vias. Next, the number of vias was minimized by setting the cost to 1,000, which dominates the total cost of all "regular" arcs, ensuring that a global minimum is reached. Finally, the cost of each via was set to 0.001, effectively minimizing the number of "regular" arcs. This results in solutions that have the same number of arcs as reported in [19] for the Manhattan one-layer model.

Interestingly, the number of vias is constant for *aug-dense-2*, *pedabox-2*, *modifieddense-3*, and *dense-3*. For the other instances, a minimization of the number of vias always results in detours, i.e., a higher total number of arcs used.

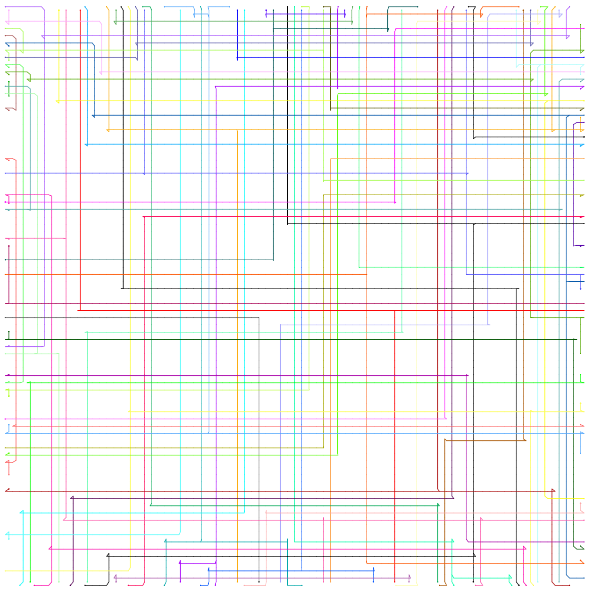
Table 2 Results for the node disjoint multi-aligned-layer model

Name	Size	N	$\|T\|$	B&B Nodes	Time [s]	Via-cost	LP relaxation	Vias	Arcs	Vias+Arcs
aug-dense-2	16×18	19	59	1	32	1	504.0	35	469	504
aug-dense-2	16×18	19	59	1	20	1000	35,469.0	35	469	504
aug-dense-2	16×18	19	59	1	67	0.001	469.035	35	469	504
difficult-2	23×15	24	66	1	23	1	526.0	56	470	526
difficult-2	23×15	24	66	7	181	1000	50,310.2727	51	484	535
difficult-2	23×15	24	66	1	50	0.001	468.8363	63	469	532
more-diff-2	22×15	24	65	5	113	1	518.60	61	461	522
more-diff-2	22×15	24	65	37	705	1000	50,276.8465	53	481	534
more-diff-2	22×15	24	65	1	38	0.001	460.9917	61	461	522
pedabox-2	15×16	22	56	1	10	1	390.0	47	343	390
pedabox-2	15×16	22	56	11	34	1000	45,885.1333	47	343	390
pedabox-2	15×16	22	56	14	78	0.001	341.4601	47	343	390
termintens-2	23×16	24	77	1	28	1	596.0	59	537	596
termintens-2	23×16	24	77	1	31	1000	55,562.0	55	562	617
termintens-2	23×16	24	77	1	28	0.001	537.059	59	537	596
dense-3	15×17	19	59	1	30	1	471.0	35	436	471
dense-3	15×17	19	59	1	21	1000	35,436.0	35	436	471
dense-3	15×17	19	59	1	44	0.001	436.035	35	436	471
mod-dense-3	16×17	19	59	1	24	1	485.0	35	450	485
mod-dense-3	16×17	19	59	1	26	1000	35,450.0	35	450	485
mod-dense-3	16×17	19	59	1	33	0.001	450.035	35	450	485

Table 3 STPP new instances

Name	Size	N	$\|T\|$	Variables	Constrains	Non-zeros
Node disjoint two-aligned-layer model						
sb80-80	81 × 81	60	158	6,168,636	5,150,535	19,965,324
sb99-99	100 × 100	70	183	10,906,800	9,052,440	35,250,720

Fig. 10 sb80-80



New Instances All the instances presented so far are relatively old and can be solved in less than 12 minutes with CPLEX. To get an outlook on how far our approach will take us, we tried some new instances, see Table 3. *sb80-80* and *sb99-99* are randomly generated switchbox instances. *sb80-80* is about 35 times the size of the largest "classical" instance, and the resulting IP has more than six million variables and almost twenty million non-zero entries in the constraint matrix. *sb99-99* is again about 2 times larger than *sb80-80*. As discussed in [21] and [25], for several instances it was faster to solve the LP relaxations from scratch with the barrier algorithm than to reoptimize with the dual simplex algorithm. This setting is used for the new instances with a newer version of CPLEX, namely, CPLEX 12.4.

For sb80-80 the value of the LP relaxation turned out to be equal to the value of the integer optimal solution, namely 6533, and only one branch and bound node is needed. CPLEX takes 94,926 seconds to solve the root relaxation and finishes after 94,232 seconds, i.e., 26.18 hours, with the optimal solution, which has 226 vias and 6307 normal arcs, see Fig. 10.

The time for solving the root relaxation of sb99-99 is 356,664 seconds, i.e., 4.13 days. After more than 2 weeks CPLEX cannot either find a feasible solution or prove that the problem is infeasible. Gurobi 5.0 also experiences the same difficulty with this instance.

These computations show the bottleneck of our approach. The resulting IPs are large and solving their LP relaxations is already very hard. Any improvement in solving LP will turn out directly to an improvement in our approach.

3.3.3 Results for the Node Disjoint Multi-crossed-layer Model

Finally, we will have a look at the crossed-layer models. For the instances listed in Table 4, except for *dense-3* and *mod-dense-3*, the heuristics presented in Sect. 3.2 was used to provide CPLEX with a starting solution. For *dense-3* and *mod-dense-3* CPLEX quickly found solutions. Therefore, employing the heuristics provided no advantage. For all instances the provided solution turned out to be already optimal. The time needed to compute these initial solution is given under *Heur*.

To solve the instances we used 8 threads in opportunistic mode, the total times needed including the heuristics is reported in column *Total* as minutes of wall clock time. The *optimization emphasis* of CPLEX was set to "optimality", cut generation was set "aggressive" for Gomory-, 0–1/2-, and cover-cuts, all other cuts were set to "moderate". Furthermore, we explicitly added cuts (17)–(25) presented in Sect. 3.1 to the User-Cut pool of CPLEX.

While it is possible to find reasonable solutions with our heuristics, proving optimality is still a hard task. The table is ordered by an increasing difference between the LP relaxation and the optimum value. As can be seen this is reflected quite well in the number of B&B nodes needed to prove optimality. While the cuts we presented in this paper proved quite helpful, the main reason for the long running time is that solving the node LPs is very time consuming. For example, the number of B&B nodes that CPLEX needs to solve the instance *termintens-2* is merely 3,453. However, it takes 51.7 hours to solve the problem and 43.3 hours for the first 2,000 nodes.

Figure 11 shows the primal and dual bounds during the solving process of the instance *pedabox-2* with CPLEX using default setting and CPLEX using our heuristics, valid cuts, and variables elimination. Clearly, our approach improves both primal and dual bounds. For *pedabox-2* with unit via-cost, our heuristics found an optimal solution after 17.8 minutes, while CPLEX alone could not find the optimal value after more than 80 hours. For this problem our approach (using crossing cuts and special heuristics) gives a dual bound of value 358.2311 directly after the root node, while default CPLEX reaches this value only after 46.1 hours and 158,507 nodes. We stop solving with default CPLEX after 83.33 hours and 300,516 nodes, and obtain a dual bound of 358.7628. This value is already reached by our approach after 49.6 minutes and 471 nodes.

Table 4 Results for the node disjoint multi-crossed-layer model

Name	Size	N	$\|T\|$	Heur [m]	Total [m]	B&B Nodes	LP relaxation	Vias	Arcs	Vias+Arcs
dense-3	15×17	19	59	–	55	77	461.8062	30	434	464
mod-dense-3	16×17	19	59	–	88	121	476.9078	28	451	479
aug-dense-2	16×18	19	59	108	269	227	492.6260	29	469	498
pedabox-2	15×16	22	56	18	112	3,027	353.4275	26	336	362
difficult-2	23×15	24	66	28	822	2,214	492.5417	39	464	503
termintens-2	23×16	24	77	97	3,103	3,453	573.1981	46	538	584
more-diff-2	22×15	24	65	60	30,727	24,713	481.1991	38	455	493

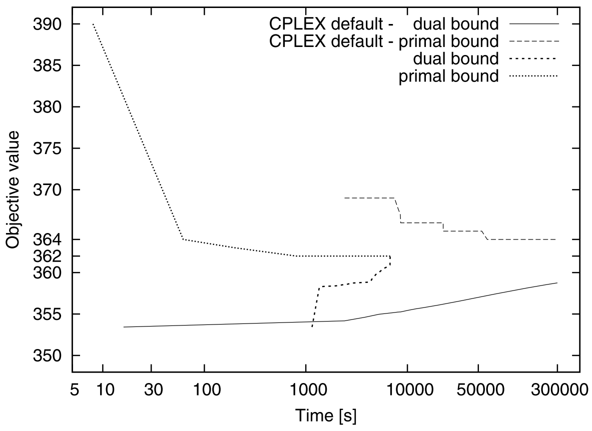


Fig. 11 pedabox-2 with unit via cost—primal and dual bounds

4 Conclusion and Outlook

The Steiner connectivity and the Steiner tree packing problem are just two examples of challenging and important questions to find the best possible connection of a set of terminals in a graph. For such problems chances are good that we can derive theoretical insights and come up with quite powerful solution algorithms, guided by our knowledge of the basic case. We are admittedly still working on individual problem variants, far from anything like a universal solution engine, and, in fact, going one step further into the direction of industrial models, e.g., in line planning or chip design, immediately makes things much more difficult, and even more, since problem sizes of real-world problems are growing fast. But such is life, would Martin Grötschel say. And he could still come, 20 years after Alexander Martin finished his thesis, into the office of one of his Ph.D. students and talk enthusiastically about an interesting and open Steiner problem.

Acknowledgements We thank an anonymous referee and the editors for helpful comments and suggestions that improved the presentation of this paper. The work of Marika Karbstein was supported by the DFG Research Center MATHEON "Mathematics for key technologies".

References

1. Achterberg, T., Raack, C.: The MCF-separator—detecting and exploiting multi-commodity flows in MIPs. Math. Program. Comput. **2**, 125–165 (2010)
2. Boit, C.: Personal communication (2004)
3. Borndörfer, R., Karbstein, M.: A direct connection approach to integrated line planning and passenger routing. In: Delling, D., Liberti, L. (eds.) 12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems. OpenAccess Series in Informatics (OASIcs), vol. 25, pp. 47–57. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Wadern (2012)
4. Borndörfer, R., Karbstein, M., Pfetsch, M.E.: The Steiner connectivity problem. Math. Program., Ser. A (2012). doi:10.1007/s10107-012-0564-5

5. Brady, M.L., Brown, D.J.: VLSI routing: four layers suffice. In: Preparata, F.P. (ed.) Advances in Computing Research: VLSI Theory, vol. 2, pp. 245–258. Jai Press, London (1984)
6. Burstein, M., Pelavin, R.: Hierarchical wire routing. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **2**, 223–234 (1983)
7. Chopra, S.: Comparison of formulations and a heuristic for packing Steiner trees in a graph. Ann. Oper. Res. **50**, 143–171 (1994)
8. Chvátal, V.: A greedy heuristic for the set-covering problem. Math. Oper. Res. **4**(3), 233–235 (1979)
9. Coohoon, J.P., Heck, P.L.: BEAVER: a computational-geometry-based tool for switchbox routing. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **7**, 684–697 (1988)
10. Feige, U.: A threshold of $\ln n$ for approximating set-cover. In: Proceedings of the 28th ACM Symposium on Theory of Computing, pp. 314–318 (1996)
11. Frank, A.: Connections in Combinatorial Optimization. Oxford University Press, Oxford (2011)
12. Fulkerson, D.R.: Blocking and anti-blocking pairs of polyhedra. Math. Program. **1**, 168–194 (1971)
13. Grötschel, M., Jünger, M., Reinelt, G.: Via minimization with pin preassignments and layer preference. Z. Angew. Math. Mech. **69**(11), 393–399 (1989)
14. Grötschel, M., Martin, A., Weismantel, R.: Optimum path packing on wheels: the consecutive case. Comput. Math. Appl. **31**, 23–35 (1996)
15. Grötschel, M., Martin, A., Weismantel, R.: Packing Steiner trees: a cutting plane algorithm and computational results. Math. Program. **72**, 125–145 (1996)
16. Grötschel, M., Martin, A., Weismantel, R.: Packing Steiner trees: further facets. Eur. J. Comb. **17**, 39–52 (1996)
17. Grötschel, M., Martin, A., Weismantel, R.: Packing Steiner trees: polyhedral investigations. Math. Program. **72**, 101–123 (1996)
18. Grötschel, M., Martin, A., Weismantel, R.: Packing Steiner trees: separation algorithms. SIAM J. Discrete Math. **9**, 233–257 (1996)
19. Grötschel, M., Martin, A., Weismantel, R.: The Steiner tree packing problem in VLSI design. Math. Program. **78**(2), 265–281 (1997)
20. Held, S., Korte, B., Rautenbach, D., Vygen, J.: Combinatorial optimization in VLSI design. In: Chvátal, V. (ed.) Combinatorial Optimization—Methods and Applications. NATO Science for Peace and Security Series—D: Information and Communication Security, vol. 31, pp. 33–96 (2011)
21. Hoàng, N.D., Koch, T.: Steiner tree packing revisited. Math. Methods Oper. Res. **76**(1), 95–123 (2012)
22. Jørgensen, D.G., Meyling, M.: Application of column generation techniques in VLSI design. Master's thesis, Department of Computer Science, University of Copenhagen (2000)
23. Karbstein, M.: Line planning and connectivity. Ph.D. thesis, TU Berlin (2013)
24. Koch, T.:. ZIMPL. zimpl.zib.de
25. Koch, T.: Rapid mathematical programming. Ph.D. thesis, Technische Universität Berlin (2004)
26. Korte, B., Prömel, H.J., Steger, A.: Steiner trees in VLSI-layout. In: Korte, B., Lovász, L., Prömel, H.J., Schrijver, A. (eds.) Paths, Flows, and VLSI-Layout. Springer, Berlin (1990)
27. Lengauer, T.: Combinatorial Algorithms for Integrated Circuit Layout. Wiley, New York (1990)
28. Lipski, W.: On the structure of three-layer wireable layouts. In: Preparata, F.P. (ed.) Advances in Computing Research: VLSI Theory, vol. 2, pp. 231–244. Jai Press, London (1984)
29. Luk, W.K.: A greedy switch-box router. Integration **3**, 129–149 (1985)
30. Martin, A.: Packen von Steinerbäumen: Polyedrische Studien und Anwendungen. Ph.D. thesis, Technische Universität Berlin (1992)
31. Oxley, J.G.: Matroid Theory. Oxford University Press, Oxford (1992)
32. Polzin, T.: Algorithms for the Steiner problem in networks. Ph.D. thesis, Universität des Saarlandes (2003)

33. Prömel, H., Steger, A.: The Steiner Tree Problem. Vieweg, Wiesbaden (2002)
34. Raack, C., Koster, A.M.C.A., Orlowski, S., Wessäly, R.: On cut-based inequalities for capacitated network design polyhedra. Networks **57**(2), 141–156 (2011)
35. Raghavan, S., Magnanti, T.: Network connectivity. In: Dell'Amico, M., Maffioli, F., Martello, S. (eds.) Annotated Bibliographies in Combinatorial Optimization, pp. 335–354. Wiley, Chichester (1997)
36. Robacker, J.T.: Min-Max theorems on shortest chains and disjunct cuts of a network. Research Memorandum RM-1660, The RAND Corporation, Santa Monica, CA (1956)
37. Wolsey, L.A.: An analysis of the greedy algorithm for the submodular set covering problem. Combinatorica **2**(4), 385–393 (1982)
38. Wong, R.T.: A dual ascent approach for Steiner tree problems on a directed graph. Math. Program. **28**, 271–287 (1984)

The Maximum Weight Connected Subgraph Problem

Eduardo Álvarez-Miranda, Ivana Ljubić, and Petra Mutzel

Abstract The *Maximum (Node-) Weight Connected Subgraph Problem* (MWCS) searches for a connected subgraph with maximum total weight in a node-weighted (di)graph. In this work we introduce a new integer linear programming formulation built on node variables only, which uses new constraints based on node-separators. We theoretically compare its strength to previously used MIP models in the literature and study the connected subgraph polytope associated with our new formulation. In our computational study we compare branch-and-cut implementations of the new model with two models recently proposed in the literature: one of them using the transformation into the Prize-Collecting Steiner Tree problem, and the other one working on the space of node variables only. The obtained results indicate that the new formulation outperforms the previous ones in terms of the running time and in terms of the stability with respect to variations of node weights.

1 Introduction

The *Maximum (Node-) Weight Connected Subgraph Problem* (MWCS) is the problem of finding a connected subgraph with maximum total weight in a node-weighted (di)graph. It belongs to the class of network design problems and has applications

E. Álvarez-Miranda
Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione, Università di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
e-mail: e.alvarez@unibo.it

I. Ljubić (✉)
Institut für Statistik und Operations Research, Universität Wien, Brünnerstraße 72, 1210 Vienna, Austria
e-mail: ivana.ljubic@univie.ac.at

P. Mutzel
Fakultät für Informatik, Technische Universität Dortmund, Otto-Hahn-Straße 14, 44227 Dortmund, Germany
e-mail: petra.mutzel@tu-dortmund.de

in various different areas such as forestry, wildlife preservation planning, systems biology, computer vision, and communication network design.

Lee and Dooly [23] introduced a cardinality-constrained version of the problem for building a designed fiber-optic communication network over time, where the given node weights reflect their degree of importance. They defined the *maximum-weight connected graph problem* for an undirected graph with given node weights, in which they search the connected subgraph of maximum weight consisting of exactly a predescribed number of nodes. The same problem version was considered already in [18] (the authors called it *Connected k-Subgraph Problem*) for a Norwegian off-shore oil-drilling application.

Another application arises in the area of systems biology [1, 8, 26]. Yamamoto et al. [26] suggest the cardinality-constrained MWCS in order to detect core source components in gene networks, which seem to be responsible for the difference between normal cells and mutant cells. The input graphs are constructed from gene regulation networks combined with gene expression data provided as node weights. Maximum weight connected subgraphs are considered to be good candidates for these core source components. A directed version of the MWCS has been considered in Backes et al. [1], where the most deregulated connected subnetwork in regulatory pathways with the highest sum of node scores (arising from expression data) is searched. In their model, they call a subgraph connected if all the nodes are reachable from one node, also called the *root* in the subgraph. The detected roots are likely to be the molecular *key-players* of the observed deregulation.

A budgeted version arises in conservation planning, where the task is to select land parcels for conservation to ensure species viability, also called *corridor design* (see, e.g. [7]). Here, the nodes of the graph do not only have node weights associated with the habitat suitability but also some costs, and the task is to design wildlife corridors that maximize the suitability with a given limited budget. Also in forest planning, the MWCS arises as a subproblem, e.g., for designing a contiguous site for a natural reserve or for preserving large contiguous patches of mature forest [3].

A surprising application of the MWCS arises in activity detection in video sequences. Here, a 3D graph is constructed from a video in which the nodes correspond to local video subregions and the edges to their proximity in time and space. The node weights correspond to the degree of activity of interest, and so the maximum weight connected subgraph corresponds to the portion of the video that maximizes a classifier's score [4].

All the above mentioned applications have in common that the MWCS arises with node weights only. In many papers, the MWCS has been solved by transforming the given instance to the *Prize-Collecting Steiner Tree Problem*. Here, the given graph has non-negative node weights and negative edge costs, and the task is to find a maximum weight subtree, where the weight is computed as the sum of the node and edge weights in the subtree. The Prize-Collecting Steiner Tree Problem has been studied intensively in the literature (see, e.g., [20, 24]), and the publicly available branch-and-cut (B&C) code of [24] is used in many recent applications to solve the underlying problems to optimality.

However, in their recent work, Backes et al. [1] attack the MWCS directly, which has the advantage to avoid variables for the arcs. The authors suggest a new integer linear programming formulation based on node variables only. The intention of our research was to study the MWCS straightly, and to suggest tight MIP formulations that improve the MIP models from the literature in theory and practice.

Our Contribution We propose a new MIP model for the MWCS based on the concept of node-separators in digraphs. We provide a theoretical and computational comparison of the new model with other models recently used in the literature. We show that the new model has the advantage of using only node variables while preserving the tight LP bounds of the Prize-Collecting Steiner Tree (PCStT) model. Furthermore, we study the connected subgraph polytope and show under which conditions the newly introduced inequalities are facet defining. In an extensive computational study, we compare different MIP models on a set of benchmark instances used in systems biology and on an additional set of network design instances. The obtained results indicate that the new formulation outperforms the previous ones in terms of the running time and in terms of the stability with respect to variations of node weights.

Relation to Martin Grötschel's Work Martin Grötschel is well-known for his research on network design problems, which has influenced a large body of work in the field in the past three and a half decades. Already early in his career in 1977, he has provided a complete and non-redundant linear description of the spanning arborescence problem [14]. In his seminal papers [15–17], jointly with Clyde Monma and Mechthild Stoer, he has studied polyhedra arising from network design problems with low- and with high-connectivity constraints. In his work, also Steiner cuts are used, which also appear in a directed version in our PCStT MIP model. Many recent papers on designing telecommunication networks, UMTS radio networks, broadband virtual private networks, and cellular phone networks were motivated by the results obtained by Martin Grötschel and his colleagues.

The paper is organized as follows. Section 2 contains a formal definition of the MWCS and some complexity results. The following sections provide four different MIP formulations and polyhedral studies. Our B&C algorithm and the practical experiments are discussed in Sect. 5.

2 The Maximum Weight Connected Subgraph Problem

In this section we formally introduce the MWCS for directed graphs and discuss some complexity results.

Definition 1 (The Maximum Weight Connected Subgraph Problem, MWCS) Given a digraph $G = (V, A)$, $|V| = n$, with node weights $p : V \to \mathbb{Q}$, the MWCS is the problem of finding a connected subgraph $T = (V_T, A_T)$ of G, that maximizes the score $p(T) = \sum_{v \in V_T} p_v$ and such that there exists a node $i \in V_T$ (called *root* or *key player*) such that every other node $j \in V_T$ can be reached from i by a directed path in T.

The MWCS in undirected graphs is to find a connected subgraph T that maximizes the score $p(T)$. However, if $G = (V, E)$ is an undirected graph, without loss of generality we will consider its bidirected counterpart (V, A) where A is obtained by replacing each edge by two oppositely directed arcs. Hence, it is sufficient to present results that hold for digraphs (which are more general), and the corresponding results for undirected graphs can be easily derived from them. We assume that in our MWCS instances always positive and negative node weights are present, otherwise, the solution would be trivial. Observe that any feasible solution of the MWCS contains a tree with the same solution value. Hence it is equivalent to search a maximum node-weighted tree in the given graph.

Furthermore, it can be distinguished between the *rooted* and *unrooted* MWCS, i.e., a root node r can be pre-specified or not. In this paper we will concentrate on the unrooted MWCS, or simply the MWCS in the rest of the paper.

Regarding the complexity of the MWCS, it has been shown that the problem is NP-hard (in the supplementary documentation of the paper by [19], the authors provide an NP-hardness proof sketched by R. Karp). Since it is possible to translate the problem to the Prize-Collecting Steiner tree problem, all its polynomially solvable cases carry over to the MWCS. E.g., the PCStT is solvable in polynomial time for the graph class of bounded treewidth [2].

Furthermore, one can show that the following result holds even when the MWCS is defined on undirected graphs:

Proposition 1 *It is NP-hard to approximate the optimum of the MWCS within any constant factor* $0 < \varepsilon < 1$.

Proof For a given MWCS instance, let APP be the objective function value of an approximate solution, and let OPT be the optimal solution value. Recall that for a given constant $0 < \varepsilon < 1$, a given problem can be approximated within factor ε if and only if $APP/OPT \geq \varepsilon$, for any problem instance. To prove this result for the MWCS it is sufficient to make a reduction from the SAT problem that works similarly to the one given in [9] (cf. Theorem 4.1). By doing so, we can show that for a given formula ϕ for SAT, we can build an instance G of the MWCS in polynomial time, such that: (i) if ϕ is a yes-instance, then the optimal MWCS solution on G has value $\varepsilon(1 + \varepsilon^3)$, and (ii) if ϕ is a no-instance, then the optimal MWCS solution on G has value ε^2. □

Some applications consider the *cardinality-constrained MWCS*, where the task is to find a connected subgraph with K nodes. Hochbaum and Pathria [18] have shown that this problem version is NP-hard even if all node weights are 0 or 1 and the graph is either bipartite or planar. For trees and for complete layered DAGs, it is solvable in polynomial time via dynamic programming [18, 22]. Observe that for this problem version, the node weights can be assumed to be all positive, and the maximization variant and the minimization variant are equivalent. Goldschmidt [13] noted that no approximation algorithm is known with a factor better than $O(K)$, and such an algorithm is almost trivial to find. The cardinality-constrained MWCS (and also the

MWCS) can be solved by translating it into the edge-weighted version, which has been studied as the *k-Minimum Spanning Tree Problem* (k-MST) or *k-Cardinality Tree Problem* in the literature (see, e.g., [6, 10]).

3 MIP Formulations for the MWCS

In this section we revise three MIP models for the MWCS recently presented in the literature, and propose a novel approach based on the concept of node-separators in digraphs.

The MIP formulations considered in this paper are based on the observation that if there is a path between i and any other node in $T = (V_T, A_T)$, then we will search for a subgraph which is an arborescence rooted at $i \in V_T$. In our models, two types of binary variables will be used to describe a feasible MWCS solution $T = (V_T, A_T)$: binary variables y_i associated to nodes $i \in V$ will be set to one iff $i \in V_T$, and additional binary variables x_i will be set to one iff the node $i \in V$ is the key player, i.e., if it is used as the root of the arborescence.

Notation and Preliminaries A set of vertices $S \subset V$ ($S \neq \emptyset$) and its complement $\overline{S} = V \setminus R$ induce two directed cuts: $(S, \overline{S}) = \delta^+(S) = \{(i, j) \in A \mid i \in S, j \in \overline{S}\}$ and $(\overline{S}, S) = \delta^-(S) = \{(i, j) \in A \mid i \in \overline{S}, j \in S\}$. When there is an ambiguity regarding the graph in which the directed cut is considered, we will sometimes write δ_G instead of only δ to specify that the cut is considered with respect to graph G. For a set $C \subset V$, let $D^-(C)$ denote the set of nodes outside of C that have ingoing arcs into C, i.e., $D^-(C) = \{i \in V \setminus C \mid \exists(i, v) \in A, v \in C\}$.

A digraph G is called strongly connected (or simply, *strong*) if for any two distinct nodes k and ℓ from V, there exists a (k, ℓ) path in G. A node i is a cut point in a strong digraph G if there exists a pair of distinct nodes k and ℓ from V such that there is no (k, ℓ) path in $G - i$.

For two distinct nodes k and ℓ from V, a subset of nodes $N \subseteq V \setminus \{k, \ell\}$ is called (k, ℓ) *node-separator* if and only if after eliminating N from V there is no (k, ℓ) path in G. A separator N is *minimal* if $N \setminus \{i\}$ is not a (k, ℓ) separator, for any $i \in N$. Let $\mathcal{N}(k, \ell)$ denote the family of all (k, ℓ) separators. Obviously, if $\exists(k, \ell) \in A$ or if ℓ is not reachable from k, we have $\mathcal{N}(k, \ell) = \emptyset$. Let $\mathcal{N}_\ell = \bigcup_{k \neq \ell} \mathcal{N}(k, \ell)$ be the family of all node separators with respect to $\ell \in V$ that we will refer to as ℓ-separators.

For binary variables $\mathbf{a} \in \{0, 1\}^{|F|}$, we denote by $a(F')$ the sum $\sum_{i \in F'} a_i$ for any subset $F' \subseteq F$.

3.1 The Prize-Collecting Steiner Tree Model

In [8] the authors observed that the MWCS on undirected graphs is equivalent to the Prize-Collecting Steiner Tree Problem (PCStT), in the sense that there exists a transformation from the MWCS into the PCStT such that each optimal solution of the PCStT on the transformed graph corresponds to an optimal MWCS solution from the original graph. Recall that, given an undirected graph $H = (V_H, E_H)$ with non-negative node weights $\tilde{p}_v$ and non-negative edge costs $\tilde{c}_e$, the PCStT is the problem of finding a subtree T_H of H that maximizes the function $\sum_{v \in T_H} \tilde{p}_v - \sum_{e \in T_H} \tilde{c}_e$, i.e., the difference between the collected node prizes and edge costs. The transformation from the MWCS into the PCStT is given as follows: Given an input graph G of the MWCS we set $H := G$ and $w = \min_{v \in V} p_v$ (note, that $w < 0$). In order to get non-negative node weights, we set $\tilde{p}_v := p_v - w$ for all $v \in V$ and $\tilde{c}_e = -w$, for all $e \in E$. This transformation also works for digraphs, i.e., if H is a digraph, the PCStT consists of finding a subarborescence of H (rooted at some node $i \in V$) that maximizes the given objective function. The transformation is correct, since any feasible solution is an arborescence, which has indegree 1 for every node, and the weight transformations neutralize each other.

We now present the MIP model proposed in [24] for the PCStT that is used for solving the MWCS after transforming it into the PCStT (see [8]). Consider a transformation from a (directed or undirected) PCStT instance into a rooted digraph $G_d = (V_d, A_d)$ that works as follows: If the input graph $G = (V, E)$ is undirected, then we create the arc set A by bidirecting each edge. In any case we now have a directed graph $G = (V, A)$. The vertex set $V_d = V \cup \{r\}$ contains the nodes of the input graph G *and an artificial root* vertex r. We add new arcs from the root r to nodes v whose out-degree is non-empty in order to get the arc set A_d i.e., $A_d = A \cup \{(r, v) \mid v \in V \text{ and } \delta^+(v) \neq \emptyset\}$. All arc weights are set to the weights of their undirected counterparts, and the weight of an arc $(r, v) \in A_d$ is set to w.

In the graph G_d, a subgraph $T_d = (V_{T_d}, A_{T_d})$ that forms a directed tree rooted at r is called a *rooted Steiner arborescence*. It is a feasible solution of the PCStT if the out-degree of the root is equal to one. To model feasible Steiner arborescences in G_d, we will use two types of binary variables: (a) binary variables y_i introduced above associated to all nodes $i \in V$, and (b) binary variables z_{ij}, such that $z_{ij} = 1$ if arc (i, j) belongs to a feasible Steiner arborescence T_d and $z_{ij} = 0$ otherwise, for all $(i, j) \in A_d$.

The set of constraints that characterizes the set of feasible solutions of the unrooted PCStT is given by:

$$z\big(\delta^-(i)\big) = y_i, \quad \text{for all } i \in V \setminus \{r\} \tag{1}$$

$$z\big(\delta^-(S)\big) \geq y_k, \quad \text{for all } S \subseteq V \setminus \{r\}, k \in S \tag{2}$$

$$z\big(\delta^+(r)\big) = 1 \tag{3}$$

The *in-degree* constraints (1) guarantee that the in-degree of each vertex of the tree is equal to one. The directed cut constraints (2) ensure that there is a directed path from the root r to each costumer k such that $y_k = 1$. The equality (3) makes sure that the artificial root is connected to exactly one of the nodes. Thus, the MWCS can be formulated using the following model that we will denote by (*PCStT*):

$$\max\left\{\sum_{v\in V}(p_v - w)y_v + \sum_{(i,j)\in A_d} wz_{ij} \mid (\mathbf{y},\mathbf{z}) \text{ satisfies (1)–(3)}, (\mathbf{y},\mathbf{z}) \in \{0,1\}^{n+|A_d|}\right\}.$$

The (*PCStT*) model uses node and arc variables (**y** and **z**) given that it relies on an equivalence with the PCStT. However, considering Definition 1 it seems more natural to find a formulation based only in the space of **y** variables since no arc costs are involved. In the next section we will discuss several models that enable elimination of arc variables in the MIP models.

3.2 *Model of Backes et al. 2011*

Recently, in [1] a new MIP model for the MWCS is introduced which avoids the explicit use of arc variables. Let $\mathcal{C}$ denote the family of all directed cycles in G. The new model, that we will denote by (*CYCLE*), reads as follows:

$$x(V) = 1 \tag{4}$$

$$x_i \le y_i, \quad \text{for all } i \in V \tag{5}$$

$$y\big(D^-(i)\big) \ge y_i - x_i, \quad \text{for all } i \in V \tag{6}$$

$$y(C) - x(C) - y\big(D^-(C)\big) \le |C| - 1, \quad \text{for all } C \in \mathcal{C} \tag{7}$$

$$(\mathbf{x},\mathbf{y}) \in \{0,1\}^{2n} \tag{8}$$

Inequalities (4) make sure that one node is selected as a root, and inequalities (5) state that if the node is chosen as a root, it has to belong to the solution. Constraints (6) are the *in-degree constraints*—they ensure that for each node which is not the root, at least one of the incoming neighbors needs to be taken into the solution. In a directed acyclic graph, in-degree constraints are sufficient to guarantee connectivity, but in general, imposing only the in-degree constraints may allow solutions that consist of several disconnected components. To avoid this, cycle constraints (7) are added to guarantee connectivity. These constraints make sure that whenever all nodes from a cycle are taken in a solution, and none of them is set as the root, at least one of the neighboring nodes from $D^-(C)$ has to be taken as well.

Observation 1 *Constraints* (7) *are redundant for those* $C \in \mathcal{C}$ *such that* $C \cup D^-(C) = V$.

To see this, observe that using the root constraint (4), the cycle constraints (7) can be rewritten as follows:

$$y(C) \le y\big(D^-(C)\big) + |C| - 1 + x(C) = y\big(D^-(C)\big) + |C| - x\big(D^-(C)\big),$$

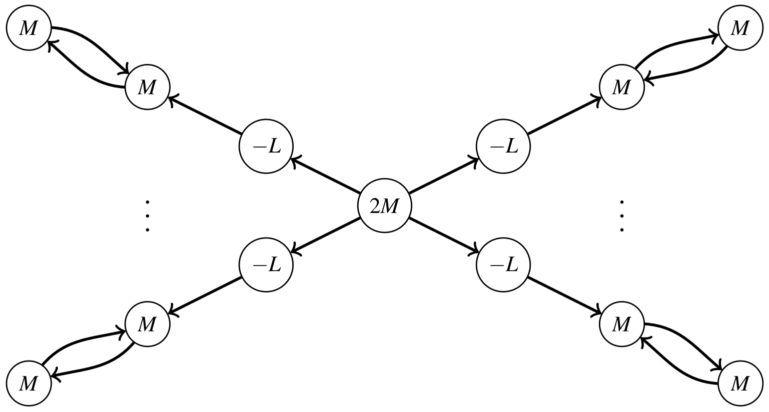


Fig. 1 An example showing that the LP bounds of the (*CYCLE*) model can be as bad as $O(n)$. *The labels of nodes* represent their weights: $M > 0$ and $L \gg M$

which is always satisfied by the model due to constraints (5) and $y_i \leq 1$, for all $i \in V$.

In this model an artificial root node r is not explicitly introduced. However, it is not difficult to see that for any feasible MWCS solution there is a one-to-one mapping between variables z_{ri} introduced above and the variables x_i, for all $i \in V$.

The following result shows that the (*CYCLE*) model provides very weak upper bounds, in general.

Lemma 1 *Given an instance of the MWCS, let OPT be the value of the optimal solution, and let UB be the upper bound obtained by solving the LP relaxation of the (CYCLE) model. Then, there exist MWCS instances for which $UB/OPT \in O(n)$.*

Proof Consider an example given in Fig. 1. The variables of the LP relaxation of the (*CYCLE*) model are set as follows: $y_i = x_i = 0$ for the nodes i with negative weights; $y_i = 1/2$ and $x_i = 0$ for the nodes i in the 2-cycles, and $x_i = y_i = 1$ for the node in the center. There are $K_n = (n-1)/3 \in O(n)$ branches in this graph. We have $UB = K_n M + 2M$ and $OPT = 2M$, which concludes the proof. □

3.3 A Model Based on (k, ℓ) Node-Separators

We now present an alternative approach to model the MWCS in the space of $(\mathbf{x}, \mathbf{y})$ variables that relies on the constraints that have been recently used by [11] and [3] to model connectivity in the context of sheet metal design and forest planning, resp. Notice that for an arbitrary pair of distinct nodes (k, ℓ) in G, if ℓ is taken into the solution and k is chosen as root, then either (i) there is a direct arc from k to ℓ, or

(ii) at least one node from any (k, ℓ) separator $N \in \mathcal{N}(k, \ell)$ has to be taken into the solution. The latter fact can be stated using the following inequalities that we will refer to as *node-separator constraints*:

$$y(N) - x(N) \geq y_\ell + x_k - 1, \quad \text{for all } k, \ell \in V, \ell \neq k, N \in \mathcal{N}(k, \ell). \tag{9}$$

If the nodes k and ℓ are connected by an arc, then $\mathcal{N}(k, \ell) = \emptyset$, in which case we need to consider the in-degree inequalities (6) to make sure k is connected to ℓ. Thus, we can formulate the unrooted MWCS as

$$(CUT)_{k,\ell} \quad \max\left\{\sum_{v \in V} p_v y_v \mid (\mathbf{x}, \mathbf{y}) \text{ satisfies (4)–(6), (9) and } (\mathbf{x}, \mathbf{y}) \in \{0, 1\}^{2n}\right\}.$$

Inequalities (9) can be separated in polynomial time in a support graph that splits nodes into arcs. Given a fractional solution $(\tilde{x}, \tilde{y})$, for each pair of nodes (k, ℓ) such that $\tilde{y}_\ell + \tilde{x}_k - 1 > 0$ we generate a graph $G_{k\ell}$ in which all nodes $i \neq k, \ell$ are replaced by arcs. Arc capacities are then set to 1, except for the arcs associated to nodes, whose capacities are set to $\tilde{y}_i - \tilde{x}_i$. If the maximum flow that can be sent from k to ℓ in $G_{k\ell}$ is less than $\tilde{y}_\ell + \tilde{x}_k - 1 > 0$, we have detected a violated inequality of type (9).

Using the root constraint (4), inequalities (9) can also be reformulated as follows:

$$y(N) \geq y_\ell + x\big(N \cup \{k\}\big) - 1 \quad \Rightarrow \quad y(N) + x\big(V \setminus \big(N \cup \{k, l\}\big)\big) \geq y_\ell - x_\ell,$$

which can be interpreted as follows: If node ℓ is in the solution and it is not the root, then for each $k \in V$ such that $\mathcal{N}(k, \ell) \neq \emptyset$ and each $N \in \mathcal{N}(k, \ell)$, either one of the nodes from N is part of the solution, or none of the nodes from $N \cup \{k\}$ is chosen as the root node.

Inequalities (9) are quite intuitive, however they are not facet defining. In the next section we will show how the (k, ℓ) node-separator constraints can be lifted to obtain facet defining inequalities.

3.4 A Model Based on Generalized Node-Separator Inequalities

Observe that the inequality (9) can be lifted as follows: Assume that $N \in \mathcal{N}(k, \ell)$ also separates another node $k' \neq k$ from ℓ. Since at most one node can be set as a root, the right-hand side of (9) can be increased as follows: $y(N) - x(N) \geq y_\ell + x_k + x_{k'} - 1$. In fact, this motivates us to introduce a generalized family of node-separator inequalities, that can be obtained by a parallel lifting of (9).

Generalized Node-Separator Inequalities Let ℓ be an arbitrary node in V and let $N \in \mathcal{N}_\ell$ be an arbitrary ℓ-separator. Let $W_{N,\ell}$ be the set of nodes i such that there is a directed (i, ℓ)-path in $G - N$. More formally:

$$W_{N,\ell} = \big\{i \in V \setminus N \mid \exists (i, \ell) \text{ path } P \text{ in } G - N\big\} \cup \{\ell\}.$$

Then, for any feasible MWCS solution, the following has to be satisfied: if node ℓ is part of a solution, then either the root of the solution is in $W_{N,\ell}$, or, otherwise, at least one of the nodes from N has to be taken. Hence, the following inequalities, that we will refer to as *generalized node-separator inequalities*, are valid for the MWCS:

$$y(N) + x(W_{N,\ell}) \geq y_\ell, \quad \text{for all } \ell \in V, N \in \mathcal{N}_\ell \tag{gNSep}$$

Notice that the in-degree inequalities (6) are a subfamily of (gNSep): The in-degree inequality can be rewritten as $\sum_{j \in D^-(\ell)} y_j + x_\ell \geq y_\ell$, i.e., they are a special case of the generalized node-separator cuts for $N = D^-(\ell)$ in which case $W_{N,\ell} = \{\ell\}$. In order to see that (gNSep) are lifted inequalities (9), notice that (gNSep) can be rewritten as follows:

$$y(N) - x(N) \geq y_\ell + x\big(V \setminus (N \cup W_{N,\ell})\big) - 1, \quad \text{for all } \ell \in V, N \in \mathcal{N}_\ell.$$

Together with this observation this proves that the following model is a valid MIP formulation for the MWCS:

$$(CUT) \quad \max\left\{\sum_{v \in V} p_v y_v \mid (\mathbf{x}, \mathbf{y}) \text{ satisfies (4)–(5), (gNSep) and } (\mathbf{x}, \mathbf{y}) \in \{0,1\}^{2n}\right\}.$$

Proposition 2 *Generalized node-separator inequalities can be separated in polynomial time.*

Proof Consider an auxiliary support graph in which the nodes are splitted as follows: each node $i \in V$ is replaced by an arc (i_1, i_2). All ingoing arcs into i are now connected to i_1, all outgoing arcs from node i are now connected to i_2. In other words, we create a graph $G' = (V', A')$ such that $V' = \{i_1 \mid i \in V\} \cup \{i_2 \mid i \in V\} \cup \{r\}$ (r is an artificial root), $A' = \{(i_2, j_1) \mid (i,j) \in A\} \cup \{(i_1, i_2) \mid i \in V\} \cup \{(r, i_1) \mid i \in V\}$. For a given fractional solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ arc capacities in G' are defined as:

$$cap_{uv} = \begin{cases} \tilde{y}_i, & \text{if } u = i_1, v = i_2, i \in V, \\ \tilde{x}_i, & \text{if } u = r, v = i_1, i \in V, \\ 1, & \text{otherwise.} \end{cases} \tag{10}$$

We calculate the maximum flow on G' between r and (ℓ_1, ℓ_2) in G' for a node ℓ such that $\tilde{y}_\ell > 0$. To check whether there are violated inequalities of type (gNSep), it only remains to show that (i) every minimum cut $(\overline{S}, S)$ in G' such that the corresponding flow is less than $\tilde{y}_\ell$ corresponds to a (gNSep) inequality for the given $\ell \in V$ and some $N \in \mathcal{N}_\ell$, or (ii) that a corresponding violated (gNSep) cut can be generated from $(\overline{S}, S)$ in polynomial time. Observe that any minimum cut $(\overline{S}, S)$ in G' which is smaller than $\tilde{y}_\ell$ can be represented as union of arcs adjacent to the root, plus union of arcs of type (i_1, i_2). Hence, each $(\overline{S}, S)$ cut implies the following inequalities:

$$\sum_{(r,j) \in \delta^-(S)} x_j + \sum_{(i_1,i_2) \in \delta^-(S)} y_i \geq y_\ell. \tag{11}$$

We can now define a partitioning (U, N, W) of the node set V such that:

$$W = \{i \in V \mid i_1, i_2 \in S\}, \qquad N = \{i \in V \mid i_1 \notin S, i_2 \in S\}, \qquad U = V \setminus (W \cup N).$$

Rewriting the inequality (11), we obtain: $x(W) + y(N) \geq y_\ell$. Observe that $U \neq \emptyset$. Indeed, if $U = \emptyset$ then $N \cup W = V$, but then we have $x(N)+y(W) \geq x(V) = 1 \geq \tilde{y}_\ell$, i.e., such cuts will never be violated. Hence, given the proper partition (U, N, W), the set N is obviously a (k, ℓ) separator for any $k \in U$ (after removing (r, i_1) arcs from G', the arcs $(i_1, i_2) \in \delta^-(S)$ are arc-separators that separate U from the rest of the graph). If W contains only nodes that can reach ℓ in $G - N$, then inequality (11) belongs to the (gNSep) family. Otherwise we reverse all arcs in $G - N$ and perform a breadth-first search from ℓ. All nodes that can be reached from ℓ (notice that they cannot belong to U), by definition, determine the set $W_{N,\ell}$. If the original cut (11) was violated, the new one with the left-hand side equal to $y(N) + x(W_{N,\ell})$ will be violated as well. □

3.5 *Some More Useful Constraints*

In this section we present additional constraints that are useful for practically solving MWCS instances.

Connected Component Inequalities In some applications of the MWCS, a K-cardinality constraint is imposed: $\sum_{i \in V} y_i = K$. For a given node $k \in V$, let P_k contain all the nodes that are further than $K - 1$ hops away from k. In that case, the following inequalities are valid for the MWCS:

$$x_k + y_\ell \leq 1, \quad \text{for all } \ell \in P_k. \tag{12}$$

Rewriting the connected component cuts, we obtain:

$$\sum_{j \neq k} x_j \geq y_\ell, \quad \text{for all } \ell \in P_k,$$

these constraints can be further strengthened by down lifting the coefficients of the left-hand side. Whenever node ℓ is in the solution, then either ℓ is the root, or the root cannot be more than $K - 1$ hops away from ℓ. Let W_ℓ be the set of such potential root nodes including ℓ. We have

$$x(W_\ell) \geq y_\ell, \quad \text{for all } \ell \in V.$$

Out-degree Inequalities The following set of inequalities state that whenever a node i such that $p_i \leq 0$ is taken into a solution, this is because it leads us to another node with positive weights:

$$y\big(D^+(i)\big) \geq y_i, \quad \text{for all } i \in V \text{ s.t. } p_i \leq 0. \tag{13}$$

Observe that these constraints are not valid if K-cardinality constraints are imposed.

Symmetry-Breaking Inequalities In case the input graph is undirected, there exist many equivalent optimal solutions with different orientations. In order to break those symmetries, we can impose the following constraint that chooses the node with the smallest index to be the root of the subgraph:

$$x_j + y_i \le 1, \quad \text{for all } i < j. \tag{14}$$

4 Polyhedral Study

Let $\mathcal{P}$ denote the connected subgraph (CS) polytope in the space of $(\mathbf{x}, \mathbf{y})$ variables:

$$\mathcal{P} = \text{conv}\big\{(\mathbf{x}, \mathbf{y}) \in \{0, 1\}^{2n} \mid (\mathbf{x}, \mathbf{y}) \text{ satisfies (4), (5), (gNSep)}\big\}.$$

In this section we compare the proposed MIP formulations with respect to their quality of LP bounds and we show that, under certain conditions, the newly introduced generalized node-separator inequalities are facet defining for the CS polytope.

4.1 Theoretical Comparison of MIP Models

Let $\mathcal{P}_{\text{LP}}(\cdot)$ denote the polytope of the LP relaxations of the MIP models presented above obtained by replacing integrality conditions by $0 \le x_i, y_i \le 1$, for all $i \in V$, and let $v_{\text{LP}}(\cdot)$ be the optimal LP values of the associated MIP relaxations. For the $\mathcal{P}_{\text{LP}}(PCStT)$ polytope, we set $Proj_{(x,y)}(\mathcal{P}_{\text{LP}}(PCStT)) = \{(\mathbf{x}, \mathbf{y}) \in \{0, 1\}^{2n} \mid x_i = z_{ri}$ and $(y, z) \in \mathcal{P}_{\text{LP}}(PCStT)\}$. We can show that:

Proposition 3

1. *We have*

$$Proj_{(x,y)}\big(\mathcal{P}_{\text{LP}}(PCStT)\big) = \mathcal{P}_{\text{LP}}(CUT) \subsetneq \mathcal{P}_{\text{LP}}(CUT_{k\ell})$$

and

$$\mathcal{P}_{\text{LP}}(CUT) \subsetneq \mathcal{P}_{\text{LP}}(CYCLE).$$

2. *Moreover, there exist MWCS instances such that*

$$v_{\text{LP}}(CYCLE)/v_{\text{LP}}(CUT) \in O(n).$$

3. *The polytopes $\mathcal{P}_{\text{LP}}(CYCLE)$ and $\mathcal{P}_{\text{LP}}(CUT_{k\ell})$ are not comparable.*

Proof 1. $Proj_{(x,y)}(\mathcal{P}_{\text{LP}}(PCStT)) = \mathcal{P}_{\text{LP}}(CUT)$:

We first show that $Proj_{(x,y)}(\mathcal{P}_{\text{LP}}(PCStT)) \subseteq \mathcal{P}_{\text{LP}}(CUT)$. Let $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ be a feasible solution for the relaxation of the $PCStT$ model, we will show that the solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ such that $\hat{x}_i = \hat{z}_{ri}$ belongs to $\mathcal{P}_{\text{LP}}(CUT)$. Let $\ell \in V$ be an arbitrary node such that

$\hat{y}_\ell > 0$, choose some $N \in \mathcal{N}_\ell$ and consider the associated $W_{N,\ell} \subset V$. Let G_d be the corresponding directed instance of the PCStT with the root r (cf. Sect. 3.1). Consider now a cut $(\overline{W}_d, W_d)$ in G_d where $W_d = N \cup W_{N,\ell}$. We have: $\delta^-_{G_d}(W_d) = \{(r,i) \in A_d \mid i \in W_{N,\ell}\} \cup \mathit{Rest}$, where $\mathit{Rest} = \{(j,i) \in A_d \mid j \in \overline{W}_d, i \in N\}$. Observe that $\mathit{Rest} \subseteq \delta^-_{G_d}(N) \subseteq \bigcup_{i \in N} \delta^-_{G_d}(i)$. Therefore, we have:

$$\hat{\mathbf{y}}(N) = \sum_{i \in N} \hat{\mathbf{z}}\big(\delta^-_{G_d}(i)\big) \geq \hat{\mathbf{z}}\big(\delta^-_{G_d}(N)\big) \geq \hat{\mathbf{z}}(\mathit{Rest}). \tag{15}$$

Since $(\overline{W}_d, W_d)$ is a Steiner cut in G_d, it holds that $\hat{\mathbf{z}}(\delta^-_{G_d}(W_d)) \geq \hat{y}_\ell$. This, together with (15) implies:

$$\hat{\mathbf{y}}(N) + \hat{\mathbf{x}}(W_{N,\ell}) \geq \hat{\mathbf{z}}(\mathit{Rest}) + \hat{\mathbf{x}}(W_{N,\ell}) = \hat{\mathbf{z}}\big(\delta^-_{G_d}(W_d)\big) \geq \hat{y}_\ell.$$

To show that $\mathcal{P}_{\mathrm{LP}}(\mathit{CUT}) \subseteq \mathit{Proj}_y(\mathcal{P}_{\mathrm{LP}}(\mathit{PCStT}))$ consider an LP solution $(\check{\mathbf{y}}, \check{\mathbf{x}}) \in \mathcal{P}_{\mathrm{LP}}(\mathit{CUT})$. We will construct a solution $(\hat{\mathbf{y}}, \hat{\mathbf{z}}) \in \mathcal{P}_{\mathrm{LP}}(\mathit{PCStT})$ such that $\check{\mathbf{y}} = \hat{\mathbf{y}}$ and $\hat{z}_{rj} = \check{x}_j$, for all $j \in V$. On the graph G' (see Proof of Proposition 2) with arc capacities of (i_1, i_2) set to $\check{y}_i$ for each $i \in V$, arc capacities of (r, j_1) set to $\check{x}_j$, and capacities set to 1 for the remaining arcs, we are able to send $\check{y}_\ell$ units of flow from the root r to every $\ell_1 \in V'$ such that $\check{y}_\ell > 0$. Let f^k_{ij} denote the amount of flow of commodity k, associated with $k_1 \in V'$, sent along an arc $(i,j) \in A'$. Let $\mathbf{f}$ be the minimal feasible multi-commodity flow on G' (i.e., the effective capacities on G' used to route the flow cannot be reduced without violating the feasibility of this flow). We now define the values of $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ as follows: $\hat{z}_{rj} = \check{x}_j$, for all $j \in V$ and

$$\hat{z}_{ij} = \begin{cases} \max_{k \in V} f^k_{i_2 j_1}, & i, j \in V \\ \max_{k \in V} f^k_{r j_1}, & i = r, j \in V \end{cases} \quad \text{for all } (i,j) \in A;$$

$$\hat{y}_i = \hat{\mathbf{z}}\big(\delta^-(i)\big), \quad \text{for all } i \in V.$$

Obviously, the constructed solution $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ is feasible for the (PCStT) model and, due to the assumption that $\mathbf{f}$ is minimal feasible, it follows that $\check{\mathbf{y}} = \hat{\mathbf{y}}$ and $\check{\mathbf{x}}$ is equivalent to $\hat{\mathbf{z}}$.

$\mathcal{P}_{\mathrm{LP}}(\mathit{CUT}) \subsetneq \mathcal{P}_{\mathrm{LP}}(\mathit{CYCLE})$:

Let $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ be an arbitrary point from $\mathcal{P}_{\mathrm{LP}}(\mathit{CUT})$. In order to prove that $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \mathcal{P}_{\mathrm{LP}}(\mathit{CYCLE})$ we only need to show that constraints (7) are satisfied (recall that in-degree inequalities (6) are contained in (gNSep)). Given the Observation 1, it is sufficient to consider cycles C such that $C \cup D^-(C) \subset V$. Since for any such cycle C the set $D^-(C)$ defines a separator for any node $\ell \in C$, from constraints (gNSep) we have that $\hat{y}(D^-(C)) + \hat{x}(C) \geq \hat{y}_\ell$. For the remaining nodes $j \in C$, $j \neq k$, we apply the bounds $1 \geq \hat{y}_j$. Summing up together these $|C|$ inequalities, we obtain (7).

2. Consider the example given in Fig. 1 for which the (CUT) model finds the optimal solution.

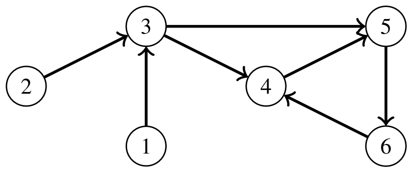


Fig. 2 An example showing that $\mathcal{P}_{\text{LP}}(CUT_{k\ell}) \not\subseteq \mathcal{P}_{\text{LP}}(CYCLE)$. The LP solution $y_4 = y_5 = y_6 = 1$, $y_1 = y_2 = y_3 = x_1 = x_2 = 1/2$ is feasible for the $(CUT_{k\ell})$ model and infeasible for $(CYCLE)$

3. The example given in Fig. 1 shows an instance for which the LP solution is feasible for the $(CYCLE)$ and infeasible for the $(CUT_{k\ell})$ model. The example given in Fig. 2 shows an instance for which the LP solution is feasible for the $(CUT_{k\ell})$ and infeasible for the $(CYCLE)$ model. □

4.2 Facets of the CS Polytope

In this section we establish under which conditions some of the presented inequalities are facet defining for the CS polytope.

Lemma 2 *If G is a strong digraph, then the dimension of the polytope $\mathcal{P}$ is* $\dim(\mathcal{P}) = 2n - 1$.

Proof We will construct the set of $2n$ feasible, affinely independent solutions as follows: Since G is strong, we can find n spanning arborescences by choosing each $i \in V$ as a root. That way, we build n affinely independent solutions. In addition, consider n single node solutions (for each $i \in V$), in which we have $x_i = y_i = 1$ and all remaining $x_j = y_j = 0$, for all $j \neq i$. The matrix obtained by merging the characteristic vectors of these solutions has full rank $2n$. □

Lemma 3 *Trivial inequalities $x_i \geq 0$ are facet defining if G is strong and i is not a cut point in G.*

Proof Consider a family $\mathcal{T}$ of spanning arborescences on the set $V \setminus \{i\}$ in which each $j \neq i$ is taken once as a root. This is possible because $G - i$ remains a strong digraph. There are $n - 1$ such solutions, and they are affinely independent. Add now to $\mathcal{T}$ single node solutions, for each $j \in V \setminus \{i\}$. Finally, add to $\mathcal{T}$ a spanning arborescence in G with a root $j \neq i$. The matrix associated to incidence vectors from $\mathcal{T}$ has full rank $2n - 1$. □

Lemma 4 *Trivial inequalities $y_i \leq 1$ are facet defining if G is strong.*

Proof Consider a spanning arborescence T rooted at i. We will then apply a *pruning technique* in order to generate n affine independent feasible MWCS solutions. We start with T in which case $\mathbf{y}$ consists of all ones. We iteratively remove one by one leaves from T, until we end up with a single root node i. Thereby, we generate a family $\mathcal{T}$ of n affinely independent solutions. We then add to $\mathcal{T}$ $n-1$ solutions obtained by choosing a spanning arborescence rooted at j, for all $j \neq i$. The matrix associated to incidence vectors from $\mathcal{T}$, has full rank $2n-1$. □

Notice that $y_i \geq 0$ are not facet defining inequalities because $y_i = 0$ implies $x_i = 0$. Similarly, $x_i \leq 1$ do not define facets of $\mathcal{P}$ because they are dominated by $x_i \leq y_i$.

Lemma 5 *Coupling inequalities $y_i \geq x_i$ are facet defining if G is strong and i is not a cut point in G.*

Proof Construct a family $\mathcal{T}$ of n affinely independent solutions by applying pruning to a spanning arborescence rooted at i. Add then to $\mathcal{T}$ additional $n-1$ arborescences on the set $V \setminus \{i\}$ in which each $j \neq i$ is taken once as a root (this is possible because $G - i$ remains strong). The matrix associated to incidence vectors from $\mathcal{T}$, has full rank $2n-1$. □

Proposition 4 *Given $\ell \in V$ and $N \in \mathcal{N}_\ell$, the associated* (gNSep) *inequality is facet defining if G is strong, N is a minimal ℓ-separator and the subgraph induced by $W_{N,\ell}$ ($|W_{N,\ell}| \geq 2$) is strong.*

Proof We prove the result by the indirect method. Let $F(\ell, N) = \{(\mathbf{x}, \mathbf{y}) \in \{0,1\}^{2n} \mid y(N) + x(W_{N,\ell}) = y_\ell\}$. Consider a facet defining inequality of the form $\mathbf{ax} + \mathbf{by} \geq a_0$. We will show that if all points in $F(\ell, N)$ satisfy

$$\mathbf{ax} + \mathbf{by} = a_0, \tag{16}$$

then (16) is a positive multiple of (gNSep). Consider $\ell' \in W$, $\ell' \neq \ell$. A path from ℓ to ℓ', completely contained in $W_{N,\ell}$ and rooted at ℓ exists in G ($W_{N,\ell}$ is strong) and it is a feasible MWCS solution that belongs to $F(\ell, N)$. Let $(\mathbf{x}^1, \mathbf{y}^1)$ be the characteristic vector of this path. A subpath obtained after removing ℓ' from this path, also rooted at ℓ, is another feasible solution from $F(\ell, N)$, and let $(\mathbf{x}^2, \mathbf{y}^2)$ be the corresponding characteristic vector. We have: $\mathbf{ax}^1 + \mathbf{by}^1 - \mathbf{ax}^2 - \mathbf{by}^2 = 0$. Therefore we have $b'_\ell = 0$, for all $\ell' \in W$, $\ell' \neq \ell$. Consider now a node $k \in U = V \setminus (N \cup W_{N,\ell})$. To show that $b_k = 0$, for all $k \in U$, we distinguish the following cases:

(1) If $D^-(k) \cap U \neq \emptyset$, then there exists an arc (k', k), $k' \in U$ that builds a feasible MWCS solution B from $F(\ell, N)$. Also, the single node solution $B' = \{k'\}$ belongs to $F(\ell, N)$. After subtracting the equations (16) with the substituted characteristic vectors of B and B', we obtain $b_k = 0$.

(2) If there exists an arc $(i,k) \in A$ for some $i \in N$, then, consider a path P from i to ℓ that does not cross $N \cup U$ (such P exists because N is minimal) and a path $P' = P \cup \{(i,k)\}$, in both of them we set i as root. Both P and P' belong to $F(\ell, N)$. After subtracting the equations (16) with the substituted characteristic vectors of P and P', we obtain $b_k = 0$.
(3) Finally, if there exists an arc $(j,k) \in A$ for some $j \in W_{N,\ell}$, we consider a path Q from ℓ to j in $W_{N,\ell}$ (such path exists because $W_{N,\ell}$ is strong) and a path $Q' = Q \cup \{(j,k)\}$. Both Q and Q' belong to $F(\ell, N)$. After subtracting the equation (16) with the substituted characteristic vectors of Q and Q', we obtain $b_k = 0$. Hence, the equation (16) can be rewritten as $\mathbf{ax} + \sum_{i \in N \cup \{\ell\}} b_i x_i = a_0$. Notice that a single node solution $\{k\}$ belongs to $F(\ell, N)$, for each $k \in U$. By plugging the associated vector into (16), it follows that $a_k = a_0$, for all $k \in U$. Consider now two spanning arborescences in $W_{N,\ell}$, one rooted at ℓ, the other rooted at arbitrary $\ell' \neq \ell$ (this is possible, because $W_{N,\ell}$ is strong). After subtracting the equation (16) with the substituted characteristic vectors of those two arborescences, we obtain $a_{\ell'} = a_\ell = \alpha$, for all $\ell' \in W_{N,\ell}$. Since $N \in \mathcal{N}_\ell$ and it is minimal, for each $i \in N$ there exist $k \in U$ such that there exist a path P_k from k to ℓ that crosses N exactly at the node i. Let P'_k be a subpath of P_k from i to ℓ. Both paths belong to $F(\ell, N)$ and after subtracting the associated equations (16), it follows that $a_i = a_k$, and hence $a_i = a_0$, for all $i \in N$.

So far, (16) can be rewritten as $a_0 x(\overline{W}_{N,\ell}) + \alpha x(W_{N,\ell}) + \sum_{i \in N \cup \{k\}} b_i y_i = a_0$. After plugging in the characteristic vector of P'_k into this equation, it follows that $a_0 + b_i + b_\ell = a_0$, and therefore we have $b_i = -b_\ell = \beta$, for all $i \in N$. Equation (16) becomes now $a_0 x(\overline{W}_{N,\ell}) + \alpha x(W_{N,\ell}) + \beta y(N) - \beta y_\ell = a_0$. Notice that solution $\{\ell\}$ also belongs to $F(\ell, N)$, which implies that $\alpha - \beta = a_0$. Finally, substituting a_0 in the previous equation, and using the equation (4), $x(V) = 1$, we end up with the following form of (16):

$$\beta\left[-x(\overline{W}_{N,\ell}) + y(N) - y_\ell = -1\right],$$

which together with equation (4) concludes the proof. □

5 Computational Results

For testing the computational performance of the presented formulations we have considered both directed and undirected MWCS instances. The (*CYCLE*) model of Backes et al. [1] has been developed for directed graphs (regulatory networks) with K-cardinality constraints, i.e., any feasible solution has to be comprised by exactly K nodes (for a given $K > 1$). Executables of this implementation are available online (see [12]). For the (*PCStT*) and (*CUT*) models we have developed our own B&C implementations that work with and without cardinality constraints. The real-world instances used in [1] require K-cardinality constraints. Therefore, in the part of our computational study conducted on digraphs, we impose cardinality constraints for all three models, (*PCStT*), (*CUT*) and (*CYCLE*). For the other set of instances we take the size of the unconstrained optimal solution (obtained by the (*CUT*) model) and provide the corresponding value of K as input to the (*CYCLE*) model.

In the following, we describe (i) components of the designed B&C algorithms and some implementation details, (ii) a testbed used for the experiments, and (iii) an extensive analysis of the obtained results.

5.1 Branch-and-Cut Algorithms

Separation of Inequalities For the (*PCStT*) model, connectivity inequalities (2) are separated within the B&C framework by means of the maximum flow algorithm given by [5]. The separation problem is solved on a support graph whose arc capacities are given by the current LP value of $\mathbf{z}$ variables. We randomly select a terminal $v \in V$ such that $p_v > 0$ and $y_v > 0$, and calculate the maximum flow between the artificial root and v, and insert the corresponding constraint (2), if violated.

For the (*CUT*) formulation, the separation of (gNSep) is performed by solving the maximum flow problems as described in the proof of Proposition 2, with arc capacities given by (10).

In all cases, instead of adding a single violated cut per iteration, we use *nested*, *back-flow* and *minimum cardinality* cuts (see also [21, 24]) to add as many violated cuts as possible. We restrict the number of inserted cuts within each separation callback to 25.

Primal Heuristic Our primal heuristic finds feasible solutions using the information available from the current LP solution in a given node of the branch-and-bound tree. Although we develop two different B&C algorithms, derived from two MIP models, the embedded primal heuristics are based on the same idea. We select a subset of potential "key-players" (nodes with a positive outgoing degree and with sufficiently large $\mathbf{y}$ values) and run a restricted breadth-first search (BFS) from each of them. Out of the constructed connected components, i.e., feasible solutions of the MWCS, we select the one with the largest total weight.

MIP Initialization We initialize the (*PCStT*) model with the root out-degree constraints (3). For the undirected MWCS, we also add symmetry-breaking constraints (similar to (14)) and inequalities $z_{ji} + z_{ij} \leq y_i$, for all $e : \{i, j\} \in E$ since they avoid too frequent calls of the maximum flow procedure. For the variants where no cardinality constraint is defined, we also include the flow-balance constraints: $z(\delta^-(i)) \leq z(\delta^+(i))$, for all $i \in V$ such that $p_i \leq 0$. These constraints ensure that a node with non-positive weight can not be a leaf in an optimal PCStT solution.

We initialize the (*CUT*) model with the constraints (4), (5), (6). For the cases where no cardinality constraint is imposed, the out-degree constraints (13) are also included. Finally, the symmetry-breaking constraints (14) are added for the undirected case.

Implementation The proposed approaches were implemented using CPLEX™ 12.3 and Concert Technology. All CPLEX parameters were set to their default values, except the following ones: (i) CPLEX cuts were turned off, (ii) CPLEX heuristics were turned off, (iii) CPLEX preprocessing was turned off, (iv) the time limit was set to 1800 seconds (except for the instances from [1]), and (v) higher branching priorities were given to $\mathbf{y}$ variables, in the case of the (*PCStT*) models, and to $\mathbf{x}$ variables, in the case of the (*CUT*) model. All the experiments were performed on a Intel Core2 Quad 2.33 GHz machine with 3.25 GB RAM, where each run was performed on a single processor.

5.2 Benchmark Instances

We have considered two sets of benchmark instances arising from applications in systems biology and from network design.

Systems Biology Instances We have considered instances used in [8] and [1]. In [8], only a single protein-protein interaction network is considered. The instance is presented as an undirected graph comprised by 2034 nodes (proteins) and 8399 edges (interactions). The considered protein-protein interaction network corresponds to a well studied human one and the protein scores come from a lymphoma microarray dataset (LYMPH). The instance is available at [25].

In [1], six instances of regulatory networks, i.e., directed graphs, were considered. These instances have the same underlying network (KEGG human regulatory network of protein complexes), which is a graph comprised by 3917 nodes and 133 310 arcs. The differences between the six benchmark instances of this set are the scores associated to the proteins (or protein complexes) which depend on the pathogenic process under consideration. All the instances are available online (see [12]). For providing a valid comparison with the method proposed in [1], it is necessary to impose cardinality constraints to the solutions. Values $K \in \{10, 11, \dots, 25\}$ are considered. This leads to 16 different instances for each of the six different score settings.

Network Design Instances These are Euclidean random instances which are generated as proposed by Johnson, Minkoff, and Phillips in their paper on the Prize-Collecting Steiner Tree Problem [20]. The topology of these instances is similar to street networks. First, n nodes are randomly located in a unit Euclidean square. A link between two nodes i and j is established if the Euclidean distance d_{ij} between them is no more than $\alpha/\sqrt{n}$, for a fixed $\alpha > 0$. To generate node weights, we performed the following procedure: δ % of the nodes are randomly selected to be associated with non-zero weights. Out of them, ε % are associated with a weight taken uniformly randomly from $[-10, 0]$ and the remaining ones are associated with a weight taken uniformly randomly from $[0, 10]$.

When generating these instances we do not impose whether links are directed or not. When reading the input files we define if the link between i and j corresponds

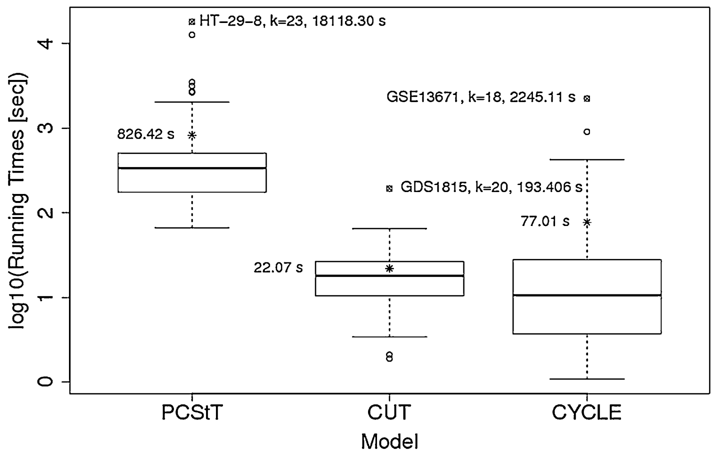


Fig. 3 Box plots of $\log_{10}$-values of the running times [sec] (instances from [1], $K \in \{10, \dots, 25\}$)

to an edge $e : \{i, j\}$ or to an arc $a : (i, j)$. This allows us to use the same set of instances for both, the directed and the undirected case.

For the computational experiments we considered $n \in \{500, 750, 1000, 1500\}$, $\alpha \in \{0.6, 1.0\}$, $\delta \in \{0.25, 0.50, 0.75\}$, $\varepsilon \in \{0.25, 0.50, 0.75\}$. This leads to 18 instances for each fixed value of n.

5.3 Algorithmic Performance

MWCS on Digraphs We consider the instances GSE13671, GDS1815, HT-29-8, HT-29-24, HT-116-8, HT-116-24 from [1] and our randomly generated instances.

In Fig. 3, using the box plots we show the $\log_{10}$-values of the running times for the three approaches considering all instances of [1] and all values of K. There are $16 \times 6 = 96$ problems in total for each approach. The values marked with an asterisk correspond to the $\log_{10}$-values of the mean running time (shown as the label next to the asterisk). The values marked with symbol $\times$ correspond to the $\log_{10}$-values of the maximum running times (the label next to it shows the name of the instance, K, and the running time). The obtained results indicate that, for this group of instances, the approach with the worst performance is (*PCStT*), since most of the running times are at least one order of magnitude larger than the ones of the other two approaches. When comparing (*CUT*) and (*CYCLE*), one can observe that the distribution of the running times of the (*CYCLE*) model has a larger dispersion (the *box* is wider) and its outliers are almost one order of magnitude larger than the maximum running times of the (*CUT*) model. In a few cases however the (*CYCLE*) model solves some instances faster than the (*CUT*) model (which can be seen from the minimum values and the values in the first-quartile). Overall, the mean value of the running times of the (*CUT*) model is 22 seconds, which is almost three times

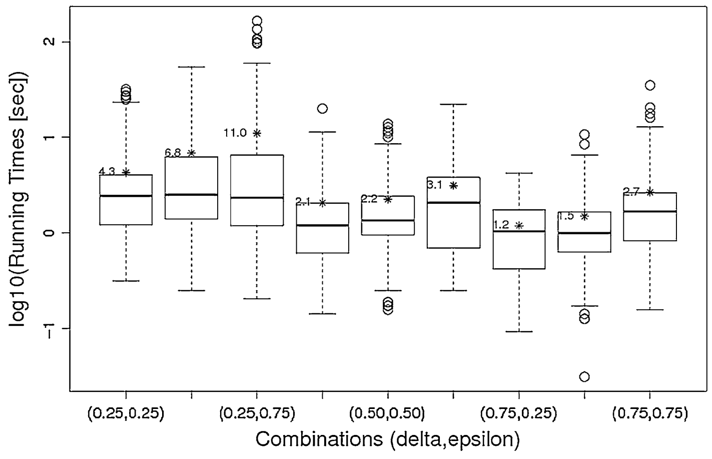
Table 1 Average values for instances from [1] ($K \in \{10, \dots, 25\}$)

Instance	δ	ε	(*PCStT*)		(*CUT*)		(*CYCLE*)	
			Time (sec)	#(2)	Time (sec)	#(gNSep)	Time (sec)	#(7)
GSE13671	0.89	0.73	176.11	1206	17.85	97	341.95	3754
GDS1815	0.92	0.64	878.63	3565	46.09	225	37.95	1264
HT-29-8	0.92	0.66	2846.36	5400	22.03	182	14.17	178
HT-29-24	0.92	0.61	196.56	1292	11.40	61	60.59	1330
HT-116-8	0.92	0.54	623.10	2214	15.26	108	3.21	129
HT-116-24	0.92	0.55	237.78	1149	19.82	93	4.19	130
Average			826.42	2471	22.07	128	77.01	1131

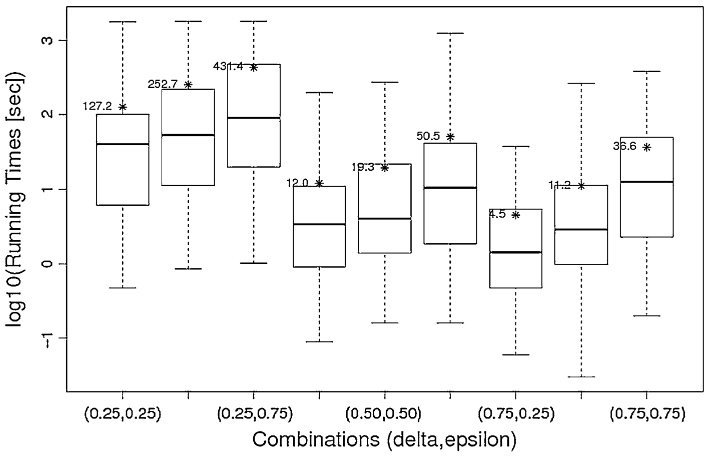
smaller than the mean running time of the (*CYCLE*) model (77 seconds). The value of the maximum running time of the (*CUT*) model is 193 seconds, which is more than 10 times smaller than the maximum running time of the (*CYCLE*) model (2245 seconds, reached for $K = 18$ for the instance GSE13671, see Fig. 3). The fact that the box of the (*CUT*) model is considerably narrower than the box of the (*CYCLE*) model, indicates that the (*CUT*) approach is more robust regarding the variation of the scores of protein complexes and the value of K.

In Table 1 we report for each instance from [1] the average values (over all $K \in \{10, \dots, 25\}$) of the running times and the average number of cuts added for each of the (*PCStT*), (*CUT*) and (*CYCLE*) models (cf. columns Time (sec), #(2), #(gNSep) and #(7), respectively). In column δ we show the fraction of nodes with a score different than 0 and in column ε the fraction of them with a negative score. The results indicate that the performance of the (*CYCLE*) model strongly depends on the instances under consideration (the average running times of GSE13671 are two orders of magnitude larger that the ones of HT-116-8), which also explains the dispersion shown in Fig. 3. Likewise, for the (*PCStT*) model, the average running time for the instance HT-29-8 is an order of magnitude larger than for the instance GSE13671. In contrast to the unstable performance of (*PCStT*) and (*CYCLE*) models, the (*CUT*) model seems to be more independent on the type of considered instances. From the same table we may conclude that the number of cuts needed to prove the optimality is one order of magnitude smaller for the (*CUT*) model than for the other two models. This means that the (gNSep) cuts are more effective in closing the gap than the (7) and (2) cuts. Regarding δ and ε, it seems that the (*CUT*) model is not sensitive to their values, while the (*CYCLE*) model performs better when ε is smaller.

For the set of Euclidean network instances, running times of the (*CUT*) and (*CYCLE*) model are given in Fig. 4(a) and Fig. 4(b), respectively (for many instances we reached the time-limit for the (*PCStT*) model, so we do not consider it here). This time we group instances according to different combinations of (δ, ε) values. Each box contains $16 \times 8 = 128$ values obtained for the settings: $K \in \{10, \dots, 25\}$,



(a)



(b)

Fig. 4 Dependence of the running times on the (δ, ε) settings. **a** Influence of δ and ε on the performance of the (CUT) model (random instances, $K \in \{10, \dots, 25\}$). **b** Influence of δ and ε on the performance of the $(CYCLE)$ model (random instances, $K \in \{10, \dots, 25\}$)

$n \in \{500, 750, 1000, 1500\}$ and $\alpha \in \{0.6, 1.0\}$. Comparing Fig. 4(a) and Fig. 4(b) we observe that although the average running times (marked with asterisk) of the (CUT) model are in general one order of magnitude smaller than those of the $(CYCLE)$ model, both of them present a similar pattern: (i) For a given δ, the increase of ε from 0.25 to 0.75 produces a worsening of the algorithmic performance. This worsening is visible not only in the increase of the running times, but also in their higher dispersion (wider boxes and more outliers). Increasing ε (for a fixed δ), means that a larger proportion of nodes has a negative weight; since our goal is to find a connected component of exactly K nodes the more nodes with negative weight, the

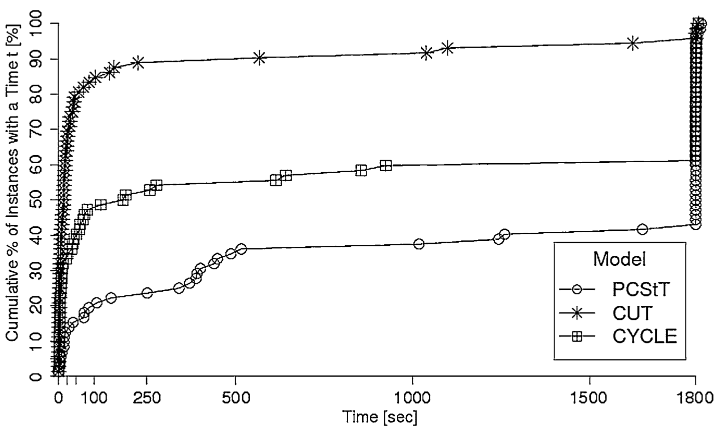


Fig. 5 Performance profile of running times on random undirected instances

more difficult is the task of reaching the "attractive" nodes that lead to a better solution. (ii) On the other hand, increasing δ from 0.25 to 0.75 produces an improvement of the algorithmic performance, i.e., the more nodes with non-zero weights, the easier the problems. One possible reason for this could be the symmetries induced by a large portion of nodes with zero weight (as it is the case for $\delta = 0.25$). Hence, by decreasing this portion (i.e., increasing δ) the cutting-planes that are added through the separation become more effective, and the primal heuristic is able to find more diverse, and eventually better, incumbent solutions.

MWCS on Undirected Graphs For this computational comparison we do not impose cardinality constraints. In order to be able to perform a comparison with the (*CYCLE*) model that requires a digraph G and K as its input, we run the (*CYCLE*) model with (i) G transformed into a digraph, and (ii) with the value of K set to be the size of the optimal unconstrained MWCS solution (obtained by, e.g., the (*CUT*) model). For these graphs we impose a time limit of 1800 seconds. Figure 5 shows the performance profile of the three approaches regarding the total running time. Figure 6 shows the performance profile of the achieved gaps within this time limit. We observe that also in the case of undirected graphs, the (*CUT*) approach significantly outperforms the (*CYCLE*) and the (*PCStT*) approach: While the (*CUT*) approach produces solutions of less than 1 % of gap in almost 100 % of the instances, the (*PCStT*) approach produces solutions with more than 15 % of gap in more than 40 % of the instances. The (*CYCLE*) approach solves about 50 % of instances to optimality, with most of the gaps of the unsolved instances being below 15 %.

In Table 2 we provide more details on these results. Each row corresponds to a fixed value of n, with 18 different instances obtained by varying δ, ε and α. Column #NOpt indicates how many out of those 18 instances were not solved to optimality within the imposed time limit of 1800 seconds. For a given n, and for each of the

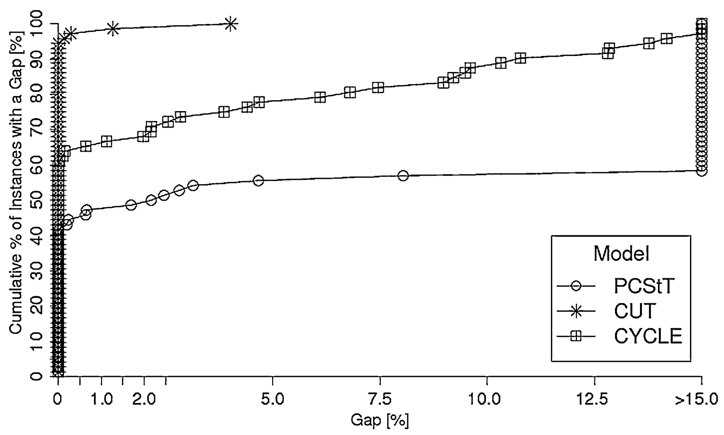


Fig. 6 Performance profile of final gaps (%) on random undirected instances

three approaches we additionally report on the following values: the average running time (cf. column Time (sec)); the average gap of those instances that were not solved to optimality (cf. column Gap (%)), and the average number of inserted cutting planes (cf. columns #(2), #(gNSep), #(7), respectively). These results show that the (*CUT*) model is by far more effective than the (*CYCLE*) model for this group of instances. The average running times of the (*CUT*) model are one order of magnitude smaller than those of the (*PCStT*) and (*CYCLE*) model. All but four instances can be solved by the (*CUT*) model to optimality, while in the case of the (*CYCLE*) and (*PCStT*) model, 29 and 42 instances remain unsolved, respectively. The number of cutting planes of type (gNSep) needed to close the gap is one order magnitude smaller than the number of cuts of type (7) or (2).

So far, it seems clear that for the considered instances the (*CUT*) model significantly outperforms the (*PCStT*) approach. However for the LYMPH instance studied in [8], for which $\delta = 1.0$ and $\varepsilon = 0.97$, the (*PCStT*) model takes only 3.19 seconds to find the optimal solution while the (*CYCLE*) model takes 15.56 seconds, and the (*CUT*) model 50.70 seconds. The optimal solution, whose objective value is 70.2, is comprised by 37 nodes with positive weight and 9 with negative weight. It is not easy to derive a concrete answer of why, for this particular instance, the (*PCStT*) model is faster than the (*CUT*) model. The following two factors could be responsible for this behavior: (i) the sparsity of the graph (the number of edges is approximately four times the number of nodes, while in random instances this ratio is almost 10) which means that the number of $\mathbf{z}$ variables is not too large, and (ii) there are significantly less symmetries due to the fact that there are no nodes with zero weight. These factors might explain why, in this particular case, it becomes easier to solve the problem with the Prize-Collecting Steiner Tree reformulation, rather than directly looking for a connected component that maximizes the objective function.

Table 2 Average values for different values of n (random instances, $\alpha \in \{0.6, 1.0\}$, δ, $\varepsilon \in \{0.25, 0.50, 0.75\}$, 18 problems for each n)

#nodes	#arcs	(*PCStT*)				(*CUT*)				(*CYCLE*)			
		Time (sec)	Gap (%)	#(2)	#NOpt	Time (sec)	Gap (%)	#(gNSep)	#NOpt	Time (sec)	Gap (%)	#(7)	#NOpt
500	4558	677	>15	1055	5	15	–	69	0	615	5.50	4289	6
750	7021	1243	>15	1552	11	108	1.27	99	1	471	2.64	1721	4
1000	9108	1304	>15	1955	12	150	0.29	201	1	990	6.76	3176	9
1500	14095	1526	>15	2021	14	453	2.08	373	2	1086	10.55	2139	10

6 Conclusion

Our work was motivated by the wide range of applications of the MWCS and a recent work of Backes et al. [1] who were the first ones to propose a MIP model for the MWCS derived on the set of node variables only. In this paper we were able to provide a tight MIP model that outperforms the model from [1] both theoretically and computationally. The new model also works on the space of node variables and is valid for all previously studied variants of the MWCS (cardinality constrained, budget constrained and undirected/directed one). We have studied the CS polytope and we have shown that the newly introduced family of generalized node-separator inequalities is facet defining. Our computational study has shown that the new approach outperforms the previously proposed ones, in particular if the inputs are digraphs with non-empty subsets of zero-weight nodes.

Acknowledgements We are deeply thankful to Christina Backes from the Department of Human Genetics, Saarland University, who helped in the understanding and interpretation of the regulatory network instances considered in this paper. This research is partially conducted during the research stay of Ivana Ljubić at the TU Dortmund, supported by the APART Fellowship of the Austrian Academy of Sciences. This support is greatly acknowledged. Eduardo Álvarez-Miranda thanks the Institute of Advanced Studies of the Università di Bologna from where he is a Ph.D. Fellow.

References

1. Backes, C., Rurainski, A., Klau, G., Müller, O., Stöckel, D., Gerasch, A., Küntzer, J., Maisel, D., Ludwig, N., Hein, M., Keller, A., Burtscher, H., Kaufmann, M., Meese, E., Lenhof, H.: An integer linear programming approach for finding deregulated subgraphs in regulatory networks. Nucleic Acids Res. **1**, 1–13 (2011)
2. Bateni, M., Chekuri, C., Ene, A., Hajiaghayi, M., Korula, N., Marx, D.: Prize-collecting Steiner problems on planar graphs. In: Randall, D. (ed.) Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, CA, USA, January 23–25, pp. 1028–1049 (2011)
3. Carvajal, R., Constantino, M., Goycoolea, M., Vielma, J., Weintraub, A.: Imposing connectivity constraints in forest planning models. Oper. Res. (2013). doi:10.1287/opre.2013.1183
4. Chen, C.Y., Grauman, K.: Efficient activity detection with max-subgraph search. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, June 16–21, pp. 1274–1281 (2012)
5. Cherkassky, B.V., Goldberg, A.V.: On implementing push-relabel method for the maximum flow problem. Algorithmica **19**, 390–410 (1994)
6. Chimani, M., Kandyba, M., Ljubic, I., Mutzel, P.: Obtaining optimal k-cardinality trees fast. ACM J. Exp. Algorithmics **14**, 5 (2009)
7. Dilkina, B., Gomes, C.: Solving connected subgraph problems in wildlife conservation. In: Lodi, A., Milano, M., Toth, P. (eds.) CPAIOR. LNCS, vol. 6140, pp. 102–116. Springer, Berlin (2010)
8. Dittrich, M., Klau, G., Rosenwald, A., Dandekar, T., Müller, T.: Identifying functional modules in protein-protein interaction networks: an integrated exact approach. Bioinformatics **24**, i223–i231 (2008)
9. Feigenbaum, J., Papadimitriou, C.H., Shenker, S.: Sharing the cost of multicast transmissions. J. Comput. Syst. Sci. **63**(1), 21–41 (2001)

10. Fischetti, M., Hamacher, H.W., Jørnsten, K., Maffioli, F.: Weighted k-cardinality trees: complexity and polyhedral structure. Networks **24**(1), 11–21 (1994)
11. Fügenschuh, A., Fügenschuh, M.: Integer linear programming models for topology optimization in sheet metal design. Math. Methods Oper. Res. **68**(2), 313–331 (2008)
12. genetrail.bioinf.uni-sb.de/ilp/. Accessed 10 September 2012
13. Goldschmidt, O., Hochbaum, D.S.: k-edge subgraph problems. Discrete Appl. Math. **74**(2), 159–169 (1997)
14. Grötschel, M.: Polyedrische Charakterisierungen Kombinatorischer Optimierungsprobleme. Mathematical Systems in Economics, vol. 36. Verlag Anton Hain, Meisenheim am Glan (1977)
15. Grötschel, M., Monma, C.L.: Integer polyhedra arising from certain network design problems with connectivity constraints. SIAM J. Discrete Math. **3**(4), 502–523 (1990)
16. Grötschel, M., Monma, C.L., Stoer, M.: Facets for polyhedra arising in the design of communication networks with low-connectivity constraints. SIAM J. Optim. **2**(3), 474–504 (1992)
17. Grötschel, M., Monma, C.L., Stoer, M.: Polyhedral and computational investigations for designing communication networks with high survivability requirements. Oper. Res. **43**(6), 1012–1024 (1995)
18. Hochbaum, D.S., Pathria, A.: Node-optimal connected k-subgraphs. Manuscript, UC Berkeley (1994)
19. Ideker, T., Ozier, O., Schwikowski, B., Siegel, A.: Discovering regulatory and signalling circuits in molecular interaction networks. Bioinformatics **18**(Suppl. 1), s233–s240 (2002)
20. Johnson, D.S., Minkoff, M., Phillips, S.: The prize-collecting Steiner tree problem: theory and practice. In: Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms, SODA 2000, San Francisco, CA, USA, 9–11 January, pp. 760–769 (2000)
21. Koch, T., Martin, A.: Solving Steiner tree problems in graphs to optimality. Networks **32**, 207–232 (1998)
22. Lee, H., Dooly, D.R.: Algorithms for the constrained maximum-weight connected graph problem. Nav. Res. Logist. **43**, 985–1008 (1996)
23. Lee, H., Dooly, D.: Decomposition algorithms for the maximum-weight connected graph problem. Nav. Res. Logist. **45**, 817–837 (1998)
24. Ljubić, 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. Math. Program., Ser. B **105**, 427–449 (2006)
25. www.planet-lisa.net/. Accessed 10 September 2012
26. Yamamoto, T., Bannai, H., Nagasaki, M., Miyano, S.: Better decomposition heuristics for the maximum-weight connected graph problem using betweenness centrality. In: Gama, J., Costa, V., Jorge, A., Brazdil, P. (eds.) Discovery Science. LNCS, vol. 5808, pp. 465–472. Springer, Berlin (2009)

Exact Algorithms for Combinatorial Optimization Problems with Submodular Objective Functions

Frank Baumann, Sebastian Berckey, and Christoph Buchheim

Abstract Many combinatorial optimization problems have natural formulations as submodular minimization problems over well-studied combinatorial structures. A standard approach to these problems is to linearize the objective function by introducing new variables and constraints, yielding an extended formulation. We propose two new approaches for constrained submodular minimization problems. The first is a linearization approach that requires only a small number of additional variables. We exploit a tight polyhedral description of this new model and an efficient separation algorithm. The second approach uses Lagrangean decomposition to create two subproblems which are solved with polynomial time algorithms; the first subproblem corresponds to the objective function while the second consists of the constraints. The bounds obtained from both approaches are then used in branch and bound-algorithms. We apply our general results to problems from wireless network design and mean-risk optimization. Our experimental results show that both approaches compare favorably to the standard techniques.

1 Introduction

Many combinatorial optimization problems can be naturally modeled as an integer program in which the objective function is not linear but submodular. Submodularity is a property of set functions. Given a set S, a function $f\colon 2^S \to \mathbb{R}$ is called *submodular*, if for each pair of subsets $A, B \subseteq S$ the property

F. Baumann · C. Buchheim (✉)
Fakultät für Mathematik, Technische Universität Dortmund, Vogelpothsweg 87, 44227 Dortmund, Germany
e-mail: christoph.buchheim@tu-dortmund.de

F. Baumann
e-mail: frank.baumann@tu-dortmund.de

S. Berckey
INFORM Institut für Operations Research und Management GmbH, Pascalstr. 23, 52076 Aachen, Germany
e-mail: sebastian.berckey@inform-software.com

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$$

holds. It is easy to see that the class of submodular functions comprises the class of linear set functions. Submodularity can be interpreted as *diminishing returns*: to see this consider the equivalent definition

$$f\big(A \cup \{x\}\big) - f(A) \geq f\big(B \cup \{x\}\big) - f(B),$$

for $A \subseteq B \subseteq S$ and $x \notin B$. Including the element x into a larger set generates less additional profit. A simple example of a submodular function is the maximum function. Given a weight w_s for each element s of S, the function

$$f(A) = \max_{s \in A} w_s$$

returns the weight of the heaviest element in the subset.

It is an important property of submodular functions that they can be minimized efficiently if the set of feasible solutions is not restricted. The first strongly polynomial time algorithm for submodular function minimization (SFM) was proposed by Grötschel et al. [13], using the ellipsoid method. The problem of finding a combinatorial algorithm was open for more than a decade. It was finally resolved independently by Schrijver [28] and Iwata et al. [16]. Since then, several fully combinatorial algorithms were devised [15, 26].

In the presence of linear constraints, however, the problem often becomes *NP*-hard. This is the case even if optimizing a linear objective function subject to the same constraints is easy. In this paper we consider such combinatorial optimization problems of the form

$$\begin{aligned} &\min f(x) \\ &\text{s.t.} \quad x \in X \subset \{0,1\}^S, \end{aligned} \tag{1}$$

where $f: 2^S \to \mathbb{R}$ is a submodular function on a set S. Without loss of generality we can assume $f(\emptyset) \geq 0$. We associate each binary variable x_i with an element of S. The set X contains all feasible solutions of the problem. In this paper, we generally assume that a *linear* objective function can be optimized over X in polynomial time or, equivalently, that the separation problem for $P := \operatorname{conv} X$ is polynomially solvable. From a practical point of view, the methods proposed can also be applied to problems where the linear optimization problem over X can be solved sufficiently fast or where a tight polyhedral description of P is known.

Commonly, model (1) is either solved directly, as a nonlinear integer program, or it is reformulated as an integer linear program (ILP). ILP models have the advantage that they are well studied and state-of-the-art solvers are extremely efficient. A downside of considering an extended linear formulation is that the linearization often can only be achieved by introducing a large number of new variables and linear constraints to the model, reducing the advantage of using linear solvers considerably. Additionally, such reformulations often not only affect the objective function but also the original constraints, obscuring or even destroying the combinatorial structure of the problem.

In this paper, we aim at generic approaches for submodular combinatorial optimization problems. We present two such general techniques: a polyhedral approach using a compact linear model in Sect. 2 and a Lagrangean decomposition approach in Sect. 3. The first approach relies on efficient separation algorithms for the underlying combinatorial problem. In combination with an efficient separation algorithm for a polyhedron corresponding to the submodular objective function, we obtain a strong polyhedral description for Problem (1). On contrary, in the Lagrangian approach the two subproblems are solved directly. Here we use the existence of efficient algorithms for SFM and for the linear version of Problem (1). Indeed, the assumptions made in both approaches are equivalent: by a well-known result of Grötschel et al. [13], the existence of efficient separation algorithms implies the existence of efficient optimization algorithms and vice versa.

In Sect. 4, we describe how to apply our results to range assignment problems and a problem from portfolio optimization, risk-averse capital budgeting. We present computational results and compare the effectiveness of the proposed techniques to existing state-of-the-art algorithms in Sect. 5.

An extended abstract covering some of the results presented in this paper has appeared in the proceedings of ISCO 2010 [4].

2 A Cutting Plane Approach

In the following, we study the polyhedral structure of submodular combinatorial optimization problems. We describe a class of linear inequalities that gives a complete description of the corresponding polyhedron in the unconstrained case and a corresponding efficient separation algorithm. Combined with the polyhedral description of the set of feasible solutions X we obtain an LP-relaxation of Problem (1).

Starting from the unconstrained nonlinear model

$$\begin{aligned} &\min f(x) \\ &\text{s.t.} \quad x \in \{0,1\}^S \end{aligned}$$

we introduce a single new variable $y \in \mathbb{R}$ to replace the objective function. Clearly, the resulting model

$$\begin{aligned} &\min y \\ &\text{s.t.} \quad y \geq f(x) \\ &\qquad\ x \in \{0,1\}^S \\ &\qquad\ y \in \mathbb{R} \end{aligned}$$

is equivalent to the original one, since we consider a minimization problem. Now consider the convex hull of feasible points:

$$P_f = \operatorname{conv}\bigl\{(x,y) \in \{0,1\}^S \times \mathbb{R} \mid y \geq f(x)\bigr\}.$$

The polyhedron P_f is the epigraph of the so-called Lovász-extension of f. The following result by Edmonds [9] and Lovász [19] gives a complete polyhedral description of P_f.

Theorem 1 *Let $|S| = n$ and let $f\colon 2^S \to \mathbb{R}$ be a submodular function with $f(\emptyset) \geq 0$. Then the separation problem for P_f can be solved in $O(n \log n)$ time. The facets of P_f are either induced by trivial inequalities $0 \leq x_i \leq 1$, $i \in S$, or by an inequality $a^\top x \leq y$ with*

$$a_{\sigma(i)} = f(S_i) - f(S_{i-1}) \quad \forall i \in \{1, \ldots, n\}, \tag{2}$$

where $\sigma\colon \{1, \ldots, n\} \to S$ is any bijection and $S_i = \{\sigma(j) \mid j \in \{1, \ldots, i\}\}$.

In the presence of constraints the above theorem does not yield a complete polyhedral description anymore, but it still provides strong dual bounds on the LP-relaxation, as we will see later. The number of facets of P_f is exponential in $n = |S|$, but the separation problem can be solved efficiently by a simple greedy algorithm. Indeed, violation of the trivial facets is checked in linear time. The following algorithm produces a candidate for a separating hyperplane:

Given a fractional point $(x^\star, y^\star) \in [0, 1]^S \times \mathbb{R}$, sort the elements of S in non-increasing order according to their value in $x^\star$. Starting with the empty set, iteratively construct a chain of subsets $\emptyset = S_0 \subset S_1 \subset \cdots \subset S_n = S$ by adding the elements in this order. The potentially violated inequality $a^\top x \leq y$ is then constructed by setting $a_i = f(S_i) - f(S_{i-1})$. Obviously this algorithm constructs an inequality of the form (2) that is most violated by the given fractional point $(x^\star, y^\star)$. Either this inequality is a separating hyperplane or none such exists. A formal description of this separation procedure is given in Algorithm 1.

In many applications the submodular objective function f can be written as a conic combination of other submodular functions f_i, i.e., we have

$$f = \sum_{i=1}^{k} \alpha_i f_i, \quad \alpha_1, \ldots, \alpha_k \geq 0,\ f_1, \ldots, f_k \text{ submodular.}$$

This situation can be exploited by modeling each function f_i separately, introducing a new continuous variable y_i modeling $f_i(x)$ for each $i \in \{1, \ldots, k\}$. Such an approach could be preferable if, e.g., the values $f_i(x)$ are used at other points in the model or if the functions f_i have much smaller domains than f. In the latter case, the total number of inequalities needed to describe the unconstrained problem can be reduced significantly.

We obtain

$$\begin{aligned} \min \ & \sum_{i=1}^{k} \alpha_i y_i \\ \text{s.t.} \ & y_i \geq f_i(x) \quad \text{for all } i \in \{1, \ldots, k\} \\ & x \in \{0, 1\}^S \\ & y \in \mathbb{R}^k. \end{aligned} \tag{3}$$

Algorithm 1: Separation Algorithm for P_f

```
Input  : a fractional solution $(x^\star, y^\star) = (x_1^\star, \dots, x_n^\star, y^\star)$
Output: a hyperplane $a^\top x \le y$ separating $(x^\star, y^\star)$ from $P_f$, if one exists

Sort the elements of $S$ into a list $\{l_1, \dots, l_n\}$ by non-increasing value of $x^\star$
$i \leftarrow 1$
$S_0 \leftarrow \emptyset$
repeat
    $S_i \leftarrow S_{i-1} \cup \{l_i\}$
    $a_i = f(S_i) - f(S_{i-1})$
    $i \leftarrow i + 1$
until $i = n$
if $y^\star < a^\top x^\star$ then
    return $a$
else
    return no constraint found
end
```

Our next aim is to show that the separation algorithm detailed above can still be used to generate a complete description for Problem (3). First note that Theorem 1 yields a complete description of the polytope P_{f_i} for each $i \in \{1, \dots, k\}$. For the following, define

$$P = \bigcap_{i \in \{1,\dots,k\}} P_{f_i},$$

where each P_{f_i} is trivially extended from $\{0,1\}^S \times \mathbb{R}$ to $\{0,1\}^S \times \mathbb{R}^k$. We will show that each vertex (x, y) of P satisfies $x \in \{0,1\}^S$ and $y_i = f_i(x)$, and hence is feasible for Problem (3). In other words, the separation problem corresponding to (3) can be reduced to the single separation problems for each P_{f_i}.

Lemma 1 *For any submodular function $f\colon \{0,1\}^S \to \mathbb{R}$ and $j \in S$, there is a submodular function $g\colon \{0,1\}^{S\setminus\{j\}} \to \mathbb{R}$ such that $\{x \in P_f \mid x_j = 0\} = P_g$.*

Proof For $x \in \{0,1\}^{S\setminus\{j\}}$, let $\bar{x}$ be its extension to $\{0,1\}^S$, setting $\bar{x}_j = 0$. Defining $g(x) = f(\bar{x})$ yields the desired submodular function. □

Lemma 2 *For any submodular function $f\colon \{0,1\}^S \to \mathbb{R}$ and $j \in S$, there is a submodular function $g\colon \{0,1\}^{S\setminus\{j\}} \to \mathbb{R}$ such that $\{x \in P_f \mid x_j = 1\} = e_j + P_g$, where e_j denotes the unit vector corresponding to x_j.*

Proof For $x \in \{0,1\}^{S\setminus\{j\}}$, let $\bar{x}$ be its extension to $\{0,1\}^S$, setting $\bar{x}_j = 1$. Defining $g(x) = f(\bar{x})$ yields the desired submodular function. □

Lemma 3 *If* $(x, y) \in P$ *with* $x \in (0, 1)^S$, *then* (x, y) *is not a vertex of* P.

Proof Let the all-ones vector in $\mathbb{R}^S$ be denoted by $\mathbf{1}_S$ and choose $\varepsilon > 0$ such that $x \pm \varepsilon \mathbf{1}_S \in [0, 1]^S$. Define $c \in \mathbb{R}^k$ by $c_i = f_i(S) - f_i(\emptyset)$ and consider

$$z_1 = (x - \varepsilon \mathbf{1}_S, y - \varepsilon c), \qquad z_2 = (x + \varepsilon \mathbf{1}_S, y + \varepsilon c).$$

As $(x, y) = \frac{1}{2}(z_1 + z_2)$, it suffices to show $z_1, z_2 \in P$. This reduces to showing $(x \pm \varepsilon \mathbf{1}_S, y_i \pm \varepsilon c_i) \in P_{f_i}$ for all $i \in \{1, \dots, k\}$. By Theorem 1, the polyhedron P_{f_i} is completely described by trivial inequalities and by inequalities of the type $a^\top x \le y_i$ with

$$a_{\sigma(j)} = f_i(S_j) - f_i(S_{j-1}) \quad \forall j \in \{1, \dots, n\}$$

where $\sigma \colon \{1, \dots, n\} \to S$ is any bijection and $S_j = \{\sigma(1), \dots, \sigma(j)\}$. We obtain in particular that $a^\top \mathbf{1}_S = f_i(S) - f_i(\emptyset) = c_i$. As $(x, y) \in P$ and therefore $(x, y_i) \in P_{f_i}$, we derive

$$a^\top (x \pm \varepsilon \mathbf{1}_S) = a^\top x \pm \varepsilon a^\top \mathbf{1}_S \le y_i \pm \varepsilon c_i.$$

Hence $z_1, z_2 \in P_{f_i}$. □

Theorem 2 *The vertices of* P *are exactly the points* $(x, y) \in \{0, 1\}^S \times \mathbb{R}^k$ *with* $y_i = f_i(x)$ *for all* $i \in \{1, \dots, k\}$.

Proof It is clear that every such point is a vertex of P. We show that every vertex (x', y') of P is of this form. Since y_i is not bounded from above, every vertex must satisfy $y_i = f_i(x)$ for all $i \in \{1, \dots, k\}$. Now assume that at least one component of x' is fractional. Define

$$S_0 = \{j \in S \mid x'_j = 0\}, \qquad S_1 = \{j \in S \mid x'_j = 1\}, \qquad T = S \setminus \{S_0 \cup S_1\}$$

and consider the face

$$\begin{aligned} F &= \{(x, y) \in P \mid x_j = 0 \text{ for all } j \in S_0, x_j = 1 \text{ for all } j \in S_1\} \\ &= \bigcap_{i=1}^{k} \{(x, y) \in P_{f_i} \mid x_j = 0 \text{ for all } j \in S_0, x_j = 1 \text{ for all } j \in S_1\}. \end{aligned}$$

By Lemma 1 and Lemma 2, the polyhedron F is an intersection of polyhedra $\mathbf{1}_{S_1} + P_{g_i}$ for suitable submodular functions $g_i \colon \{0, 1\}^T \to \mathbb{R}$. Since $x'_i \in (0, 1)$ for all $i \in T$, the point $(x' - \mathbf{1}_{S_1}, y')$ is not a vertex of $\bigcap_{i=1}^{k} P_{g_i}$ by Lemma 3. It follows that (x', y') is not a vertex of F and hence not a vertex of P. □

Note that the last theorem can also be shown in a more general context [11]. It implies that the polyhedron P_f is a projection of P, given by the linear transformation $y := \sum_{i=1}^{k} \alpha_i y_i$. Moreover, it follows that each facet of P is obtained from a facet of one of the polyhedra P_{f_i}. In particular, the separation problem for (3) can be reduced to the respective separation problems for each polyhedron P_{f_i} as follows: to separate a point $x^\star$ from the polytope P_f it is sufficient to check if $x^\star$ violates any of the inequalities characterizing the polyhedra P_{f_i}. This can be done by applying Algorithm 1 to each P_{f_i} in turn.

So far we have only considered unconstrained submodular optimization problems. Recall that the original Problem (1) was given as

$$\begin{aligned} &\min f(x) \\ &\text{s.t.} \quad x \in X \subset \{0,1\}^S, \end{aligned}$$

where X is the set of feasible solutions. The problem can thus be formulated as

$$\begin{aligned} &\min y \\ &\text{s.t.} \quad (x, y) \in (X \times \mathbb{R}) \cap P_f \subset \{0,1\}^S \times \mathbb{R}. \end{aligned}$$

In this case our results remain applicable but do not necessarily give a complete polyhedral description of the problem anymore. Even if the complete linear description of X (or an exact separation algorithm for X) is available, the combination of the inequalities describing X and P_f in general does not yield a full description of the intersection $(X \times \mathbb{R}) \cap P_f$. For an exact algorithm the generation of cutting planes can be embedded into a branch and bound-approach. In each node of the branch and bound-tree the LP-relaxation is solved. If the solution is fractional, the separation routines for X and P_f are used to generate cutting planes. This process is iterated until the solution is integral or no more violated inequalities are found. In this case a branching step on one of the binary variables is performed.

3 A Lagrangean Decomposition Approach

In this section we avoid the linearization of the submodular objective function and capitalize on the existence of polynomial time algorithms for SFM in the unconstrained case. We use Lagrangean decomposition to separate the objective function from its constraints. For this purpose we add a new variable set x_2 of the same size as the original variable set to Problem (1):

$$\begin{aligned} &\min f(x_1) \\ &\text{s.t.} \quad x_1 = x_2 \\ &\qquad\ \ x_1 \in \{0,1\}^S \\ &\qquad\ \ x_2 \in X \subset \{0,1\}^S. \end{aligned} \tag{4}$$

Next we eliminate the coupling constraint $x_1 = x_2$ by Lagrangean relaxation and then decompose the problem into two parts which are separately solvable. We obtain

$$\begin{aligned} Z(\lambda) = \min\ & f(x_1) - \lambda^T x_1 + \min \lambda^T x_2 \\ \text{s.t.}\ \ & x_1 \in \{0,1\}^S \qquad \text{s.t.} \quad x_2 \in X \subset \{0,1\}^S. \end{aligned} \tag{5}$$

It is well known that $Z(\lambda)$ is a lower bound of Problems (1) and (4) for every Lagrangean multiplier $\lambda \in \mathbb{R}^S$. The second part of (5) is an integer linear problem which is assumed to be solvable in polynomial time in our context. The term $-\lambda^T x_1$ is modular, hence for fixed λ the first part is an unconstrained submodular minimization problem. As mentioned earlier several efficient combinatorial algorithms for SFM have been proposed [15, 26]. Therefore Lagrangean decomposition enables us to calculate dual bounds for Problem (1) in polynomial time for given Lagrangean multipliers $\lambda \in \mathbb{R}^S$. To obtain the best dual bound from the Lagrangean decomposition we can use the so-called Lagrangean dual, i.e. the maximum over all possible Lagrangean multipliers:

$$\begin{aligned} Z_D := \max\ & Z(\lambda) \\ \text{s.t.}\quad & \lambda \in \mathbb{R}^S \end{aligned}$$

$Z(\lambda)$ is a concave function in λ and can be maximized using the subgradient method. To apply this method we need a supergradient of $Z(\lambda)$ for all $\lambda \in \mathbb{R}^S$.

Lemma 4 *For a given point $\lambda^\star \in \mathbb{R}^S$, let $x_1^\star$ and $x_2^\star$ be minimizers of the two components of* (5)*, respectively. Then $x_2^\star - x_1^\star$ is a supergradient of Z in $\lambda^\star$.*

Proof $Z(\lambda)$ is concave and for $\lambda \in \mathbb{R}^S$ we have

$$Z(\lambda^\star) + (\lambda - \lambda^\star)^T (x_2^\star - x_1^\star) = f(x_1^\star) + \lambda^T (x_2^\star - x_1^\star) \geq Z(\lambda). \qquad \square$$

These dual bounds can replace the LP-based bounds in the branch and bound-approach described in Sect. 2. Furthermore, the second part of Problem (5) yields feasible solutions of Problem (1), which can be used to compute upper bounds.

The following theorem states that the best lower bound obtainable by Lagrangean relaxation in our setting is the same as the LP bound discussed in Sect. 2. However, the Lagrange approach might allow a faster computation of this bound in practice, depending on the problem structure. We investigate this in Sect. 5.

Theorem 3 *Let Z_D be defined as above, then*

$$\begin{aligned} Z_D = \min\ & y \\ \textit{s.t.}\quad & (x, y) \in P_f \\ & x \in \operatorname{conv} X. \end{aligned}$$

Proof We can rewrite (4) as

$$\begin{aligned} \min\ & y \\ \text{s.t.}\quad & x_1 = x_2 \\ & y \geq f(x_1) \\ & x_1 \in \{0,1\}^S \\ & x_2 \in X \subset \{0,1\}^S. \end{aligned}$$

By a general result for Lagrangean relaxation [12], Z_D is the minimum of y over the intersection of the two sets $\operatorname{conv}\{(x,y) \mid y \geq f(x), x \in \{0,1\}^S\} = P_f$ and $\operatorname{conv}\{(x,y) \mid x \in X\} = (\operatorname{conv} X) \times \mathbb{R}$. □

4 Applications

4.1 Range Assignment Problems

As a first application we study a class of problems from wireless network design, so-called *range assignment problems*. When designing an ad-hoc wireless network one main objective is to minimize transmission costs subject to certain requirements concerning the network topology. In traditional wired networks, these transmission costs are roughly proportional to the length of all connections installed, so that the aim is to minimize the total length of all connections. In wireless networks, the transmission costs depend on the transmission ranges assigned to the nodes. The main difference lies in the so-called *multicast advantage*: if a node v reaches another node w, then it also reaches each node u that is closer to v than w, at no additional cost. Accordingly, the objective function of range assignment problems, i.e. the overall transmission power of the network needed to establish the specified connections, is nonlinear as a function of the connections.

Range assignment problems have been studied intensively in recent years and several exact algorithms have been proposed. Fuchs [10] showed that the problem of finding a minimum-power connected network with bidirectional links (the *symmetric connectivity problem*) is *NP*-hard. Althaus et al. [1, 2] proposed an ILP formulation for this problem which is based on a linear extended formulation. For each node of the network they introduce a new binary variable for each value the transmission power of the node can take in an optimal solution and express the objective function in terms of these new variables. Montemanni and Gambardella [22] apply a very similar technique, modeling the transmission power levels of the nodes incrementally, as the sum of artificial binary variables. A comparison of different formulations for the symmetric connectivity problem can be found in [24]. Note that all models mentioned above are extended linear formulations of the original problem and do not exploit the submodularity of the objective function directly. We do not know of any approach in the literature that needs only a constant number of artificial variables.

A second important variant of the range assignment problem is the *minimum power multicast problem*. Here the objective is to construct a network that allows unidirectional communication from a designated source node to a set of receiving nodes. All nodes of the network, including the receiving stations, can function as relay nodes, thereby passing on a signal on its way from the source node to the receivers. Special cases are the *unicast problem* and the *broadcast problem*. In the former, communication is directed to a single receiving node; in the latter, all nodes except the source are addressed. The general minimum power multicast problem is *NP*-hard [10]. The unicast problem, on the other hand, is efficiently solvable. With only a single receiving station the problem reduces to finding a shortest path through the directed network from the source to the destination node. The linear variant of the broadcast problem is also known as the *optimum branching problem*. Several authors independently presented efficient algorithms to compute an optimal solution [5, 6, 8].

Many of the algorithms for the symmetric connectivity case can be easily adapted to multicasting. Additionally, Leggieri et al. [18] investigate the multicasting problem specifically and present a set covering formulation, as well as preprocessing techniques to reduce the problem size [25]. There are also flow-based ILP-formulations. One example can be found in the paper by Min et al. [21]. In the same paper the authors present two exact iterative algorithms which use LP-relaxations to compute lower bounds. An overview over existing IP models for the multicast problem can be found in [7].

We model the general range assignment problem in graph theoretic terms. The communication stations correspond to the set of nodes V of the graph, potential links between the stations to the set of weighted edges E. For the symmetric connectivity problem, the graph $G = (V, E, c)$ is undirected with edge costs c, for the multicast problem it is directed. The objective is to compute a subset of the edges such that certain restrictions on the topology of the network are satisfied and the overall transmission costs are minimal. Common to both models is the objective function: given a subset of edges, for each node only the most expensive incident/outgoing edge is taken into account. Summing up these values gives the overall costs. Associating a binary variable x_{vw} to each edge $e = (v, w) \in E$, the objective function can be written as

$$f(x) = \sum_{v \in V} \max\{c_{vw} x_{vw} \mid vw \in E\}. \tag{6}$$

A central property of this objective function is that it is submodular:

Theorem 4 *For each $v \in V$ and for arbitrary $c \in \mathbb{R}^E$, the function*

$$f_v(x) = \max\{c_{vw} x_{vw} \mid vw \in E\}$$

is submodular. In particular, the function $f(x) = \sum_{v \in V} f_v(x)$ is submodular.

Proof By definition, f_v is submodular if

$$f_v(A \cup B) + f_v(A \cap B) \leq f_v(A) + f_v(B)$$

for arbitrary sets $A, B \subseteq E$. We distinguish two cases:

(a) if $f_v(A) \geq f_v(B)$, then $f_v(A \cup B) = f_v(A)$ and $f_v(A \cap B) \leq f_v(B)$
(b) if $f_v(A) \leq f_v(B)$, then $f_v(A \cup B) = f_v(B)$ and $f_v(A \cap B) \leq f_v(A)$

In both cases, it follows that $f_v(A \cup B) + f_v(A \cap B) \leq f_v(A) + f_v(B)$. Finally, the function f is submodular, because it is a conic combination of submodular functions. $\square$

The desired network topology is described by a set of feasible vectors X. Combining objective function and constraints, the general IP formulation for range assignment problems reads

$$\begin{aligned} \min \sum_{v \in V} & \max\{c_{vw} x_{vw} \mid vw \in E\} \\ \text{s.t.} \quad & x \in X. \end{aligned} \tag{7}$$

4.1.1 The Standard Model

As mentioned earlier, the standard linearization for this model found in the wireless networking literature is due to Althaus et al. [1]. They introduce new binary variables which model the possible values of the nonlinear terms in optimal solutions and add constraints linking the original variables to the new ones. The resulting problem reads

$$\begin{aligned} \min \sum_{vw \in E} & c_{vw} y_{vw} \\ \text{s.t.} \quad & \sum_{vw \in E} y_{vw} \leq 1 \quad \text{for all } v \in V \\ & \sum_{c_{vu} \geq c_{vw}} y_{vu} \geq x_{vw} \quad \text{for all } vw \in E \\ & x \in X \\ & y \in \{0,1\}^E. \end{aligned} \tag{8}$$

In this model, the binary variable y_{vw} is thus set to one if and only if the transmission power of node v is just enough to reach node w. Note that, depending on the network topology described by X, the first set of constraints can be strengthened to equations. This is the case when all feasible edge-induced subgraphs are connected. In this case, each node has to reach at least one other node. In general, this is not true, so that for some v all variables y_{vw} can be zero. The number of variables in this model is $2|E|$.

A closely related model appearing in the literature [23] uses binary variables in an incremental way: again, a variable $y'_{vw} \in \{0,1\}$ is used for each pair of nodes v and w, now set to one if and only if node v can reach node w. It is easy to see that the two models are isomorphic by the transformation

$$y'_{vw} = \sum_{c_{vu} \geq c_{vw}} y_{vu}.$$

Because of this, the two models are equivalent from a polyhedral point of view and it suffices to consider the first model in the following.

4.1.2 New Mixed-Integer Models

The general formulation for range assignment problems we gave above is already very similar to the model we studied in Sect. 2. We can now introduce a single artificial variable $y \in \mathbb{R}$ to move the objective function into the constraints. The corresponding model is

$$\begin{aligned} \min\ & y \\ \text{s.t.}\quad & y \geq \sum_{v \in V} \max\{c_{vw}x_{vw} \mid vw \in E\} \\ & x \in X \\ & y \in \mathbb{R}. \end{aligned} \tag{9}$$

From Theorem 4 we know that the objective is submodular; this means that Theorem 1 is applicable and we have an efficient separation algorithm for (9).

Theorem 4 showed that in the case of range assignment problems the objective function is not only submodular itself but also the sum of submodular functions. We can thus use the slightly larger mixed-integer model (3), which in our application reads

$$\begin{aligned} \min\ & \sum_{v \in V} y_v \\ \text{s.t.}\quad & y_v \geq f_v(x) \quad \text{for all } v \in V \\ & x \in X \\ & y \in \mathbb{R}^V. \end{aligned} \tag{10}$$

We know from Theorem 2 that we can again separate efficiently when ignoring the problem-specific constraint $x \in X$.

4.1.3 Polyhedral Relations

In the following, we investigate the polyhedral properties of the standard model and the new mixed-integer models. First, we show how the corresponding polyhedra are related to each other. For this, let $P_1(X)$, $P_2(X)$, and $P_3(X)$ denote the polyhedra given as the convex hulls of feasible solutions in the models (8), (9), and (10), respectively. Note that $P_1(X)$ is a convex hull of binary vectors, so in particular it is a polytope and all its integral points are vertices. On the other hand, the polyhedra $P_2(X)$ and $P_3(X)$ are unbounded by definition. It is easy to see that $P_3(X)$ arises from the convex hull of

$$\left\{(x, y) \in X \times \mathbb{R}^V \mid y_v = \max\{c_{vw}x_{vw} \mid vw \in E\}\ \forall v \in V\right\}$$

by adding arbitrary nonnegative multiples of unit vectors for the variables y_v. Similarly, $P_2(X)$ arises from the convex hull of

$$\left\{(x,y) \in X \times \mathbb{R} \;\middle|\; y = \sum_{v \in V} \max\{c_{vw}x_{vw} \mid vw \in E\}\right\}$$

by adding arbitrary nonnegative multiples of the unit vector for y.

Theorem 5 *The convex hull of all vertices of $P_3(X)$ is a projection of an integer subpolytope of $P_1(X)$.*

Proof Consider the projection π_1 given by

$$y_v := \sum_{vw \in E} c_{vw} y_{vw}.$$

Let $(x,y) \in X \times \mathbb{R}^V$ be a vertex of $P_3(X)$. Then $y_v = \max\{c_{vw}x_{vw} \mid vw \in E\}$ for all $v \in V$. Thus setting $y_{vw} = 1$ for exactly one w with $y_v = c_{vw}$ yields a vertex of $P_1(X)$ that is mapped to (x,y) under π_1. □

In Sect. 2, we have shown that $P_2(X)$ is a projection of the polyhedron $P_3(X)$, so that Theorem 5 also holds if $P_3(X)$ is replaced by $P_2(X)$. These results show that for every reasonable objective function the optimal faces of all three polyhedra are projections of each other. The first model can thus be considered an extended formulation of the second and third one, and the third model can be considered an extended formulation of the second.

Note that in general $P_1(X)$ contains vertices that are not mapped to the convex hull of vertices of $P_2(X)$ or $P_3(X)$. These vertices cannot be optimal for any of the considered objective functions.

4.2 Risk-Averse Capital Budgeting

As a second application we study the risk-averse capital budgeting problem. In portfolio theory an important concept is to not only consider the expected return when choosing a set of investments but also take into account the risk associated with investments. Such *mean-risk optimization problems* can be modeled using stochastic objective functions. Potential investment decisions are represented by independent random variables that have an associated mean value μ as well as a variance σ^2. The mean value stands for the expected return of the investments, σ^2 models the uncertainty inherent in the investment, i.e. the risk that the real return deviates from the expected. The case of continuous variables is well studied whereas the case of discrete variables has received relatively little attention yet.

We concentrate on the *risk-averse capital budgeting problem* with binary variables [3]. In this variant of the mean-risk optimization problem a set of possible investments characterized by their costs, expected return values and variances and a number ε are given as input. The number $\varepsilon > 0$ characterizes the level of risk the investor is willing to take. Investment decisions are binary, this means one can choose to make a certain investment or not. The only constraint in the risk-averse capital budgeting problem is a limit on the available budget. An optimal solution of the problem is a set of investment decisions and a solution value z. The choice of investments guarantees that with probability $1-\varepsilon$ the portfolio will return at least a profit of z.

The corresponding nonlinear IP-model is

$$\begin{aligned} z = \max \sum_{i\in I} \mu_i x_i - \sqrt{\frac{1-\varepsilon}{\varepsilon}\sum_{i\in I}\sigma_i^2 x_i^2} \\ \text{s.t.} \quad \sum_{i\in I} a_i x_i \le b \\ x \in \{0,1\}^I, \end{aligned}$$

where I is the set of available investments, a_i the cost of investment $i \in I$, and b the amount of capital that can be invested. The vector μ represents the expected returns of the investments and σ^2 the variance of the expected returns.

To apply the polyhedral results from Sect. 2 we need to rewrite the above model as a minimization problem and show that the objective function is submodular. Note that since the x-variables are binary we have $x_i^2 = x_i$. The problem now reads

$$\begin{aligned} z = -\min - \sum_{i\in I} \mu_i x_i + \sqrt{\frac{1-\varepsilon}{\varepsilon}\sum_{i\in I}\sigma_i^2 x_i} \\ \text{s.t.} \quad \sum_{i\in I} a_i x_i \le b \\ x \in \{0,1\}^I. \end{aligned} \tag{11}$$

The first part of the objective function

$$f(A) = -\sum_{i\in A} \mu_i + \sqrt{\frac{1-\varepsilon}{\varepsilon}\sum_{i\in A}\sigma_i^2}$$

is obviously modular. The second part is the composition of a nondecreasing modular function and a nondecreasing concave function. It is easy to prove submodularity for a slightly more general class of functions:

Theorem 6 *Let $f\colon \mathbb{R} \to \mathbb{R}$ be a concave function and $g\colon 2^S \to \mathbb{R}$ a submodular function on the set S. If both f and g are nondecreasing, the composition*

$$h\colon 2^S \to \mathbb{R}, \quad h(A) = f\big(g(A)\big)$$

is submodular and nondecreasing.

Proof The composition $f \circ g$ obviously is nondecreasing. To see that it is submodular, note that

$$\begin{aligned} A \cap B \subseteq B \subseteq A \cup B \quad &\Rightarrow \quad g(A \cap B) \leq g(B) \leq g(A \cup B) \\ &\Rightarrow \quad \exists t \in [0,1] : g(B) = tg(A \cap B) + (1-t)g(A \cup B). \end{aligned}$$

We have

$$\begin{aligned} f\big(g(A)\big) &\geq f\big(g(A \cup B) + g(A \cap B) - g(B)\big) \\ &= f\big(tg(A \cup B) + (1-t)g(A \cap B)\big) \\ &\geq tf\big(g(A \cup B)\big) + (1-t)f\big(g(A \cap B)\big) \\ &= f\big(g(A \cup B)\big) + f\big(g(A \cap B)\big) \\ &\quad - \big(tf\big(g(A \cap B)\big) + (1-t)f\big(g(A \cup B)\big)\big) \\ &\geq f\big(g(A \cup B)\big) + f\big(g(A \cap B)\big) - f\big(tg(A \cap B) + (1-t)g(A \cup B)\big) \\ &= f\big(g(A \cup B)\big) + f\big(g(A \cap B)\big) - f\big(g(B)\big), \end{aligned}$$

since f is concave. □

Corollary 1 *The objective function of model* (11) *is submodular.*

Corollary 1 shows that we can address the risk-averse capital budgeting problem (11) by the techniques described above.

5 Computational Results

In the following, we report results of branch and bound-algorithms based on the cutting plane approach of Sect. 2 and the Lagrangean approach of Sect. 3, respectively. For the implementation, we use the exact optimization software library SCIL [29]. The LP-relaxations at each node of the enumeration tree are solved with CPLEX 12.1. The subgradient method for the Lagrangean relaxation approach is implemented using the ConicBundle library v0.3.8 [14]. To calculate the subgradients, i.e. to optimize the second partial problem, we used an implementation of Edmonds' algorithm [31] for the broadcast problem and the Boost Graph Library 1.46.1 for graph modeling and basic graph algorithms [30].

All experiments were run on a 2.6 GHz AMD Opteron 252 processor. We set a time limit of one hour for each instance.

5.1 Symmetric Connectivity

As mentioned earlier the symmetric connectivity problem is a range assignment problem on an undirected graph G. To establish a connection between nodes u

and v, the transmission range of node u must be large enough to reach node v and vice versa. The set X in the general nonlinear model (1) specializes to the set of spanning subgraphs of G.

In this case, all three IP-formulations (8), (9), and (10) can be significantly strengthened. First of all, the set X can be restricted to the set of spanning trees in G without loss of generality. This is equivalent to introducing an additional constraint $\sum_{e\in E} x_e = |V| - 1$. This stronger formulation does not change the optimum of our problem but improves the quality of the bounds obtained from the LP-relaxations and thus reduces running time. In our experiments we used the subtour formulation of the spanning tree polytope.

Another way to strengthen the model is related to the fact that in a connected subgraph (on at least two nodes) each node has at least one incident edge. For the standard model, this means that the constraints $\sum_{uv\in E} y_{uv} \le 1$ can be strengthened to equations $\sum_{uv\in E} y_{uv} = 1$, for all $u \in V$. In the mixed-integer models (9) and (10) we can eliminate one variable from each maximum term. As the transmission power for each node v has to be at least the smallest weight $c_v^{\min}$ of the incident edges, this constant can be extracted from the corresponding maximum term. The constraints of model (10) become

$$y_v \ge c_v^{\min} + \max\big\{\big(c_{vw} - c_v^{\min}\big)x_{vw} \mid vw \in E\big\} \quad \text{for all } v \in V,$$

so that at least one entry in the maximum can be removed. In the compact model (9), the constraint that bounds the overall transmission power from below can be strengthened analogously. Both replacements lead to stronger LP-relaxations if the separation algorithms derived in Sect. 2 are now applied to the remaining maximum terms.

Turning to the Lagrangean relaxation approach, the structure of the set X, i.e. the set of all spanning trees, allows to apply fast combinatorial algorithms like Kruskal's [17] or Prim's [27] to the second problem in (5). The first problem is a special submodular function minimization problem. Even though currently no specialized combinatorial algorithm for this kind of submodular function is available in the case of undirected graphs, there exists one for directed graphs first described by Miller [20]. The algorithm is based on the fact that for directed graphs the minimization of the corresponding submodular function can be decomposed into $|V|$ smaller minimization problems

$$\sum_{v\in V} \min_{x\in\{0,1\}^{\delta(v)}} \left(\max_{e\in\delta(v)} \{c_e x_e\} - \sum_{e\in\delta(v)} \lambda_e x_e \right),$$

where $\delta(v) = \{vw \in E \mid w \in V\}$. This is due to the fact that each variable x_{vw} appears in only one of the minima and the variables are not linked by any constraints. The partial problems can be solved by Algorithm 2, which for each $e \in \delta(v)$ computes the optimal solution x satisfying $x_e = 1$ and $x_f = 0$ for $c_f > c_e$.

We mention that Algorithm 2 can also be implemented to run in linear time after sorting the coefficients c_e; the latter can be done in a preprocessing step, as it does not depend on the Lagrangean multipliers λ. To take advantage of this algorithm

Algorithm 2: Solution of partial problem

```
Input  : objective function $f_v(x) := \max_{e\in\delta(v)}\{c_e x_e\} - \sum_{e\in\delta(v)} \lambda_e x_e$
Output: optimal solution of $\min_{x\in\{0,1\}^{\delta(v)}} f_v$
$x^\star \leftarrow 0$
$opt \leftarrow 0$
for $e \in \delta(v)$ do
    $x \leftarrow 0$
    $sum \leftarrow c_e$
    for $f \in \delta(v)$ do
        if $c_f \le c_e$ and $\lambda_f > 0$ then
            $x_f \leftarrow 1$
            $sum \leftarrow sum - \lambda_f$
        end
    end
    if $sum < opt$ then
        $opt \leftarrow sum$
        $x^\star \leftarrow x$
    end
end
return $x^\star$
```

we will consider a directed version of the symmetric connectivity problem. To gain an equivalent directed formulation we double the variables and introduce new constraints $x_{vw} = x_{wv}$ for all $vw \in E$, where E is now the set of all directed edges between nodes in V. These new constraints will become part of the second problem in (5), so that a spanning tree algorithm can still be applied to the corresponding undirected graph where the weights (Lagrangean multipliers) of two edges vw and wv are summed up.

As an alternative, one could use an algorithm for general submodular function minimization or a linear programming approach to solve the first problem in (5) directly on the undirected instance. However, experiments show that the directed and the undirected models give similar bounds, while the computation of the Lagrangean dual is much faster when Algorithm 2 can be applied. The following results for the Lagrangean decomposition approach are therefore based on the directed version of the symmetric connectivity problem.

To speed up the subgradient method, we use a warm start approach, using the best Lagrangean multipliers from the corresponding parent node as starting points. This leads to a much lower number of iterations in general. Note that in most instances over 50 % of the total time was spent in the root node to compute a good initial set of Lagrangean multipliers.

We generated random range assignment instances by randomly placing points on a 10000×10000 grid, as proposed in [1]. For each size, 50 instances were

Table 1 Results for the symmetric connectivity range assignment problem

n	Subs	LPs/LCs	t_{sep}/s	t_{tot}/s	# solved
Standard IP model (8)					
10	29.48	32.20	0.00	0.08	50
15	1147.96	1217.28	0.18	12.28	50
20	*4048.22*	*4461.83*	*3.63*	*114.65*	46
25	*2248.40*	*2513.51*	*4.35*	*117.99*	43
MIP model (10)					
10	23.28	70.70	0.01	0.08	50
15	823.04	2597.38	0.98	7.29	50
20	*2820.51*	*11049.13*	*16.17*	*100.79*	45
25	*3353.15*	*14001.85*	*45.21*	*281.17*	41
Lagrangean decomposition					
10	18.84	282.26	–	0.63	50
15	180.00	3007.10	–	20.43	50
20	*749.83*	*11871.20*	–	*106.39*	48
25	*1668.22*	*26067.50*	–	*324.14*	41

created. The transmission power needed for node u to reach node v was chosen as $d(u,v)^2$, where $d(u,v)$ is the Euclidean distance between u and v. Table 1 summarizes our results for the symmetric connectivity problem. The first column shows the size of the instances, the second the average number of subproblems computed in the branch and cut-tree. The column *LPs/LCs* contains the average number of linear programs solved (for the cutting plane approach) and the average number of times the Lagrangean function $Z(\lambda)$ was evaluated (for the decomposition approach), respectively. The average overall time needed to solve the instance is denoted by t_{tot}/s. For the cutting plane approach we also state the time spent on separation (t_{sep}/s). The last column shows how many of the 50 instances of each size could be solved within the time limit of one hour. For the computation of averages only instances that could be solved to optimality were considered.

We do not list results for the compact model (9). It turned out that this model is not competitive. Because only a single inequality of the description of the objective function can be separated per iteration, the number of LPs grows quickly in comparison to the other models. The medium-sized model (10) gives the best results for instances up to 15 nodes, also compared to the Lagrangean decomposition approach. The number of subproblems is significantly smaller than for the standard model, which compensates for the larger number of LPs. For instance size 20 the decomposition approach performs best. For the largest instances the standard model gives the best results, because the time spent per node in the other models becomes too large. It is remarkable that several instances could not be solved at all within

Table 2 Results for the multicast range assignment problem with $|T| = \lfloor \frac{n-1}{2} \rfloor$

n	Subs	LPs	t_{sep}/s	t_{tot}/s	# solved
Standard IP model (8)					
10	32.68	57.10	0.00	0.08	50
15	117.92	241.84	0.09	0.93	50
20	2991.52	6444.50	8.43	71.18	50
25	*5773.97*	*27788.03*	*64.02*	*383.07*	39
MIP model (10)					
10	28.92	111.92	0.01	0.10	50
15	88.24	556.38	0.28	1.14	50
20	951.32	7815.54	11.39	33.28	50
25	*5650.17*	*73373.59*	*208.51*	*571.88*	46

the time limit, whereas the average solution time for the other instances is relatively small and only grows moderately with the instance size.

5.2 *Multicast*

We next investigate the min-power multicast problem. Recall that its objective is to send signals wirelessly from a designated source node to a set of receiving stations at minimum cost. Transmissions are unidirectional and all stations can relay signals through the network. Treating this problem as a graph optimization problem, there obviously is a one-to-one correspondence between feasible solutions and Steiner arborescences in the directed graph. The multicast advantage can again be expressed by (6), this time for directed graphs. We used a separation routine for the cut formulation of the Steiner arborescence polytope to model the network topology in both cutting plane models.

The given connectivity constraints can again be used to strengthen the LP-based formulations, however to a lesser extent than in the symmetric case. Only the fact that at least one edge has to leave the source node provides a way to strengthen the models.

Table 2 shows the results for the multicast problem. The number of terminal nodes is $\lfloor \frac{n-1}{2} \rfloor$. As mentioned before the decomposition approach is not applicable here, because there is no efficient algorithm for the Steiner tree problem. We used the same instances as for the symmetric connectivity problem. The source and terminal nodes were determined randomly. For this kind of problem exploiting the submodularity of the objective function clearly pays off. While for small instances both models give similar results, the better polyhedral description in the MIP model significantly reduces running times for larger instances. 46 of the largest instances could be solved to proven optimality within the time limit of one hour, compared to 39 with the standard model.

Table 3 Results for the broadcast range assignment problem

n	Subs	LPs/LCs	t_{sep}/s	t_{tot}/s	# solved
Standard IP model (8)					
10	36.16	45.02	0.00	0.09	50
15	167.24	243.04	0.07	1.31	50
20	1519.40	2801.68	4.24	36.53	50
25	*7117.19*	*16238.07*	*32.57*	*375.87*	43
MIP model (10)					
10	32.88	120.70	0.02	0.12	50
15	142.52	776.78	0.43	1.79	50
20	1051.76	8796.32	13.79	42.87	50
25	*6101.98*	*69896.47*	*200.10*	*598.74*	43
Lagrangean decomposition					
10	25.72	350.14	–	0.53	50
15	447.32	3674.34	–	7.08	50
20	2437.36	20767.40	–	55.22	50
25	*32657.40*	*245163.00*	–	*875.73*	44

5.3 *Broadcast*

Since the problem of finding a minimal Steiner arborescence is *NP*-hard the Lagrangean decomposition approach is inefficient for general multicast problems. However, the set of feasible solutions for the broadcast problem corresponds to the set of s-arborescences, for which the minimization problem can be solved in polynomial time [8]. The first problem in (5) can then again be solved by the algorithm described for the symmetric connectivity problem. Table 3 shows that the Lagrangean decomposition approach is able to solve the highest number of the large instances, while remaining competitive for the smaller instances. The MIP approach is slowed down by the large number of LPs and the resulting high number of calls to the separation routines.

5.4 *Risk-Averse Capital Budgeting*

In contrast to the range-assignment problem presented before, no linear model for this problem is known. Atamtürk and Narayanan [3] present a solution approach that solves the above model using second-order cone programming embedded in a branch and bound-algorithm. They use the inequalities of Theorem 1 to strengthen the relaxation in each node of the enumeration tree. Table 4 shows our results for the risk-averse capital budgeting problem. We solved the set of random instances

Table 4 Results for the risk-averse capital budgeting problem

n	ε	Subs	LPs	Time/s	n	ε	Subs	LPs	Time/s
25	0.10	52.60	147.00	0.07	500	0.10	515.00	1847.80	3.93
	0.05	43.40	182.40	0.10		0.05	1083.40	6768.60	14.38
	0.03	21.40	151.80	0.06		0.03	895.00	8714.00	20.61
	0.02	6.20	74.60	0.03		0.02	1479.00	20914.60	54.44
	0.01	4.60	37.60	0.02		0.01	2420.60	73099.80	267.04
50	0.10	145.40	409.80	0.14	600	0.10	907.00	3295.80	8.34
	0.05	107.80	528.00	0.15		0.05	1277.80	7710.80	20.65
	0.03	67.00	409.40	0.12		0.03	658.20	6830.40	20.38
	0.02	77.80	634.60	0.20		0.02	2207.40	28467.80	96.58
	0.01	19.00	513.00	0.20		0.01	1378.60	38536.80	197.85
100	0.10	210.60	625.60	0.28	700	0.10	1531.40	7823.20	23.47
	0.05	132.20	852.40	0.42		0.05	1064.20	6240.80	19.90
	0.03	251.40	2276.00	1.21		0.03	1391.80	18167.40	65.54
	0.02	252.60	2015.80	1.19		0.02	1970.20	32083.20	135.11
	0.01	205.00	4877.40	3.34		0.01	1616.60	51970.80	336.48
200	0.10	315.40	1041.60	0.91	800	0.10	922.20	3778.20	13.20
	0.05	387.80	2012.60	1.78		0.05	1648.60	11625.00	43.78
	0.03	323.40	2928.20	2.78		0.03	1623.80	14574.20	59.50
	0.02	415.40	4644.60	5.10		0.02	1612.60	22405.00	98.44
	0.01	407.00	18369.20	29.35		0.01	2330.20	83383.80	553.18
300	0.10	623.80	3103.40	3.86	900	0.10	690.20	1710.80	7.03
	0.05	324.60	2377.20	3.10		0.05	456.60	1715.60	7.53
	0.03	391.40	3475.80	4.81		0.03	1049.40	5700.20	27.65
	0.02	411.40	6940.00	11.24		0.02	3505.80	20868.00	117.43
	0.01	364.20	20375.60	59.27		0.01	6004.60	120758.40	929.79
400	0.10	682.20	2884.40	4.83	1000	0.10	1601.40	3730.20	17.77
	0.05	990.60	9454.00	16.33		0.05	1313.80	4198.80	21.11
	0.03	1094.60	12793.20	23.64		0.03	480.60	2496.60	14.31
	0.02	311.40	6138.20	12.83		0.02	1968.20	12609.60	77.33
	0.01	2409.40	56058.00	165.37		0.01	2925.40	71134.20	636.85

from [3], which have between 25 and 100 variables. The larger instances with up to 1000 variables were generated using the same method as for the smaller instances. The expected returns μ and the costs a are independent random numbers between 0 and 100. The variances σ are chosen as the expected returns multiplied by an independent random number between 0 and 1. The available budget is $\frac{1}{2}\sum_{i\in I} a_i$.

This ensures the existence of feasible solutions and at the same time excludes the trivial case where the budget is large enough to make all investments.

We generated five instances of each size and solved each instance for the values of ε given in the table. All values given (number of subproblems, number of linear programs and the running time in seconds) are averages over these five instances.

It can be observed that the value of ε has a strong impact on the running times. For decreasing ε the problem becomes harder to solve. This was already observed in [3]. A direct comparison of our results with the results for the second-order cone programming approach in [3] shows that the number of nodes in the branch and bound-tree is much smaller when our MIP model is used. It is also remarkable that in our model the number of violated inequalities separated is much higher.

As can be seen from Table 4, we were able to solve instances of size 50 in 0.2 seconds on average. The times reported in [3] for the same instances vary between 2 and 79 seconds for different values of ε. Also for $n = 100$ our algorithm is significantly faster for all values of ε. While with the second-order cone programming approach only instances of up to 100 variables could be solved within half an hour, our cutting plane approach easily solves instances of size 1000. Especially remarkable is the fact that the number of subproblems grows only moderately with the instance size.

We did not apply the Lagrangean relaxation approach to risk-averse capital budgeting problems, as we do not know of any fast algorithm for solving the first problem of the decomposition (5) in this case. Using a general purpose submodular function minimizer did not yield satisfactory results.

6 Conclusion

We propose two exact algorithms for solving combinatorial optimization problems with submodular objective functions. Both approaches are tailored for problems that become tractable whenever the submodular objective function is replaced by a linear function. Our algorithms are based on a branch and bound-scheme, where bounds are computed by either a cutting plane approach or by Lagrangean relaxation. The performance of both approaches depends on the underlying problem structure and on the given submodular objective function. If the latter can be minimized very efficiently (ignoring the problem constraints), as is typically the case for range assignment problems, the Lagrangean approach turns out to be very effective. The LP-based approach is applicable to general submodular functions; it yields a flexible and fast solution method, as demonstrated by our results for the risk-averse capital budgeting problem. Our experiments show that treating the objective function and the underlying constraints separately still yields tight relaxations.

References

1. Althaus, E., Calinescu, G., Mandoiu, I.I., Prasad, S., Tchervenski, N., Zelikovsky, A.: Power efficient range assignment in ad-hoc wireless networks. In: WCNC'03, pp. 1889–1894 (2003)
2. Althaus, E., Calinescu, G., Mandoiu, I.I., Prasad, S., Tchervenski, N., Zelikovsky, A.: Power efficient range assignment for symmetric connectivity in static ad hoc wireless networks. Wirel. Netw. **12**(3), 287–299 (2006)
3. Atamtürk, A., Narayanan, V.: Polymatroids and mean-risk minimization in discrete optimization. Oper. Res. Lett. **36**(5), 618–622 (2008)
4. Baumann, F., Buchheim, C.: Submodular formulations for range assignment problems. In: Haouari, M., Mahjoub, A.R. (eds.) Proceedings of ISCO 2010. ENDM, vol. 36, pp. 239–246 (2010)
5. Bock, F.: An algorithm to construct a minimum directed spanning tree in a directed network. In: Developments in Operations Research, pp. 29–44 (1971)
6. Chu, Y., Liu, T.: On the shortest arborescence of a directed graph. Sci. Sin. **14**, 1396–1400 (1965)
7. Das, A.K., Marks, R.J., El-Sharkawi, M., Arabshahi, P., Gray, A.: Minimum power broadcast trees for wireless networks: integer programming formulations. In: INFOCOM 2003, vol. 2, pp. 1001–1010 (2003)
8. Edmonds, J.: Optimum branchings. J. Res. Natl. Bur. Stand. B, Math. Math. Phys. **71B**, 233–240 (1967)
9. Edmonds, J.: Submodular functions, matroids, and certain polyhedra. In: Jünger, M., Reinelt, G., Rinaldi, G. (eds.) Combinatorial Optimization—Eureka, You Shrink! LNCS, vol. 2570, pp. 11–26. Springer, Berlin (2003)
10. Fuchs, B.: On the hardness of range assignment problems. Networks **52**(4), 183–195 (2008)
11. Fujishige, S.: Submodular Functions and Optimization, 2nd edn. Annals of Discrete Mathematics, vol. 58. Elsevier, Amsterdam (2005)
12. Geoffrion, A.M.: Lagrangean relaxation for integer programming. Math. Program. Stud. **2**, 82–114 (1974)
13. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Algorithms and Combinatorics, vol. 2. Springer, Berlin (1988)
14. Helmberg, C.: The ConicBundle library for convex optimization. www-user.tu-chemnitz.de/~helmberg/ConicBundle (2011)
15. Iwata, S., Orlin, J.B.: A simple combinatorial algorithm for submodular function minimization. In: Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'09, pp. 1230–1237 (2009)
16. Iwata, S., Fleischer, L., Fujishige, S.: A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. In: Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, STOC'00, pp. 97–106. ACM, New York (2000)
17. Kruskal, J.B. Jr.: On the shortest spanning subtree of a graph and the traveling salesman problem. Proc. Am. Math. Soc. **7**(1), 48–50 (1956)
18. Leggieri, V., Nobili, P., Triki, C.: Minimum power multicasting problem in wireless networks. Math. Methods Oper. Res. **68**, 295–311 (2008)
19. Lovász, L.: Submodular functions and convexity. In: Mathematical Programming: The State of the Art (Bonn, 1982), pp. 235–257. Springer, Berlin (1983)
20. Miller, P.: Exakte und heuristische Verfahren zur Lösung von Range-Assignment-Problemen. Master's thesis, Universität zu Köln (2010)
21. Min, M., Prokopyev, O., Pardalos, P.: Optimal solutions to minimum total energy broadcasting problem in wireless ad hoc networks. J. Comb. Optim. **11**, 59–69 (2006)
22. Montemanni, R., Gambardella, L.M.: Minimum power symmetric connectivity problem in wireless networks: a new approach. In: MWCN, pp. 497–508 (2004)
23. Montemanni, R., Gambardella, L.M.: Exact algorithms for the minimum power symmetric connectivity problem in wireless networks. Comput. Oper. Res. **32**, 2891–2904 (2005)

24. Montemanni, R., Gambardella, L.M., Das, A.: Mathematical models and exact algorithms for the min-power symmetric connectivity problem: an overview. In: Wu, J. (ed.) Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks, pp. 133–146. CRC Press, Boca Raton (2006)
25. Nobili, P., Oprea, S., Triki, C.: Preprocessing techniques for the multicast problem in wireless networks. In: MTISD 2008, pp. 131–134 (2008)
26. Orlin, J.B.: A faster strongly polynomial time algorithm for submodular function minimization. Math. Program., Ser. A **118**(2), 237–251 (2007)
27. Prim, R.: Shortest connection networks and some generalizations. Bell Syst. Tech. J. **36**, 1389–1401 (1957)
28. Schrijver, A.: A combinatorial algorithm minimizing submodular functions in strongly polynomial time. J. Comb. Theory, Ser. B **80**, 346–355 (2000)
29. SCIL—symbolic constraints in integer linear programming (2011). scil-opt.net
30. Siek, J.G., Lee, L., Lumsdaine, A.: The Boost Graph Library: User Guide and Reference Manual (C++ In-Depth Series). Addison-Wesley, Reading (2001)
31. Sjölund, E., Tofigh, A.: Edmonds' algorithm (2010). edmonds-alg.sourceforge.net

A Primal Heuristic for Nonsmooth Mixed Integer Nonlinear Optimization

Martin Schmidt, Marc C. Steinbach, and Bernhard M. Willert

Abstract Complex real-world optimization tasks often lead to mixed-integer nonlinear problems (MINLPs). However, current MINLP algorithms are not always able to solve the resulting large-scale problems. One remedy is to develop problem specific primal heuristics that quickly deliver feasible solutions. This paper presents such a primal heuristic for a certain class of MINLP models. Our approach features a clear distinction between nonsmooth but continuous and genuinely discrete aspects of the model. The former are handled by suitable smoothing techniques; for the latter we employ reformulations using complementarity constraints. The resulting mathematical programs with equilibrium constraints (MPEC) are finally regularized to obtain MINLP-feasible solutions with general purpose NLP solvers.

1 Introduction

Mixed-integer nonlinear optimization is a highly versatile tool for modeling application problems in many areas: it covers both discrete aspects of decision making and nonlinear real-world phenomena. However, state-of-the-art algorithms for mixed-integer nonlinear problems (MINLPs) are still far from offering the reliability, performance and robustness of solvers for mixed-integer linear problems (MIPs) or nonlinear optimization problems (NLPs). As a consequence, hard MINLPs that cannot be solved directly are frequently tackled by one of the following approaches:

1. *MIP-based approach*: Nonlinearities are replaced by local linearizations or by piecewise linear global approximations. This yields MIP models for which the

M. Schmidt · M.C. Steinbach (✉) · B.M. Willert
Institut für Angewandte Mathematik, Leibniz Universität Hannover, Welfengarten 1, 30167 Hannover, Germany
e-mail: mcs@ifam.uni-hannover.de

M. Schmidt
e-mail: mschmidt@ifam.uni-hannover.de

B.M. Willert
e-mail: willert@ifam.uni-hannover.de

number of discrete variables is often drastically increased due to the linearization techniques; see, e.g., [8, 24, 28, 42] and the references therein.
2. *NLP-based approach*: Discrete aspects are reformulated with continuous variables and constraints or approximated to obtain an NLP model; see Sect. 2.2 for a brief literature review.

Both approaches offer specific advantages and suffer from inherent drawbacks. The MIP approach shows its strength if the MINLP is dominated by discrete variables and incorporates only a few nonlinear constraints. This will typically lead to a slight increase of the number of discrete variables only. Moreover, standard MIP solvers deliver global solutions (of the linearized problem). The NLP approach is often superior for MINLPs with only few discrete variables but a large number of nonlinear constraints, but in general it delivers only local minima. From a numerical point of view, the MIP approach tends to be more robust in terms of starting points, scaling of the problem, etc., whereas the NLP approach can be very fast.

Of course, ultimately one would like to solve hard MINLP models directly. An essential and generally difficult subtask of dedicated MINLP algorithms consists in finding (approximately) feasible solutions to obtain upper bounds. In this paper we present a primal heuristic for that purpose. It is inspired by the NLP-based approach sketched above, and we will refer to it as the *MPEC-based approach*. The MINLP models that can be handled have the key property that their (moderate number of) discrete aspects possess equivalent reformulations with problem-specific *complementarity constraints*, yielding nonlinear *mathematical programs with equilibrium constraints* (MPEC). The MPECs are finally regularized by standard techniques so that they can be solved by NLP algorithms. The goal is to deliver approximate MINLP-feasible solutions quickly (or to fail quickly).

Our approach aims at large-scale optimization models arising in real-life applications. Such models frequently involve nonsmooth constraints (continuous but only piecewise C^2) that can be converted to smooth (C^2-)constraints with artificial discrete variables to identify the smoothness domains. Solving MINLP models of this type tends to be extremely hard: algorithms have been observed to spend a lot of time (or fail) to get the artificial discrete "decisions" right, without making any progress toward a minimum. An important aspect of the approach proposed here lies in avoiding this (inappropriate) modeling of nonsmooth constraints. Instead we consider C^0-MINLPs and employ suitable smoothing techniques to obtain C^2-constraints. A precursor of this approach has been successfully applied in operative planning of water networks [4, 5]. In the application part of this paper (Sect. 4) we demonstrate that the proposed MPEC-based approach delivers approximately MINLP-feasible solutions and hence can be used as a primal heuristic in MINLP solver frameworks.

The motivation for the techniques developed here originates in several application projects from gas and water management, on which the second author worked as a PostDoc in the research group of Martin Grötschel at Zuse Institute Berlin (ZIB). Previous results of this work include [4, 5] and [12, 13, 40]. The application example presented in Sect. 4 will address a problem from a current joint project with the research group of Martin Grötschel.

The paper is organized as follows. Section 2 gives a formal definition of the problem classes arising in the reformulation of nonsmooth mixed-integer nonlinear problems as smoothed and regularized NLP models. In Sect. 3 the relations between these models are discussed. As a proof of concept the real-world application of validation of nominations in gas transport networks is presented in Sect. 4. Finally we give a brief summary in Sect. 5.

2 A Hierarchy of Optimization Models

Here the reformulation of a given nonsmooth MINLP is presented step by step in order to discuss certain properties of the models and to explain some model transition techniques. The sequence of reformulations finally leads to an NLP model.

In all models we denote constraints by c. A vector of constraints is indexed with a corresponding index set. For instance, $c_E := (c_i)_{i \in E}$, is the vector of all equality constraints; c_I the vector of inequality constraints. Frequently we use superindices to indicate the semantics of constraints. A superindex d marks nonsmooth constraints, s marks smoothed constraints and r marks constraints that result from regularization techniques for MPECs. Constraints without superindex are always assumed to be twice continuously differentiable. Continuous variables are referred to as x and discrete ones as z. Objective functions are denoted by f.

2.1 Standard Mixed-Integer Nonlinear Problems

The general MINLP model is given by

$$\min_{x,z} f(x,z) \tag{1a}$$

$$\text{s.t.} \quad c_E(x,z) = 0, \qquad c_I(x,z) \geq 0, \tag{1b}$$

$$x \in \mathbb{R}^{n_x}, \qquad z \in \mathbb{Z}^{n_z}. \tag{1c}$$

Instead of $z \in \mathbb{Z}^{n_z}$ we may have $z \in \{0,1\}^{n_z}$, i.e., z is further restricted to be binary. The objective f and constraints $c = (c_E, c_I)$ are assumed to be twice continuously differentiable.

As discussed in the introduction, many applications do not satisfy the smoothness assumption. While jump discontinuities are properly handled by mixed-integer techniques involving artificial discrete variables and additional (big-M) constraints, this approach typically yields unnecessarily hard MINLPs when applied to nonsmooth but continuous functions. Therefore we consider the refined problem class C^0-MINLP defined by

$$\min_{x,z} f(x,z) \tag{2a}$$

$$\text{s.t.} \quad c_E(x,z) = 0, \qquad c_I(x,z) \geq 0, \tag{2b}$$

$$c_E^d(x,z) = 0, \qquad c_I^d(x,z) \geq 0, \tag{2c}$$

$$x \in \mathbb{R}^{n_x}, \qquad z \in \mathbb{Z}^{n_z}. \tag{2d}$$

Here we split the constraints into smooth and nonsmooth ones: $c = (c_E, c_I) \in C^2$, $c^d = (c_E^d, c_I^d) \in C^0$ and piecewise C^2. For the objective we still assume $f \in C^2$ without loss of generality: nonsmooth terms can always be moved to the constraints c^d. Problem (2a)–(2d) will be the basis of the following sequence of reformulations.

2.2 From C^0-MINLP to C^0-MPEC: Complementarity Constraints

The primary difficulties in the C^0-MINLP (2a)–(2d) are the discrete variables z and the nonsmooth constraints c^d. In this section, we reformulate (2a)–(2d) with continuous variables and additional (smooth) constraints to obtain an equivalent problem without discrete variables. The original nonsmooth constraints c^d will be kept in this step.

Typical reformulation approaches [21, 39] make use of so called *NCP-functions* (see [41] for an overview). In particular, the *Fischer–Burmeister* function is used to restrict a continuous variable $x \in \mathbb{R}$ to $B := \{0, 1\}$ or to $\tilde{B} := \{0\} \cup [1, \infty)$. However, since NCP-functions are nonsmooth we prefer an alternative approach that works directly with complementarity constraints [1]. Ultimately it is based on the trivial fact that

$$x(x-1) = 0 \quad \iff \quad x \in B. \tag{3}$$

Since this formulation is very ill-behaved numerically, a lifted version with an additional continuous variable y is usually preferred, yielding the standard MPEC formulation

$$xy = 0, \quad x, y \geq 0. \tag{4}$$

Here, the cases $x = 0$, $y > 0$ and $x > 0$, $y = 0$ correspond to $x = 0$ and $x = 1$, respectively, but the improved numerical behavior comes at the price of an undecided third state, $x = y = 0$.

Fortunately, many applications with discrete alternatives share a property that can be exploited in a more useful way: the alternatives can be represented as subsets of a space of *continuous* variables. Formally, let A be some model aspect with a finite set of states $A_1, \ldots, A_a$ that correspond to constraint sets for some vector $x_A \in \mathbb{R}^{n_A}$,

$$c_{E,A_i}(x_A) = 0, \qquad c_{I,A_i}(x_A) \geq 0, \quad i = 1, \ldots, a. \tag{5}$$

Then aspect A has a generic MINLP formulation as part of (1a)–(1c) or (2a)–(2d), using binary variables

$$z_A = (z_{A_i})_{i=1}^a \in \{0,1\}^a \tag{6}$$

together with big-M and SOS-1 constraints,

$$M_{E,A_i}(1 - z_{A_i}) - c_{E,A_i}(x_A) \geq 0, \quad i = 1, \dots, a, \tag{7a}$$

$$M_{E,A_i}(1 - z_{A_i}) + c_{E,A_i}(x_A) \geq 0, \quad i = 1, \dots, a, \tag{7b}$$

$$M_{I,A_i}(1 - z_{A_i}) + c_{I,A_i}(x_A) \geq 0, \quad i = 1, \dots, a, \tag{7c}$$

$$\sum_{i=1}^{a} z_{A_i} = 1, \quad z_{A_i} \in \{0,1\}, \quad i = 1, \dots, a. \tag{7d}$$

An equivalent general disjunctive programming formulation [17, 32] is given by

$$\bigvee_{i=1}^{a} \begin{pmatrix} z_{A_i} = 1 \\ c_{E,A_i}(x_A) = 0 \\ c_{I,A_i}(x_A) \geq 0 \end{pmatrix}. \tag{8}$$

For the following we need the key concept of *non-disjunctive* states: states whose constraint sets *overlap*. The formal definition involves characteristic functions.

Definition 1 Let A be a model aspect with states A_i, $i = 1, \dots, a$, represented by variables and constraints as in (5) and (6).

1. A function $\chi_{A_i} : \mathbb{R}^{n_A} \to \mathbb{R}$ is called a *characteristic function* of state A_i if

$$\begin{aligned} &\chi_{A_i}(x) = 0 \quad \text{if } c_{E,A_i}(x) = 0 \text{ and } c_{I,A_i}(x) \geq 0, \\ &\chi_{A_i}(x) > 0 \quad \text{else.} \end{aligned} \tag{9}$$

2. Two states A_i and A_j are called *non-disjunctive* if there exists $x \in \mathbb{R}^{n_A}$ such that

$$\chi_{A_i}(x) = \chi_{A_j}(x) = 0. \tag{10}$$

In what follows we only consider C^0-MINLP models where all discrete aspects have *two* non-disjunctive states and refer to this class as *2-state-C^0-MINLP*. As a direct consequence of the above definition, we can then state the following Lemma.

Lemma 1 *Let A_1, A_2 be non-disjunctive states of a model aspect A that is modeled with variables (x_A, z_A) and constraint sets c_{E,A_i}, c_{I,A_i}, $i = 1, 2$. Let χ_{A_i} denote corresponding characteristic functions. Then the MINLP model of A can be equivalently replaced by the MPEC model*

$$\chi_{A_1}(x_A)\chi_{A_2}(x_A) = 0. \tag{11}$$

Proof Let x_A^* be a solution of the reformulated MPEC model and let $\chi_{A_1}(x_A^*) = 0$. Then it follows with (9) that $c_{E,A_1}(x_A^*) = 0$ and $c_{I,A_1}(x_A^*) \geq 0$. By setting $z_{A_1} = 1$ and $z_{A_2} = 0$ we have constructed a feasible solution to (7a)–(7d). The case $\chi_{A_2}(x_A^*) = 0$ and the reverse direction are analogous. □

Now, an equivalent reformulation of the general 2-state-C^0-MINLP model as a C^0-MPEC model according to Lemma 1 can be written

$$\min_{x\in\mathbb{R}^{n_x}} \; f(x) \tag{12a}$$

$$\text{s.t.}\quad c_E(x) = 0, \qquad c_I(x) \geq 0, \tag{12b}$$

$$c_E^d(x) = 0, \qquad c_I^d(x) \geq 0, \tag{12c}$$

$$\phi_i(x)\psi_i(x) = 0, \quad i = 1, \dots, p, \tag{12d}$$

$$\phi_i(x), \psi_i(x) \geq 0, \quad i = 1, \dots, p. \tag{12e}$$

Here $\phi_i, \psi_i : \mathbb{R}^{n_x} \to \mathbb{R}$ are the complementarity constraint pairings constructed from characteristic functions and p is the number of 2-state model aspects A_i,

$$\phi_i = \chi_{A_{i,1}}, \qquad \psi_i = \chi_{A_{i,2}}, \quad i = 1, \dots, p. \tag{13}$$

Note that the "undecided third state" of (4) does not pose a problem here: the crucial property of non-disjunctive states is that their continuous variables can be identical.

2.3 From C^0-MPEC to C^2-MPEC: Smoothing

This section addresses the remaining major difficulty of (12a)–(12e): the nonsmooth constraints c^d. Quite often the nonsmoothness arises from the absolute value function or from functions that can be expressed in terms of it, like $\min(x, y)$ and $\max(x, y)$.

As already mentioned, first-order discontinuities should not be modeled by artificial discrete variables in the current context; the constraints c^d should rather be replaced with sufficiently smooth approximations c^s. We use approximations that depend on a smoothing parameter, $c^s(x; \tau) \approx c^d(x)$ with $\tau > 0$, satisfying at least a *pointwise approximation property*:

$$\forall x: \quad \lim_{\tau\to 0} c^s(x; \tau) = c^d(x). \tag{14}$$

This provides control over the approximation quality by adjusting the smoothing parameter τ. For the nonsmooth functions mentioned above we actually have uniformly convergent approximations,

$$|x| \approx v(x; \tau) = \sqrt{x^2 + \tau}, \tag{15}$$

$$\min(x, y) \approx y - \frac{1}{2}\big(v(x - y; \tau) - (x - y)\big), \tag{16}$$

$$\max(x, y) \approx y + \frac{1}{2}\big(v(x - y; \tau) + (x - y)\big). \tag{17}$$

Of course, the value of τ should not be chosen too small because this would introduce numerical instabilities and ill-conditioning. Unfortunately there is no general rule for choosing parameters like τ; they have to be tuned separately for each model. Moreover, one often needs problem specific smoothing techniques. We will give an example for an application in gas network optimization in Sect. 4.

The smoothed C^0-MPEC model (12a)–(12e) will be referred to as C^2-MPEC; it reads

$$\min_{x \in \mathbb{R}^{n_x}} \; f(x) \tag{18a}$$

$$\text{s.t.} \quad c_E(x) = 0, \qquad c_I(x) \geq 0, \tag{18b}$$

$$c_E^s(x;\tau) = 0, \qquad c_I^s(x;\tau) \geq 0, \tag{18c}$$

$$\phi_i(x)\psi_i(x) = 0, \quad i = 1, \ldots, p, \tag{18d}$$

$$\phi_i(x), \psi_i(x) \geq 0, \quad i = 1, \ldots, p. \tag{18e}$$

2.4 From C^2-MPEC to C^2-NLP: Regularization

We now have to solve the smooth MPEC model (18a)–(18e). It is well-known that standard NLP constraint qualifications, such as the Mangasarian–Fromowitz constraint qualification (MFCQ) or the linear independence constraint qualification (LICQ), do not hold at *any* feasible point of the MPEC. To apply standard NLP algorithms without losing their good convergence properties, various regularization schemes have therefore been developed. There are basically three groups of existing schemes:

1. relaxation schemes,
2. penalization schemes,
3. smoothing schemes.

The common idea of all regularization schemes is to replace the MPEC constraints (18d) and (18e) with NLP constraints that depend on a regularization parameter μ. Then one solves a sequence NLP(μ_k) with $\mu_k \to 0$ whose solutions converge to an MPEC solution under suitable assumptions.

One of the first regularization schemes is the relaxation scheme of Scholtes [38], which relaxes the complementarity constraints (18d) to

$$\phi_i(x)\psi_i(x) \leq \mu, \quad i = 1, \ldots, p. \tag{19}$$

This yields a regularized problem NLP(μ) with feasible set $F(\mu)$, such that the original MPEC-feasible set is $F(0)$ and $F(\mu_0) \subset F(\mu)$ for all $\mu > \mu_0 \geq 0$. If NLP(μ) is to be solved by an interior point method, the relaxation (19) has the drawback that it lacks strict interior points in the limit. DeMiguel et al. address this problem in [10] by additionally relaxing the nonnegativity constraints (18e) to

$$\phi_i(x), \psi_i(x) \geq -\theta, \quad i = 1, \ldots, p. \tag{20}$$

They propose a method that drives either θ or μ to zero in the limit but not both.

Penalization schemes remove the complementarity constraints completely from the constraints set and introduce a weighted penalty term in the objective instead. Thus (18d) is dropped from (18a)–(18e), and (18a) is replaced with

$$f(x) + \frac{1}{\mu}\Pi\bigl(\phi(x), \psi(x)\bigr). \tag{21}$$

Theoretical results for a general class of penalty functions Π can be found in [19]. In particular, these results include the concrete instance

$$\Pi\bigl(\phi(x), \psi(x)\bigr) = \sum_{i=1}^{p} \phi_i(x)\psi_i(x) \tag{22}$$

that is most frequently used in practice.

Other regularization approaches use nonsmooth reformulations of the complementarity constraints $\phi_i(x)\psi_i(x) = 0$, such as

$$\min\bigl\{\phi_i(x), \psi_i(x)\bigr\} = 0, \tag{23}$$

and employ nonsmooth optimization techniques to solve the resulting problem.

Finally there are smoothing techniques using modified NCP-functions like the perturbed Fischer–Burmeister function (first proposed in [14]),

$$\zeta(\phi, \psi; \tau) = \phi + \psi - \sqrt{\phi^2 + \psi^2 + \tau} = 0. \tag{24}$$

In what follows, we concentrate on relaxation and penalization schemes since these performed best on the application problem presented below. Both approaches of regularizing the MPEC model (18a)–(18e) yield a C^2-NLP that satisfies a standard constraint qualification and can be written in the general form

$$\min_{x\in\mathbb{R}^{n_x}} \; f(x) + g(x; \mu) \tag{25a}$$

$$\text{s.t.} \quad c_E(x) = 0, \qquad c_I(x) \geq 0, \tag{25b}$$

$$c_E^s(x; \tau) = 0, \qquad c_I^s(x; \tau) \geq 0, \tag{25c}$$

$$c_E^r(x; \mu) = 0, \qquad c_I^r(x; \mu) \geq 0, \tag{25d}$$

$$\phi_i(x), \psi_i(x) \geq -\theta, \quad i = 1, \dots, p. \tag{25e}$$

For the relaxation scheme (19), possibly extended by (20), c_E^r vanishes and we have $g = 0$, $\theta \geq 0$, $c_I^r(x; \mu) = (\phi_i(x)\psi_i(x) - \mu)_{i=1}^p$. The penalization scheme (21) has $g(x; \mu) = \frac{1}{\mu}\Pi(\phi(x), \psi(x))$, $\theta = 0$, and $c_E^r(x; \mu)$, $c_I^r(x; \mu)$ are not needed. Defining $\bar{c}_E := (c_E, c_E^s, c_E^r)$, $\bar{c}_I := (c_I, c_I^s, c_I^r, \phi, \psi)$ and $\bar{f} := f + g$ now yields the (parameterized) standard NLP formulation

$$\begin{aligned}
&\min_{x \in \mathbb{R}^{n_x}} \ \bar{f}(x;\mu) \\
&\text{s.t.} \quad \bar{c}_E(x;\tau,\mu) = 0, \qquad \bar{c}_I(x;\tau,\mu) \geq 0.
\end{aligned} \tag{26}$$

In the context of our primal heuristic a final remark is in order. Since we are primarily interested in finding feasible solutions of the original MINLP quickly rather than solving the approximating MPEC with high accuracy, we do not attempt to solve an entire sequence NLP(μ_k) with $\mu_k \to 0$. Instead we take a more aggressive approach and try to solve only a single instance NLP(μ) where the parameter $\mu > 0$ is fixed at a carefully chosen problem-specific value.

3 Relations of the Model Classes

This section briefly highlights basic theoretical relations between the models presented in the last section. We focus on feasible points and not on optimality and stationarity because our main topic here is the primal heuristic.

The first result gives a relation between feasible points of the original model 2-state-C^0-MINLP and its first reformulation C^0-MPEC.

Lemma 2 *Let P be a 2-state-C^0-MINLP in the form (2a)–(2d) and let Q be a reformulation as C^0-MPEC in the form (12a)–(12e). Then for every Q-feasible point x^*_Q there exists a P-feasible point (x^*_P, z^*_P). Conversely, if there is no Q-feasible point, P is also infeasible. Thus, there is a one-to-one correspondence of feasible points between 2-state-C^0-MINLP models and their C^0-MPEC reformulations.*

Proof The claim follows directly from Definition 1 and Lemma 1. □

To obtain binary decisions z^*_P from x^*_Q the complementarity constraints of C^0-MPEC are evaluated at the solution x^*_Q. Then the states are determined according to Definition 1 and (4). If a complementarity constraint is biactive, i.e. both characteristic functions evaluate to zero, the discrete state is arbitrary.

The second transition step, from C^0-MPEC to C^2-MPEC, has a genuinely heuristic flavor: pointwise convergence of the smoothing functions for $\tau \to 0$ does not necessarily imply any useful convergence of C^2-MPEC-feasible sets to C^0-MPEC-feasible sets. Moreover, the smoothing parameter τ is not driven to zero but has to be fixed at some positive value. For complex application problems one will typically try problem-specific smoothings and parameter tuning anyway, and the smoothing error is often smaller than other model inaccuracies. If the overall heuristic still fails, one simply has to rely on computationally more expensive rigorous methods.

The properties of the transition from C^2-MPEC to C^2-NLP depend on the regularization scheme being used. We will discuss penalization and relaxation in the following. To this end, some basic MPEC theory is needed [19, 35, 38] and we consider the standard MPEC formulation:

$$\min_x f(x) \tag{27a}$$

$$\text{s.t.} \quad c_E(x) = 0, \qquad c_I(x) \geq 0, \tag{27b}$$

$$\phi_i(x)\psi_i(x) = 0, \quad i = 1, \dots, p, \tag{27c}$$

$$\phi_i(x), \psi_i(x) \geq 0, \quad i = 1, \dots, p. \tag{27d}$$

Definition 2 We say that the *MPEC linear independence constraint qualification* holds for an MPEC-feasible point x if and only if the standard LICQ holds for the entire constraints system with the exception of complementarity constraints (27c):

$$c_E(x) = 0, \qquad c_I(x) \geq 0, \qquad \phi_i(x) \geq 0, \qquad \psi_i(x) \geq 0. \tag{28}$$

For the following, we define several sets of active indices,

$$\mathcal{A}_c(x) = \big\{ i \in I : c_i(x) = 0 \big\}, \tag{29a}$$

$$\mathcal{A}_\phi(x) = \big\{ i \in \{1, \dots, p\} : \phi_i(x) = 0 \big\}, \tag{29b}$$

$$\mathcal{A}_\psi(x) = \big\{ i \in \{1, \dots, p\} : \psi_i(x) = 0 \big\}. \tag{29c}$$

The next theorem extends standard first-order KKT conditions for NLP to MPEC. A proof can be found in [35].

Theorem 1 *Let x^* be a minimizer of* (27a)–(27d) *and let MPEC-LICQ hold at x^*. Then there exist dual variables $\lambda_E^* \in \mathbb{R}^{|E|}$, $\lambda_I^* \in \mathbb{R}^{|I|}$ and $\gamma_\phi^*, \gamma_\psi^* \in \mathbb{R}^p$ so that*

$$\nabla f^* - \nabla c_E^{*T} \lambda_E^* - \nabla c_I^{*T} \lambda_I^* - \nabla \phi^{*T} \gamma_\phi^* - \nabla \psi^{*T} \gamma_\psi^* = 0, \tag{30a}$$

$$c_E^* = 0, \qquad c_I^* \geq 0, \qquad \phi_i^* \geq 0, \qquad \psi_i^* \geq 0, \tag{30b}$$

$$\phi_i^* = 0 \quad \textit{or} \quad \psi_i^* = 0, \quad i = 1, \dots, p, \tag{30c}$$

$$c_i^* \lambda_{I_i}^* = 0, \quad i \in I, \qquad \phi_i^* \gamma_{\phi_i}^* = 0 \quad \textit{and} \quad \psi_i^* \gamma_{\psi_i}^* = 0, \quad i = 1, \dots, p, \tag{30d}$$

$$\lambda_{I_i}^* \geq 0, \quad i = 1, \dots, p, \tag{30e}$$

$$\gamma_{\phi_i}^* \geq 0, \qquad \gamma_{\psi_i}^* \geq 0, \quad i \in \mathcal{A}_\phi^* \cap \mathcal{A}_\psi^*. \tag{30f}$$

The superindex * indicates function evaluation at x^*. Condition (30a) corresponds to standard dual feasibility, (30b) and (30c) cover primal feasibility of (27a)–(27d), and (30d) is the standard complementarity of inequalities and their multipliers. Finally (30e) and (30f) correspond to nonnegativity of the multipliers of inequality constraints. Note that the last condition is only required for so called *corner pairings* [23], i.e. complementarity pairings satisfying $\phi_i(x^*) = \psi_i(x^*) = 0$.

Theorem 1 is the basis of MPEC stationarity concepts [35]:

Definition 3 Let x^* be MPEC-feasible, i.e. (30b) and (30c) hold, and assume that there exist dual variables λ_E^*, λ_I^*, γ_ϕ^*, γ_ψ^* satisfying (30a)–(30e). Then x^* is called

1. *strongly stationary* if in addition (30f) holds,
2. *C-stationary* if in addition $\gamma_{\phi_i}^* \gamma_{\psi_i}^* \geq 0$.

After these preparations we can discuss the main convergence results of MPEC regularization schemes for $\mu \to 0$.

For the *relaxation* scheme (19) it is shown in [38] that the sequence of stationary points of the relaxed MPECs converge to C-stationary points if the MPEC-LICQ condition holds in the limit. In [18] it is shown that this scheme in fact converges to C-stationary points under the milder assumption of MPEC-MFCQ.

Theoretical results for the *penalization* scheme can be found in [19]. For the penalty objective (21) in particular, convergence to C-stationary points is obtained if MPEC-LICQ holds in the limit.

Stronger convergence results can be proved under stronger assumptions such as the *weak second order necessary condition* or *upper level strict complementarity*. In particular, both schemes deliver MPEC-feasible accumulation points in these cases, which is sufficient for our purpose of constructing a primal heuristic.

4 Application: Gas Network Optimization

The techniques just presented are now applied as a primal heuristic for a planning problem in gas transport. We model the gas network as a directed graph $G = (\mathbb{V}, \mathbb{A})$. The node set $\mathbb{V}$ consists of entries $\mathbb{V}_+$, exits $\mathbb{V}_-$ and junctions $\mathbb{V}_0$, and the arc set $\mathbb{A}$ consists of pipes $\mathbb{A}_{\text{pi}}$, resistors $\mathbb{A}_{\text{re}}$, valves $\mathbb{A}_{\text{va}}$, control valves $\mathbb{A}_{\text{cv}}$ and compressor groups $\mathbb{A}_{\text{cg}}$. A *nomination* defines the amounts of all flows that are supplied or discharged by entry and exit customers. In addition, a nomination determines bounds for the gas pressure as well as specific values of certain gas quality parameters.

We address a problem referred to as *validation of nominations (NoVa)*, which is to decide whether a given nomination can be served by some feasible stationary network operation. Because of discrete decisions at active elements (status of valves, control valves and compressor groups) and the mostly nonlinear and partly non-smooth physical models of network elements this task leads to a nonsmooth MINLP *feasibility* problem. If successful, our heuristic will thus deliver solutions directly.

Various related problems of the gas transport industry have been addressed in the literature. In [6, 29, 44] one finds first (often heuristic) attempts at mixed-integer nonlinear optimization, addressing single compressor groups under fixed operating conditions. Later research incorporates more detailed physical models [3, 45], and more recently also additional discrete aspects and network elements [7]. For large-scale real-world network models MIP-driven approaches have been developed in [9, 25, 26, 31] together with problem-specific heuristic enhancements [27]. NLP-oriented investigations include [2, 33, 34] for stationary optimization, [12, 13, 40] for the transient case and [11] where specific MIP and NLP approaches are compared. The work presented here results from the large industry project ForNe that aims at developing mathematical methods for all kinds of network planning problems. Publications in preparation related to the ForNe project include [15, 20, 30, 36, 37]. ForNe is funded by Open Grid Europe GmbH. The scientific project partners are Friedrich-Alexander Universität Erlangen-Nürnberg, Konrad Zuse Zentrum für Informationstechnik Berlin (ZIB), Universität Duisburg-Essen, Weierstraß Institut für Angewandte Analysis und Stochastik (WIAS), Humboldt Universität zu Berlin, Technische Universität Darmstadt and Leibniz Universität Hannover.

4.1 Model

In this section, the C^0-MINLP model of the NoVa problem is presented along with the smoothed MPEC reformulation C^2-MPEC. As we wish to obtain results quickly, we use a reasonably simplified model rather than the highly detailed model developed in [36]. For instance, the model presented is isothermal, i.e. all temperatures are considered constant. If an approximate solution with correct discrete decisions is found, the accuracy of the continuous variables can still be increased by an extra optimization run with a refined model. On the other hand, real-world instances may contain discrete aspects of global nature like interdependencies of decisions that can currently not be handled by the model.

We introduce every network element model separately and show that the discrete decisions lead to a 2-state-C^0-MINLP in the form (2a)–(2d). The notation is similar to the previous sections except that subindices now refer to network elements or sets thereof. For instance, x_i denotes the variables of the component model of node i, and $\mathbf{c}_{\mathbb{A}_{\text{pi}}}$ denotes the constraints of the component models of all pipes.

4.1.1 Nodes

Every node $i \in \mathbb{V}$ has a gas pressure variable with simple bounds, $p_i \in [p_i^-, p_i^+]$. The flows at node i satisfy a mass balance equation

$$0 = c_i^{\text{flow}}(x) = \sum_{a \in \delta_i^-} q_a - \sum_{a \in \delta_i^+} q_a + d_i, \tag{31}$$

where d_i is the externally supplied flow:

$$d_i \geq 0 \quad \text{for } i \in \mathbb{V}_+, \qquad d_i = 0 \quad \text{for } i \in \mathbb{V}_0, \qquad d_i \leq 0 \quad \text{for } i \in \mathbb{V}_-. \tag{32}$$

The complete (smooth) node model reads

$$0 = \mathbf{c}_i(x) = c_i^{\text{flow}}(x), \quad x_i = p_i. \tag{33}$$

4.1.2 Pipes

Gas dynamics in pipes $a = ij \in \mathbb{A}_{\mathrm{pi}}$ are properly described by the Euler equations for compressible fluids: a PDE system involving mass, momentum and energy balances. Consider a cylindrical pipe with diameter D, cross-sectional area A, roughness k and slope $s \in [-1, +1]$ (the tangent of the inclination angle). For the isothermal and stationary case considered here, the mass balance (continuity equation) yields constant mass flow q along the pipe, the energy equation is not needed, and we are left with the *stationary momentum equation*

$$\frac{\partial p}{\partial x} + \frac{q^2}{A^2}\partial_x \frac{1}{\rho} + g\rho s + \lambda(q)\frac{|q|q}{2A^2 D\rho} = 0. \tag{34}$$

Here g denotes gravitational acceleration, and the *friction coefficient* $\lambda(q)$ is given in terms of the *Reynolds number* $\mathrm{Re}(q)$: for laminar flow by the law of Hagen–Poiseuille,

$$\lambda^{\mathrm{HP}}(q) = \frac{64}{\mathrm{Re}(q)}, \qquad \mathrm{Re}(q) = \frac{D}{A\eta}|q|, \quad |q| \le q_{\mathrm{crit}} \tag{35}$$

and for turbulent flow by the empirical implicit model of Prandtl–Colebrook,

$$\frac{1}{\sqrt{\lambda^{\mathrm{PC}}(q)}} = -2\log_{10}\left(\frac{2.51}{\mathrm{Re}(q)\sqrt{\lambda^{\mathrm{PC}}(q)}} + \frac{k}{3.71D}\right), \quad |q| > q_{\mathrm{crit}}. \tag{36}$$

The state quantities pressure p, density ρ and temperature T in (34) are coupled by an *equation of state*; we use the thermodynamical standard equation

$$\rho = \rho(p, T) = \frac{p}{R_s z(p, T)T}, \tag{37}$$

where R_s is the specific gas constant. The *compressibility factor* $z(p, T)$ is given by an empirical model; here we use a formula of the American Gas Association (AGA),

$$z(p, T) = 1 + 0.257\frac{p}{p_c} - 0.533\frac{p/p_c}{T/T_c}, \tag{38}$$

where p_c and T_c denote the pseudocritical gas pressure and temperature.

The ODE (34) essentially yields the pressure loss along pipe a, for which various approximation formulas exist. We use a quadratic approximation of Weymouth type,

$$0 = c_a^{\text{p-loss}}(x) = p_j^2 - \left(p_i^2 - \Lambda_a z_{a,m}\lambda_a q_a|q_a|\frac{e^{S_a} - 1}{S_a}\right)e^{-S_a}, \tag{39}$$

$$0 = c_a^{\text{slope}}(x) = S_a z_{a,m} - \frac{2L_a g}{R_s T}s_a. \tag{40}$$

The coefficient Λ_a and inclination variable S_a depend on pipe data like length L_a and slope s_a, and on an approximate mean value $z_{a,m}$ of the compressibility factor,

$$0 = c_a^{\text{z-mean}}(x) = z_{a,m} - z(p_{a,m}, T), \tag{41}$$

$$0 = c_a^{\text{p-mean}}(x) = p_{a,m} - \frac{2}{3}\left(p_i + p_j - \frac{p_j p_j}{p_i + p_j}\right). \tag{42}$$

The friction variable λ_a in $c_a^{\text{p-loss}}$ (39) has to satisfy the nonsmooth constraint

$$0 = c_a^{\text{HPPC}}(x) = \lambda_a - \begin{cases} \lambda^{\text{HP}}(q_a), & q_a \le q_{\text{crit}}, \\ \lambda^{\text{PC}}(q_a), & q_a > q_{\text{crit}}. \end{cases} \tag{43}$$

The complete pipe model then reads

$$0 = \mathbf{c}_a(x) = \begin{pmatrix} c_a^{\text{p-loss}}(x) \\ c_a^{\text{p-mean}}(x) \\ c_a^{\text{z-mean}}(x) \\ c_a^{\text{HPPC}}(x) \\ c_a^{\text{slope}}(x) \end{pmatrix}, \qquad x_a = \begin{pmatrix} q_a \\ z_{a,m} \\ p_{a,m} \\ \lambda_a \\ S_a \end{pmatrix}. \tag{44}$$

4.1.3 Pipe Model Reformulation: Smoothing

The pipe model is discontinuous at $q_a = q_{\text{crit}}$ due to c_a^{HPPC}, and second-order discontinuous at $q_a = 0$ due to the term $q_a|q_a|$ in $c_a^{\text{p-loss}}$. We replace the term $\lambda_a q_a |q_a|$ in (39) and constraint (43) by a new variable ϕ_a defined by a smooth constraint,

$$0 = c_a^{\text{p-loss-s}}(x) = p_j^2 - \left(p_i^2 - \Lambda_a z_{a,m} \phi_a \frac{e^{S_a} - 1}{S_a}\right) e^{-S_a}, \tag{45}$$

$$0 = c_a^{\text{HPPC-s}}(x) = \phi_a - r_a q_a \left(\sqrt{q_a^2 + e_a^2} + b_a + \frac{c_a}{\sqrt{q_a^2 + d_a^2}}\right). \tag{46}$$

This provides an asymptotically correct second-order approximation of $\lambda_a q_a |q_a|$ if the parameters r_a, b_a, c_a, d_a, e_a are suitably chosen [4, 36]. In summary, we obtain the smooth pipe model

$$0 = \mathbf{c}_a^{\text{smooth}}(x) = \begin{pmatrix} c_a^{\text{p-loss-s}}(x) \\ c_a^{\text{p-mean}}(x) \\ c_a^{\text{z-mean}}(x) \\ c_a^{\text{HPPC-s}}(x) \\ c_a^{\text{slope}}(x) \end{pmatrix}, \qquad x_a^{\text{smooth}} = \begin{pmatrix} q_a \\ z_{a,m} \\ p_{a,m} \\ \phi_a \\ S_a \end{pmatrix}. \tag{47}$$

4.1.4 Resistors

Resistors $a = ij \in \mathbb{A}_{\text{rs}}$ are fictitious network elements modeling the approximate pressure loss across gadgets, partly closed valves, filters, etc. The pressure loss has the same sign as the mass flow and is either assumed to be (piecewise) constant,

$$0 = c_a^{\text{p-loss-lin}}(x) = p_i - p_j - \xi_a \operatorname{sign}(q_a), \tag{48}$$

or (piecewise) quadratic according to the law of Darcy–Weisbach,

$$0 = c_a^{\text{p-loss-nl}}(x) = p_i - p_j - \frac{8\zeta_a}{\pi^2 D_a^4} \frac{q_a |q_a|}{\rho_{a,k}}. \tag{49}$$

Here ζ_a is the resistance coefficient and $\rho_{a,k}$ is the inflow gas density according to the equation of state (37),

$$0 = c_a^{\text{dens-in}}(x) = \rho_{a,k} - \rho(p_k, T) \quad \text{with } k := \begin{cases} i, & q_a \ge 0, \\ j, & q_a < 0. \end{cases} \tag{50}$$

The compressibility factor z has to be evaluated at the inflow node as well,

$$0 = c_a^{\text{z-in}}(x) = z_{a,k} - z(p_k, T). \tag{51}$$

In summary, the piecewise constant resistor model ($a \in \mathbb{A}_{\text{lin-rs}}$) reads

$$0 = \mathbf{c}_a(x) = c_a^{\text{p-loss-lin}}(x), \qquad x_a = q_a, \tag{52}$$

and the piecewise quadratic resistor model ($a \in \mathbb{A}_{\text{nonlin-rs}}$) reads

$$0 = \mathbf{c}_a(x) = \begin{pmatrix} c_a^{\text{p-loss-nl}}(x) \\ c_a^{\text{dens-in}}(x) \\ c_a^{\text{z-in}}(x) \end{pmatrix}, \qquad x_a = \begin{pmatrix} q_a \\ z_{a,k} \\ \rho_{a,k} \end{pmatrix}. \tag{53}$$

4.1.5 Resistor Model Reformulation: Smoothing

The resistor models (52) and (53) are nonsmooth because of three reasons:

1. the discontinuous sign function in (48),
2. the second-order discontinuous term $|q_a|q_a$ in (49),
3. the direction dependence of the inflow gas density $\rho_{a,k}$ in (49).

Note that items 1 and 3 violate the assumptions made so far. However, resistors play a minor role in the NoVa problem and it suffices to include a coarse approximation in the model, so we just proceed with a problem-specific smoothing.

In the piecewise constant resistor model, we use the identity $\operatorname{sign}(x) = x/|x|$ together with the standard smoothing of $|x|$. For $a \in \mathbb{A}_{\text{lin-rs}}$ this yields

$$0 = \mathbf{c}_a^{\text{smooth}}(x) = c_a^{\text{p-loss-lin-s}}(x) = p_i - p_j - \xi_a \frac{q_a}{\sqrt{q_a^2 + \tau}}. \tag{54}$$

The same approximation of the absolute value function is applied to the piecewise quadratic resistor model (49):

$$0 = c_a^{\text{p-loss-nl-s}}(x) = p_i - p_j - \frac{8\zeta_a}{\pi^2 D_a^4} \frac{q_a\sqrt{q_a^2 + \tau}}{\rho_{a,k}}. \tag{55}$$

Finally, the direction dependence of the inflow gas density $\rho_{a,k}$ is addressed by using the mean density

$$0 = c_a^{\text{dens-mean}}(x) = \rho_{a,m} - \frac{1}{2}(\rho_{a,\text{in}} + \rho_{a,\text{out}}). \tag{56}$$

As a consequence, we need to evaluate the equation of state and the compressibility factor at both nodes i and j,

$$c_a^{\text{dens-in}} = \rho_{a,\text{in}} - \rho(p_i, T), \qquad c_a^{\text{dens-out}} = \rho_{a,\text{out}} - \rho(p_j, T), \tag{57}$$

$$c_a^{\text{z-in}} = z_{a,\text{in}} - z(p_i, T), \qquad c_a^{\text{z-out}} = z_{a,\text{out}} - z(p_j, T). \tag{58}$$

This yields for $a \in \mathbb{A}_{\text{nonlin-rs}}$ the smoothed model

$$0 = \mathbf{c}_a^{\text{smooth}}(x) = \begin{pmatrix} c_a^{\text{p-loss-nl-s}}(x) \\ c_a^{\text{dens-in}}(x) \\ c_a^{\text{dens-out}}(x) \\ c_a^{\text{dens-mean}}(x) \\ c_a^{\text{z-in}}(x) \\ c_a^{\text{z-out}}(x) \end{pmatrix}, \qquad x_a^{\text{smooth}} = \begin{pmatrix} q_a \\ z_{a,\text{in}} \\ z_{a,\text{out}} \\ \rho_{a,\text{in}} \\ \rho_{a,\text{out}} \\ \rho_{a,m} \end{pmatrix}. \tag{59}$$

4.1.6 Valves

Valves $a = ij \in \mathbb{A}_{\text{vl}}$ have two discrete states: *open* and *closed*. Across open valves, the pressures are identical and the flow is arbitrary within its technical bounds,

$$p_j = p_i, \qquad q_a \in [q_a^-, q_a^+]. \tag{60}$$

Closed valves block the gas flow and the pressures are arbitrary within their bounds,

$$q_a = 0, \qquad p_i \in [p_i^-, p_i^+], \qquad p_j \in [p_j^-, p_j^+]. \tag{61}$$

This behavior can be modeled with one binary variable $z_a \in \{0, 1\}$ together with big-M constraints:

$$0 \le c_a^{\text{flow-lb}}(x,z) = q_a - z_a q_a^-, \tag{62}$$

$$0 \le c_a^{\text{flow-ub}}(x,z) = -q_a + z_a q_a^+, \tag{63}$$

$$0 \le c_a^{\text{p-coupl-1}}(x,z) = M_{a,1}(1-z_a) - p_j + p_i, \tag{64}$$

$$0 \le c_a^{\text{p-coupl-2}}(x,z) = M_{a,2}(1-z_a) - p_i + p_j. \tag{65}$$

The resulting valve MINLP model reads

$$0 \le \mathbf{c}_a(x,z) = \begin{pmatrix} c_a^{\text{flow-lb}}(x,z) \\ c_a^{\text{flow-ub}}(x,z) \\ c_a^{\text{p-coupl-1}}(x,z) \\ c_a^{\text{p-coupl-2}}(x,z) \end{pmatrix}, \qquad x_a = q_a. \tag{66}$$

4.1.7 Valve Model Reformulation: Complementarity Constraints

It is easily seen that (66) directly fits into the concept of 2-state model aspects. Here, the model aspect valve has the two states $A_1 = \textit{open}$ and $A_2 = \textit{closed}$. They are non-disjunctive if $0 \in [q_a^-, q_a^+]$ and $[p_i^-, p_i^+] \cap [p_j^-, p_j^+] \neq \emptyset$. The characteristic functions are

$$\chi_a^{\text{open}}(x) = p_j - p_i, \qquad \chi_a^{\text{closed}}(x) = q_a. \tag{67}$$

According to Sect. 2, the 2-state-MINLP model (66) can be equivalently reformulated using a complementarity constraint:

$$0 = \mathbf{c}_a^{\text{mpec}}(x) = c_a^{\text{vl-state}}(x) = \chi_a^{\text{open}}(x)\chi_a^{\text{closed}}(x), \qquad x_a^{\text{mpec}} = q_a. \tag{68}$$

It offers two advantages: no binary variables are required and the number of constraints reduces from four to one.

4.1.8 Control Valves

Control valves $a = ij \in \mathbb{A}_{\text{cv}}$ are used to decrease the gas pressure in a technically prescribed direction (which we define as the graph direction $i \to j$). They possess three discrete states: *active*, *bypass* and *closed*. An *active* control valve reduces the inflow pressure by a certain amount,

$$p_j = p_i - \Delta p_a, \quad \Delta p_a \in \left[\Delta p_a^-, \Delta p_a^+\right], \qquad q_a \in \left[q_a^-, q_a^+\right] \cap \mathbb{R}_+. \tag{69}$$

A *closed* control valve acts like a closed regular valve, leading to the simple state model (61). A control valve in *bypass* mode acts like an open regular valve, with arbitrary flow direction and without decreasing the pressure, cf. (60). Our complete mixed-integer linear model is based on the variable vector $x_a = (q_a, \Delta p_a)^T$ and $z_a = (z_{1,a}, z_{2,a})^T$, where $z_{1,a}$ defines if the control valve is open ($z_{1,a} = 1$) or closed ($z_{1,a} = 0$) and $z_{2,a}$ defines if it is active ($z_{2,a} = 1$) or not ($z_{2,a} = 0$). In terms of the constraints

$$0 \le c_a^{\text{flow-lb-open}}(x,z) = q_a - z_{a,1} q_a^-, \tag{70a}$$

$$0 \le c_a^{\text{flow-ub-open}}(x,z) = -q_a + z_{a,1} q_a^+, \tag{70b}$$

$$0 \le c_a^{\text{flow-lb-active}}(x,z) = q_a - (1 - z_{a,2}) q_a^-, \tag{70c}$$

$$0 \le c_a^{\text{p-coupl-1}}(x,z) = M_{a,1}(1 - z_{a,1}) + \Delta p_a^+ z_{a,2} - (p_i - p_j), \tag{70d}$$

$$0 \le c_a^{\text{p-coupl-2}}(x,z) = M_{a,2}(1 - z_{a,1}) - \Delta p_a^- z_{a,2} - (p_j - p_i), \tag{70e}$$

$$0 \le c_a^{\text{consistent-states}}(x,z) = z_{a,1} - z_{a,2}, \tag{70f}$$

the resulting mixed-integer model then becomes

$$0 \le \mathbf{c}_a(x,z) = \begin{pmatrix} c_a^{\text{flow-lb-open}}(x,z) \\ c_a^{\text{flow-ub-open}}(x,z) \\ c_a^{\text{flow-lb-active}}(x,z) \\ c_a^{\text{p-coupl-1}}(x,z) \\ c_a^{\text{p-coupl-2}}(x,z) \\ c_a^{\text{consistent-states}}(x,z) \end{pmatrix}, \qquad x_a = \begin{pmatrix} q_a \\ \Delta p_a \end{pmatrix}, \qquad z_a = \begin{pmatrix} z_{1,a} \\ z_{2,a} \end{pmatrix}. \tag{71}$$

4.1.9 Control Valve Model Reformulation: Complementarity Constraints

For our reformulation, we require $\Delta p_a^- = 0$. However, this appears to be a moderate restriction in practice: it holds in all cases we have encountered. With this, we can model control valves as a model aspect with two non-disjunctive states $A_1 = \mathit{open}$ and $A_2 = \mathit{closed}$ and the characteristic functions

$$\chi_a^{\text{open}}(x) = p_j - p_i + \Delta p_a, \qquad \chi_a^{\text{closed}}(x) = q_a. \tag{72}$$

The state *open* can then be distinguished in *active* or *bypass* depending on the value of Δp_a. Thus, we have the MPEC reformulation

$$0 = c_a^{\text{cv-state}}(x) = \chi_a^{\text{open}}(x) \chi_a^{\text{closed}}(x). \tag{73}$$

In addition, the restriction to nonnegative flows in the active state is modeled as

$$0 \le c_a^{\text{cv-active-flow}}(x) = \Delta p_a q_a. \tag{74}$$

The complete MPEC type control valve model now reads

$$0 = \mathbf{c}^{\text{mpec}}_{a,E}(x) = c_a^{\text{cv-state}}(x), \qquad 0 \le \mathbf{c}^{\text{mpec}}_{a,I}(x) = c_a^{\text{cv-active-flow}}(x),$$
$$x_a = \begin{pmatrix} q_a \\ \Delta p_a \end{pmatrix}. \tag{75}$$

4.1.10 Compressor Groups

Compressor groups $a = ij \in \mathbb{A}_{\text{cg}}$ typically consist of several compressor units of different types that can be combined in various configurations to increase the gas pressure; see [36] and the upcoming publications [15, 20, 30].

For our primal heuristic we use a substantially simplified model where compressor groups can work in the same states as control valves: *open*, *closed* and *active*. The only difference is the sign of the pressure control variable Δp_a in the characteristic function (72). Thus we have sign changes in (70d) and (70e), yielding adapted mixed-integer and MPEC formulations corresponding to (71) and (75), respectively.

4.2 *Model Summary*

In the preceding sections, we have described components of gas transport networks, both as nonsmooth nonlinear mixed-integer models and, if necessary, as smooth MPEC reformulations. Now we combine the components into complete models.

The complete feasibility problem in C^0-MINLP form reads

$$\exists ?(x, z): \quad \mathbf{c}_E(x) = 0, \qquad \mathbf{c}_I(x, z) \ge 0, \tag{76}$$

where

$$\mathbf{c}_E(x, z) = \begin{pmatrix} \mathbf{c}_{\mathbb{V}}(x) \\ \mathbf{c}_{\mathbb{A}_{\text{pi}}}(x) \\ \mathbf{c}_{\mathbb{A}_{\text{lin-rs}}}(x) \\ \mathbf{c}_{\mathbb{A}_{\text{nonlin-rs}}}(x) \end{pmatrix}, \qquad \mathbf{c}_I(x, z) = \begin{pmatrix} \mathbf{c}_{\mathbb{A}_{\text{va}}}(x, z) \\ \mathbf{c}_{\mathbb{A}_{\text{cv}}}(x, z) \\ \mathbf{c}_{\mathbb{A}_{\text{cg}}}(x, z) \end{pmatrix} \tag{77}$$

and

$$x = (x_{\mathbb{V}}, x_{\mathbb{A}_{\text{pi}}}, x_{\mathbb{A}_{\text{lin-rs}}}, x_{\mathbb{A}_{\text{nonlin-rs}}}, x_{\mathbb{A}_{\text{va}}}, x_{\mathbb{A}_{\text{cv}}}, x_{\mathbb{A}_{\text{cg}}}), \qquad z = (z_{\mathbb{A}_{\text{va}}}, z_{\mathbb{A}_{\text{cv}}}, z_{\mathbb{A}_{\text{cg}}}). \tag{78}$$

Note that the equality constraints do not contain any discrete aspects in our case. Here, nonsmooth aspects arise in all passive elements: $\mathbf{c}_{\mathbb{A}_{\text{pi}}}(x)$, $\mathbf{c}_{\mathbb{A}_{\text{lin-rs}}}(x)$, $\mathbf{c}_{\mathbb{A}_{\text{nonlin-rs}}}(x)$. Discrete decisions (with "genuine" binary variables) arise in the active elements: $\mathbf{c}_{\mathbb{A}_{\text{va}}}(x, z)$, $\mathbf{c}_{\mathbb{A}_{\text{cv}}}(x, z)$, $\mathbf{c}_{\mathbb{A}_{\text{cg}}}(x, z)$. The node model $\mathbf{c}_{\mathbb{V}}(x)$ is smooth and will be kept in its original form.

Collecting all smoothed and complementarity constrained components yields the following C^2-MPEC model:

$$\exists ?x: \quad \text{s.t.} \quad \mathbf{c}_E(x) = 0, \qquad \mathbf{c}_I(x) \geq 0, \tag{79}$$

where

$$\mathbf{c}_E(x) = \begin{pmatrix} \mathbf{c}_{\mathbb{V}}(x) \\ \mathbf{c}^{\text{smooth}}_{\mathbb{A}_{\text{pi}}}(x) \\ \mathbf{c}^{\text{smooth}}_{\mathbb{A}_{\text{lin-rs}}}(x) \\ \mathbf{c}^{\text{smooth}}_{\mathbb{A}_{\text{nonlin-rs}}}(x) \\ \mathbf{c}^{\text{mpec}}_{\mathbb{A}_{\text{va}}}(x) \\ \mathbf{c}^{\text{mpec}}_{\mathbb{A}_{\text{cv}},E}(x) \\ \mathbf{c}^{\text{mpec}}_{\mathbb{A}_{\text{cg}},E}(x) \end{pmatrix}, \qquad \mathbf{c}_I(x) = \begin{pmatrix} \mathbf{c}^{\text{mpec}}_{\mathbb{A}_{\text{cv}},I}(x) \\ \mathbf{c}^{\text{mpec}}_{\mathbb{A}_{\text{cg}},I}(x) \end{pmatrix} \tag{80}$$

and

$$x = \left(x_{\mathbb{V}}, x^{\text{smooth}}_{\mathbb{A}_{\text{pi}}}, x_{\mathbb{A}_{\text{lin-rs}}}, x^{\text{smooth}}_{\mathbb{A}_{\text{nonlin-rs}}}, x^{\text{mpec}}_{\mathbb{A}_{\text{va}}}, x^{\text{mpec}}_{\mathbb{A}_{\text{cv}}}, x^{\text{mpec}}_{\mathbb{A}_{\text{cg}}}\right). \tag{81}$$

Finally, the C^2-MPEC model (79) is regularized by any of the techniques from Sect. 2. The reformulation is generic except for one aspect. Some of the complementarity constraint pairings in $\mathbf{c}_E(x)$ do not have to be nonnegative as in the standard MPEC (27a)–(27d). For instance, this can happen for flow variables with a negative lower bound. In this case we square the corresponding functions so that condition (9) in Definition 1 is satisfied. We denote the regularization of (79) by C^2-NLP.

The primal heuristic for our concrete application actually involves additional problem-specific steps. As already mentioned in Sect. 4.1.10, we use an idealized compressor group model that disregards individual compressor units. Solutions of the above C^2-NLP (*stage-1*) are therefore refined by solving a second NLP (*stage-2*) that incorporates discrete decisions of individual compressor units by a special convexification approach; see [20] for details. If both stages are successful, we finally check the stage-2 feasible solution with a highly accurate validation NLP [36] to decide whether it is sufficiently accurate to be used in practice. Full details of these and other aspects of the application problem will be given in the future papers [15, 20, 30, 36, 37].

4.3 *Numerical Results*

We have tested the primal heuristic on the northern high-calorific gas network of our industry partner Open Grid Europe GmbH. The network model contains 452 pipes, 9 resistors, 35 valves, 23 control valves and 6 compressor groups. Gas is supplied at 31 entry nodes and discharged at 129 exit nodes. The intermediate C^2-MPEC models are regularized by penalization; the resulting NLP models are formulated with

Table 1 Success rate of primal heuristic on NoVa test sets

Test set	Size	Success rate (%)		
		stage-1	stage-2	NLP
SN2	3 882	100.0	90.34	72.1
SN3	4 077	100.0	85.45	65.0
SN4	4 227	99.97	88.24	59.0
Expert	40	100.0	47.50	30.0

the modeling language GAMS v23.8.2 [16] and solved with the interior point code Ipopt v3.10 [43] on a Desktop PC with an Intel i7 920 CPU and 12 GB RAM. The C++ software framework LaMaTTO++ [22] is used to implement the models and to interface the problem data. LaMaTTO++ is a framework for modeling and solving mixed-integer nonlinear programming problems on networks. It was originally developed by the working groups of Jens Lang and Alexander Martin and is now being used and extended within the ForNe project.

The test set includes some 12 000 NoVa instances of four different types: the sets SN i, $i = 2, 3, 4$, and Expert. SN i, $i = 2, 3, 4$, are automatically generated nominations. The generation process depends on the current set of contracts with supplying and discharging customers and historical data about nominated entry and exit capacities. The three sets of nominations mainly differ in how the contracts are modeled within the generation process. The sets SN i are of increasing degree of difficulty (see [20] for full details). The set Expert contains 40 manually designed nominations from our industry partner that are intended to represent *hard* instances.

The success rates are given in Table 1. The first column states the name of the NoVa test set and the second column gives the number of instances in the set. The following three columns show the percentage of instances that have successfully passed stage-1, stage-2, and the final validation NLP, respectively. Table 2 offers statistics of the CPU times of successful runs. For both stages and the validation NLP, the minimum, maximum and average CPU times of the different test sets are displayed. Similarly, Table 3 shows the minimum, maximum and average numbers of iterations of the successful instances. The maximum allowed number of iterations was set to 3 000.

In addition, the profiles in Fig. 1 and Fig. 2 display the distribution of the required iterations and CPU time. More formally, if P denotes one of the test sets and if t_p,

Table 2 Min, max, and average Ipopt CPU time (seconds) of primal heuristic on NoVa test sets

Test set	stage-1			stage-2			NLP		
	min	max	avg	min	max	avg	min	max	avg
SN2	1.8	26.3	7.5	0.3	65.5	1.0	0.8	11.7	1.1
SN3	2.0	27.6	10.3	0.2	51.6	1.1	0.5	4.0	1.4
SN4	2.3	28.6	10.5	0.3	50.4	1.2	0.9	5.3	2.1
Expert	5.0	40.8	11.4	0.9	4.7	2.0	1.1	1.7	1.3

Table 3 Min, max, and average Ipopt iterations of primal heuristic on NoVa test sets

Test set	stage-1			stage-2			NLP		
	min	max	avg	min	max	avg	min	max	avg
SN2	74	913	263.4	11	2 345	39.0	22	256	31.2
SN3	78	969	365.2	6	3 000	44.3	11	62	33.6
SN4	90	996	371.0	13	1 700	46.7	25	68	34.4
Expert	157	692	322.1	29	137	61.9	28	40	31.4

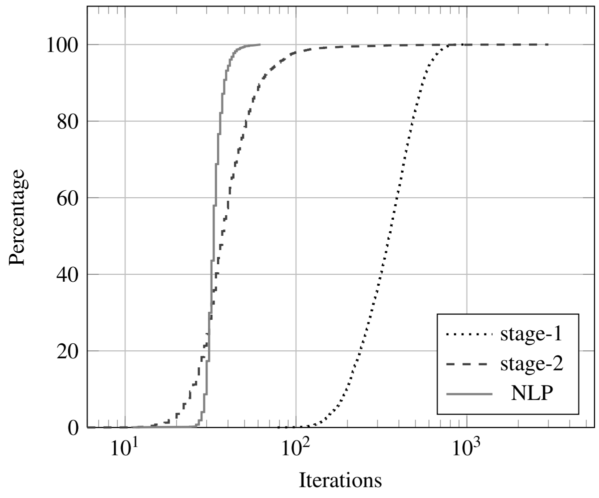


Fig. 1 Profiles of Ipopt iterations in stages 1, 2, and validation NLP on test set SN3

$p \in P$, is the considered performance measure for problem p (here: the number of iterations or the CPU time), the plots show the graph $\tau \mapsto 100|\{p \in P : t_p \leq \tau\}|/|P|$. Thus, the graphs display the percentage of feasible instances that need at most τ iterations (or τ seconds) to be solved. The displayed data represents stage-1, stage-2 and the validation NLP on test set SN3. The profiles of the remaining test sets look essentially similar.

We see that stage-1 is successful on all instances but one, in spite of the smoothings, MPEC regularization and other approximations that are involved. This is primarily due to the model simplifications employed here, in particular the idealized compressor group model. With the detailed compressor group model in stage two, discrete decisions for individual compressor units are found for 85 % to 90 % of the statistical nominations and for less than half of the expert nominations. Finally, 59 % to 72 % of all statistical nominations and 30 % of the expert nominations pass the high-accuracy validation NLP.

It is apparent how the success rate slowly decreases with increasing difficulty of the test sets (see Table 1). In particular, the expert nominations really prove to be hard from stage-2 on. This is because of the central role of the compressor units: in hard cases they have to be operated close to their limits, and a highly accurate

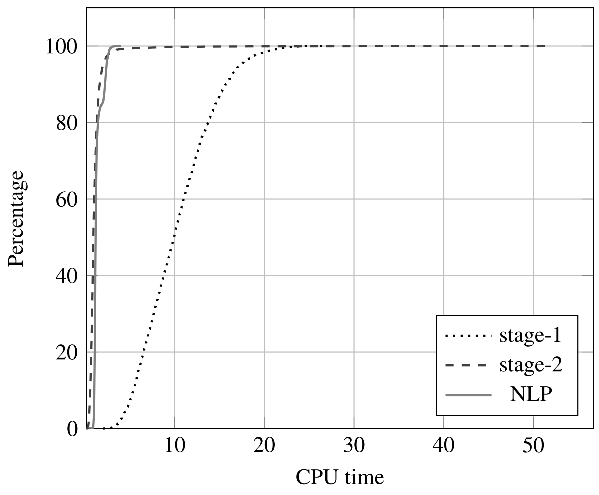


Fig. 2 Profiles of Ipopt CPU time (seconds) in stages 1, 2, and validation NLP on test set SN3

model is required to distinguish feasible and infeasible operating points. To lower the risk of missing feasible solutions early on, we therefore use increasingly restrictive compressor models in the successive stages.

A further reason of failure can occur in case of approximately biactive complementarity constraints, i.e. complementarity constraints with small non-zero values of both characteristic functions. Infeasible discrete decisions may then be deduced; for instance, a valve may be considered to be closed although a very small flow is actually required.

A comparison of the CPU times and iterations shows that most of the computational effort is spent on stage-1, taking about 10 s, while stage-2 and the validation NLP roughly require another second each. This indicates that the simplified stage-1 model is still reasonably hard (it encompasses the major difficulties) and that stage-2 can actually be considered as a refinement step. The profile plots support this interpretation, since the stage-1 curves are located significantly to the right of both the stage-2 and the validation NLP curves.

5 Summary

We have proposed a general reformulation technique as a primal heuristic for a certain class of nonsmooth mixed-integer nonlinear optimization problems. Our approach explicitly distinguishes discrete aspects and nonsmooth but continuous aspects. The former are handled with complementarity constraints; the latter are handled by generic or problem-specific smoothing techniques. Additional regularizations are applied to obtain a smooth and regular nonlinear optimization problem that can be solved by standard NLP solvers to produce (approximately) feasible solutions of the underlying nonsmooth MINLP efficiently. As a proof of concept, we have successfully applied our heuristic to the problem of validation of nominations in real-life gas networks.

Acknowledgements This work has been funded by the Federal Ministry of Economics and Technology owing to a decision of the German Bundestag. We would next like to thank our industry partner Open Grid Europe GmbH and the project partners in the ForNe consortium. The authors are also indebted to an anonymous referee whose comments and suggestions greatly improved the quality of the paper. Finally, the second author would like to express his gratitude for the challenging and stimulating scientific environment and the personal support that Martin Grötschel provided to him as a PostDoc in his research group at ZIB, and for the continued fruitful cooperation that he is now experiencing as a ZIB Fellow with his own research group at Leibniz Universität Hannover. We dedicate this paper to Martin Grötschel on the occasion of his 65th birthday.

References

1. Baumrucker, B.T., Renfro, J.G., Biegler, L.T.: MPEC problem formulations and solution strategies with chemical engineering applications. Comput. Chem. Eng. **32**(12), 2903–2913 (2008)
2. Borraz-Sánchez, C., Ríos-Mercado, R.Z.: A hybrid meta-heuristic approach for natural gas pipeline network optimization. In: Blesa, M., Blum, C., Roli, A., Sampels, M. (eds.) Hybrid Metaheuristics. Lecture Notes in Computer Science, vol. 3636, pp. 54–65. Springer, Berlin (2005). doi:10.1007/11546245_6
3. Boyd, E.A., Scott, L.R., Wu, S.: Evaluating the quality of pipeline optimization algorithms. In: Pipeline Simulation Interest Group 29th Annual Meeting, Tucson, AZ, paper 9709 (1997)
4. Burgschweiger, J., Gnädig, B., Steinbach, M.C.: Optimization models for operative planning in drinking water networks. Optim. Eng. **10**(1), 43–73 (2009). doi:10.1007/s11081-008-9040-8
5. Burgschweiger, J., Gnädig, B., Steinbach, M.C.: Nonlinear programming techniques for operative planning in large drinking water networks. Open Appl. Math. J. **3**, 14–28 (2009). doi:10.2174/1874114200903010014
6. Carter, R.G.: Compressor station optimization: computational accuracy and speed. In: Pipeline Simulation Interest Group 28th Annual Meeting, paper 9605 (1996)
7. Cobos-Zaleta, D., Ríos-Mercado, R.Z.: A MINLP model for a problem of minimizing fuel consumption on natural gas pipeline networks. In: Proc. XI Latin-Ibero-American Conference on Operations Research, paper A48-01, pp. 1–9 (2002)
8. Dantzig, G.B.: On the significance of solving linear programming problems with some integer variables. Econometrica **28**(1), 30–44 (1960). www.jstor.org/stable/1905292
9. de Wolf, D., Smeers, Y.: The gas transmission problem solved by an extension of the simplex algorithm. Manag. Sci. **46**(11), 1454–1465 (2000)
10. DeMiguel, A.V., Friedlander, M.P., Nogales, F.J., Scholtes, S.: A two-sided relaxation scheme for mathematical programs with equilibrium constraints. SIAM J. Optim. **16**(1), 587–609 (2005). doi:10.1109/TIT.2005.860448
11. Domschke, P., Geißler, B., Kolb, O., Lang, J., Martin, A., Morsi, A.: Combination of nonlinear and linear optimization of transient gas networks. INFORMS J. Comput. **23**(4), 605–617 (2011). doi:10.1287/ijoc.1100.0429
12. Ehrhardt, K., Steinbach, M.C.: KKT systems in operative planning for gas distribution networks. Proc. Appl. Math. Mech. **4**(1), 606–607 (2004)
13. Ehrhardt, K., Steinbach, M.C.: Nonlinear optimization in gas networks. In: Bock, H.G., Kostina, E., Phu, H.X., Rannacher, R. (eds.) Modeling, Simulation and Optimization of Complex Processes, pp. 139–148. Springer, Berlin (2005)
14. Fischer, A.: A special Newton-type optimization method. Optimization **24**(3–4), 269–284 (1992). doi:10.1080/02331939208843795
15. Fügenschuh, A., Geißler, B., Gollmer, R., Hayn, C., Henrion, R., Hiller, B., Humpola, J., Koch, T., Lehmann, T., Martin, A., Mirkov, R., Morsi, A., Römisch, W., Rövekamp, J.,

Schewe, L., Schmidt, M., Schultz, R., Schwarz, R., Schweiger, J., Stangl, C., Steinbach, M.C., Willert, B.M.: Mathematical optimization for challenging network planning problems in unbundled liberalized gas markets. Technical report ZR 13-13, ZIB (2013)
16. GAMS—A Users Guide. Redwood City (1988)
17. Grossmann, I.E.: Review of nonlinear mixed-integer and disjunctive programming techniques. Optim. Eng. **3**(3), 227–252 (2002). doi:10.1023/A:1021039126272
18. Hoheisel, T., Kanzow, C., Schwartz, A.: Theoretical and numerical comparison of relaxation methods for mathematical programs with complementarity constraints. Preprint 299, Institute of Mathematics, University of Würzburg (2010)
19. Hu, X.M., Ralph, D.: Convergence of a penalty method for mathematical programming with complementarity constraints. J. Optim. Theory Appl. **123**, 365–390 (2004)
20. Koch, T., Bargmann, D., Ebbers, M., Fügenschuh, A., Geißler, B., Geißler, N., Gollmer, R., Gotzes, U., Hayn, C., Heitsch, H., Henrion, R., Hiller, B., Humpola, J., Joormann, I., Kühl, V., Lehmann, T., Leövey, H., Martin, A., Mirkov, R., Möller, A., Morsi, A., Oucherif, D., Pelzer, A., Pfetsch, M.E., Schewe, L., Römisch, W., Rövekamp, J., Schmidt, M., Schultz, R., Schwarz, R., Schweiger, J., Spreckelsen, K., Stangl, C., Steinbach, M.C., Steinkamp, A., Wegner-Specht, I., Willert, B.M., Vigerske, S. (eds.): From Simulation to Optimization: Evaluating Gas Network Capacities. In preparation
21. Kraemer, K., Marquardt, W.: Continuous reformulation of MINLP problems. In: Diehl, M., Glineur, F., Jarlebring, E., Michiels, W. (eds.) Recent Advances in Optimization and Its Applications in Engineering, pp. 83–92. Springer, Berlin (2010). doi:10.1007/978-3-642-12598-0_8
22. LaMaTTO++. www.mso.math.fau.de/edom/projects/lamatto.html
23. Leyffer, S., López-Calva, G., Nocedal, J.: Interior methods for mathematical programs with complementarity constraints. SIAM J. Optim. **17**, 52–77 (2004)
24. Markowitz, H.M., Manne, A.S.: On the solution of discrete programming problems. Econometrica **25**(1), 84–110 (1957). www.jstor.org/stable/1907744
25. Martin, A., Möller, M.: Cutting planes for the optimization of gas networks. In: Bock, H.G., Kostina, E., Phu, H.X., Rannacher, R. (eds.) Modeling, Simulation and Optimization of Complex Processes, pp. 307–329. Springer, Berlin (2005)
26. Martin, A., Möller, M., Moritz, S.: Mixed integer models for the stationary case of gas network optimization. Math. Program., Ser. B **105**(2–3), 563–582 (2006). doi:10.1007/s10107-005-0665-5
27. Martin, A., Mahlke, D., Moritz, S.: A simulated annealing algorithm for transient optimization in gas networks. Math. Methods Oper. Res. **66**(1), 99–115 (2007). doi:10.1007/s00186-006-0142-9
28. Meyer, R.R.: Mixed integer minimization models for piecewise-linear functions of a single variable. Discrete Math. **16**(2), 163–171 (1976). doi:10.1016/0012-365X(76)90145-X
29. Osiadacz, A.: Nonlinear programming applied to the optimum control of a gas compressor station. Int. J. Numer. Methods Eng. **15**(9), 1287–1301 (1980). doi:10.1002/nme.1620150902
30. Pfetsch, M.E., Fügenschuh, A., Geißler, B., Geißler, N., Gollmer, R., Hiller, B., Humpola, J., Koch, T., Lehmann, T., Martin, A., Morsi, A., Rövekamp, J., Schewe, L., Schmidt, M., Schultz, R., Schwarz, R., Schweiger, J., Stangl, C., Steinbach, M.C., Vigerske, S., Willert, B.M.: Validation of nominations in gas network optimization: models, methods, and solutions. Technical report ZR 12-41, ZIB (2012)
31. Pratt, K.F., Wilson, J.G.: Optimization of the operation of gas transmission systems. Trans. Inst. Meas. Control **6**(5), 261–269 (1984). doi:10.1177/014233128400600411
32. Raman, R., Grossmann, I.E.: Modeling and computational techniques for logic based integer programming. Comput. Chem. Eng. **18**(7), 563–578 (1994)
33. Ríos-Mercado, R.Z., Wu, S., Scott, L.R., Boyd, A.E.: A reduction technique for natural gas transmission network optimization problems. Ann. Oper. Res. **117**, 217–234 (2002)
34. Ríos-Mercado, R.Z., Kim, S., Boyd, A.E.: Efficient operation of natural gas transmission systems: a network-based heuristic for cyclic structures. Comput. Oper. Res. **33**(8), 2323–2351 (2006). doi:10.1016/j.cor.2005.02.003

35. Scheel, H., Scholtes, S.: Mathematical programs with complementarity constraints: stationarity, optimality and sensitivity. Math. Oper. Res. **25**, 1–22 (2000)
36. Schmidt, M., Steinbach, M.C., Willert, B.M.: High detail stationary optimization models for gas networks—part 1: model components. IfAM preprint 94, Inst. of Applied Mathematics, Leibniz Universität Hannover (2012, submitted)
37. Schmidt, M., Steinbach, M.C., Willert, B.M.: High detail stationary optimization models for gas networks—part 2: validation and results (2013, in preparation)
38. Scholtes, S.: Convergence properties of a regularization scheme for mathematical programs with complementarity constraints. SIAM J. Optim. **11**(4), 918–936 (2001)
39. Stein, O., Oldenburg, J., Marquardt, W.: Continuous reformulations of discrete-continuous optimization problems. Comput. Chem. Eng. **28**(10), 1951–1966 (2004)
40. Steinbach, M.C.: On PDE solution in transient optimization of gas networks. J. Comput. Appl. Math. **203**(2), 345–361 (2007). doi:10.1016/j.cam.2006.04.018
41. Sun, D., Qi, L.: On NCP-functions. Computational optimization—a tribute to Olvi Mangasarian, part II. Comput. Optim. Appl. **13**(1–3), 201–220 (1999). doi:10.1023/A:1008669226453
42. Vielma, J.P., Nemhauser, G.L.: Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. In: Lodi, A., Panconesi, A., Rinaldi, G. (eds.) Integer Programming and Combinatorial Optimization. Lecture Notes in Computer Science, vol. 5035, pp. 199–213. Springer, Berlin (2008). doi:10.1007/978-3-540-68891-4_14
43. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006). doi:10.1007/s10107-004-0559-y
44. Wright, S., Somani, M., Ditzel, C.: Compressor station optimization. In: Pipeline Simulation Interest Group 30th Annual Meeting, paper 9805 (1998)
45. Wu, S., Ríos-Mercado, R.Z., Boyd, A.E., Scott, L.R.: Model relaxations for the fuel cost minimization of steady-state gas pipeline networks. Technical report TR-99-01, University of Chicago (1999)

A New Algorithm for MINLP Applied to Gas Transport Energy Cost Minimization

Björn Geißler, Antonio Morsi, and Lars Schewe

Abstract In this article, we present a new algorithm for the solution of nonconvex mixed-integer nonlinear optimization problems together with an application from gas network optimization, the gas transport energy cost minimization problem. Here, the aim is to transport gas through the network at minimum operating cost. The proposed algorithm is based on the adaptive refinement of a new class of MIP-relaxations and has been developed within an industry project on gas network optimization. Since therefore the implementation is not as general as it could be, our computational results are restricted to instances from gas network optimization at this point of time. However, as these problems are real-world applications and turn out to be rather hard to solve with the aid of state-of-the-art MINLP-solvers we believe that our computational results reveal the potential of this new approach and motivate further research on the presented techniques.

1 Introduction

In Martin Grötschel's work, the development of the theory of mixed-integer linear programming was always pushed forward in tandem with the solution of application problems. A rich source for such problems was the design of networks, with the primary application being the design of communication networks [9–13].

In this article, we build on the vast improvements in the understanding and algorithmic tractability of mixed-integer linear programs. This solid foundation on which we build would not be available without the work of Martin Grötschel and his coauthors.

B. Geißler (✉) · A. Morsi · L. Schewe
Department Mathematik, Friedrich-Alexander-Universität Erlangen-Nürnberg, Cauerstr. 11, 91058 Erlangen, Germany
e-mail: bjoern.geissler@math.uni-erlangen.de

A. Morsi
e-mail: antonio.morsi@math.uni-erlangen.de

L. Schewe
e-mail: lars.schewe@math.uni-erlangen.de

The problem we consider, though a rather special mixed-integer nonlinear program, offers interesting structure regularly found in other problems. We aim at optimizing flow in a natural gas network with supplies and demands. The problem structure then is essentially determined by conservation laws from physics, tailored to the commodity natural gas. The objective is to route the gas through the network with the help of compressors such that the energy cost for the operation of the compressors is minimized. An essential simplification we will make here is to consider the stationary case as in e.g. [22, 30]. This is justified in so far as it can be of interest for planning purposes of a network operator to get good solutions for the general supply/demand scenario without taking into account transient fluctuations.

We propose an algorithm for the solution of nonconvex mixed-integer nonlinear optimization problems, i.e., optimization problems of the form below, where even after relaxing the integrality restrictions the remaining nonlinear problem remains nonconvex. This property precludes the use of standard branch-and-bound techniques as the underlying nonlinear optimization problem is already hard to solve on its own. The general thrust of our approach is to reformulate the problem in such a way that we replace the difficult nonconvexities in the original problem with comparatively well-understood nonconvexities, namely integrality restrictions. In short, we reformulate the mixed-integer *nonlinear* problem as mixed-integer *linear* problem. We have already shown such a reformulation [8] that was a proper relaxation of the underlying mixed-integer nonlinear program. The main problem of this approach was, however, that we had to fix the precision of the relaxation a priorily and, thus, had to compute a relaxation of the problem that achieves this bound over the whole domain. In this article, we show how to extend this approach to be able to refine the given problem adaptively. This allows us to start with a rather coarse relaxation of the underlying problem and to refine the relaxation only where needed.

Optimization problems in the design and operation of natural gas networks have already been considered in the optimization and the engineering literature since the 1960s. Specifically, the minimization of operating costs has been an active topic of research. For a thorough overview over the literature we refer to the survey [29] and to the overviews in [16] and [28].

There are a variety of possible solution approaches to general mixed-integer nonlinear programming problems. For an excellent overview we refer to [33]. The algorithm described in [17] is probably closest to our approach. The main difference is that there the nonconvexities are not expressed via integrality restrictions but are instead tackled mainly via branching.

We compare our approach to two state-of-the-art general mixed-integer nonlinear programming solvers: Baron [32] and SCIP [1, 21, 33]. Both use the same general solving paradigm: Spatial branch-and-bound using linear programming relaxations. They differ in the way the LP relaxations of the problems are built.

We will first discuss a mixed-integer nonlinear programming model for the gas transport energy cost minimization problem. The next section is then devoted to describe the basic relaxation approach from [8]. Building on that, we will describe our adaptive approach and conclude with computational results on real-world test instances.

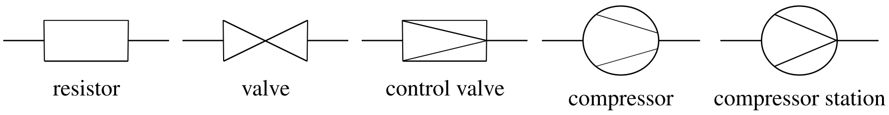


Fig. 1 Technical symbols of the network components

2 MINLP-Model

In this section, we derive an MINLP model for the Gas Transport Energy Cost Minimization Problem. The model we give is an extended version of the model that is used to model the problem of validating a nomination, i.e. a vector of supplies and demands. For a more thorough introduction we refer to [16] or [28]. The difficulty of the problem has two sources: Nonconvex nonlinear constraints and integrality constraints. The former are mainly needed to model the physics of the gas flow in the pipes, the compressors, and the resistors. The integrality constraints are needed to model switching behavior introduced through valves and furthermore through restrictions on the possible operation modes of subnetworks. As all of these components have a non-negligible influence on the solution of the problem, we have to model it as a mixed-integer nonlinear programming problem and cannot get by with a simpler model.

2.1 Network Model

We model a gas network by means of a directed finite graph $G = (\mathcal{V}, \mathcal{A})$. The set $\mathcal{A}$ of arcs is partitioned into the set $\mathcal{A}_P$ of *pipes*, the set $\mathcal{A}_S$ of *shortcuts*, the set $\mathcal{A}_R$ of *resistors*, the set $\mathcal{A}_V$ of *valves*, the set $\mathcal{A}_{CV}$ of *control valves* and the set $\mathcal{A}_{CS}$ of *compressors*. Each control valve $a \in \mathcal{A}_{CV}$ is part of a so-called *control valve station* and each compressor $a \in \mathcal{A}_{CS}$ is part of a so-called *compressor station*. Compressor stations and control valve stations are series-parallel subgraphs of G. We denote the set of control valve stations by $\mathcal{H}_{CV}$ and the set of compressor stations by $\mathcal{H}_{CS}$. Compressor and control valve stations have pairwise disjoint arc sets, i.e., for all $S, S' \in \mathcal{H}_{CV} \cup \mathcal{H}_{CS}$ with $S \neq S'$, we have $\mathcal{A}(S) \cap \mathcal{A}(S') = \emptyset$. A summary of the technical symbols of the components is given in Fig. 1.

The set $\mathcal{V}$ of nodes consists of the set $\mathcal{V}_S$ of *entries*, the set $\mathcal{V}_D$ of *exits* and the set $\mathcal{V}_I$ of *intersection points* of segments, which are neither entries nor exits. Entries are considered as gas delivering points and exits reflect gas demands, specified by the quantities flow and pressure. In the following, the gas flow on an arc $a = (i, j) \in \mathcal{A}$ is denoted by q_a and the pressure at a node $v \in \mathcal{V}$ is denoted by p_v. We associate positive gas flow with flow in arc direction, i.e., if gas flows from node i to node j, we have $q_a \geq 0$ and if gas flows from node j to node i we have $q_a < 0$. Gas flow and pressure are the most important variables. In our model, pressure is measured in bar and gas flow is given in thousands of standard cubic meters, which is the most commonly used unit for flow in gas economy. One standard cubic meter of gas denotes the quantity of gas, which has a volume of one cubic meter under standard conditions, i.e., at 1.01325 bar, 0 °C and 0 % of humidity according to ISO 2533 or DIN 1343. Nevertheless, in the remainder of this section, we assume all physical values given in coherent SI units [3], in order to avoid confusing constants in the formulas, which are due to unit conversions. In our model, we do not distinguish between different gas mixtures. All parameters depending on the chemical composition of the gas are considered constant.

For each node $v \in \mathcal{V}$ of the network, we assume minimal and maximal pressure values p_v^- and p_v^+, the height h_v of the node in terms of meters above mean sea level and a lower and upper bound d_v^- and d_v^+ on the demand d_v in terms of gas flow as given. To model a node, we bound the pressure variable p_v and the demand flow variable d_v from below and above.

$$p_v^- \le p_v \le p_v^+ \quad \forall v \in \mathcal{V}, \tag{1}$$

$$d_v^- \le d_v \le d_v^+ \quad \forall v \in \mathcal{V}. \tag{2}$$

Further, the first law of Kirchhoff must hold. This physical law ensures that the amount of gas flowing into a node v is equal to the amount of gas flowing away from v minus the amount of gas taken out of the network at v.

$$\sum_{e \in \delta^-(v)} q_e - \sum_{e \in \delta^+(v)} q_e = d_v \quad \forall v \in \mathcal{V}. \tag{3}$$

We remark that according to our definition of different node types d_v^- is non-negative for exits $v \in \mathcal{V}_D$, d_v^+ is non-positive for entries $v \in \mathcal{V}_S$ and $d_v^- = d_v^+ = 0$ for intersection points $v \in \mathcal{V}_I$.

For each arc $a \in \mathcal{A}$ of the network, we assume lower and upper bounds q_a^-, q_a^+ for the gas flow to be given and bound the corresponding flow variables.

$$q_a^- \le q_a \le q_a^+ \quad \forall a \in \mathcal{A}. \tag{4}$$

We proceed with a detailed description of each type of network arc starting with pipes.

2.2 Pipes

Pipes are the main components of a gas network. A pipe $a = (i, j) \in \mathcal{A}_P$ is specified by its *length* L_a, *diameter* D_a and the *roughness* k_a of the pipe wall. In our model, we assume all pipes to be straight and of cylindrical shape. Gas flow in such a pipe is governed by the system of Euler equations supplemented by a suitable equation of state. Since pipes in Germany are typically at least one meter beneath the ground, it is reasonable to assume the *temperature* T to be constant. In such a situation, isothermal flow is an appropriate model. Taking into account a non-ideal gas behavior, the relevant equations reduce to the continuity and momentum equation, together with the equation of state for real gases [4]. The *continuity equation*

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho v}{\partial x} = 0 \tag{5}$$

states that the rate of change of *gas density* ρ in time is proportional to the *mass flow*

$$M = \rho_0 q \tag{6}$$

across the pipe boundaries. Here,

$$v = \frac{\rho_0}{A_a} \frac{q}{\rho} \tag{7}$$

denotes the *velocity* of the gas, where $A_a = \frac{D_a^2 \pi}{4}$ is the *cross sectional area* of the pipe and ρ_0 the *gas density under standard conditions*.

Since our model is intended to be used from a planner's perspective, where in contrast to, e.g., real-time optimal control problems for gas networks transient effects can be safely neglected. Here, a *stationary model* for the gas flow, where all time derivatives are zero, is reasonable.

In this case the continuity equation simplifies to

$$\frac{\partial \rho v}{\partial x} = \frac{\rho_0}{A_a} \frac{\partial q}{\partial x} = 0 \quad \Leftrightarrow \quad \frac{\partial q}{\partial x} = 0 \tag{8}$$

meaning that gas flow is constant within a pipe. The *momentum equation*

$$\frac{\partial \rho v}{\partial t} + \frac{\partial \rho v^2}{\partial x} + \frac{\partial p}{\partial x} + g\rho \frac{dh}{dx} + \frac{\lambda_a}{2D_a} \rho |v| v = 0 \tag{9}$$

relates all forces acting on gas particles to each other. Here, g denotes the constant *acceleration due to gravity*, $\frac{dh}{dx}$ is the *slope of the pipe* and λ_a is the so-called *friction factor*. Investigating the addends on the left-hand side of the momentum equation from left to right, the first term represents the flow rate change over time. The second term is the so-called impact pressure, followed by the pressure gradient. The fourth term represents the impact of gravitational forces, which are influenced by the slope of the pipe. Finally, the last term is the most important one. It represents the friction forces acting on the gas particles due to rough pipe walls. These forces are responsible for the major part of pressure drop within pipes. To calculate the friction factor λ_a, we use the formula of Nikuradse [27]

$$\lambda_a = \left(2 \log_{10} \left(\frac{D_a}{k_a} \right) + 1.138 \right)^{-2}, \tag{10}$$

which is suitable for large Reynolds numbers. Since we want to perform optimization on large transport networks, where we typically have to deal with highly turbulent flows, Nikuradse's formula matches our requirements [25]. Moreover, for the derivation of an algebraic pressure loss equation, we have to introduce the *equation of state* linking gas pressure and density

$$\rho = \frac{\rho_0 z_0 T_0}{p_0} \frac{p}{z(p,T)T}. \tag{11}$$

Here, T_0 and p_0 are the *standard temperature* and *standard pressure*. The *compressibility factor* (also called z-factor) $z(p,T)$ characterizes the deviation of a real gas from ideal gas. The *compressibility factor under standard conditions* is denoted by z_0. The z-factor depends on gas pressure and temperature. Since there is no exact formula to compute the z-factor, we use the so-called Papay Formula

$$z(p,T) = 1 - 3.52 p_r e^{-2.26 T_r} + 0.247 p_r^2 e^{-1.878 T_r} \tag{12}$$

in our model, which is suitable for pressures up to 150 bar [18]. The *relative pressure* $p_r = \frac{p}{p_c}$ and the *relative temperature* $T_r = \frac{T}{T_c}$ are calculated using the *pseudocritical pressure* p_c and the *pseudocritical temperature* T_c. Below the pseudocritical temperature, a gas may be liquefied under pressure. Above the pseudocritical temperature, this is impossible. At the *pseudocritical point* (T_c, p_c) in a pressure-temperature phase diagram, differences between the aggregate phases vanish. Altogether this enables us to derive an algebraic equation from the Momentum Equation (9) that describes pressure loss along a pipe.

$$p_j^2 = \left(p_i^2 - \Lambda_a |q_a| q_a \frac{e^{S_a} - 1}{S_a} \right) e^{-S_a} \quad \forall a \in \mathcal{A}_P. \tag{13}$$

The constants used in Eq. (13) are given by

$$S_a = 2g(h_j - h_i) \frac{\rho_0 z_0 T_0}{p_0 z_m T}, \tag{14}$$

$$\Lambda_a = \left(\frac{4}{\pi}\right)^2 \frac{L_a \lambda_a \rho_0 p_0 z_m T}{D_a^5 z_0 T_0}, \quad \text{and} \tag{15}$$

$$z_m = z\left(\frac{\min\{p_i^-, p_j^-\} + \max\{p_i^+, p_j^+\}}{2}, T \right), \tag{16}$$

for each pipe $a = (i,j) \in \mathcal{A}_P$. For a derivation of Eq. (13), we refer to [2, 19].

A specialized type of pipe with negligible pressure loss are so-called shortcuts. Shortcuts do not exist in reality, but are widely used by practitioners to obtain more descriptive network models. Moreover, they are numerically more stable than using very short pipes and less error prune for the modeler than using pipes with length $L_a = 0$ for that purpose. The explicit introduction of shortcuts also has the major advantage to easily separate between pipes that introduce additional complexity and shortcuts which actually do not affect the complexity. To model a shortcut, we just require the pressure values at its endpoints to be equal

$$p_i = p_j \quad \forall (i,j) \in \mathcal{A}_S. \tag{17}$$

2.3 Resistors

Besides pipes and shortcuts a gas network contains further passive, i.e., non-controllable elements like measuring stations, filtration plants, gas-preheaters and gas-coolers. The only property of these facilities, relevant for our model is that due to their viscous drag, they cause a loss of pressure. In the following, we subsume all these elements under the term resistor.

The set of resistors $\mathcal{A}_R = \mathcal{A}_{R_v} \cup \mathcal{A}_{R_f}$ is further subdivided into the set $\mathcal{A}_{R_v}$ of resistors with variable, i.e., flow dependent pressure drop and the set $\mathcal{A}_{R_f}$ of resistors causing a fixed loss of pressure.

Each resistor with variable pressure loss $a = (i, j) \in \mathcal{A}_{R_v}$ is given together with its *drag factor* ξ_a and *diameter* D_a. According to [18], the pressure loss caused by such a resistor is given by

$$p_i - p_j = \begin{cases} c_a \frac{q_a^2 z(p_i, T_i)}{p_i} & \text{for } q \geq 0 \\ -c_a \frac{q_a^2 z(p_j, T_j)}{p_j} & \text{for } q \leq 0 \end{cases} \quad \forall a = (i, j) \in \mathcal{A}_{R_v}, \tag{18}$$

and with $c_a = \frac{8\rho_0 p_0}{\pi^2 z_0 T_0} \frac{\xi_a T}{D_a^4}$. Similar to the case of pipes, we assume constant temperature $T_a = T$, for $a = (i, j)$, and approximate the z-factors $z(p_i, T_i)$ and $z(p_j, T_j)$ by a mean z-factor z_m as defined in Eq. (16). Since pressure loss depends on the flow direction, we introduce a binary variable $o_a \in \{0, 1\}$ indicating whether there is a flow in the opposite direction of a or not. Thus the variable o_a should be equal to zero, if gas flows in direction of the arc a, i.e., $q_a > 0$, and the variable o_a should be set to one if gas flows in the opposite direction, i.e., $q_a < 0$. To reflect the pressure loss according to (18) we add

$$p_i^2 - p_j^2 + |\Delta_a|\Delta_a = 2c_a z_m |q_a| q_a, \tag{19}$$

$$\Delta_a = p_i - p_j, \tag{20}$$

$$p_i - p_j \leq \left(p_i^+ - p_j^-\right)(1 - o_a), \tag{21}$$

$$p_j - p_i \leq \left(p_j^+ - p_i^-\right) o_a, \tag{22}$$

$$q_a^+(1 - o_a) \geq q_a, \quad \text{and} \tag{23}$$

$$q_a^- o_a \leq q_a \tag{24}$$

to our model. Here, Δ_a can be considered as an auxiliary variable for pressure loss.

For each resistor with constant pressure loss $a = (i, j) \in \mathcal{A}_{R_f}$, pressure loss equals an a priori given constant $\overline{\Delta}_a$, if there is a non-negligible positive flow $q_a \geq q_\varepsilon$ through the resistor. If there is a non-negligible negative flow $q_a \leq -q_\varepsilon$, then pressure loss is equal to $-\overline{\Delta}_a$. For an absolute flow value less than q_ε pressure loss is a linear interpolation between those two values $-\overline{\Delta}_a$ and $\overline{\Delta}_a$. In particular, there is no pressure loss if the flow is zero. The pressure loss equation for resistors with fixed pressure loss is summarized by

$$p_i - p_j = \begin{cases} \overline{\Delta}_a & \text{for } q \geq q_\varepsilon \\ \overline{\Delta}_a + \frac{\overline{\Delta}_a}{q_\varepsilon}(q - q_\varepsilon) & \text{for } q_\varepsilon \geq q \geq -q_\varepsilon \\ -\overline{\Delta}_a & \text{for } q \leq -q_\varepsilon \end{cases} \quad \forall a = (i, j) \in \mathcal{A}_{R_f}. \tag{25}$$

To model the relationship according to (25), in the most general form, i.e., for $q_a^- < -q_\varepsilon$ and $q_a^+ < q_\varepsilon$, we use a univariate piecewise linear function with four breakpoints $(q_a^-, -\Delta_a)$, $(-q_\varepsilon, -\Delta_a)$, $(q_\varepsilon, \Delta_a)$, (q_a^+, Δ_a). The parameter $q_\varepsilon > 0$ indicates an almost zero flow and is typically chosen as 10^{-2}. The piecewise linear function is then modeled using the incremental model, which means to introduce auxiliary continuous variables $\delta_{a,1}$, $\delta_{a,2}$, $\delta_{a,3}$ and binary variables $z_{a,1}$, $z_{a,2}$ and link them with additional constraints

$$q_a = q_a^- + \big(q_\varepsilon - q_a^-\big)\delta_{a,1} + 2q_\varepsilon\delta_{a,2} + \big(q_a^+ - q_\varepsilon\big)\delta_{a,3}, \tag{26}$$

$$\Delta_{ij} = -\Delta_a + 2\Delta_a\delta_{a,2}, \tag{27}$$

$$\delta_{a,2} \leq z_{a,1} \leq \delta_{a,1}, \tag{28}$$

$$\delta_{a,3} \leq z_{a,2} \leq \delta_{a,2}, \tag{29}$$

$$\delta_{a,1} \leq 1, \tag{30}$$

$$\delta_{a,3} \geq 0. \tag{31}$$

2.4 Valves

The simplest type (at least from a discrete point of view) of controllable elements in a gas network is a valve. Valves can either be open or closed. An open valve causes no pressure drop, while a closed valve restricts gas from passing. The pressures at the end nodes of the valve are decoupled in this case.

To model the state of a valve $a = (i, j) \in \mathcal{A}_V$, we introduce a switching variable $s_a \in \{0, 1\}$. The variable s_a is equal to one, if and only if the valve is open. A valve is represented in our model as follows:

$$q_a^+ s_a \geq q_a, \tag{32}$$

$$q_a^- s_a \leq q_a, \tag{33}$$

$$\big(p_j^+ - p_i^-\big)s_a + p_j - p_i \leq p_j^+ - p_i^-, \tag{34}$$

$$\big(p_i^+ - p_j^-\big)s_a + p_i - p_j \leq p_i^+ - p_j^-. \tag{35}$$

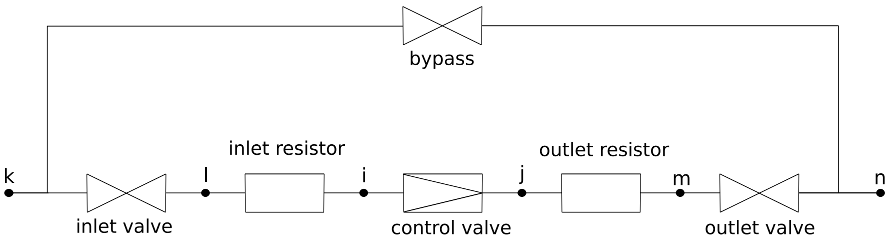


Fig. 2 A control valve station

2.5 Control Valves

For example, when gas is transported from a large conveyor pipeline into a regional subnetwork, it is necessary to reduce pressure manually. For that purpose, control valves are usually located at such transition points.

Just like a valve, a control valve can be either closed or open. When it is closed, gas cannot pass and the pressure values at its incident nodes are decoupled. With an open control valve, pressure can be reduced within a given range $[\Delta_a^-, \Delta_a^+]$. As in case of a valve, we introduce a binary variable $s_a \in \{0, 1\}$ for all control valves $a = (i, j) \in A_{CV}$, to model whether it is open or closed. To reflect the properties of control valves in our model, we add the following constraints:

$$q_a^+ s_a \geq q_a \quad \forall a = (i, j) \in \mathcal{A}_{CV}, \tag{36}$$

$$q_a^- s_a \leq q_a \quad \forall a = (i, j) \in \mathcal{A}_{CV}, \tag{37}$$

$$(p_j^+ - p_i^- + \Delta_a^-) s_a + p_j - p_i \leq p_j^+ - p_i^- \quad \forall a = (i, j) \in \mathcal{A}_{CV}, \tag{38}$$

$$(p_i^+ - p_j^- - \Delta_a^+) s_a + p_i - p_j \leq p_i^+ - p_j^- \quad \forall a = (i, j) \in \mathcal{A}_{CV}. \tag{39}$$

We remark that control valves are unidirectional elements, i.e., $q_a^- \geq 0$ and that it is only possible to reduce pressure in flow direction. Moreover, in real-world networks (and not necessarily in our model) a control valve is always located within a so-called control valve station.

A control valve station $S \in \mathcal{H}$ is a series-parallel subgraph of G that consists of a parallel circuit of a valve (bypass valve) and a series circuit of a valve (inlet valve), a resistor (inlet resistor), a control valve, a resistor (outlet resistor) and a valve (outlet valve). Such a subgraph is depicted in Fig. 2. Pressure drop due to vicious drag of fittings and facilities like, e.g., gas pre-heaters at the inlet and outlet of a control valve is represented by the inlet and outlet resistor. The bypass valve is used to transport gas without pressure reduction in both directions. The inlet and outlet valves are used since each active control valve requires a minimum inlet pressure $p_{S,\text{in}}^-$ and a maximum outlet pressure $p_{S,\text{out}}^+$. Altogether a control valve station has three valid states. It might be active, which means the bypass valve is closed, the inlet and outlet valves and the control valve are open. It might be in bypass, which means that the inlet and outlet valves are closed and the bypass valve is open. Finally, the station might be closed, which means that all three valves are closed. Not all combinations of these states are valid. They are subject to further restrictions. In larger subnetworks, e.g., plants with multiple compressors and valves, only a subset of possible states can actually be used. These further restrictions are discussed in Sect. 2.7.

2.6 Compressors

Since pressure gets lost due to friction, a large gas transport network needs to contain some kind of facilities to compensate for pressure losses. Those facilities are called compressors. In reality, the operating range of a compressor is defined by its characteristic diagram, which is given by a set of curves obtained by (bi)quadratic least squares fits from measured data points. Here, as in [5, 22, 24, 26], we restrict ourselves to a so-called idealized compressor model, where we describe the operating range of a compressor by an approximation of its characteristic diagram in terms of bounds for the compression ratio, pressure increase, and power consumption. The process of compression is considered adiabatic. *Adiabatic compression* means that no heat transfer between the gas and its surroundings takes place [23].

We assume each compressor $a = (i, j) \in \mathcal{A}_{CS}$ to be given together with its *adiabatic efficiency* $\eta_{ad,a}$. According to [26], the *power* P_a a compressor consumes to bring a certain amount of gas to a higher pressure level is given by

$$P_a = c_a T_i z(p_i, T_i)\left(\left(\frac{p_j}{p_i}\right)^{\gamma} - 1\right)q_a \quad \forall a = (i, j) \in \mathcal{A}_{CS}, \tag{40}$$

with $c_a = \frac{\rho_0 R}{\gamma \eta_{ad,a} m}$ and $\gamma = \frac{\kappa - 1}{\kappa}$. Here, the *adiabatic exponent* κ is the ratio of *heat capacity at constant pressure* to *heat capacity at constant volume*, m is the *molecular weight* of gas and R is the *universal gas constant*. Again, we assume constant temperature $T_i = T$, substitute $z(p_i, T_i)$ by an approximation $\tilde{z}_i = z(\frac{p_i^- + p_i^+}{2}, T)$, and add the continuous variables P_a for the compressors' power together with the constraints

$$P_{a,1} = p_j^{\gamma}, \tag{41}$$

$$P_{a,2} = p_i^{-\gamma}, \tag{42}$$

$$P_{a,3} = P_{a,1} P_{a,2}, \tag{43}$$

$$\frac{1}{c_a T \tilde{z}_i} P_a = q_a (P_{a,3} - 1), \tag{44}$$

for all $a = (i, j) \in \mathcal{A}_{CS}$ to our model. The auxiliary continuous variables $P_{a,1}$, $P_{a,2}$, $P_{a,3}$ are here used to express the trivariate power function by a recursive composition of univariate and bivariate nonlinear subexpressions. Just like a control valve, a compressor may be either closed or open. If it is closed, no gas flows through the compressor and the pressures at its inlet and outlet are decoupled. Obviously, a closed compressor cannot increase pressure and therefore consumes no power. Whenever a compressor is open, it consumes power while it increases the pressure of gas flowing in arc direction, i.e., $p_j \geq p_i$ and $q_a \geq 0$. Flow against the direction of a compressor arc is impossible.

In order to model the different states of a compressor, we introduce a binary variable s_a for each compressor $a \in \mathcal{A}_{CS}$, where $s_a = 1$ means that a is open and $s_a = 0$ means that a is closed. Besides the set of equations for the power consumption (41)–(44), the following constraints are added to our model to describe the approximated characteristic diagram together with the combinatorial aspects of a compressor:

$$q_a \geq s_a q_a^- \quad \forall a \in \mathcal{A}_{CS}, \tag{45}$$

$$q_a \leq s_a q_a^+ \quad \forall a \in \mathcal{A}_{CS}, \tag{46}$$

$$P_a \geq s_a P_a^- \quad \forall a \in \mathcal{A}_{CS}, \tag{47}$$

$$P_a \leq s_a P_a^+ \quad \forall a \in \mathcal{A}_{CS}, \tag{48}$$

$$p_j - p_i \geq \Delta_a^- s_a + \left(p_j^- - p_i^+\right)(1 - s_a) \quad \forall a = (i, j) \in \mathcal{A}_{CS}, \tag{49}$$

$$p_j - p_i \leq \Delta_a^+ s_a + \left(p_j^+ - p_i^-\right)(1 - s_a) \quad \forall a = (i, j) \in \mathcal{A}_{CS}, \tag{50}$$

$$p_j \geq r_a^- p_i - (1 - s_a)\left(r_a^- p_i^+ + p_j^-\right) \quad \forall a = (i, j) \in \mathcal{A}_{CS}, \tag{51}$$

$$p_j \leq r_a^+ p_i - (1 - s_a)\left(r_a^+ p_i^- - p_j^+\right) \quad \forall a = (i, j) \in \mathcal{A}_{CS}. \tag{52}$$

Here, P_a^- and P_a^+ are bounds for the power consumption of an active compressor. The minimum and maximum pressure increases are denoted by Δ_a^- and Δ_a^+. Finally, the compression ratio is bounded by $r_a^- \leq \frac{p_j}{p_i} \leq r_a^+$, whenever the compressor is running. Similar to control valves, a compressor is not a stand-alone element. Typically, a couple of compressors, together with a bypass valve allowing backward flow, form a so-called compressor station.

Likewise a control valve station, a compressor station $S = (\mathcal{V}(S), \mathcal{A}(S)) \in \mathcal{H}_{CS}$ is a series-parallel subgraph of G. It consists of a parallel circuit of a valve (bypass valve) and a series circuit of a valve (inlet valve), a resistor (inlet resistor), a parallel circuit of one or more configurations, a resistor (outlet resistor), and a valve (outlet valve). A configuration again is a series composition of a valve, one or more compressor stages, and a valve. Finally, a compressor stage is a parallel composition of one or more compressors. An example of a compressor station with two configurations is given in Fig. 3. One configuration consists of a single stage with two parallel compressors, while the second configuration has two stages, with a single compressor unit in each of them.

Also, exactly as in case of a control valve station, each compressor station $S \in \mathcal{H}_{CS}$ is given together with a minimum inlet pressure $p_{S,\text{in}}^-$ and a maximum outlet pressure $p_{S,\text{out}}^+$. Whenever any configuration in $\mathcal{C}_S$ is active, pressure at the tail node of the inlet resistor must not fall below $p_{S,\text{in}}^-$ and pressure at the head node of the outlet resistor must not exceed $p_{S,\text{out}}^+$. Again, we do not enforce these restric-

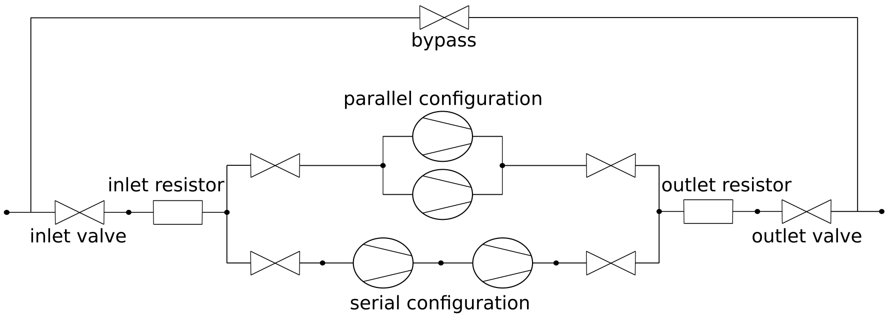


Fig. 3 A compressor station with two configurations

tions by additional constraints, but rather use them to strengthen the corresponding variable bounds. This is possible due to the existence of the inlet and outlet valves.

A compressor station is either closed, in bypass mode, or exactly one of its configurations is active. In turn, a compressor is active if and only if the configuration it is contained in is active. These decisions are again subject to further restrictions, which are detailed in the next section.

2.7 Combinatorics of Subnetwork Operation Modes

We have seen that with compressor and control valve stations, there are subgraphs of gas networks forming kind of functional units. In addition, a number of compressor (stations), control valve (stations) or valves may form so-called decision groups. Roughly speaking, a decision group can be considered as a set of network elements sharing the property that each single element can be opened and closed, but not all combinations of open and closed elements are possible. These restrictions usually are given by practitioners to avoid solutions which are not practically realizable due to technical restrictions or which are not reasonable according to the current contract situation. A third reason for introducing a decision group is symmetry breaking, like in the trivial example with two parallel valves. These two valves can be in four different switching states, but from a physical point of view it makes no difference whether one valve is open and the other one is closed or whether both valves are open. Thus, it would be reasonable to introduce a decision group containing both valves, where only two switching states are allowed, namely that both valves are closed and for example both valves are open.

Formally, we denote the set of decision groups by $\mathcal{G}$ and every decision group consists of a number of control valves, compressors and valves, together with a set of admissible switching states, i.e., each decision group $D = (\mathbf{n}, B) \in \mathcal{G}$, is a tuple, where $\mathbf{n} = (n_1, \dots, n_{k_D})$ with $n_i \in \mathcal{A}_{CV} \cup \mathcal{A}_{CS} \cup \mathcal{A}_V$, for $i = 1, \dots, k_D$, is a vector containing the respective network elements and $B = (\mathbf{b}_1, \dots, \mathbf{b}_{l_D}) \subset \{0, 1\}^{k_D}$ is a sequence of binary decision vectors describing the possible switching states of the elements in $\mathbf{n}$. We further introduce the index sets $N_0^D(j) := \{i : b_{ij} = 0\}$ containing the indices i of binary decision vectors, where n_j has to be closed if decision i is chosen and the set $N_1^D(j) := \{i : b_{ij} = 1\}$ contains the indices i of binary decision vectors, where n_j has to be open, if decision i is chosen for $i = 1, \ldots, l_D$ and $j = 1, \ldots, k_D$. Here, b_{ij} denotes the j-th component of $\mathbf{b}_i$. To model restrictions on the decisions, we introduce binary variables $b_i^D \in \{0, 1\}$ for each $D \in \mathcal{G}$ and $i = 1, \ldots, l_D$ and extend our model by the following constraints:

$$s_{n_j} \le \sum_{i \in N_1^D(j)} b_i^D \quad \forall D \in \mathcal{G}, j \in \mathcal{A}_V^D \cup \mathcal{A}_{CV}^D \cup \mathcal{A}_{CS}^D, \tag{53}$$

$$(1 - s_{n_j}) \le \sum_{i \in N_0^D(j)} b_i^D \quad \forall D \in \mathcal{G}, j \in \mathcal{A}_V^D \cup \mathcal{A}_{CV}^D \cup \mathcal{A}_{CS}^D. \tag{54}$$

Here, $A_V^D := \{1, \ldots, k_D\} \cap \{k : n_k \in \mathcal{A}_V\}$ is the set of valves belonging to decision group D. The set $\mathcal{A}_{CV}^D := \{1, \ldots, k_D\} \cap \{k : n_k \in \mathcal{A}_{CV}\}$ contains all control valves belonging to decision group D and analogously $\mathcal{A}_{CS}^D := \{1, \ldots, k_D\} \cap \{k : n_k \in \mathcal{A}_{CS}\}$ is the set of all compressors of decision group D for all $D \in \mathcal{G}$. Constraints (53) state that each valve, control valve, or compressor n_j can only be open if a decision is chosen in which it is allowed to be open. Inequalities (54) cause that a valve, control valve, or compressor can only be closed if a decision is chosen, where it is allowed to be closed. Finally, exactly one decision has to be chosen per decision group,

$$\sum_{i=1}^{l_D} b_i^D = 1 \quad \forall D \in \mathcal{G}. \tag{55}$$

With the aid of these groups it is then also possible to enforce the valid states of control valve stations, namely to be closed, active, or in bypass. Therefore we just introduce a single decision group $D_S = (\mathbf{n}_S, B_S)$ for each such station $S \in \mathcal{H}_{CV}$, where $\mathbf{n}_S$ consists of the bypass valve, the inlet and outlet valve, and the control valves of the station. The only allowable binary states of these four elements are then $B_S = \{(0, 0, 0, 0), (0, 1, 1, 1), (1, 0, 0, 0)\}$. The first binary vector reflects the closed state of the station, the second one the active state, and the third one represents the bypass mode of the station.

Similarly, the valid states of a compressor station graph $S \in \mathcal{H}_{CS}$ are modeled by a decision group $D_S = (\mathbf{n}_S, B_S)$. Recall that a compressor station might be closed, in bypass, or exactly one of its configurations is active. If a configuration is active, all compressors contained in that configuration are active and the inlet and outlet valves of the configuration are open. In the following, we denote the number of configurations of a compressor station by m_S. To reflect these states the vector $\mathbf{n}_S$

of the decision group consists of the bypass valve, the inlet and outlet valves of the station, and the inlet and outlet valves and compressors of each configuration. The set of admissible binary vectors is then given by

$$\begin{aligned} B_S = \big\{ & (0,0,0,\dots,0), \\ & (1,0,0,\underbrace{0,\dots,0}_{\text{config } 1},\dots,\underbrace{0,\dots,0}_{\text{config } m_S}), \\ & (0,1,1,\underbrace{1,\dots,1}_{\text{config } 1},\dots,\underbrace{0,\dots,0}_{\text{config } m_S}), \\ & \dots, \\ & (0,1,1,\underbrace{0,\dots,0}_{\text{config } 1},\dots,\underbrace{1,\dots,1}_{\text{config } m_S})\big\}, \end{aligned}$$

representing the states that the station is closed, in bypass mode, or exactly one of its configuration is active.

2.8 Objective Function

We would like to find a control, which is optimal in the sense of practical interest. Practitioners are interested in finding an operating cost minimal control. Operating costs arise due to energy costs caused by active compressors. Thus we assume that for each compressor $a \in \mathcal{A}_{CS}$ an *energy cost coefficient* $c_a \geq 0$ is given. The most obvious form is to just use $c_a = 1$ and minimize the overall power consumption of the compressors. In general, energy costs may vary from compressor to compressor due to different types of drives. For example, so-called turbo compressors are usually driven by gas turbines, whereas piston compressors are typically shipped with electric or gas driven motors. With such given energy cost coefficients the objective is to minimize the overall energy costs:

$$\begin{aligned} & \min \sum_{a \in \mathcal{A}_{CS}} c_a P_a \\ & \text{subject to} \\ & \quad (\mathbf{P}, \mathbf{p}, \mathbf{q}, \mathbf{d}, \mathbf{s}, \mathbf{o}, \mathbf{b}) \text{ satisfies} \\ & \qquad (1)\text{–}(4), (13), (17), (19)\text{–}(24), (26)\text{–}(39), (41)\text{–}(55). \end{aligned} \tag{P}$$

For the sake of clarity we explicitly remark that due to the finiteness of all variables bounds the Gas Transport Energy Cost Minimization Problem (P) is bounded by definition.

3 Basic MIP-Relaxation

In this section, we will introduce a framework for the construction of MIP-relaxations of the MINLP-model for the problem introduced in Sect. 2. The basis of our discussion is the approach described in [8], which we review here for completeness.

We will first construct piecewise linear approximations for each nonlinearity present in the system of constraints of Problem (P). In a second step we will show how to construct an MIP-model from such an approximation which constitutes a proper relaxation of the underlying MINLP. In addition, we will explain how to make these MIP-relaxations comply to any a priori given bound $\varepsilon > 0$ on the approximation error.

As starting point for our considerations, we assume the following situation: Let $\mathcal{D} \subseteq \mathbb{R}^d$ and $f : \mathcal{D} \to \mathbb{R}$ be some continuous function. Further, let $\phi : \mathcal{P} \to \mathbb{R}$ be a piecewise linear approximation of f over some convex polytope $\mathcal{P} \subseteq \mathcal{D}$. We assume ϕ to interpolate f on the vertices of some triangulation of $\mathcal{P}$. Thus, for a triangulation with simplex set $\mathcal{S}$, we can define affine functions $\phi_i : \mathbf{x} \to \mathbf{a}_i^T \mathbf{x} + \mathbf{b}_i$ for each simplex $S_i \in \mathcal{S}$ with $\phi_i(\mathbf{x}) = f(\mathbf{x})$ for every vertex $\mathbf{x}$ of S_i such that we can write $\phi(\mathbf{x}) = \phi_i(\mathbf{x})$ for $\mathbf{x} \in S_i$. To construct such an approximation satisfying some a priori given error bound, we suggest Algorithm 1.

It is easy to see that Algorithm 1 always terminates with a triangulation $\mathcal{S}$ that corresponds to a piecewise linear function ϕ, that interpolates f on the vertices of $\mathcal{S}$. Since the projection of $\mathcal{S}$ onto the space of the $\mathbf{x}$-variables yields a triangulation of $\mathcal{P}$ and since the error bound ε is satisfied for any $S \in \mathcal{S}$, the function ϕ approximates f such that the approximation error is less than or equal to ε for every $\mathbf{x} \in \mathcal{P}$.

So once we are able to control the approximation error within a simplex, we are able to control the overall approximation error. What remains unclear is how to compute $\max_{\mathbf{x} \in S_i} |f(\mathbf{x}) - \phi_i(\mathbf{x})|$ together with a point, where the maximum error is attained. For explaining how this can be done, we denote the maximum interpolation error of a function f over a simplex S by $\varepsilon(f, S) = \max\{\varepsilon_u(f, S), \varepsilon_o(f, S)\}$, where $\varepsilon_u(f, S) = \max\{f(\mathbf{x}) - \phi(\mathbf{x}) : \mathbf{x} \in S\}$ and $\varepsilon_o(f, S) = \max\{\phi(\mathbf{x}) - f(\mathbf{x}) : \mathbf{x} \in S\}$ are the maximum underestimation and the maximum overestimation of f by its linear interpolation ϕ at the vertices of S.

Since the maximum overestimation $\varepsilon_o(f, S)$ of f over S with respect to ϕ is equal to the maximum underestimation $\varepsilon_u(-f, S)$ of $-f$ over S with respect to $-\phi$, we only show how to compute the maximum overestimation in the remainder of this section. To this end, we introduce the notion of a convex underestimating function.

Definition 1 The elements of $\mathcal{U}(f, S) := \{\mu : S \to \mathbb{R} : \mu \text{ convex and } \mu(\mathbf{x}) \leq f(\mathbf{x}), \forall \mathbf{x} \in S\}$ are called *convex underestimators* of f over S and the function $\text{vex}_S[f] : S \to \mathbb{R}$, defined as $\text{vex}_S[f](\mathbf{x}) := \sup\{\mu(\mathbf{x}) : \mu \in \mathcal{U}(f, S)\}$ is called the *convex envelope* of f over S.

Obviously, $\text{vex}_S[f]$ is the tightest convex underestimator of f over S. According to Theorem 1 [6, 8], the maximum overestimation $\varepsilon_o(f, S)$ can be computed

Algorithm 1: Adaptive piecewise linear interpolation

```
Input  : A convex polytope $\mathcal{P} \subset \mathcal{D} \subseteq \mathbb{R}^d$, a continuous function
         $f : \mathcal{D} \subset \mathbb{R}^d \to \mathbb{R}$, and an upper bound $\varepsilon > 0$ for the approximation
         error.
Output: A triangulation $\mathcal{S}$ of $\mathcal{P}$ corresponding to a piecewise linear
         interpolation $\phi$ of $f$ over $\mathcal{P}$ with $\phi(\mathbf{x}) = f(\mathbf{x})$ for all vertices $\mathbf{x}$ of
         simplices in $\mathcal{S}$ and $\phi = \phi_i$ for $S_i \in \mathcal{S}$ and $i = 1, \dots, n$.

Set $\mathcal{V} = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{x}$ is a vertex of $\mathcal{P}\}$;
Construct an initial triangulation $\mathcal{S}$ of $\mathcal{V}$ and the corresponding piecewise
linear interpolation $\phi$ of $f$ with $\phi(\mathbf{x}) = \phi_i(\mathbf{x})$ for $\mathbf{x} \in S_i$ for all $S_i \in \mathcal{S}$;
while $\exists S_i \in \mathcal{S}$, $S_i$ unmarked do
    if $\max_{\mathbf{x} \in S_i} |f(\mathbf{x}) - \phi_i(\mathbf{x})| > \varepsilon$ then
        Add a point where the maximum error is attained to $\mathcal{V}$;
        Set $\mathcal{S} \leftarrow \mathcal{S} \setminus \{S_i\}$;
        Update $\mathcal{S}$ w.r.t. $\mathcal{V}$;
    else
        Mark $S_i$;
    end
end
return $\mathcal{S}$
```

by solving a convex optimization problem in d variables once $\text{vex}_S[f]$ is known. Again, from [6] we known that this is indeed the case for all nonlinearities occurring in the MINLP-model from Sect. 2.

Theorem 1 *Let $\mathcal{M}_o := \{\mathbf{x} \in S : \phi(\mathbf{x}) - f(\mathbf{x}) = \varepsilon_o(f, S)\}$ be the set of global maximizers for the overestimation of f by ϕ over S and let $\mathcal{N}_o := \{\mathbf{x} \in S : \phi(\mathbf{x}) - \text{vex}_S[f](\mathbf{x}) = \varepsilon_o(\text{vex}_S[f], S)\}$ be the set of global maximizers for the overestimation of the convex envelope of f by ϕ over S. Then we get $\varepsilon_o(f, S) = \varepsilon_o(\text{vex}_S[f], S)$ and $\mathcal{N}_o = \text{conv}(\mathcal{M}_o)$.*

It has also been shown in [6, 8] that a point, where the maximum overestimation is attained can be computed by solving at most d convex optimization problems in dimension less than or equal to d.

So once Algorithm 1 has been terminated with a triangulation $\mathcal{S} = \{S_1, \dots, S_n\}$, we are ready to replace each nonlinear expression $y = f(\mathbf{x})$ from (P) by the mixed-integer linear constraint set (56)–(62) that is a slightly modified version of the well-known incremental model for piecewise linear functions [6, 8, 20].

$$\mathbf{x} = \bar{\mathbf{x}}_0^{S_1} + \sum_{i=1}^{n} \sum_{j=1}^{d} \left(\bar{\mathbf{x}}_j^{S_i} - \bar{\mathbf{x}}_0^{S_i}\right) \delta_j^{S_i}, \tag{56}$$

$$y = \bar{y}_0^{S_i} + \sum_{i=1}^{n} \sum_{j=1}^{d} \left(\bar{y}_j^{S_i} - \bar{y}_0^{S_i}\right) \delta_j + e, \tag{57}$$

$$\sum_{j=1}^{d} \delta_j^{S_i} \le 1, \qquad \text{for } i = 1, \dots, n, \tag{58}$$

$$\sum_{j=1}^{d} \delta_j^{S_{i+1}} \le z_i, \qquad \text{for } i = 1, \dots, n-1, \tag{59}$$

$$z_i \le \delta_d^{S_i}, \qquad \text{for } i = 1, \dots, n-1, \tag{60}$$

$$\delta_j^{S_i} \ge 0, \qquad \text{for } i = 1, \dots, n \text{ and } j = 1, \dots, d, \tag{61}$$

$$z_i \in \{0, 1\}, \qquad \text{for } i = 1, \dots, n-1. \tag{62}$$

The main difference to the traditional incremental model lies in the fact that an extra variable e is added to the right-hand side of Eq. (57). This additional variable is intended to express the approximation errors, which can be achieved by adding the inequalities

$$\varepsilon_u(f, S_1) + \sum_{i=1}^{n-1} z_i \big(\varepsilon_u(f, S_{i+1}) - \varepsilon_u(f, S_i)\big) \ge e, \tag{63}$$

$$-\varepsilon_o(f, S_1) - \sum_{i=1}^{n-1} z_i \big(\varepsilon_o(f, S_{i+1}) - \varepsilon_o(f, S_i)\big) \le e. \tag{64}$$

Note that for every feasible solution to the constraints (56)–(64) there is some index j with $z_i = 1$ for all $i < j$ and $z_i = 0$ for all $i \ge j$. This means that all terms $\varepsilon_u(f, S_i)$ on the left-hand side of (63) and all terms $\varepsilon_o(f, S_i)$ on the left-hand side of (64) with $i \ne j$ either cancel out or are multiplied by 0. Therefore, we get $-\varepsilon_o(f, S_j) \le e \le \varepsilon_u(f, S_j)$ as desired.

The projection of the polytope described by (56)–(64) onto the $(\mathbf{x}, y)$-hyperplane can be interpreted as a piecewise polyhedral envelope of the graph of f. An illustrative example for the case of the expression $c|q|q$ appearing in the pressure loss equations for pipes is given in Fig. 4. The feasible set of the MIP-relaxation is given by the union of the parallelograms depicted with dotted lines.

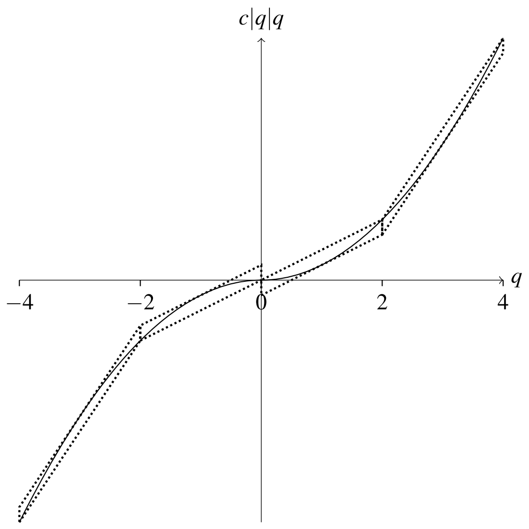


Fig. 4 MIP-relaxation of $c|q|q$ with $c = 0.25$ on $[-4, 4]$ with 5 breakpoints

4 Adaptive Refinement of the Relaxation

In order to solve Problem (P) our idea is to first construct some initial MIP-relaxation as described in Sect. 3 which satisfies some a-priori given (potentially coarse) error bound. After that, we compute optimal solutions to a sequence $(\Pi^i)_{i\in\mathbb{Z}_+}$ of successively refined MIP-relaxations until all nonlinear constraints are satisfied up to the desired accuracy $\varepsilon > 0$. These steps are summarized in Algorithm 2.

Since Π^i is a relaxation of P, Algorithm 2 never detects infeasibility erroneously. Moreover, for $\varepsilon \to 0$, Algorithm 2 converges to a global optimum of P if one exists [6, 7].

The detailed steps necessary to perform the local refinement of the approximations depend on the chosen triangulation algorithm. For the computations presented in Sect. 5 we applied an incremental algorithm for Delaunay triangulations [31]. The refinement step in iteration i is performed such that all simplices S with a distance $\min\{\|\mathbf{x}^i_{opt} - y\|_2 : y \in S\} < \delta$ are refined by adding an element of $\arg\max\{|f(\mathbf{x}) - \phi_S(\mathbf{x})| : x \in S\}$ to the set of breakpoints, i.e., to the vertices of the triangulation. Here ϕ_s denotes the linear interpolation of f on the vertices of S and $\delta > 0$. Fur further details we again refer to [6, 7].

In addition, we try to further enhance the performance of Algorithm 2 by applying a heuristic procedure to obtain high quality feasible solutions as early as possible. To this end, we set up an auxiliary NLP-model every time the upper bound of the current MIP-relaxation improves. The NLP-model is constructed from the underlying MINLP by fixing the integer variables to their respective values in the new incumbent solution of the current MIP-relaxation. This auxiliary NLP is thereafter solved with an algorithm for convex nonlinear programming. Thus, whenever the NLP-solver converges to a feasible point which has an objective value that is lower

Algorithm 2: Adaptive refinement of MIP-relaxations

```
Input  : An instance $P$ of Problem (P) and an upper $\varepsilon > 0$ on the absolute
         constraint violation.
Output: If there exists a point which satisfies all constraints of Problem (P) up
         to $\varepsilon$, an optimal solution to Problem (P) is returned. Otherwise
         infeasibility is reported.
1  Compute an initial MIP-relaxation $\Pi^0$ of $P$;
2  Solve $\Pi^0$;
3  Set $i \leftarrow 0$;
4  while Problem $\Pi^i$ has an optimal solution $\mathbf{x}^i_{opt}$ with finite objective value do
5      if $\mathbf{x}^i_{opt}$ satisfies all constraints of Problem $P$ up to $\varepsilon$ then
6          return $\mathbf{x}^i_{opt}$;
7      else
8          Find the set $\mathcal{F}$ of nonlinear constraints which are violated by $\mathbf{x}^i_{opt}$ by
           more than $\varepsilon$;
9          Construct $\Pi^{i+1}$ by refining the approximations of $f \in \mathcal{F}$ locally
           around $\mathbf{x}^i_{opt}$;
10     end
11     $i \leftarrow i + 1$;
12 end
13 return "P is infeasible";
```

than the one of the best feasible solution found so far, a new incumbent solution for the MINLP has been found.

Finally Algorithm 2 is preceded by a bound strengthening procedure, where the bounds of the flow- and pressure variables are propagated according to the flow conservation constraints (3) and according to the natural interval extension of the nonlinear pressure loss equations (13), (18), (25). Further details on these preprocessing steps are again given in [6].

5 Computational Results

In this section we describe the computational results obtained for three test instances of the gas transport energy cost minimization problem. We investigate one rather small network, a medium-sized network, and one large real-world instance. An overview of the sizes of the networks is given in Table 3. Schematic plots of the network topologies are given in Figs. 5, 6, 7, where entries are depicted as discs and exits are drawn as circles.

The first test network is given in Fig. 5. It consists of thirteen pipes and three compressor stations CS1, CS2, and CS3. Altogether 1,348,000 m^3/h of gas have to

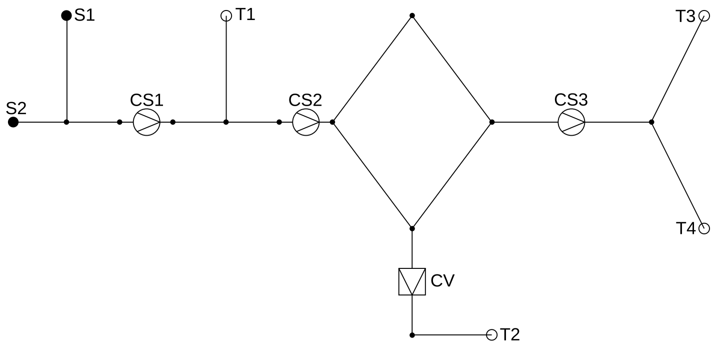


Fig. 5 The first test network

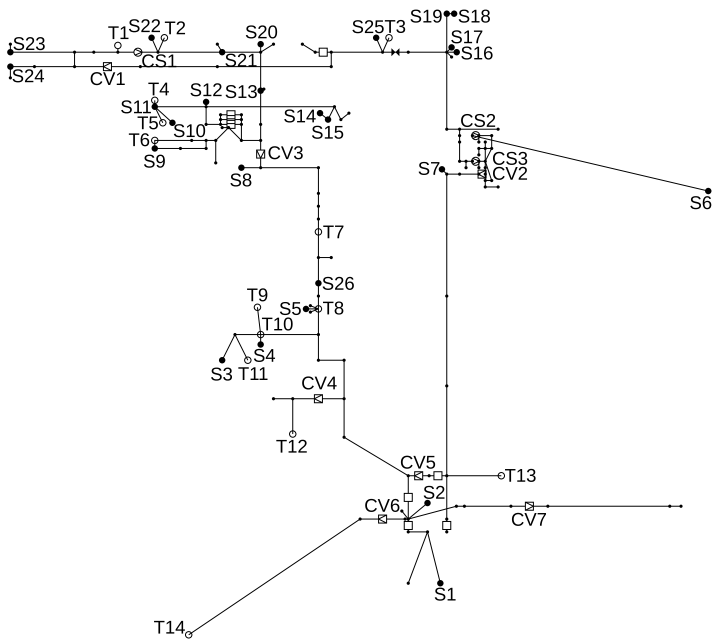


Fig. 6 The second test network

be transported from the two entries S1 and S2 located to the left to the four customers represented by the exits T1, T2, T3, and T4. In front of exit T3, a control valve station CV is located in order to reduce pressure if necessary. Compressor station CS1 can be operated in three different configurations, while both other stations CS2 and CS3 only have a single configuration. Gas is at a pressure level of at most 70 bar at the entries, while it has to be delivered to the exits T1 and T2 with at least 62 bar and to T3 and T4 with at least 70 bar. Pressure at exit T2 must also not exceed 63 bar.

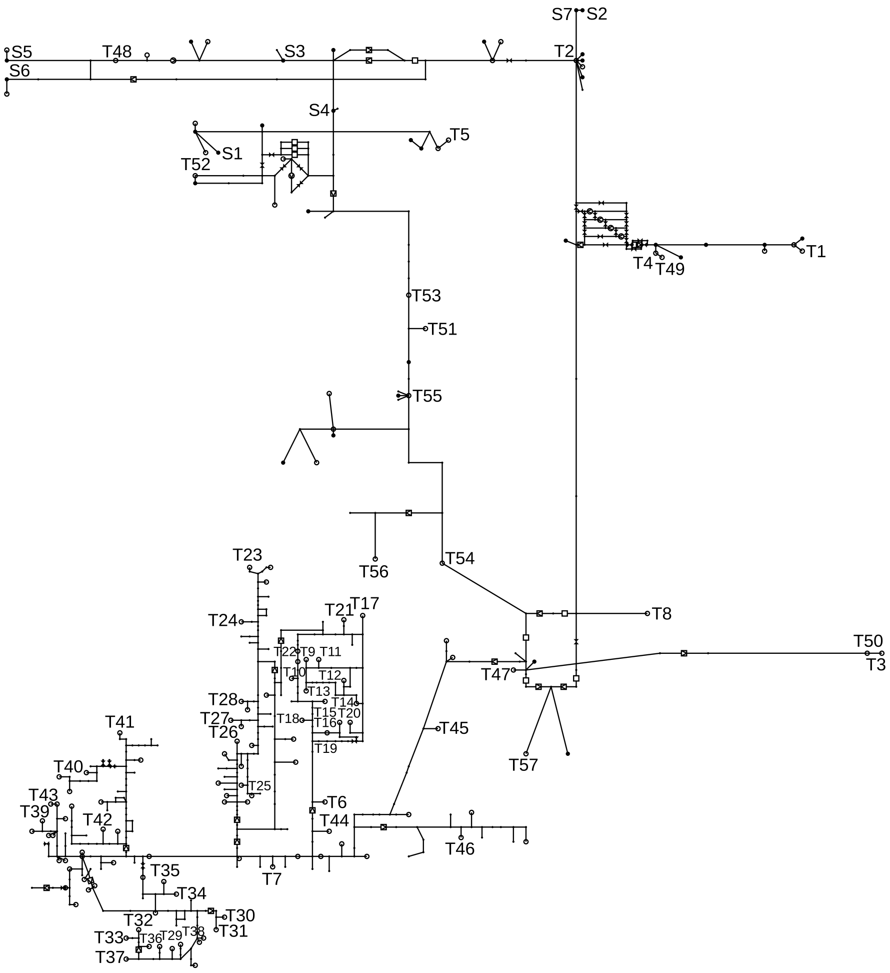


Fig. 7 The third test network

Our second test network is made up of the most important transport pipelines of Open Grid Europe's northern German network for the transport of high caloric natural gas. Besides some minor simplifications, only a couple of regional subnets have been cut off from the real network in order to create this test instance (cf. Fig. 7). With 69 pipes, 67 shortcuts, 8 resistors, 3 compressor stations and 7 control valve stations it already has a considerable degree of complexity. Gas is fed into the network at 26 entries and customers withdraw gas from the network at 14 exits.

Compressor station CS1 can be operated in 14 different configurations. For the compressor stations CS2 and CS3 on the right-hand side of Fig. 6 we have to choose between two distinct configurations. The nomination considered for the second test network is given in Table 1. All nodes for which no data is provided in the table nei-

Table 1 The nomination considered for the second test network

v	p^-	p^+	d	v	p^-	p^+	d
T1	43	56	8056	T12	16	69	35964
T2	2	85	96698	T13	67	80	157166
T3	2	85	241509	T14	1	121	338487
T4	75	82	529532	S1	66	79	−232710
T5	72	85	141345	S6	1	122	−396757
T6	75	82	1070961	S14	2	90	−141976
T7	2	85	19772	S15	2	90	−257181
T8	41	85	52015	S18	2	91	−100031
T9	65	78	15317	S19	78	91	−1530962
T10	65	78	137653	S23	2	82	−275755
T11	71	85	418477	S24	44	57	−327580

ther feed in nor withdraw gas in the considered nomination. In Table 1 the columns labeled p^- and p^+ contain the lower and upper bounds for the pressure values at the respective nodes in bar. The columns labeled with d contain the demands of the nodes in terms of m^3/h.

Our third test network is given in Fig. 7. It is the complete northern German network for the transport of high caloric gas of the Open Grid Europe GmbH. It consists of 452 pipes, 98 shortcuts, 9 resistors, 23 control valve stations, 6 compressor stations and 34 valves. Gas is fed into the network at 31 entries and customers have the possibility to withdraw gas at 129 exits. The overall length of the pipelines is approximately 1,241 km. The nomination considered for this network is given in Table 2. The basic properties of the three test networks are summarized in Table 3.

To solve these problems we employ Algorithm 2 from Sect. 4 and compare our results with those obtained from the state-of-the-art general purpose MINLP-solvers Baron 10.2.0 [32] and SCIP 2.1.1 [1, 21, 33].

All computations presented in this section are carried out using at most 4 cores of a machine with two six core AMD Opteron™2435 processors with 64 GB of main memory. The installed Linux operating system is Debian Squeeze with kernel version 2.6.32-5-amd64. As MIP-solver within Algorithm 2 we use Gurobi 5.0 [14], where all parameters are set to their default values. The occurring NLP-problems are solved with Ipopt 3.10 [34, 35] with ma27 [15] as linear solver. The number of Ipopt iterations is limited to 300. We compare our results to those obtained from the MINLP-solvers Baron 10.2.0 [32] and SCIP 2.1.1 [1, 21, 33].

For all computations, we employ a time limit of 2 hours. In Algorithm 2, we choose 10.0 bar as upper bounds for the initial linearization errors for the pressure loss equations and 10 MW for the equations describing the power consumption of the compressors. The parameter controlling the refinement step is set to $\delta = 10^{-6}$. The maximal constraint violation is set to $\varepsilon = 10^{-4}$.

The results obtained with SCIP, Baron, and Algorithm 2 for the three test instances are given in Table 4. The columns labeled $\text{lb}(x)$ and $\text{ub}(x)$ contain the best

Table 2 The nomination considered for the third test network

v	p^-	p^+	d	v	p^-	p^+	d
T1	1	122	149935	T33	2	8	2325
T2	2	86	156454	T34	11	41	116
T3	41	68	108800	T35	2	41	652
T4	2	86	1213451	T36	4	41	2634
T5	2	86	138000	T37	2	41	36
T6	3	8	15525	T38	2	41	107
T7	2	86	284	T39	2	695	3322
T8	2	86	47	T40	9	51	561
T9	2	8	2470	T41	26	51	19678
T10	5	9	2624	T42	16	51	25
T11	5	9	3890	T43	36	69	9608
T12	2	86	1223	T44	2	69	11031
T13	3	9	177	T45	2	69	448
T14	5	9	177	T46	2	51	8200
T15	2	9	102	T47	2	85	87347
T16	5	9	324	T48	2	71	33483
T17	5	9	8120	T49	51	74	6775
T18	5	9	33	T50	41	68	68713
T19	5	9	1425	T51	41	85	5823
T20	5	9	3790	T52	74	81	7362
T21	5	9	2739	T53	2	85	14352
T22	5	9	3035	T54	41	85	165022
T23	2	5	907	T55	41	85	28650
T24	2	5	71	T56	16	69	14385
T25	2	5	2090	T57	2	86	1259051
T26	2	5	215	S1	2	86	−46268
T27	2	5	233	S2	2	86	−1213451
T28	2	5	629	S3	2	86	−1347
T29	2	41	3340	S4	2	86	−3281
T30	6	41	1149	S5	2	50	−204090
T31	6	41	2193	S6	2	50	−489212
T32	2	8	1012	S7	2	84	−1616521

Table 3 Numbers of network elements and overall pipe length of the three test networks

Net	$\|\mathcal{V}_S\|$	$\|\mathcal{V}_D\|$	$\|\mathcal{V}_I\|$	$\|\mathcal{A}_P\|$	$\|\mathcal{A}_S\|$	$\|\mathcal{A}_{CS}\|$	$\|\mathcal{A}_{CV}\|$	$\|\mathcal{A}_V\|$	$\|\mathcal{A}_R\|$	$\sum_{a\in\mathcal{A}_P} L_a$
1	2	4	11	13	1	3	1	0	0	620 km
2	3	5	16	20	4	3	1	0	1	820 km
3	31	129	432	452	98	6	23	34	9	1241 km

Table 4 Results obtained with SCIP, Baron and Algorithm 2 after 2 hours

Net	lb(SCIP)	lb(Baron)	lb(MIP)	ub(SCIP)	ub(Baron)	ub(MIP)	gp(SCIP)	gp(Baron)	gp(MIP)
1	456.23	467.85	462.45	467.85	467.85	467.85	2.55 %	0.00 %	1.17 %
2	1484.10	1259.02	1529.78	1571.16	1610.67	1568.68	5.87 %	27.93 %	2.54 %
3	0.00	0.00	453.79	–	–	517.96	∞	∞	14.10 %

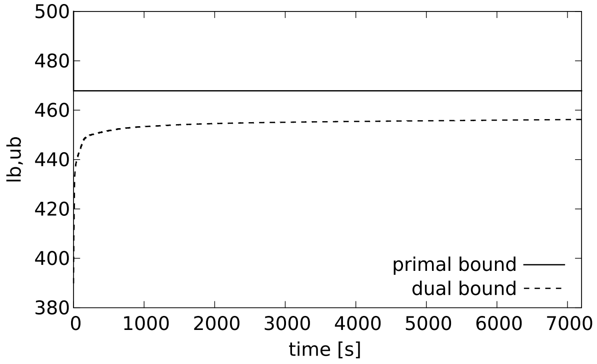


Fig. 8 Bounds for the 1st net (SCIP)

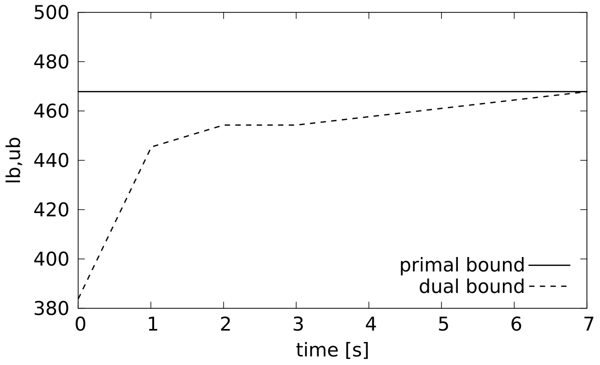


Fig. 9 Bounds for the 1st net (Baron)

lower bound and the best feasible solution obtained from solver x after 7200 seconds. The entries of the last three columns are the relative optimality gaps, which have been computed as $(\text{ub} - \text{lb})/\text{lb}$.

For the small test network Baron was able to prove optimality of a solution with objective value 467.85 within 7 s. Although all three approaches find the optimal solution within a second (cf. Figs. 8, 9, 10), SCIP and our new approach are not able to prove optimality within the time limit. However, the lower bound obtained from Algorithm 2 is slightly better than the lower bound obtained from SCIP. We believe

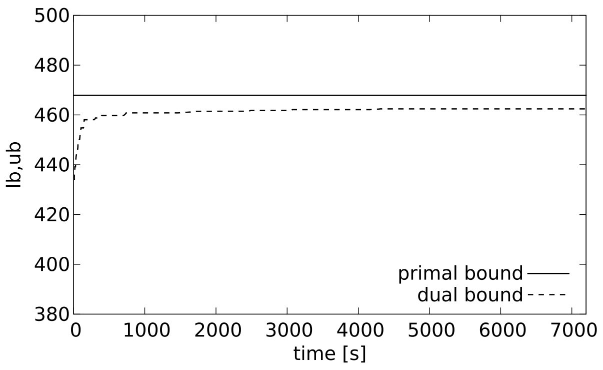


Fig. 10 Bounds for the 1st net (Algorithm 2)

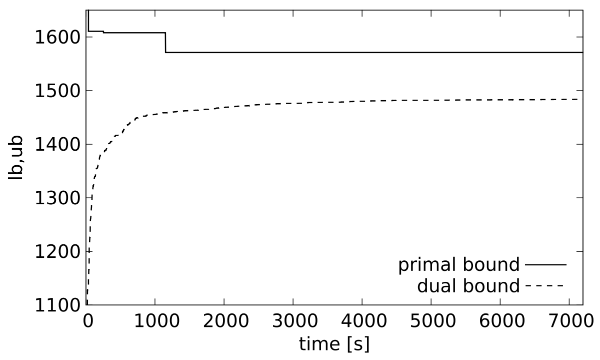


Fig. 11 Bounds for the 2nd net (SCIP)

that the very short running time of Baron in this case results from the application of a variety of bound strengthening techniques, which are not yet implemented in our new algorithm. For the two larger test networks none of the three algorithms was able to prove optimality of a solution but the best lower and upper bounds are obtained with Algorithm 2. For the large real-world instance neither SCIP nor Baron were able to even find a feasible solution and both were not able to push the lower bound significantly above the trivial lower bound of zero within the time limit. In contrast, with our new approach significantly tighter lower bounds were obtained rather quickly. A more detailed view on how the lower and upper bounds evolve over time is given in Figs. 8–16 for all test instances and solvers. A comparative view on the evolution of the relative optimality gaps is presented in Figs. 17, 18, 19.

One important detail to point out is the seemingly piecewise constant development of the lower bound during a run of Algorithm 2. This is mainly due to the fact that we cold-start the MIP-solver after each iteration and can thus not benefit from all the information inherent to the branch-and-bound tree computed in the preceding iteration. At this point, there is clearly space for improvements. However, the lower bounds obtained from Algorithm 2 are systematically better than those obtained

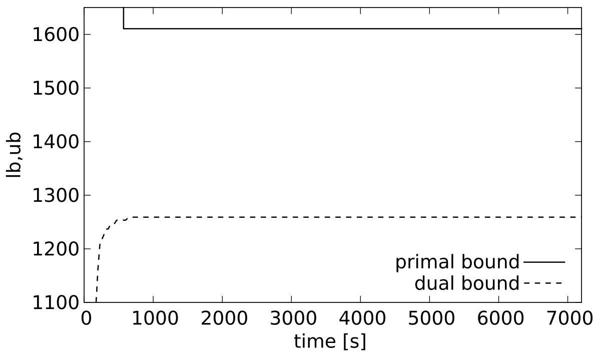


Fig. 12 Bounds for the 2nd net (Baron)

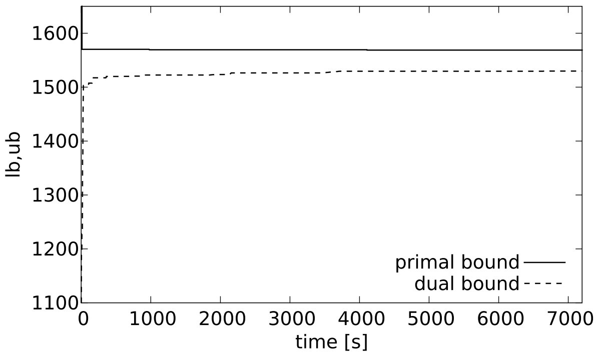


Fig. 13 Bounds for the 2nd net (Algorithm 2)

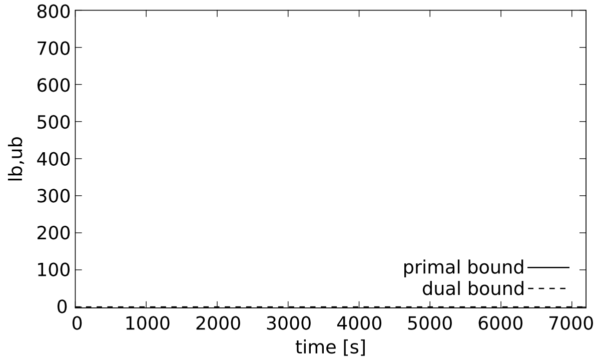


Fig. 14 Bounds for the 3rd net (SCIP)

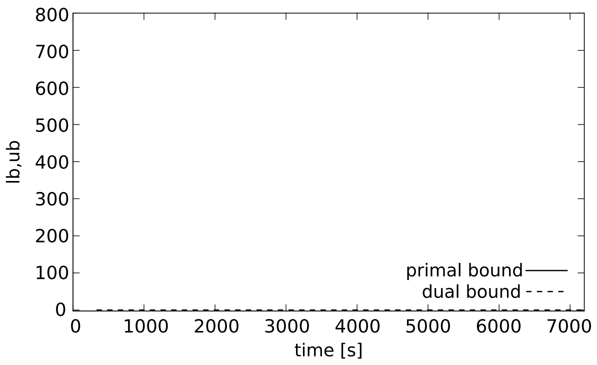


Fig. 15 Bounds for the 3rd net (Baron)

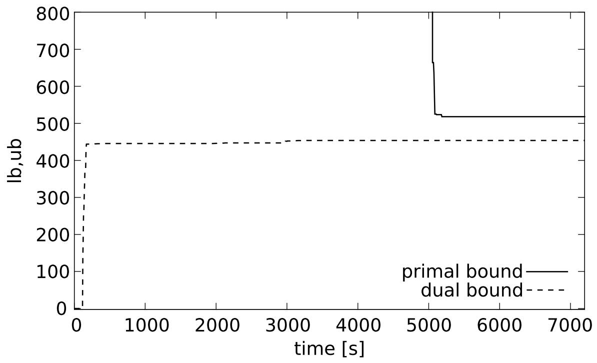


Fig. 16 Bounds for the 3rd net (Algorithm 2)

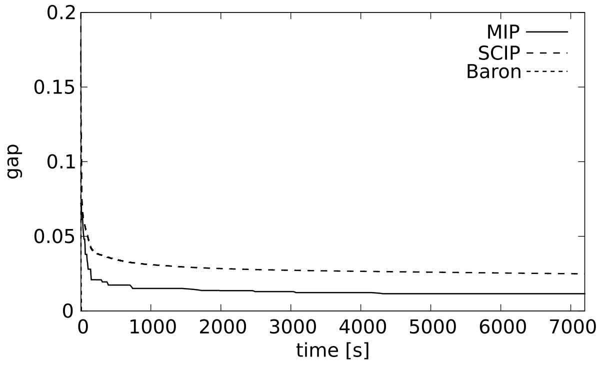


Fig. 17 Gap for the 1st net

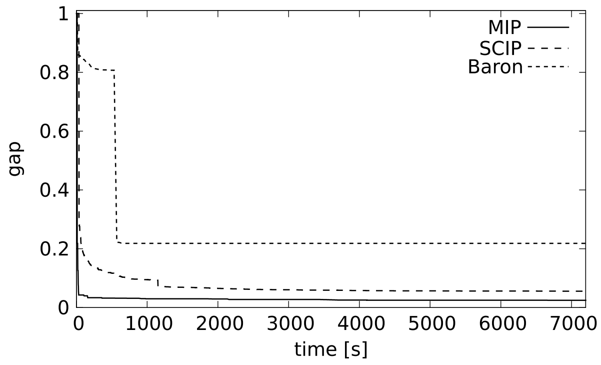


Fig. 18 Gap for the 2nd net

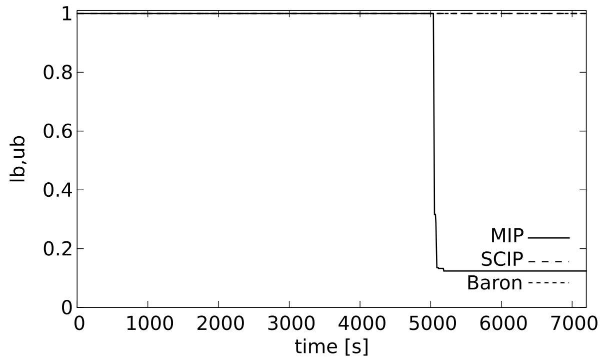


Fig. 19 Gap for the 3rd net

from SCIP and despite for the small test network the same holds in comparison with Baron. Moreover, we are able to determine feasible solutions even for the large real-world instance and better feasible solutions for the medium-sized instance. We believe that the latter is mainly possible due to the tightness of the employed type of MIP-relaxations. Since these relaxations are nonconvex, they are potentially tighter than any existing convex relaxation and we do not have to rely on (spatial) branching alone to obtain high accuracy solutions in regions of interest. This is due to the fact that the nonconvexities are modeled via integrality restrictions only. Thus, also polyhedral techniques could be exploited to determine a feasible assignment, especially for the discrete variables of the underlying MINLP.

Finally, we would like to give insight into the complexity of the MIP-relaxation models solved during the iterations of Algorithm 2. For the first network this is summarized in Table 5. For the two other networks the same information is given in Table 6 and Table 7. Here, the i-th row corresponds to the i-th iteration of Algorithm 2, i.e., to the solution of a single MIP-relaxation. The iteration number is given in the first column. In iteration 0 the initial MIP-relaxation is solved. The second column

Table 5 Log of Algorithm 2 on the 1st net

i	ref	bins	reals	cons	nodes	sec
0	0	56	379	812	1	1
1	17	82	423	890	36	3
2	15	106	465	962	5128	6
3	15	134	515	1046	7580	8
4	15	172	581	1160	123092	28
5	13	194	619	1226	45709	19
6	14	219	664	1301	103132	36
7	14	244	709	1376	30377	12
8	10	268	753	1448	65018	30
9	13	296	805	1532	222502	73
10	8	315	842	1589	221772	82
11	10	338	887	1658	243782	90
12	8	363	936	1733	712089	342
13	8	382	973	1790	2457946	980
14	6	397	1002	1835	596648	263
15	8	418	1043	1898	1102106	503
16	7	435	1076	1949	365960	159
17	7	453	1110	2003	887336	420
18	4	463	1130	2033	2172757	1235
19	4	475	1154	2069	>4533060	>2889

contains the number of nonlinear expressions, whose piecewise polyhedral outer approximations have been refined in the end of iteration i. The three subsequent columns contain the number of binary and continuous variables and the number of linear constraints of the respective MIP-relaxation, while the last two columns show the number of branch-and-bound nodes and the number of seconds needed to solve the i-th MIP-relaxation.

Clearly, with each iteration a number of refinement steps are performed and thus the number of rows and columns is strictly increasing with the iteration number. As one might expect, the number of branch-and-bound nodes as well as the number of seconds needed to solve the MIP-relaxations also tends to increase with the number of iterations. However, the number of refinement steps is mainly decreasing from iteration i to $i+1$ such that the number of additional rows and columns tends to decrease with i. The latter shows an extremely favorable property of Algorithm 2. Although in early iterations, where the global optimum of a (rather coarse) relaxation is typically far away from an optimum of the underlying MINLP, a lot of refinement steps are performed, after a while the optimum of the relaxation approaches an MINLP-optimum and the remaining refinements are typically performed in the neighborhood of this point. From there on the number of rows and columns of the MIP-relaxations do not increase significantly anymore. Of

Table 6 Log of Algorithm 2 on the 2nd net

i	ref	bins	reals	cons	nodes	sec
0	0	292	2027	4070	818	10
1	73	367	2104	4295	1242	10
2	74	442	2181	4520	2057	12
3	79	525	2272	4769	3960	33
4	75	606	2361	5012	6992	42
5	75	683	2440	5243	12281	53
6	68	759	2524	5471	4671	37
7	69	832	2605	5690	4178	56
8	64	900	2681	5894	17423	104
9	55	961	2750	6077	17061	68
10	49	1014	2811	6236	19045	115
11	35	1055	2860	6359	55620	179
12	51	1107	2914	6515	53281	186
13	30	1143	2960	6623	107601	447
14	24	1175	3002	6719	175484	554
15	53	1231	3062	6887	43167	239
16	47	1279	3112	7031	453691	1554
17	23	1308	3153	7118	33625	225
18	39	1349	3198	7241	973103	2721
19	13	1368	3225	7298	>62765	>535

Table 7 Log of Algorithm 2 on the 3rd net

i	ref	bins	reals	cons	nodes	sec
0	0	907	4487	9848	302932	1523.61
1	221	1142	4738	10553	72778	735.76
2	216	1367	4979	11228	323738	2694.89
3	207	1582	5210	11873	>291328	>2242.0

course one might construct problems where it takes arbitrary long to arrive at this point, but at least for the problems considered here this does not seem to be the case.

6 Discussion

The computational results obtained with our algorithm are extremely encouraging: Our approach outperforms the state-of-the-art MINLP-solvers Baron and SCIP on the two larger problems. We stress that we did not incorporate any problem specific knowledge in the algorithm so far to keep the comparison with the solvers fair. We also have not yet implemented more refined preprocessing techniques except for a straight-forward bound strengthening.

In our view, the problem we studied has special structure that makes it particularly amenable to our approach. Most of the nonlinear functions that appear are low-dimensional. The pressure loss equation for pipes also can be seen to be convex on one half of the real line and concave on the other. This also makes the problem comparatively well-structured. We also note that the problem is far more difficult than just checking the feasibility of a nomination, the so-called nomination validation problem [28]. The main additional difficulty is the description of the compressors, especially their power requirement (40). We, nevertheless, think that the approach outlined here can be generalized to other problems, which we will tackle in future work. The hope is generally that we will, in the future, identify a large class of mixed-integer nonlinear problems, that is solvable in practice. The way will involve looking at special cases and general insight about the structure of the problems, just as in the case of mixed-integer linear problems.

Acknowledgements The authors are grateful to Alexander Martin for fruitful discussions and comments. We thank our industry partner Open Grid Europe GmbH and the other partners of the ForNe project. The work was partially funded by the German Federal Ministry of Economics and Technology under grant number 0328006B.

References

1. Achterberg, T.: Constraint integer programming. Ph.D. thesis, TU Berlin (2007)
2. Bales, P.: Hierarchische Modellierung der Eulerschen Flussgleichungen in der Gasdynamik. Master's thesis, Technische Universität Darmstadt (2005)
3. Bureau International des Poids et Mesures: The International System of Units (SI), 8th edn. (2006)
4. Domschke, P., Geißler, B., Kolb, O., Lang, J., Martin, A., Morsi, A.: Combination of nonlinear and linear optimization of transient gas networks. INFORMS J. Comput. **23**(4), 605–617 (2011)
5. Ehrhardt, K., Steinbach, M.C.: Nonlinear optimization in gas networks. In: Bock, H.G., Kostina, E., Phu, H.X., Ranacher, R. (eds.) Modeling, Simulation and Optimization of Complex Processes, pp. 139–148. Springer, Berlin (2005)
6. Geißler, B.: Towards globally optimal solutions of MINLPs by discretization techniques with applications in gas network optimization. Ph.D. thesis, FAU Erlangen-Nürnberg (2011)
7. Geißler, B., Martin, A., Morsi, A., Schewe, L.: Solving MINLPs using adaptively refined MIPs (2012, to be published)
8. Geißler, B., Martin, A., Morsi, A., Schewe, L.: Using piecewise linear functions for solving MINLPs. In: Lee, J., Leyffer, S. (eds.) Mixed Integer Nonlinear Programming. The IMA Volumes in Mathematics and Its Applications, vol. 154, pp. 287–314. Springer, New York (2012)
9. Grötschel, M., Monma, C., Stoer, M.: Polyhedral approaches to network survivability. In: Reliability of Computer and Communication Networks, New Brunswick, NJ, 1989. DIMACS Ser. Discrete Math. Theoret. Comput. Sci., vol. 5, pp. 121–141. Am. Math. Soc., Providence (1991)

10. Grötschel, M., Monma, C.L., Stoer, M.: Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. Oper. Res. **40**(2), 309–330 (1992). doi:10.1287/opre.40.2.309
11. Grötschel, M., Monma, C.L., Stoer, M.: Facets for polyhedra arising in the design of communication networks with low-connectivity constraints. SIAM J. Optim. **2**(3), 474–504 (1992). doi:10.1137/0802024
12. Grötschel, M., Monma, C.L., Stoer, M.: Design of survivable networks. In: Network Models. Handbooks Oper. Res. Management Sci., vol. 7, pp. 617–672. North-Holland, Amsterdam (1995). doi:10.1016/S0927-0507(05)80127-6
13. Grötschel, M., Monma, C.L., Stoer, M.: Polyhedral and computational investigations for designing communication networks with high survivability requirements. Oper. Res. **43**(6), 1012–1024 (1995). doi:10.1287/opre.43.6.1012
14. Gurobi Optimization, Inc.: Gurobi Optimizer Reference Manual. Version 5.0. Gurobi Optimization, Inc., Houston, TX, USA (2010)
15. HSL: a collection of Fortran codes for large scale scientific computation (2011). www.hsl.rl.ac.uk
16. Koch, T. (ed.): From simulation to optimization: evaluating gas network capacities (2012, in preparation)
17. Leyffer, S., Sartenaer, A., Wanufelle, E.: Branch-and-refine for mixed-integer nonconvex global optimization. Technical report ANL/MCS-P1547-0908, Mathematics and Computer Science Division, Argonne National Laboratory (2008)
18. LIWACOM Informations GmbH and SIMONE Research Group s.r.o.: Gleichungen und Methoden. Benutzerhandbuch (2004)
19. Lurie, M.V.: Modeling of Oil Product and Gas Pipeline Transportation. Wiley-VCH, Weinheim (2008)
20. Markowitz, H.M., Manne, A.S.: On the solution of discrete programming problems. Econometrica **25**, 84–110 (1957)
21. Martin, A.: Integer programs with block structure. Habilitation treatise, Zuse Institute Berlin (1999)
22. Martin, A., Möller, M., Moritz, S.: Mixed integer models for the stationary case of gas network optimization. Math. Program., Ser. B **105**, 563–582 (2006)
23. Menon, E.: Gas Pipeline Hydraulics. Taylor & Francis, Boca Raton (2005)
24. Möller, M.: Mixed integer models for the optimisation of gas networks in the stationary case. Ph.D. thesis, Technische Universität Darmstadt (2004)
25. Moody, L.: Friction factors for pipe flow. Trans. Am. Soc. Mech. Eng. **66**(8), 671–677 (1944)
26. Moritz, S.: A mixed integer approach for the transient case of gas network optimization. Ph.D. thesis, Technische Universität Darmstadt (2007)
27. Nikuradse, J.: Strömungsgesetze in rauhen Rohren. Forschungsheft auf dem Gebiete des Ingenieurwesens. VDI-Verlag, Düsseldorf (1933)
28. Pfetsch, M.E., Fügenschuh, A., Geißler, B., Geißler, N., Gollmer, R., Hiller, B., Humpola, J., Koch, T., Lehmann, T., Martin, A., Morsi, A., Rövekamp, J., Schewe, L., Schmidt, M., Schultz, R., Schwarz, R., Schweiger, J., Stangl, C., Steinbach, M.C., Vigerske, S., Willert, B.M.: Validation of nominations in gas network optimization: models, methods, and solutions. Technical report ZR 12-41, ZIB (2012)
29. Ríos-Mercado, R.Z., Borraz-Sánchez, C.: Optimization problems in natural gas transmission systems: a state-of-the-art survey. Technical report ZR 12-41, ZIB (2012)
30. Ríos-Mercado, R.Z., Wu, S., Scott, L.R., Boyd, E.A.: A reduction technique for natural gas transmission network optimization problems. Ann. Oper. Res. **117**(1), 217–234 (2002)
31. Shewchuk, J.: Triangle: engineering a 2D quality mesh generator and Delaunay triangulator. In: First ACM Workshop on Applied Computational Geometry (1996)
32. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**(2), 225–249 (2005)
33. Vigerske, S.: Decomposition in multistage stochastic programming and a constraint integer programming approach to mixed-integer nonlinear programming. Ph.D. thesis, Humboldt-

Universität zu Berlin (2012). www.math.hu-berlin.de/~stefan/diss.pdf
34. Wächter, A., Biegler, L.T.: Line search filter methods for nonlinear programming: motivation and global convergence. SIAM J. Optim. **16**(1), 1–31 (2005)
35. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program., Ser. A **106**, 25–57 (2006)

Solving k-Way Graph Partitioning Problems to Optimality: The Impact of Semidefinite Relaxations and the Bundle Method

Miguel F. Anjos, Bissan Ghaddar, Lena Hupp, Frauke Liers, and Angelika Wiegele

Abstract This paper is concerned with computing global optimal solutions for maximum k-cut problems. We improve on the SBC algorithm of Ghaddar, Anjos and Liers in order to compute such solutions in less time. We extend the design principles of the successful BiqMac solver for maximum 2-cut to the general maximum k-cut problem. As part of this extension, we investigate different ways of choosing variables for branching. We also study the impact of the separation of clique inequalities within this new framework and observe that it frequently reduces the number of subproblems considerably. Our computational results suggest that the proposed approach achieves a drastic speedup in comparison to SBC, especially when $k = 3$. We also made a comparison with the orbitopal fixing approach of Kaibel, Peinhardt and Pfetsch. The results suggest that, while their performance is better for sparse instances and larger values of k, our proposed approach is superior for smaller k and for dense instances of medium size. Furthermore, we used CPLEX for solv-

M.F. Anjos
Canada Research Chair in Discrete Nonlinear Optimization in Engineering, GERAD, École Polytechnique de Montréal, Montréal, QC, Canada H3C 3A7
e-mail: anjos@stanfordalumni.org

B. Ghaddar
Centre for Operational Research and Analysis, Defence Research and Development Canada, Department of National Defence, 101 Colonel By Drive, Ottawa, ON, Canada K1A 0K2
e-mail: bghaddar@uwaterloo.ca

L. Hupp · F. Liers (✉)
Department Mathematik, Friedrich-Alexander-Universität Erlangen-Nürnberg, Cauerstr. 11, 91058 Erlangen, Germany
e-mail: frauke.liers@math.uni-erlangen.de

L. Hupp
e-mail: lena.hupp@math.uni-erlangen.de

A. Wiegele
Institut für Mathematik, Alpen-Adria-Universität Klagenfurt, Universitätsstr. 65-67, 9020 Klagenfurt, Austria
e-mail: angelika.wiegele@aau.at

ing the ILP formulation underlying the orbitopal fixing algorithm and conclude that especially on dense instances the new algorithm outperforms CPLEX by far.

1 Introduction

The impact of Martin Grötschel on the field of optimization is hard to overstate. His work combines mathematics and computer science by both aiming to develop a deep mathematical understanding of the structure of a problem and then turning this understanding into high-performance algorithms.

We are delighted to have the opportunity to contribute to this Festschrift on the occasion of Martin Grötschel's 65th birthday. Liers is an academic grandchild of Grötschel via Michael Jünger, while Hupp is a doctoral student of Liers and is thus on the way to become a descendant of Grötschel. We also mention that the research collaboration between Liers and Anjos began when Jünger hosted Anjos as a post-doctoral fellow in Köln, and that Ghaddar is an academic child of Anjos.

It is fitting that this paper proposes an effective exact algorithm for the maximum k-cut problem that combines a semidefinite relaxation with polyhedral insights and state-of-the-art bundle algorithms. Grötschel, together with several of his academic descendents, was among the first to study graph partitioning and its application in areas such as physics and VLSI. This is but one of the important classes of NP-hard problems on which he has had a lasting impact. Furthermore, together with Lovász and Schrijver, Grötschel proposed the first polynomial-time algorithm for solving semidefinite optimization problems, namely the ellipsoid method. While it is not competitive in practice, it remains the only truly proven polynomial-time algorithm for solving semidefinite problems.

Yet another aspect in which this paper follows the inspiration of Martin Grötschel is in the iterative addition of cutting planes. Again with Lovász and Schrijver, Grötschel pioneered the fundamental principle of the equivalence of separation and optimization. He undertook polyhedral studies of exact linear optimization formulations of various NP-hard problems, including the traveling salesman problem, the maximum-cut problem and linear ordering problems. In this way, he laid solid foundations for modern exact branch-and-bound algorithms.

The maximum k-cut (max-k-cut) problem is a graph partitioning problem concerned with finding an optimal k-way partitioning of the set of nodes of an undirected simple graph with weights on the edges. An edge is cut if its endpoints are in different sets of the partition, and a partition is also called a cut of the graph. Thus, the weight of a cut is equal to the sum of the weights on the edges cut by its corresponding partition. There are a number of different versions of graph partitioning problems in the literature, depending on the number of sets allowed in a partition, on the types of edge weights allowed, and on the possible presence of additional side constraints such as restrictions on the number of nodes allowed in each partition. Most versions are known to be NP-hard. Graph partitioning problems have myriad applications in areas as varied as telecommunications network planning [18], VLSI circuit design [6], sports scheduling [19, 44], and statistical physics [37].

The special case of max-k-cut with $k = 2$ is known as the max-cut problem. The max-cut problem has been extensively studied; in particular, it is known to be equivalent to quadratic unconstrained binary optimization. Among the numerous references for max-cut, we point out Barahona and Mahjoub [5], Deza and Laurent [15], and Boros and Hammer [9]. A prominent application of max-cut is in determining energy-minimum states, i.e., ground states, of Ising spin glasses. The first exact branch-and-cut approach for its solution was presented in [6] and developed further in [37]. Extending the number of shores to $k > 2$, maximum k-cuts need to be computed when determining ground states of Potts glasses. In the physics literature, ground states are usually computed heuristically, but more reliable conclusions can be drawn by analyzing exact solutions.

The max-k-cut problem is sometimes also called the minimum k-partition problem by noting that maximizing the k-cut is equivalent to minimizing the sum of the weights of the edges connecting nodes in the same partition. It was studied in [11] by Chopra and Rao who identified several valid and facet-defining inequalities for the k-partition polytope. Further results can be found in Chopra and Rao [12] and Deza, Grötschel, and Laurent [16].

Armbruster et al. [4] consider the minimum bisection problem, where $k = 2$ and the number of nodes in both partitions has to be less than a given value $F \leq \frac{n}{2}$. The case $F = \lceil \frac{n}{2} \rceil$ corresponds to a minimum equipartition problem since the sizes of both partitions then have to be (as close as possible to) equal. Alternatively, this latter constraint added to the max-cut problem gives the equicut problem, which can be motivated by an application to Coulomb glasses in theoretical physics. Motivated by this application, Anjos et al. [3] recently proposed an enhanced branch-and-cut algorithm for equicut based on an approach proposed by Brunetta et al. [10].

More generally, the k-way equipartition problem is a minimum k-partition problem with the additional constraint that the k partitions have to be of the same size. Mitchell [43] applied a branch-and-cut algorithm based on linear programming (LP) to the k-way equipartition problem with application to a sports league realignment problem. Lisser and Rendl [39] considered an application of k-way equipartition in telecommunications and investigated both semidefinite and linear relaxations of the problem with iterative cutting plane algorithms.

Strong approximation guarantees have been obtained for several of these NP-hard problems. A famous example is the randomized approximation algorithm for max-cut proposed by Goemans and Williamson [23] that uses a semidefinite programming (SDP) relaxation. Frieze and Jerrum [21] extended the approach of Goemans and Williamson to max-k-cut and obtained a polynomial-time approximation algorithm and a corresponding rounding technique. In particular, they proved the existence of constants α_k, $k \geq 2$, such that

$$\mathbf{E}\big(w(\mathcal{V}_k)\big) \geq \alpha_k w\big(\mathcal{V}_k^*\big)$$

where $w(\mathcal{V}_k) = \sum_{1 \leq r < s \leq k} \sum_{i \in V_r, j \in V_s} w_{ij}$, $\mathcal{V}_k^*$ determines an optimal cut, and $\mathbf{E}$ denotes the expected value. For small values of k, the best-known lower bounds for these constants are given by de Klerk et al. [14]. The improved SDP relaxation for max-cut of Anjos and Wolkowicz [2] provides very tight bounds for max-cut and perfectly captures the faces of dimension 1 of the cut polytope [1]. This ability to capture portions of the structure of the underlying polytope was proved for the whole Lasserre hierarchy by Laurent [32]. Eisenblätter [18] used an SDP relaxation for the minimum k-partition problem and proved that all the feasible solutions for the SDP problem cannot violate the (facet-defining) triangle and clique inequalities for the k-partition polytope by more than a small amount, thus showing that an SDP relaxation can closely approximate the structure of the k-partition polytope.

Our interest is in computing global optimal solutions for max-k-cut problems. Computationally speaking, SDP relaxations often yield stronger bounds than LP relaxations. However, this strength usually comes at the expense of long running times. Thus, it is not clear beforehand whether linear or semidefinite relaxations lead to best performance.

For max-cut, sparse instances can usually be solved efficiently with LP-based methods for large graphs. The web-based Spin Glass Server [50] is especially designed for fast solutions of instances defined on grids that arise in statistical physics [37]. For instance, for a two-dimensional lattice with $L \leq 80$ and periodic boundary conditions, one ground-state computation takes less than two minutes on average on a SUN Opteron (2.2 GHz) machine; for 120^2 lattices the computation takes 28 minutes [38]. On the other hand, SDP-based methods perform better for dense instances of max-cut [45]. The SDP-based web server BiqMac [7] can solve max-cut instances with arbitrary structure with up to 100 vertices [47].

For the k-way equipartition problem, the LP-based branch-and-cut algorithm of Mitchell [43] found the optimal solution for the NFL realignment problem where $k = 8$ and $n = 32$, whereas a percentage gap of less than 2.5 % was given for graphs of sizes 100 to 500. Lisser and Rendl [39] found that for graph sizes ranging from 100 to 900 vertices and for $k = 5$ and $k = 10$, the SDP approach produced a gap between 4–6 % from the optimal solution and had overall better performance than the LP approach.

For sparse instances of minimum bisection, the computational results of Armbruster et al. [4] suggest that SDP relaxations are superior to the corresponding LP relaxations. On the other hand, Anjos et al. [3] compared basic LP and SDP relaxations for the equicut problem, and found that linear bounds can be competitive with the semidefinite ones and can be computed much faster. While their results appear to contradict the above observations, it is important to note that they focus on dense instances coming from the physics application, and that their specialized relaxation includes constraints that are not valid for the minimum bisection polytope in general.

In this paper, we focus on max-k-cut for $k \geq 3$. Our motivation is that while effective computational procedures that yield globally optimal solution for arbitrary instances with up to 100 vertices and sparse graphs of considerably larger sizes have been implemented for the $k = 2$ case, to the best of our knowledge, most of the procedures proposed in the literature either cannot be applied for general k, provide no guarantee of global optimality, or enforce additional constraints.

Among the exceptions, Kaibel, Peinhardt and Pfetsch [29, 30] applied a symmetry-breaking method called orbitopal fixing (OF) to graph partitioning problems within an LP-based branch-and-cut. Symmetry arises in graph partition problems because different feasible solutions may represent the same partition. The feasible set of the problem can thus be partitioned into orbits so that all the solutions in an orbit represent the same partition. This structure is exploited by OF through choosing one representative solution from each orbit, namely the lexicographically maximal one, and the branching and pruning steps are adjusted to restrict the enumeration to only such solutions. This is a specialization to partition problems of the isomorphism pruning technique of Margot [40, 41]. The authors present results for the minimum k-partition problem in sparse graphs with up to 50 nodes and a few hundred edges [29].

Another exception is the SDP-based branch-and-cut algorithm for the minimum k-partition problem proposed by Ghaddar, Anjos and Liers [22]. Their SBC algorithm combines the SDP relaxation proposed by Eisenblätter [18] with valid inequalities for the k-partition polytope and with a novel iterative clustering heuristic (ICH) that finds feasible solutions using the SDP optimal solution. The computational results reported in [22] show that ICH consistently provides feasible solutions that are better than those obtained using the hyperplane rounding techniques of Goemans and Williamson (for $k = 2$) and of Frieze and Jerrum (for $k \geq 3$). Ghaddar et al. presented results showing that SBC computes globally optimal solutions for dense graphs with up to 60 nodes, for (sparse) grid graphs with up to 100 nodes, and for different values of $k \geq 3$.

In this paper, we combine the approach of Ghaddar et al. with the design principles of BiqMac [47] to compute globally optimal solutions for max-k-cut more efficiently. We refer to this new approach as bundleBC. Although straightforward in principle, this combination raises several challenges, including the following:

- branching decisions may lead to subproblems that are infeasible or that have no interior (this can be avoided for max-cut by appropriate switching and shrinking steps);
- a wider variety of cutting planes needs to be generated and managed dynamically; and
- the constraints of the SDP relaxation itself need to be handled differently, i.e., we relax the lower bound constraints of the initial SDP relaxation and treat them as cuts.

Our computational results suggest that bundleBC achieves a drastic speedup in comparison to SBC, especially when $k = 3$. Furthermore, a comparison with the results reported by Kaibel et al. [29], suggests that for $k = 3$ and medium-sized dense instances (30 nodes), our approach performs better than their OF approach, whereas their performance is better for sparse instances and for larger values of k. Additionally, we used CPLEX to evaluate the ILP model underlying the OF approach.

This paper is organized as follows. In Sect. 2, we state the formal definition of the max-k-cut problem and briefly summarize the relevant formulations and relaxations in the literature. The proposed exact algorithm is described in Sect. 3. Section 3.1 is concerned with the upper bound computation using a bundle to solve the

SDP relaxations, and Sect. 3.2 describes the heuristic we use for computing lower bounds. Section 3.3 describes the 6 branching rules that we tested, and how we handle the possibility that branching sometimes yields SDP subproblems that are infeasible or that have no interior. Section 4 presents the basic implementation details of bundleBC. Computational results are reported in Sect. 5. Section 5.1 describes the benchmark sets of instances that we used, Sect. 5.2 reports the performance of the 6 branching rules that we considered, and Sect. 5.3 studies the impact of clique inequalities on the performance of bundleBC. Section 5.4 presents comparisons of bundleBC with SBC, and Sect. 5.5 compares the performance of bundleBC with the orbitopal fixing approach and presents the CPLEX results. Section 6 concludes the paper.

2 Problem Description, Formulations and Relaxations

An instance of the max-k-cut problem is specified by fixing an undirected graph $G = (V, E)$ with edge weights w_{ij} of the edges, and a positive integer $k \geq 2$. The objective is to find a partition of V into at most k disjoint partitions $V_1, \ldots, V_k$ such that the sum of the weights of edges joining different partitions is maximized. We assume without loss of generality that G is a complete graph (missing edges can be added with a corresponding weight of zero).

From a semidefinite perspective, the max-k-cut problem can be formulated as:

$$\max \sum_{i,j \in V, i<j} w_{ij} \frac{(k-1)(1-X_{ij})}{k} \tag{1}$$

$$\text{s.t.} \quad X_{ii} = 1 \quad \forall i \in V \tag{2}$$

$$X_{ij} \in \left\{ \frac{-1}{k-1}, 1 \right\} \quad \forall i, j \in V, i < j \tag{3}$$

$$X \succeq 0,$$

where $X_{ij} = \frac{-1}{k-1}$ if vertices i and j are in different partitions, and $X_{ij} = 1$ if they are in the same partition. Replacing the binary constraint (3) by $\frac{-1}{k-1} \leq X_{ij} \leq 1$ results in a semidefinite relaxation. However, the constraint $X_{ij} \leq 1$ can be dropped since it is enforced implicitly by the constraints $X_{ii} = 1$ and $X \succeq 0$. We end up with the following SDP relaxation:

$$(\mathrm{SM}k\mathrm{C}) \quad \max \sum_{i,j \in V, i<j} w_{ij} \frac{(k-1)(1-X_{ij})}{k} \tag{4}$$

$$\text{s.t.} \quad X_{ii} = 1 \quad \forall i \in V \tag{5}$$

$$X_{ij} \geq \frac{-1}{k-1} \quad \forall i, j \in V, i < j \tag{6}$$

$$X \succeq 0$$

or, alternatively, using the Laplace matrix L of the graph and rewriting the constraints $X_{ii} = 1$, we end up with

$$\begin{aligned} (\text{SM}k\text{C}) \quad & \max \frac{k-1}{2k}\langle L, X\rangle && (7)\\ & \text{s.t.} \quad \operatorname{diag}(X) = e && (8)\\ & X_{ij} \geq \frac{-1}{k-1} \quad \forall i, j \in V, i < j && (9)\\ & X \succeq 0 \end{aligned}$$

with e being the vector of all ones of length $|V|$. Note that if we fix $k = 2$ in (SMkC), we obtain the SDP relaxation used for max-cut by Goemans and Williamson [23].

The relaxation (SMkC) was first used by Frieze and Jerrum [21]; it is the basis of the SBC algorithm of Ghaddar et al. [22]. In that algorithm, the SDP relaxation was further tightened by adding valid inequalities. The two types of valid inequalities used in SBC are the triangle and the clique inequalities. The triangle inequalities are based on the observation that if any two nodes i and j are in the same partition, and j and another node k are in the same partition, then also nodes i and k necessarily have to be in the same partition. For the SDP formulation, the $3\binom{|V|}{3}$ triangle inequalities have the form:

$$X_{ij} + X_{jh} - X_{ih} \leq 1, \qquad (10)$$

where i, j, and $h \in V$. The $\binom{|V|}{k+1}$ clique inequalities ensure that for every subset of $k + 1$ nodes, at least two of the nodes must belong to the same partition:

$$\sum_{i,j \in Q, i<j} X_{ij} \geq -\frac{k}{2} \quad \forall Q \subseteq V \text{ where } |Q| = k + 1.$$

Together with the constraints (10), this implies that there are at most k partitions.

3 Proposed Exact Algorithm

We use a branch-and-bound framework to solve the max-k-cut problem to global optimality. To set up the framework, the following three issues must be addressed:

- how to obtain upper bounds;
- how to obtain lower bounds, i.e., high-quality cuts; and
- how to branch.

The computation of the upper bounds is the subject of Sect. 3.1. Computing lower bounds is discussed in Sect. 3.2, and the question how to branch is addressed in Sect. 3.3. Algorithm 1 gives the steps as they are executed at each node of the branch-and-bound tree.

Algorithm 1: One node of the branch-and-bound algorithm

```
1. Initialize $\gamma$ and solve the problem (SMC$_{\text{basic}}$) using the oracle. Obtain the
   primal matrix $X^*$ and the upper bound $ub$.
2. Apply a heuristic to the current $X^*$ to obtain a $k$-cut and a lower bound $lb$.
3. Separate triangle inequalities.
4. While progress is made

     a. Do a descent step, i.e., obtain improved $ub$.
     b. If number of descent steps $mod 10 = 0$, apply a heuristic to the current
        $X^*$ and obtain a $k$-cut and a lower bound $lb$.
     c. If $lb \geq ub$ then stop: return and fathom node.
     d. Remove triangles and cliques if non-binding.
     e. Separate triangle and clique inequalities.

5. Apply a heuristic to the current $X^*$ to obtain a $k$-cut and a lower bound $lb$.
6. If $lb \geq ub$ then stop: return and fathom node.
7. Choose an edge for branching and return.
```

3.1 Computing Upper Bounds

It is well known that the bounds obtained by the LP relaxation of the ILP formulation are often weaker than the bounds obtained using the SDP relaxation (see e.g. [18, 22]). However, their computation is usually time consuming. In this work, we focus on an approach that maintains the strength of the relaxations using a fast approximation procedure to speed up computing times.

To this end, we make use of the SDP relaxation (SM*k*C) tightened by facets of the partition polytope. Specifically, we use triangle and clique inequalities. Solving the resulting relaxation is not trivial because the number of inequalities (for large graphs) is too large and the SDP problem becomes intractable for interior point methods. Thus, we need an alternative machinery to obtain this bound, namely a dynamic version of the bundle method.

3.1.1 Bundle Methods

The bundle method was first proposed by Lemarechal [35], and later on further investigated and refined by several authors, e.g., [31, 36, 49]. It has been developed for finding the approximate minimizer of a non-smooth convex function $f(\gamma)$ over $\gamma \in R^n$. In order to apply the bundle method, it is necessary to be able to obtain for any given γ the function value $f(\gamma)$ and a subgradient $g \in \partial f(\gamma)$. We assume that an oracle is available to return these values (see Sect. 3.1.3 for the specifics of the oracle that we use). This information is collected for different γs in a so-called “bundle” and used to construct a minorizing cutting plane model $\hat{f}$ of f.

To find a new value of γ, the displacement from the current point $\hat{\gamma}$ is penalized by adding a term proportional to $\|\gamma - \hat{\gamma}\|$ to the cutting plane model $\hat{f}$. Thus, the bundle algorithm requires minimizing

$$\hat{f}(\gamma) + \frac{1}{2\sigma}\|\gamma - \hat{\gamma}\| \tag{11}$$

with σ being some suitably chosen weight. Solving this problem is done by solving a sequence of convex quadratic problems of "small" dimension, i.e. dimension equal to the size of the bundle. The minimizer gives a new trial point $\tilde{\gamma}$ for which the oracle supplies the function value and a subgradient. This new information is added to the bundle and used to improve the cutting plane model. Then the whole process is repeated until the subgradient at the current point is sufficiently close to zero.

The bundle method as a tool for solving SDP problems has already been used by Poljak and Rendl [46] for solving the basic SDP relaxation for max-cut. Later on, the spectral bundle method has been introduced [25, 27]. In [24] a variant of the spectral bundle method is developed that allows adding and deleting cutting planes on the fly, and convergence of this method is proved.

Here we follow the concept of Fischer et al. [20]. Their idea is to apply the bundle method to the partial Lagrangian dual function, partial in the sense that only some of the constraints are dualized and lifted into the objective function by using Lagrangian multipliers, whereas constraints that are considered to be "easy" are handled directly inside the oracle. In other words, an oracle call amounts to solving a semidefinite program having only "easy" constraints.

In contrast to interior point methods, bundle methods are capable of solving semidefinite programs with a few thousand constraints. The price one pays for this is in the accuracy of the solution. However, the results in [20] demonstrate that the bundle algorithm often returns a reasonably accurate approximation.

3.1.2 Conic Bundle

Our aim is to solve the semidefinite program

$$(\mathrm{SM}k\mathrm{C}_{\text{strengthened}}) \quad \max \frac{(k-1)}{2k}\langle L, X\rangle \tag{12}$$

$$\text{s.t.} \quad \operatorname{diag}(X) = e \tag{13}$$

$$\mathcal{A}(X) \le b \tag{14}$$

$$X \succeq 0$$

where we collect in $\mathcal{A}(X) \le b$ all the bound constraints, i.e., $X_{ij} \ge \frac{-1}{k-1}$, for all $i, j \in V$, $i < j$, the set of triangle-inequalities and the set of clique-inequalities. Dualizing all the inequality constraints, we obtain the partial Lagrangian:

$$\mathcal{L}(X;\gamma) = \frac{(k-1)}{2k}\langle L, X\rangle + \gamma^\top\big(b - \mathcal{A}(X)\big) = b^\top\gamma + \left\langle \frac{(k-1)}{2k}L - \mathcal{A}^\top(\gamma), X\right\rangle,$$

and the dual functional reads

$$
\begin{aligned}
f(\gamma) &= \max_{X \succeq 0, \operatorname{diag}(X)=e} \mathcal{L}(X;\gamma) \\
&= b^\top \gamma + \max_{X \succeq 0, \operatorname{diag}(X)=e} \left\langle \frac{(k-1)}{2k} L - \mathcal{A}^\top(\gamma), X \right\rangle
\end{aligned} \tag{15}
$$

The number of inequality constraints is too large to handle, even after they are dualized. Our approach is to include all the bound constraints, and to add only those inequalities that are active at the optimum. Since this information is not known at the beginning, the choice of triangle- and clique-inequalities is updated in the course of the algorithm, as described below in Sect. 3.1.4.

The function (15) is then minimized over $R^n_{\geq 0}$ using the bundle method. We use the Conic Bundle software of Helmberg [13]. This implementation of the bundle method supports the minimization of the function arising from a Lagrangian dual, as in our case, i.e., it allows to generate primal solutions. Furthermore, it offers the possibility of adding and removing constraints in the course of the algorithm, as needed for our purposes.

3.1.3 Oracle

As already mentioned, in each bundle iteration an oracle is called to compute the function value and a subgradient at the current γ. The function evaluation amounts to solving the SDP problem

$$
\begin{aligned}
(\text{SMC}_{\text{basic}}) \quad & \max \left\langle \frac{(k-1)}{2k} L - \mathcal{A}^T(\gamma), X \right\rangle \\
& \text{s.t.} \quad X_{ii} = 1 \quad \forall i \in V \\
& \qquad\;\; X \succeq 0.
\end{aligned}
$$

The optimal solution $\tilde{X}$ of this SDP is then used to compute the function value

$$
f(\gamma) = b^\top \gamma + \left\langle \frac{(k-1)}{2k} L - \mathcal{A}^\top(\gamma), \tilde{X} \right\rangle
$$

and a subgradient

$$
g(\gamma) = b - \mathcal{A}(\tilde{X}).
$$

Note that the cost matrix depends on γ and therefore changes at each iteration. The feasible set of (SMC$_{\text{basic}}$) is the so-called elliptope, which has been well studied, see e.g. [33, 34]. The problem (SMC$_{\text{basic}}$) can thus be solved efficiently by interior point methods, even for large dimension. We implemented the primal-dual interior point method proposed in [28].

After branching, we have to add equality constraints of the form $X_{ij} = \frac{-1}{k-1}$ to (SMC$_{\text{basic}}$), as explained in Sect. 3.3 below. Since the number of these constraints is small, we can still use an interior point method to solve the SDP problem. However, we may end up with a problem having no interior, for example if we have a k-clique for which all the edge variables $X_{ij} = \frac{-1}{k-1}$. If this happens, we solve the SDP problem using CSDP [8] since its infeasible interior point algorithm runs well in this situation.

3.1.4 Adding Valid Inequalities

Once the SDP relaxation (SMkC) is solved, one can look for violated inequalities and add them to the relaxation, hence improving the upper bound. Triangle and clique inequalities are added at each iteration of the bundle algorithm and non-binding inequalities are detected and removed. Looking for violated triangle inequalities by complete enumeration is not computationally expensive. We describe in Sect. 4.1 how we manage the search and addition of violated triangle inequalities.

On the other hand, exact separation of clique inequalities is an NP-hard problem, and complete enumeration becomes intractable already for small values of k. Therefore, we use a separation heuristic that generates inequalities that are promising. It does not necessarily determine a violated inequality whenever one exists, however the algorithm is fast and yields good bounds.

The clique inequalities that are binding at optimality usually cover the whole graph, and each vertex in the graph is contained in several different clique inequalities. The separation heuristic is designed to have a similar behavior. For each vertex v in the graph, the algorithm grows a clique of size $k+1$ containing v. Vertices are added to the cliques in a greedy fashion. In each iteration, a vertex is added to a clique of size smaller than $k+1$ that contributes the smallest amount to the left-hand side of the corresponding clique inequality. The heuristic is described in detail in [22]. For a graph with n vertices, this procedure generates n clique inequalities. Violated ones are added to the problem formulation.

3.2 Lower Bound Heuristic

Using the conic bundle, we can generate approximate primal solutions in the course of the minimization algorithm. We use a heuristic method to compute a feasible k-cut from these approximate primal matrices X^*. This way we produce lower bounds which are useful for fathoming in the branch-and-cut tree.

There are two heuristics for extracting k-cuts from a primal solution X^*. The first one is the heuristic proposed by Frieze and Jerrum [21] and called FJ in the following. It works as follows:

1. Compute unit vectors $v_1, \ldots, v_n \in \mathbb{R}^n$ satisfying $v_i^T v_j = X_{ij}^*$ where $i, j \in V$.
2. Randomly generate k vectors $r_1, \ldots, r_k \in \mathbb{R}^n$ with their kn components drawn from independent and identically distributed random variables with a standard normal distribution with mean 0 and variance 1.

3. Partition V into $\mathcal{V}_k = \{V_1, \dots, V_k\}$ according to

$$V_j = \left\{ i : v_i \cdot r_j \geq v_i \cdot r_{j'}, \text{ for } j \neq j' \right\} \quad \text{for } 1 \leq j \leq k.$$

The second heuristic is called ICH. It was proposed by Ghaddar et al. [22] and used in their SBC algorithm. ICH works by aggregating information from X^* corresponding to subgraphs of G. Specifically, ICH sums the X^*_{ij} values on the edges between each of the $\binom{n}{k}$ subsets of k vertices, then sorts the resulting list of values, and places a subset of k vertices all in the same partition (or all in different partitions) when the sum is one of the largest values (or one of the smallest). The intuition behind this approach is that aggregated information is more reliable than single elements of data.

The implementation of the lower bound computation is described in Sect. 4.2 below.

3.3 *Branching*

The final ingredient of a branch-and-bound algorithm is how to subdivide the set of feasible solutions. It is well known that part of the success of a branch-and-bound algorithm depends on the choice of the branching variable X_{ij}.

3.3.1 Branching Rules

We use the information in the solution X^* of the SDP relaxation of the current node to choose a branching variable X^*_{ij}. We adapt the rules R1–R4 of [26] for max-cut in order to derive different choices for branching variables.

There are two important differences with respect to the max-cut case in [26] that we must address. First, while for max-cut the entries in X^* are all in the interval $[-1, 1]$, the SDP relaxation (SMkC) restricts the entries in X^* to the interval $[-\frac{1}{k-1}, 1]$. Second, since we dualize the bound constraints $X^*_{ij} \geq -\frac{1}{k-1}$, some values of X^*_{ij} may lie outside this interval. We considered different ways to deal with these differences.

Rules R1 and R3 are adapted most easily. Rule R1 chooses the "most decided" variable, i.e., we simply branch on the edge ij that is closest to $-\frac{1}{k-1}$ or to 1. By choosing an edge that seems to be already decided, the hope is that for the opposite decision the node will be fathomed quickly. This results in a deep but narrow branch-and-bound tree.

Rule R3 branches on the variable that is "least decided", i.e., we branch on the edge ij for which X^*_{ij} is closest to the middle of the interval $[-\frac{1}{k-1}, 1]$. If all the variables are either nearly 1 or less than or equal to $-\frac{1}{k-1}$, we choose ij corresponding to the minimum value of X^*_{ij}. By branching on the most undecided edge, we hope that the upper bounds will improve quickly.

Rules R1 and R3 do not distinguish between the variables with values outside the interval $[-\frac{1}{k-1}, 1]$ and the others.

Rules R2 and R4 are more elaborate. Instead of working with individual entries, these rules are based on the closeness of the rows of the matrix X^* to $\{-\frac{1}{k-1}, 1\}$ vectors.

Rule R2 looks for the two rows i' and j' that are closest to a $\{-\frac{1}{k-1}, 1\}$ vector. Let m denote the middle of the interval $[-\frac{1}{k-1}, 1]$. The branching edge i', j' is chosen as

$$i' = \operatorname*{argmin}_{1 \le i \le n} \sum_{r \ne i, r=1}^{n} \left((1-m) - \left|X^*_{ir} - m\right|\right)^2$$

$$j' = \operatorname*{argmin}_{1 \le j \le n, j \ne i'} \sum_{r \ne j, r=1}^{n} \left((1-m) - \left|X^*_{jr} - m\right|\right)^2.$$

Rule R4 looks for rows i' and j' such that i' is closest to a $\{-\frac{1}{k-1}, 1\}$ vector whereas j' is farthest from being feasible. Here i' and j' are chosen such that

$$i' = \operatorname*{argmin}_{1 \le i \le n} \sum_{r \ne i, r=1}^{n} \left((1-m) - \left|X^*_{ir} - m\right|\right)^2$$

$$j' = \operatorname*{argmin}_{1 \le j \le n} \sum_{r \ne j, r=1}^{n} \left(X^*_{jr} - m\right)^2.$$

Concerning the variables that are outside the interval $[-\frac{1}{k-1}, 1]$, we investigated two options for each of R2 and R4. The first option is to treat them just like the others (this corresponds to our rules R2 and R4). The second option for R2 is our rule R2a according to which we do not consider these variables as branching candidates unless all the variables inside the feasible interval are equal to 1 or $-\frac{1}{k-1}$. When this is the case, rule R2a selects the variable outside the interval with the smallest value.

Similarly, rule R4a works just as R4 but first considers only variables with values in the interval $[-\frac{1}{k-1}, 1]$ as candidates. If all those variables are already equal to 1 or $-\frac{1}{k-1}$, R4a selects the variable X^*_{ij} with the smallest value.

3.3.2 Shrinking and SDP Relaxations Without Interior

In the case where we fix $X_{i'j'} = 1$ at a particular node of the branch and bound tree, the resulting problem is equivalent to maximum k-cut of dimension $n-1$. Hence we can shrink the graph, i.e., we reduce the graph size by eliminating the vertex j'. The Laplacian matrix $\tilde{L}$ for the shrunken graph has entries $\tilde{l}_{ij}, i, j \in \{1, \dots, n\} \backslash \{j'\}$, as follows:

$$\tilde{l}_{ij} = \begin{cases} l_{ij} & \text{if } i, j \neq i' \\ l_{ii'} + l_{ij'} & \text{if } i \neq i', j = i' \\ l_{i'j} + l_{j'j} & \text{if } i = i', j \neq i' \\ l_{i'i'} + 2l_{i'j'} + l_{j'j'} & \text{if } i, j = i' \end{cases}$$

When we fix $X_{ij} = \frac{-1}{k-1}$, we cannot shrink the graph immediately, but we could shrink the graph as soon as there is a k-clique with all the values on its edges fixed to $\frac{-1}{k-1}$. However, performing this shrinking would require either expensive clique searches or more than two branches at each node of the branch-and-bound tree. Neither possibility is attractive, and moreover good cuts found on the shrunken graph cannot be extended to the original graph in a straightforward way. Therefore we omit these shrinkings, but as a consequence the SDP relaxation to be solved by the oracle may have no interior. When this happens, we solve the relaxations using CSDP [8] as mentioned earlier in Sect. 3.1.3.

4 Implementation Details

In this section, we explain how we set various parameters for the overall algorithm.

We used the Conic Bundle with its default settings. In particular, the relative precision requirement for successful termination was set to the default value of 10^{-5}.

The following subsections describe the preliminary experiments we performed to decide on a strategy for adding triangle inequalities and possibly clique inequalities, and a heuristic for computing lower bounds. In principle, there are several different parameter values and their combinations to test. We focused on the instances on complete graphs with Gaussian or bimodal distribution, and always averaged over five instances of the same size. In the course of our experiments we found that the resulting settings also worked well for the other types of graphs.

4.1 Adding Triangle Inequalities

In the course of the bundle iterations, we have to find a good set of triangle inequalities to add. Since enumeration of all triangle inequalities is cheap, we do this after every descent step of the bundle algorithm. The tolerance for considering an inequality as violated is 10^{-3}, and we build a heap of (at most) 5000 most-violated triangle inequalities.

Then we want to add m violated inequalities. We experimented with doing this in three different ways:

- selecting m inequalities randomly among the 5000;
- selecting the $\frac{m}{2}$ most violated ones and $\frac{m}{2}$ randomly from the remaining;

- selecting the m most violated.

It turned out that none of these options clearly stood out from the others, though the second option seemed to be slightly better. Thus, we chose the second strategy for our algorithm.

As for the choice of m, we ran experiments with $m = 500$ and $m = 1000$. Again, there was no clear winner, but $m = 500$ was slightly better so we chose this value.

4.2 Computing Lower Bounds

As mentioned in Sect. 3.2, we have two candidates for computing lower bounds, namely the heuristics ICH and FJ. While the computational results for minimum k-partition in [22] suggest that ICH consistently provides better k-cuts than FJ, its running times are much longer. For this reason, and because we want to solve large instances of max-k-cut, we choose to use FJ.

We experimented with how often to run the heuristic at each node of the branch-and-bound tree. While calling the heuristic often is time-consuming, not having a good lower bound at hand can cause the tree to be much larger. We tried three different settings for the frequency of the heuristic calls at each node:

- after every descent step;
- after every 10th descent step;
- only at the beginning and at the end.

The computational results did not give strong evidence that one of the above mentioned options is better than the others. The second setting improved slightly over the others, therefore this was our choice for our algorithm.

5 Computational Results

Our 32-Bit executables were run on 2.3 GHz Intel Xeon processors with 32 GB memory. For each instance, we allowed a maximum CPU time of 10 hours. We refer to our new algorithm as bundleBC.

5.1 The Benchmark Sets of Instances

We used the following sets of instances for our computational results:

Set A To have instances with varying number of vertices, we generated graphs with $|V|$ ranging from $10, 20, \ldots, 50$. Edges are chosen randomly such that we yield graphs with edge densities 25 % , 50 %, and 100 %. The weights on the edges are randomly chosen, either following a Gaussian or a bimodal ± 1 distribution. For each combination of $|V|$ and edge density, we generated 5 different instances and we always report averages over the 5 instances with the same values of $|V|$, edge density, and weight distribution.

Set B We also considered the instances from [42] for our numerical experiments. The first two classes of instances consist of complete graphs. Edge weights are either chosen as $|i - j|$ for edge (i, j), or are drawn randomly in $\{0, 1, \dots, 9\}$.
A different class of instances stems from an application in physics in which energy-minimum states of so-called Potts glasses need to be determined. In this application, instances are regular two- or three-dimensional grids with edge weights that are either Gaussian distributed around zero having variance one, or they are taken from $\{\pm 1\}$, where 50 % of the weights are negative. These instances were generated using rudy graph generator [48].
The name of an instance encodes its size followed by the distribution of the weights and the random seed that initializes rudy. E.g., the instance 2g_3_93 denotes a two-dimensional grid with Gaussian distributed weights of size 3^2 and random seed 93, whereas instance 3pm_234_234 denotes a three-dimensional grid with ± 1 weights of size $2 \times 3 \times 4$ and random seed 234.

Set C Finally, we take the set of instances from [29]. They generate instances with $|V| = 30$ and different number of edges m, namely $m = 200$ (sparse), $m = 300$ (medium), and $m = 400$ (dense). Furthermore, they consider graphs with $|V| = 50$ and $m = 560$. Edges are chosen uniformly at random until the specified number of edges is reached. The weights on these edges are drawn independently and uniformly at random from $\{1, \dots, 1000\}$. For each $|V|$ and each edge density three instances are generated.

The values of k were chosen as $k = 3$ and $k = 5$. For instances from [42] we tested additionally $k = 7$, and for the instances from [29] we consider $k \in \{3, 6, 9, 12\}$ to allow a comparison with the results in [29].

5.2 Choosing a Branching Rule

We implemented the six branching rules R1, R2, R2a, R3, R4, and R4a as they were explained in Sect. 3.3 and ran experiments using the instances described in Sect. 5.1. The different types of instances all display a similar behavior; thus we restrict our presentation to the results on the instances of benchmark set A.

We use performance profiles as proposed by Dolan and Moré [17] to facilitate the comparison of the branching rules. We set up our profiles as follows. Let P be the set of parameters we want to compare (for example the set of all different branching rules) and let I be the set of instances for which we ran our experiments with the different parameter settings $p \in P$. For each instance i and parameter $p \in P$ the performance ratio for the running time is calculated as

$$PR_{i,p}^{\text{time}} = \frac{\text{RunningTime}_{i,p}}{\min\{\text{RunningTime}_{i,p} : p \in P\}},$$

and the performance ratio for the number of subproblems is obtained as

$$PR_{i,p}^{\text{sub}} = \frac{\#\ \text{subproblems}_{i,p}}{\min\{\#\ \text{subproblems}_{i,p} : p \in P\}}.$$

If an instance i could not be solved for parameter setting p within the given time limit of 10 h = 36,000 s then we set $PR_{i,p}^{\text{time}} = PR_{\max}^{\text{time}}$ and $PR_{i,p}^{\text{sub}} = PR_{\max}^{\text{sub}}$ for suitably large values. (The specific choice of these does not affect the resulting profiles.) Our performance profiles are defined by the empirical distribution function

$$F(pr) = P\big(i \in I : \log_2(PR_{i,p}) \le pr\big)$$

where we use a $\log_2$ scale for ease of visualization.

Unlike the observations in Sect. 4.1 and Sect. 4.2, we found that the choice of branching rule has a strong influence on the computing times. Indeed the CPU times sometimes differ by more than two orders of magnitude between different rules. The main observation is that branching rule R2a is best, and that R2 is usually the second best. This dominance of R2a and R2 is independent of the size of the graph, its density, and the value of k. On the other hand, R1 usually leads to the worst performance.

The performance profiles of the CPU time for the different options of choosing a branching variable are shown in Fig. 1. The top figures represent the time comparison for complete graphs, the bottom figures for graphs with 25 % and 50 % edge density. Figures on the left refer to $k = 3$, figures on the right to $k = 5$.

These results demonstrate that the impact of the different branching strategies is greater for $k = 3$ than for $k = 5$. In other words, for $k = 5$ the lines in the profile are closer to each other. Considering complete graphs versus graphs with edge densities 25 % and 50 %, the performance profiles indicate that the branching strategy has a greater impact on the run time for complete graphs.

The number of nodes in the branch-and-bound tree is clearly related to the time needed for solving the problem. However, since in each iteration of the bundle algorithm we obtain a valid upper bound, we may be able to stop the bound computation after very few bundle iterations. This usually happens if an edge seems to be already decided whether it is cut or not: the opposite decision should lead to fathoming the node quickly. Therefore, a larger number of nodes in the branch-and-bound tree may still lead to shorter overall run times if the bound computation can be stopped early in many of the nodes.

We looked at the influence of the different branching strategies on the number of nodes in the branch-and-bound tree. The performance profiles are given in Fig. 2. We do not observe any significant differences in the performance profiles showing the CPU time (Fig. 1) and those referring to the number of nodes in the branch-and-bound tree (Fig. 2). Indeed, R2a is again the best performer and R2 is usually second best. Concerning the different characteristics of the problem, once more the impact of the branching rule is greater for $k = 3$ than for $k = 5$, and is less evident for instances having edge density 25 % and 50 %.

For all subsequent results, we fixed the branching rule to R2a because it usually gives the best results.

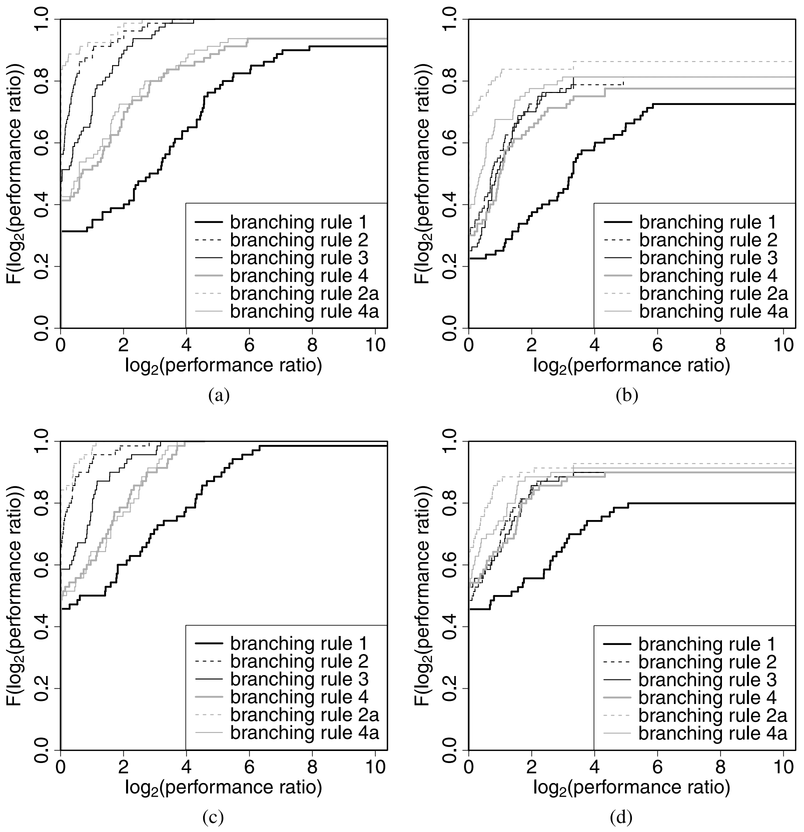


Fig. 1 Performance profiles for the 6 branching rules with respect to the running times for complete graphs (*top profiles*: **a** $k = 3$, edge density 100 %; **b** $k = 5$, edge density 100 %) and graphs with edge density 25 % and 50 % (*bottom profiles*: **c** $k = 3$, edge densities 25 % and 50 %; **d** $k = 5$, edge densities 25 % and 50 %). Edge weights follow a Gaussian or a bimodal distribution

5.3 *Separating Cliques*

In this section, we study the impact of clique separation on the performance of bundleBC. We first compared bundleBC with and without clique separation for $k = 3, 5$ for the benchmark set A of randomly generated graphs with varying edge density. The performance profiles comparing the CPU time and the number of nodes in the branch-and-bound tree for the entire benchmark set A are presented in Fig. 3.

For the bimodal instances we report detailed results in Table 1 and Table 2. (Detailed results for the instances with Gaussian distributed weights with clique separation turned on are reported in Table 4 of Sect. 5.5, where they are used for the comparison with the results from an integer LP model.) Table 1 and Table 2 report the average CPU time (in seconds) and the average number of subproblems (# subs) for solving the instances to optimality. The results in each line correspond to the

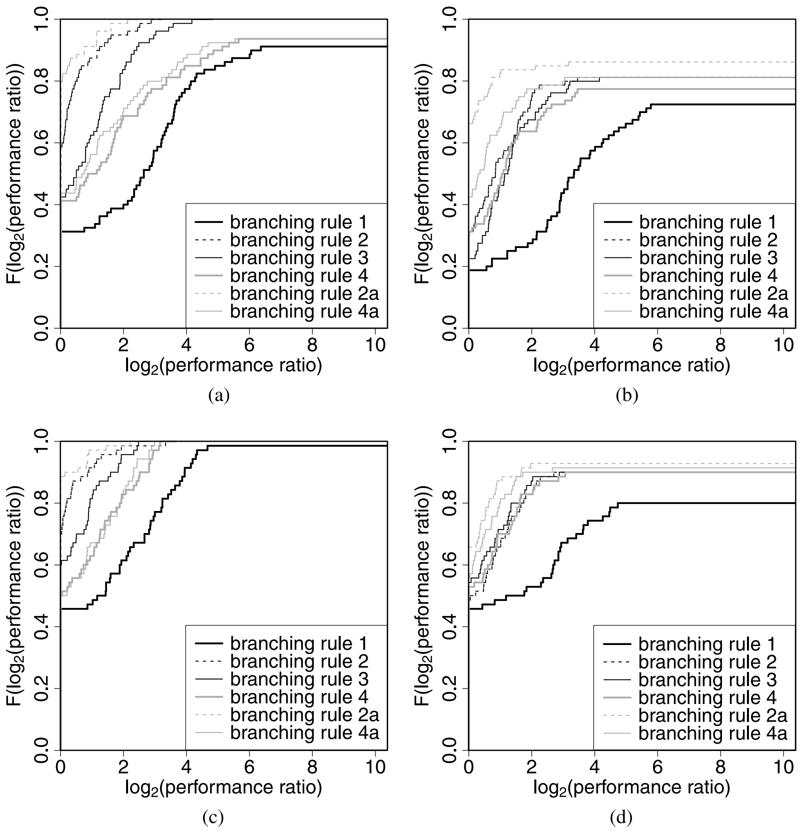


Fig. 2 Performance profiles for the 6 branching rules with respect to the number of nodes in the branch-and-bound tree for complete graphs (*top profiles*: **a** $k = 3$, edge density 100 %; **b** $k = 5$, edge density 100 %) and graphs with edge density 25 % and 50 % (*bottom profiles*: **c** $k = 3$, edge densities 25 %, 50 %; **d** $k = 5$, edge densities 25 %, 50 %). Edge weights follow a Gaussian or a bimodal distribution

average over five different instances or over those instances that could be solved within the time limit. The columns entitled nrinst report the number of instances that could be solved within the time limit.

The profiles in Fig. 3 show that, while run times do not differ significantly, the number of subproblems is smaller when clique separation is turned on. These results are explained by the fact that a subproblem takes longer to solve when clique separation is turned on than when it is turned off. However, investing this additional CPU time may pay off if the bounds are sufficiently tighter to reduce the number of subproblems, and hence the running time of the algorithm. This effect is particularly observable for the bimodal instances in Table 1 and Table 2. We see there that in most cases the number of subproblems is considerably reduced by using clique separation, and the computational time may also improve, especially for instances with edge density 25 % and 50 %. Moreover, there is one instance (of type $|V| = 40$

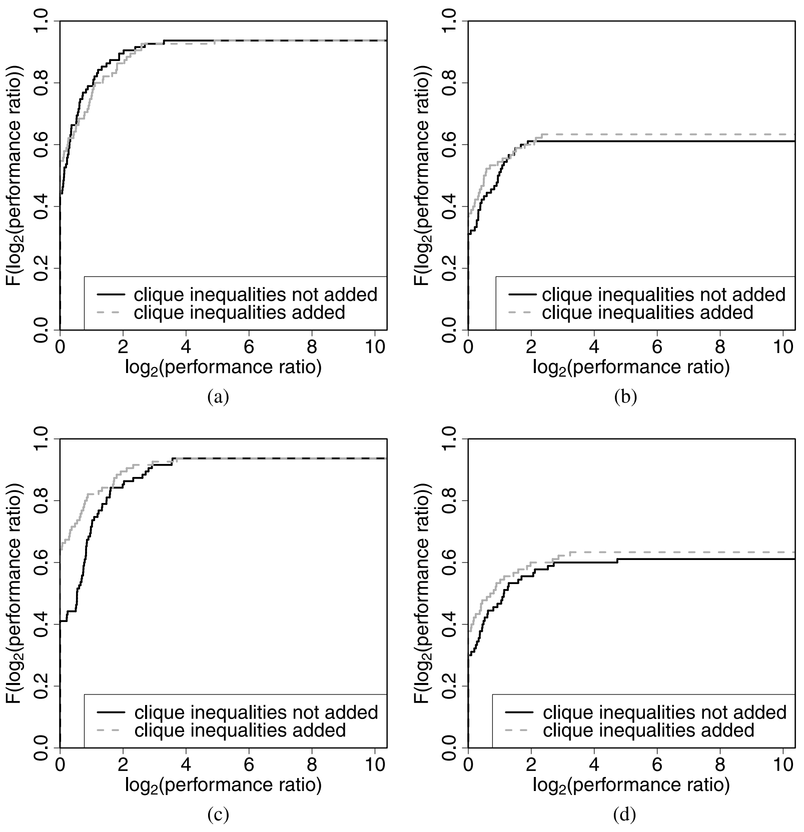


Fig. 3 Performance profiles concerning clique separation with respect to running time (*top profiles*: **a** CPU time, $k = 3$; **b** CPU time, $k = 5$) and the number of subproblems (*bottom profiles*: **c** Number of subproblems, $k = 3$; **d** Number of subproblems, $k = 5$) for the entire benchmark set A

with 50 % edge density) that can only be solved within the time limit when clique separation is used.

We also compared bundleBC with and without clique separation for $k = 3, 5, 7$ for the benchmark set B from [42]. The detailed results are reported in Table 3. These results support the same conclusions, namely that the number of subproblems is often reduced when clique separation is turned on, and the computational time frequently improved. Furthermore, three of the instances can only be solved within the time limit if clique separation is turned on (data_2g_8_37, data_2g_8_648, and data_random_40_k=3).

In summary, there are several instances for which the number of subproblems is not improved by the use of cliques in bundleBC. Although the heuristic separation of cliques is fast, for these cases the overall running time is obviously longer than without using cliques. On the other hand, for several instances the separation of cliques reduces the number of subproblems considerably. Furthermore, certain

Table 1 Results for instances of benchmark set A with 100 % edge density and edge weights following a bimodal distribution

$\|V\|$	k	Without cliques			With cliques		
		Time (sec)	# subs	nrinst	Time (sec)	# subs	nrinst
10	3	0.0	1.0	5	0.0	1.0	5
20	3	0.6	3.4	5	2.2	4.2	5
30	3	35.6	45.0	5	35.6	29.4	5
40	3	377.2	225.4	5	728.4	279.8	5
50	3	12,925.0	4,340.2	5	10,758.8	2,467.4	5
10	5	0.0	1.0	5	0.0	1.0	5
20	5	7.0	14.6	5	9.0	14.6	5
30	5	818.6	666.2	5	640.8	473.8	5
40	5	28,198.0	12,076.0	2	23,435.0	9,457.0	2

Table 2 Results for instances of benchmark set A with varying edge density (25 % and 50 %) and edge weights following a bimodal distribution

$\|V\|$	Edge density in %	k	Without cliques			With cliques		
			Time (sec)	# subs	nrinst	Time (sec)	# subs	nrinst
10	50	3	0.0	1.0	5	0.0	1.0	5
20	25	3	0.0	1.4	5	0.0	1.4	5
20	50	3	0.8	3.0	5	0.2	2.2	5
30	25	3	9.4	5.8	5	8.8	3.8	5
30	50	3	26.2	28.2	5	22.4	15	5
40	25	3	169.4	60.6	5	151.6	37.8	5
40	50	3	479.0	221.0	5	846.8	185.4	5
50	25	3	3,326.6	722.2	5	2,201.6	352.6	5
50	50	3	8,851.6	2,630.6	5	8,277.8	1,519.8	5
10	50	5	0.0	1.0	5	0.0	1.0	5
20	25	5	0.2	1.4	5	0.0	1.4	5
20	50	5	8.8	13.4	5	14.4	19.4	5
30	25	5	18.4	9.4	5	13.2	5.8	5
30	50	5	513.6	368.2	5	321.6	209.8	5
40	25	5	3,242.8	1,033.8	5	2,865.8	804.6	5
40	50	5	6,844.5	2,092.0	2	18,451.3	5,151.7	3
50	25	5	427.0	47.0	1	1,480.0	185.0	1
50	50	5	–	–	0	–	–	0

Table 3 Number of subproblems and running times for solving the instances of benchmark set B to optimality using bundleBC for $k = 3$, $k = 5$, and $k = 7$. Instances that could not be solved to optimality within the time limit are indicated by –

Instance	Without cliques						With cliques					
	$k = 3$		$k = 5$		$k = 7$		$k = 3$		$k = 5$		$k = 7$	
	CPU	# subs	CPU	# subs	CPU	# subs	CPU	# subs	CPU	# subs	CPU	# subs
data_2g_3_93	0	11	0	25	1	45	0	11	0	25	0	43
data_2g_4_164	0	3	5	601	1	7	1	21	16	359	18	1,059
data_2g_5_25	2	7	17	29	3	3	6	47	38	57	8	7
data_2g_6_366	19	27	6	5	191	81	6	45	24	13	37	11
data_2g_6_66	25	15	245	77	87	35	30	21	77	23	72	21
data_2g_6_701	7	17	6	3	5	3	9	47	21	13	6	3
data_2g_7_1034	13	3	388	85	833	243	33	7	186	39	177	23
data_2g_7_491	6	3	22,661	17,339	–	–	5	3	5,724	3,623	–	–
data_2g_7_77	27	7	127	29	567	91	44	17	193	29	52	11
data_2g_8_37	792	47	1,979	261	–	–	111	9	2,355	231	30,370	6,703
data_2g_8_648	381	29	141	7	–	–	104	5	3,035	389	169	9
data_2g_8_88	28	7	2,595	253	393	33	39	7	89	7	1,439	113
data_2g_9_819	214	9	1,128	27	12,874	627	380	17	1,106	39	150	5
data_2g_9_9211	1,768	75	523	13	2,418	65	169	5	2,246	51	11,607	691
data_2g_10_1001	106	5	–	–	–	–	182	7	–	–	–	–
data_2g_10_824	104	5	–	–	–	–	201	11	–	–	–	–
data_2pm_4_44	0	1	1	0	0	1	0	1	0	1	0	1

Table 3 (*Continued*)

Instance	Without cliques						With cliques					
	$k = 3$		$k = 5$		$k = 7$		$k = 3$		$k = 5$		$k = 7$	
	CPU	# subs	CPU	# subs	CPU	# subs	CPU	# subs	CPU	# subs	CPU	# subs
data_2pm_5_55	0	1	0	1	0	1	0	1	0	1	0	1
data_2pm_6_66	16	5	10	3	7	1	11	3	11	3	11	3
data_2pm_7_777	1	1	115	17	93	13	0	1	266	29	56	7
data_2pm_8_888	254	21	493	21	309	11	116	7	245	11	303	13
data_2pm_9_999	576	21	7,215	337	3,369	127	789	29	2,507	51	963	23
data_3g_234_234	0	3	28	47	4	5	0	3	6	7	1	3
data_3g_244_244	16	15	11	7	9	7	20	11	19	17	45	43
data_3g_333_333	1	5	10	7	21	25	0	3	26	21	13	11
data_3g_334_334	23	27	84	27	51	9	16	19	40	13	36	11
data_3g_344_344	4	13	456	119	66	5	4	13	48	5	89	7
data_3g_444_444	198	7	979	37	1,678	81	222	11	1,548	73	868	29
data_3pm_234_234	0	1	0	1	0	1	0	1	0	1	0	1
data_3pm_244_244	3	1	6	1	29	7	3	1	6	1	12	7
data_3pm_333_333	3	7	0	1	0	1	0	3	0	1	0	1
data_3pm_334_334	39	13	156	19	81	9	8	3	141	35	91	27
data_3pm_444_444	11,094	887	11,955	341	11,315	303	23,667	875	12,956	429	9,843	423
data_3pm_344_344	160	21	1,618	205	1,240	191	102	13	2,269	349	2,493	413

Table 3 (*Continued*)

Instance	Without cliques						With cliques					
	$k = 3$		$k = 5$		$k = 7$		$k = 3$		$k = 5$		$k = 7$	
	CPU	# subs	CPU	# subs	CPU	# subs	CPU	# subs	CPU	# subs	CPU	# subs
data_3pm_345_345	162	11	3,787	303	576	41	625	53	909	39	724	57
data_clique_20	0	1	0	1	3	21	0	1	0	1	1	9
data_clique_30	0	1	4	7	15	41	0	1	3	7	5	15
data_clique_40	1	3	20	13	3	5	1	3	6	5	13	17
data_clique_50	9	9	4	3	44	27	2	3	17	9	76	45
data_clique_60	3	3	7	3	91	33	4	3	9	3	79	29
data_clique_70	36	11	17	3	152	31	20	7	78	13	276	53
data_random_20_k=3	0	11	1	1	9	51	0	3	0	1	13	67
data_random_30_k=2	39	275	614	3,231	340	1,111	32	217	588	2,673	269	825
data_random_30_k=3	32	317	1,009	5,671	944	3,339	10	73	932	3,991	1,104	3,327
data_random_40_k=2	255	1,493	–	–	2,889	5,747	130	521	–	–	4,994	9,653
data_random_40_k=3	29	145	9,057	23,877	–	–	32	137	7,925	18,243	32,168	57,489
data_random_50_k=2	2,606	8,267	–	–	–	–	5,248	14,337	–	–	–	–
data_random_50_k=3	5,248	14,337	–	–	–	–	3,963	10,741	–	–	–	–

instances can only be solved within the time limit if clique separation is used. We conclude that it is beneficial to use clique separation in bundleBC, and we use it for all subsequent computations.

5.4 Comparison with SBC

In this section, we present some comparisons of the performance of bundleBC with that of SBC [22]. For this purpose, we use the results in Table 3 where we solved the instances of benchmark set B to optimality for $k = 3$, $k = 5$, and $k = 7$ using bundleBC. In contrast to SBC, we now solve the SDP-relaxations approximately in a shorter time using a bundle method. It is therefore interesting to compare SBC and bundleBC in terms of the quality of the bounds as well as the CPU times necessary to compute them. We use the percentage gap calculated with respect to the value of the best known primal solution prim as the ratio

$$\text{gap} = \left| \frac{\text{bound at root} - \text{prim}}{\text{prim}} \right|.$$

We first comment on the quality of the bounds and the running times of bundleBC at the root node only, i.e., before any branching is done. The root node bounds for SBC are typically zero or close to zero for two- and three-dimensional grids so that branching rarely takes place [22]. However, their computation can take very long. For example, it took SBC more than ten hours to reach a gap of 1 % at the root node for an instance with $k = 3$ on a 10^2 grid with Gaussian distributed edge weights (Table 5 in [22]), while bundleBC solved the instances from [42] defined on 10^2 grids to optimality within at most 4 minutes and needing only up to 11 subproblems (see Table 3). Hence, while the bounds computed by bundleBC are weaker than those determined by SBC, their calculation is much faster and this makes them more useful within a branch-and-bound procedure.

Turning to the solution of instances to optimality, we recall from [22] that SBC almost never has to branch. However, several instances could not be solved within 24 hours of computation time. While it is not surprising that bundleBC branches more often than SBC, bundleBC gains from the fact that the number of subproblems is often in the order of several hundreds only for this set of instances. Because computing one subproblem is much faster for bundleBC than for SBC for small k, bundleBC achieves a drastic speedup over SBC. On the other hand, the instances get more difficult for bundleBC as k increases, unlike what we observed for SBC in [22].

Table 4 Results for the instances of benchmark set C

$\|V\|$	$\|E\|$	k	bundleBC		
			Time (sec)	# subs	nrinst
30	200	3	60.6	99.0	3
30	300	3	41.0	102.3	3
30	400	3	13.6	50.3	3
50	560	3	17,497.5	10,167.0	2
30	200	6	472.7	681.0	3
30	300	6	324.3	558.3	3
30	400	6	627.7	1,691.0	3
50	560	6	–	–	0
30	200	9	–	–	0
30	300	9	1,514.7	2,610.3	3
30	400	9	1,181.3	2,554.3	3
50	560	9	–	–	0
30	200	12	–	–	0
30	300	12	–	–	0
30	400	12	462.0	898.3	3
50	560	12	–	–	0

5.5 *Comparison with the Orbitopal Fixing Approach of Kaibel et al.*

Kaibel et al. [29] report experimental results on max-k-cut instances for an LP-based branch-and-cut algorithm using orbitopal fixing (OF). Furthermore, some of the instances in [42] were addressed in the more recent paper [30]. In particular, several of the Gaussian distributed instances defined on regular grids were solved using OF in very short computing times.

We evaluated bundleBC (with clique separation active) on the benchmark set C of instances from [29]. Our results are reported in Table 4, where the first three columns indicate for each line the number of nodes ($|V|$), the number of edges ($|E|$) and the value of k of the instances averaged. The subsequent two columns report the number of subproblems and the CPU time of bundleBC for solving the instances to optimality. Similarly to what was done in [29], averages were taken over three instances for each row unless some of the instances could not be solved to optimality within the time limit, in which case the number of instances over which the average is taken is denoted in the last column.

We first compare the performance of bundleBC with that reported in [29]. The instances classified as 'easy' in [29] with $|V| = 30$, $|E| = 200$ need almost always zero time with the OF approach for all considered values of k, whereas for bundleBC not all of these instances are easy. In fact, for $k = 9$ and $k = 12$, bundleBC timed out

Table 5 Results for instances of benchmark set A with 100 % edge density and edge weights following a Gaussian distribution

$\|V\|$	k	CPLEX		bundleBC		
		Real time (sec)	nrinst	Real time (sec)	# subs	nrinst
10	3	0.03	5	0.0	1.0	5
20	3	17.96	5	4.2	15.8	5
30	3	8,113.31	5	72.4	38.2	5
40	3	–	0	375.6	119.8	5
50	3	–	0	8,228.0	1,582.3	3
10	5	0.14	5	0.2	7.0	5
20	5	6,674.24	3	16.8	37.4	5
30	5	–	0	1,126.8	807.0	5
40	5	–	0	11,489.0	2,919.0	3

on all three instances. On the other hand, for the instances with $|V| = 30$, $|E| = 400$ that are denser and more difficult for [29], it is clear that bundleBC needs fewer subproblems than OF. Moreover, the average number of subproblems for OF are 9,864 ($k = 3$), 159,298 ($k = 6$), 70,844 ($k = 9$), and 2,098 ($k = 12$). The fact that for bundleBC these numbers are almost always at least one order of magnitude lower shows that our bounds are considerably stronger than those generated by OF.

Computation times are trickier to compare because we used different modern machines from [29]. Nevertheless, it seems that for $k = 3$ and the medium-sized instances with 30 nodes our approach performs better than OF, especially for denser graphs. On the other hand, their performance is better for larger values of k as OF can exploit symmetries well, while these instances are more difficult for bundleBC. Finally, we cannot solve the most difficult instances from [29] that can be solved within several hours of computing time with OF.

Even though we do not have the implementation of the algorithm from [29] at hand, we implemented their integer LP model with variables x_{ip} specifying whether node i is contained in partition p or not. This formulation is polynomially-sized in the x_{ip} variables. As a comparison with our approach, we solve this integer LP problem with CPLEX 12.1 using the standard parameter settings and 6 cores per job. For each job a real time limit of 10 hours was imposed. For larger values of k, the integer LP model is not effective in practice because of the symmetries that are present. However, for small values of k such as $k = 3$ and $k = 5$, the speedup achieved by using OF is not so significant and hence the results are likely more representative of the behavior of OF [29].

We used our benchmark set A of instances to test this model. The results for both CPLEX and bundleBC on the subset of instances that have Gaussian distributed edge weights are reported in Table 5 and Table 6, where we average for each row over 5 instances or the number of instances that finished within the time limit. It turned out that CPLEX ran out of memory even for relatively small instances; for example, for an instance with $k = 3$ and $|V| = 40$, CPLEX ran out of memory after

Table 6 Results for instances of benchmark set A with varying edge density (25 % and 50 %) and edge weights following a Gaussian distribution

$\|V\|$	Edge density in %	k	CPLEX		bundleBC		
			Time (sec)	nrinst	Time (sec)	# subs	nrinst
10	50	3	0.02	5	0.0	20.6	5
20	25	3	0.03	5	1.0	17.0	5
20	50	3	1.69	5	4.6	30.6	5
30	25	3	2.61	5	11.4	8.2	5
30	50	3	184.48	5	24.6	18.2	5
40	25	3	95.90	5	108.8	147.8	5
40	50	3	–	0	265.0	455.0	5
50	25	3	9,763.68	3	2,833.4	5,774.2	5
50	50	3	–	0	10,237.6	17,323.8	5
10	50	5	0.03	5	0.4	80.2	5
20	25	5	0.08	5	16.0	174.6	5
20	50	5	7.07	5	4.4	9.4	5
30	25	5	85.94	5	50.6	28.6	5
30	50	5	–	0	124.2	67.8	5
40	25	5	–	0	1,609.2	1,672.2	5
40	50	5	–	0	2,081.6	606.6	5
50	25	5	–	0	3,780.7	24,272.3	3
50	50	5	–	0	–	–	0

more than 8000 seconds and still had a gap of 43 %. The results of CPLEX for the instances with bimodal edge weights have a similar outcome, as shown in Table 7 and Table 8. This is in marked contrast with bundleBC that solved many of these instances within a few minutes to optimality.

6 Conclusion

We extended the SBC algorithm of Ghaddar, Anjos and Liers for minimum k-partition using the design principles of the successful BiqMac solver for maximum 2-cut to obtain bundleBC, a new algorithm for computing global optimal solutions for maximum k-cut problems. As part of this extension, we investigated different ways of choosing variables for branching. We also studied the impact of the separation of clique inequalities within this new framework and observed that it frequently reduces the number of subproblems considerably. The computational results suggest that bundleBC achieves a significant speedup in comparison to SBC, especially when $k = 3$. A comparison with the results reported from the application of the OF technique by Kaibel, Peinhardt and Pfetsch suggests that while their performance

Table 7 Results for instances of benchmark set A with 100 % edge density and edge weights following a bimodal distribution. The results for solving these instances with bundleBC are given in Table 1

$\|V\|$	k	CPLEX	
		Time (sec)	nrinst
10	3	0.03	5
20	3	88.76	5
30	3	6,053.06	5
40	3	–	0
50	3	–	0
10	5	0.26	5
20	5	–	0
30	5	–	0
40	5	–	0

Table 8 Results for instances of benchmark set A with varying edge density (25 % and 50 %) and edge weights following a bimodal distribution. The results for solving these instances with bundleBC are given in Table 2

$\|V\|$	Edge density in %	k	CPLEX	
			Time (sec)	nrinst
10	50	3	0.02	5
20	25	3	0.05	5
20	50	3	2.40	5
30	25	3	4.48	5
30	50	3	1,215.32	5
40	25	3	1,504.37	5
40	50	3	–	0
50	25	3	29,817.4	1
50	50	3	–	0
10	50	5	0.04	5
20	25	5	0.27	5
20	50	5	405.55	5
30	25	5	1,624.80	4
30	50	5	4,459.24	1
40	25	5	5,299.18	3
40	50	5	–	0
50	25	5	–	0
50	50	5	–	0

is better for sparse instances and larger values of k, bundleBC is superior for $k = 3$ and for dense instances of medium size. Solving the ILP formulation for max-k-cut used by Kaibel, Peinhardt and Pfetsch with CPLEX clearly demonstrates the advantage of our semidefinite approach. The strength of bundleBC is especially evident on dense instances and small values of k.

Acknowledgements We are grateful to Vera Schmitz for providing us with her implementation of a generic branch-and-bound procedure and to Andreas Schmutzer for help with various aspects of the implementation. We thank Brian Borchers and Christoph Helmberg for support with CSDP and Conic Bundle respectively. We thank an anonymous referee for detailed criticism that helped improve the paper. We also thank Matthias Peinhardt for providing us with data for the instances from [29]. Finally we acknowledge the financial support of the German Science Foundation under contract Li 1675/1 and of the Natural Sciences and Engineering Research Council of Canada.

References

1. Anjos, M.F., Wolkowicz, H.: Geometry of semidefinite max-cut relaxations via matrix ranks. J. Comb. Optim. **6**(3), 237–270 (2002)
2. Anjos, M.F., Wolkowicz, H.: Strengthened semidefinite relaxations via a second lifting for the max-cut problem. Discrete Appl. Math. **119**(1–2), 79–106 (2002)
3. Anjos, M.F., Liers, F., Pardella, G., Schmutzer, A.: Engineering branch-and-cut algorithms for the equicut problem. Cahier du GERAD G-2012-15, GERAD, Montreal, QC, Canada (2012). In: Fields Institute Communications on Discrete Geometry and Optimization. Springer, Berlin (2013, to appear)
4. Armbruster, M., Fügenschuh, M., Helmberg, C., Martin, A.L.: LP and SDP branch-and-cut algorithms for the minimum graph bisection problem: a computational comparison. Math. Program. Comput. **4**(3), 275–306 (2012)
5. Barahona, F., Mahjoub, A.: On the cut polytope. Math. Program. **36**, 157–173 (1986)
6. Barahona, F., Grötschel, M., Jünger, M., Reinelt, G.: An application of combinatorial optimization to statistical physics and circuit layout design. Oper. Res. **36**, 493–513 (1988)
7. BiqMac solver. biqmac.uni-klu.ac.at. Accessed 07 June 2012
8. Borchers, B.: CSDP, a C library for semidefinite programming. Optim. Methods Softw. **11/12**(1–4), 613–623 (1999)
9. Boros, E., Hammer, P.: The max-cut problem and quadratic 0–1 optimization: polyhedral aspects, relaxations and bounds. Ann. Oper. Res. **33**, 151–180 (1991)
10. Brunetta, L., Conforti, M., Rinaldi, G.: A branch-and-cut algorithm for the equicut problem. Math. Program., Ser. B **78**(2), 243–263 (1997)
11. Chopra, S., Rao, M.R.: The partition problem. Math. Program. **59**, 87–115 (1993)
12. Chopra, S., Rao, M.R.: Facets of the k-partition problem. Discrete Appl. Math. **61**, 27–48 (1995)
13. Conic Bundle Library. www-user.tu-chemnitz.de/~helmberg/ConicBundle/. Accessed 28 October 2011
14. de Klerk, E., Pasechnik, D., Warners, J.: On approximate graph colouring and max-k-cut algorithms based on the ϑ-function. J. Comb. Optim. **8**(3), 267–294 (2004)
15. Deza, M., Laurent, M.: Geometry of Cuts and Metrics. Algorithms and Combinatorics. Springer, Berlin (1997)
16. Deza, M., Grötschel, M., Laurent, M.: Complete descriptions of small multicut polytopes. In: Applied Geometry and Discrete Mathematics—The Victor Klee Festschrift, pp. 205–220, Am. Math. Soc., Providence (1991)
17. Dolan, E., Moré, J.: Benchmarking optimization software with performance profiles. Math. Program., Ser. A, **91**(2), 201–213 (2002)
18. Eisenblätter, A.: The semidefinite relaxation of the k-partition polytope is strong. In: Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization. Lecture Notes in Computer Science, vol. 2337, pp. 273–290. Springer, Berlin (2002)
19. Elf, M., Jünger, M., Rinaldi, G.: Minimizing breaks by maximizing cuts. Oper. Res. Lett. **31**(5), 343–349 (2003)

20. Fischer, I., Gruber, G., Rendl, F., Sotirov, R.: Computational experience with a bundle approach for semidefinite cutting plane relaxations of max-cut and equipartition. Math. Program., Ser. B **105**(2–3), 451–469 (2006)
21. Frieze, A., Jerrum, M.: Improved approximation algorithms for max k-cut and max bisection. Algorithmica **18**, 67–81 (1997)
22. Ghaddar, B., Anjos, M.F., Liers, F.: A branch-and-cut algorithm based on semidefinite programming for the minimum k-partition problem. Ann. Oper. Res. **188**(1), 155–174 (2011)
23. Goemans, M., Williamson, D.: New $\frac{3}{4}$-approximation algorithms for the maximum satisfiability problem. SIAM J. Discrete Math. **7**(4), 656–666 (1994)
24. Helmberg, C.: A cutting plane algorithm for large scale semidefinite relaxations. In: The Sharpest Cut. MPS/SIAM Ser. Optim., pp. 233–256. SIAM, Philadelphia (2004)
25. Helmberg, C., Kiwiel, K.C.: A spectral bundle method with bounds. Math. Program., Ser. A **93**(2), 173–194 (2002)
26. Helmberg, C., Rendl, F.: Solving quadratic $(0, 1)$-problems by semidefinite programs and cutting planes. Math. Program., Ser. A **82**(3), 291–315 (1998)
27. Helmberg, C., Rendl, F.: A spectral bundle method for semidefinite programming. SIAM J. Optim. **10**(3), 673–696 (2000) (electronic)
28. Helmberg, C., Rendl, F., Vanderbei, R.J., Wolkowicz, H.: An interior-point method for semidefinite programming. SIAM J. Optim. **6**(2), 342–361 (1996)
29. Kaibel, V., Peinhardt, M., Pfetsch, M.: Orbitopal fixing. In: Fischetti, M., Williamson, D. (eds.) Integer Programming and Combinatorial Optimization. Lecture Notes in Computer Science, vol. 4513, pp. 74–88. Springer, Berlin (2007)
30. Kaibel, V., Peinhardt, M., Pfetsch, M.: Orbitopal fixing. Discrete Optim. **8**(4), 595–610 (2011)
31. Kiwiel, K.C.: Methods of Descent for Nondifferentiable Optimization. Lecture Notes in Mathematics, vol. 1133. Springer, Berlin (1985)
32. Laurent, M.: Semidefinite relaxations for max-cut. In: The Sharpest Cut. MPS/SIAM Ser. Optim., pp. 257–290. SIAM, Philadelphia (2004)
33. Laurent, M., Poljak, S.: On a positive semidefinite relaxation of the cut polytope. Linear Algebra Appl. **223/224**, 439–461 (1995)
34. Laurent, M., Poljak, S.: On the facial structure of the set of correlation matrices. SIAM J. Matrix Anal. Appl. **17**(3), 530–547 (1996)
35. Lemaréchal, C.: Bundle methods in nonsmooth optimization. In: Nonsmooth Optimization, Proc. IIASA Workshop, Laxenburg, 1977. IIASA Proc. Ser., vol. 3, pp. 79–102. Pergamon, Oxford (1978)
36. Lemaréchal, C., Nemirovskii, A., Nesterov, Y.: New variants of bundle methods. Math. Program., Ser. B **69**(1), 111–147 (1995)
37. Liers, F., Jünger, M., Reinelt, G., Rinaldi, G.: Computing exact ground states of hard Ising spin glass problems by branch-and-cut. In: New Optimization Algorithms in Physics, pp. 47–68. Wiley, New York (2004)
38. Liers, F., Lukic, J., Marinari, E., Pelissetto, A., Vicari, E.: Zero-temperature behavior of the random-anisotropy model in the strong-anisotropy limit. Phys. Rev. B **76**(17), 174423 (2007)
39. Lisser, A., Rendl, F.: Telecommunication clustering using linear and semidefinite programming. Math. Program. **95**, 91–101 (2003)
40. Margot, F.: Pruning by isomorphism in branch-and-cut. Math. Program., Ser. A, **94**(1), 71–90 (2002)
41. Margot, F.: Exploiting orbits in symmetric ILP. Math. Program., Ser. B, **98**(1–3), 3–21 (2003)
42. Max-k-cut instances. www.eng.uwaterloo.ca/~bghaddar/Publications.htm. Accessed 10 March 2011
43. Mitchell, J.: Branch-and-cut for the k-way equipartition problem. Technical report, Department of Mathematical Sciences, Rensselaer Polytechnic Institute (2001)
44. Mitchell, J.E.: Realignment in the National Football League: did they do it right? Nav. Res. Logist. **50**(7), 683–701 (2003)
45. Palagi, L., Piccialli, V., Rendl, F., Rinaldi, G., Wiegele, A.: Computational approaches to max-cut. In: Handbook on Semidefinite, Conic and Polynomial Optimization. Internat. Ser. Oper.

Res. Management Sci., vol. 166, pp. 821–847. Springer, New York (2012)
46. Poljak, S., Rendl, F.: Solving the max-cut problem using eigenvalues. Discrete Appl. Math. **62**(1–3), 249–278 (1995). doi:10.1016/0166-218X(94)00155-7
47. Rendl, F., Rinaldi, G., Wiegele, A.: Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. Math. Program. **121**, 307–335 (2010)
48. Rinaldi, G.: Rudy. www-user.tu-chemnitz.de/~helmberg/rudy.tar.gz. Accessed 07 April 2010
49. Schramm, H., Zowe, J.: A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. SIAM J. Optim. **2**(1), 121–152 (1992)
50. Spin-glass server. www.informatik.uni-koeln.de/ls_juenger/research/sgs/index.html. Accessed 07 June 2012

On Perspective Functions and Vanishing Constraints in Mixed-Integer Nonlinear Optimal Control

Michael N. Jung, Christian Kirches, and Sebastian Sager

Abstract Logical implications appear in a number of important mixed-integer nonlinear optimal control problems (MIOCPs). Mathematical optimization offers a variety of different formulations that are equivalent for boolean variables, but result in different relaxations. In this article we give an overview over a variety of different modeling approaches, including outer versus inner convexification, generalized disjunctive programming, and vanishing constraints. In addition to the tightness of the respective relaxations, we also address the issue of constraint qualification and the behavior of computational methods for some formulations. As a benchmark, we formulate a truck cruise control problem with logical implications resulting from gear-choice specific constraints. We provide this benchmark problem in AMPL format along with different realistic scenarios. Computational results for this benchmark are used to investigate feasibility gaps, integer feasibility gaps, quality of local solutions, and well-behavedness of the presented reformulations of the benchmark problem. Vanishing constraints give the most satisfactory results.

1 Introduction

Sebastian Sager (in 2006, [56]) and Christian Kirches (in 2010, [39]) received their Ph.D.s from Heidelberg University under the supervision of Gerhard Reinelt, thus

M.N. Jung · C. Kirches
Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, Ruprecht-Karls-Universität Heidelberg, Im Neuenheimer Feld 368, 69120 Heidelberg, Germany

M.N. Jung
e-mail: michael.jung@iwr.uni-heidelberg.de

C. Kirches
e-mail: christian.kirches@iwr.uni-heidelberg.de

S. Sager (✉)
Institut für Mathematische Optimierung, Otto-von-Guericke-Universität Magdeburg, Universitätsplatz 2, 02-224, 39106 Magdeburg, Germany
e-mail: sager@ovgu.de

becoming grand-sons of Martin Grötschel. Michael Jung is a Ph.D. student at Heidelberg University supervised by Reinelt and Sager, and expects to become both a grandson and a grand-grandson of Grötschel in 2013. Another interesting link is the scientific and personal socialization of all three authors in the realm of Hans Georg Bock, who, like Grötschel, is a protagonist of the University of Augsburg's golden age of applied mathematics in the late eighties.

Mixed-integer optimal control problems (MIOCPs) have been gaining significantly increased interest over the last years. This is due to the fact that the underlying processes have a high potential for optimization, while at the same time they are hard to assess manually due to their combinatorial, nonlinear, and dynamic nature. Typical examples are the choice of gears in automotive control [26, 40], water or gas networks [14, 49], traffic flow [23, 32], supply chain networks [29], distributed autonomous systems [1], and processes in chemical engineering that involve valves [38, 65]. The truck benchmark problem we present in this article is motivated by work of [35, 39, 68] on heavy duty trucks. See [59] for an open benchmark library for MIOCPs. In this article, we are especially interested in switching decisions that imply constraints. In the interest of readability we focus on the specific problem class of MIOCPs in ordinary differential equations (ODEs) of the following form.

Definition 1 (MIOCP) In this contribution a mixed-integer optimal control problem (MIOCP) refers to the switched system on a fixed time horizon $[0, t_\mathrm{f}]$ given by

$$\begin{aligned}
&\min_{x(\cdot),u(\cdot),Y(\cdot)} \quad e\bigl(x(t_\mathrm{f})\bigr) \\
&\text{s.t.} \quad \bigvee_{i\in\{1,\dots,n_\omega\}} \begin{bmatrix} Y_i(t) \\ \dot{x}(t) = f(x(t), u(t), v^i) \\ 0 \le c(x(t), u(t), v^i) \end{bmatrix} \quad \forall t \in [0, t_\mathrm{f}], \\
&\qquad x(0) = x_0, \\
&\qquad 0 \le d\bigl(x(t), u(t)\bigr) \quad \forall t \in [0, t_\mathrm{f}].
\end{aligned} \tag{1}$$

The logical operator $\bigvee_{i\in\{1,\dots,n_\omega\}}$ implies that at all times $t \in [0, t_\mathrm{f}]$ exactly one of the n_ω possible modes is chosen and is represented here by time-dependent logical literals $Y_i(\cdot)$, $1 \le i \le n_\omega$. The optimal control $u : [0, t_\mathrm{f}] \to \mathbb{R}^{n_u}$ is assumed to be measurable and of bounded variation, the differential states $x : [0, t_\mathrm{f}] \to \mathbb{R}^{n_x}$ to be uniquely determined once a switching regime $Y(\cdot)$ and the controls $u(\cdot)$ are fixed. The vectors $v^i \in \mathbb{R}^{n_v}$ comprise constant values that are specific for the given mode. We write $\Omega := \{v^1, v^2, \dots, v^{n_\omega}\}$. The objective function $e : \mathbb{R}^{n_x} \to \mathbb{R}$ of Mayer type and the constraint functions $c : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_c}$ and $d : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^d$ are assumed to be sufficiently often continuously differentiable.

For example, in Sect. 2 the vectors v^i will denote degrees of efficiency and engine speed bounds for a particular choice i of the gear. Note that problem class (1) can be generalized in many directions to include different objective functionals, multipoint constraints, algebraic variables, more general hybrid systems, and the likes, compare [57] for further details and references.

Although the first MIOCPs, namely the optimization of subway trains that are equipped with discrete acceleration stages, were already solved in the early eighties for the city of New York [12], the so-called *indirect methods* used there do not seem appropriate for generic optimal control problems with underlying nonlinear dynamic systems and mixed path-control constraints. The discussion of advantages and disadvantages of indirect approaches is ongoing since the 1980s and beyond the scope of this paper. See [57] for a discussion of the applicability of dynamic programming and global maximum principles to MIOCP and further references.

Direct and all-at-once methods, such as Bock's direct multiple shooting [13, 44] and direct collocation [5, 9, 10] have emerged as the methods of choice for the computational solution of many purely continuous optimal control problems. They are also at the heart of our approach to MIOCP. These approaches have in common that they result in highly structured nonlinear programs (NLPs). As the focus of this article is on properties of relaxed MIOCPs, we do not go into details on how these transformations are carried out, but rather refer the interested reader to [11, 44]. Nevertheless, we need some NLP definitions to illustrate and transfer important concepts.

Definition 2 (NLP) A nonlinear programming problem (NLP) is given by

$$\min_{y \in \mathbb{R}^{n_y}} E(y) \quad \text{s.t.} \quad C(y) = 0, \qquad D(y) \geq 0 \tag{2}$$

with twice continuously differentiable functions $E : \mathbb{R}^{n_y} \to \mathbb{R}$, $C : \mathbb{R}^{n_y} \to \mathbb{R}^{n_C}$, and $D : \mathbb{R}^{n_y} \to \mathbb{R}^{n_D}$.

Note that we distinguish between control problems and nonlinear programs by using lower and upper case letters respectively, and make use of the unusual E for the objective function to avoid confusion with the right hand side function $f(\cdot)$.

Definition 3 (LICQ) A nonlinear programming problem is said to satisfy the *linear independence constraint qualification* (LICQ) in a feasible point $\bar{y} \in \mathbb{R}^{n_y}$ if for all $d \neq 0 \in \mathbb{R}^{n_y}$ it holds that

$$\begin{aligned} \nabla C_i(\bar{y})^T d &\neq 0, \quad 1 \leq i \leq n_C, \\ \nabla D_i(\bar{y})^T d &\neq 0, \quad i \in \left\{1 \leq j \leq n_D : D_j(\bar{y}) = 0\right\}. \end{aligned} \tag{3}$$

In Sect. 4 we will see that certain constraint formulations do not enjoy the LICQ property. Thus, we discuss mathematical programs with complementarity structure.

Definition 4 (MPCC, MPVC) A nonlinear programming problem

$$\begin{aligned} &\min_{y\in\mathbb{R}^{n_y}} E(y) \\ &\text{s.t.} \quad G(y) \geq 0, \qquad H(y) \geq 0, \qquad G_i(y)\cdot H_i(y) = 0, \quad 1 \leq i \leq n_{GH}, \end{aligned} \tag{4}$$

with twice continuously differentiable functions E, G, and H is called a *Mathematical Program with Complementarity Constraints*, e.g., [7]. An NLP

$$\min_{y\in\mathbb{R}^{n_y}} E(y) \quad \text{s.t.} \quad H(y) \geq 0, \qquad G_i(y)\cdot H_i(y) \leq 0, \quad 1 \leq i \leq n_{GH}, \tag{5}$$

with twice continuously differentiable functions F, G, and H is called a *Mathematical Program with Vanishing Constraints*, see e.g., [2].

MPCCs and MPVCs are known to possess critical points that violate LICQ. Their computational solution with standard NLP software is prone to numerical difficulties and will often terminate in suboptimal points that possess trivial descent directions and are not local minimizers of (4) or (5).

Remark 1 (MINLP view of MPVCs) One way to see the difficulty is a look at the mixed-integer nonlinear program given by

$$\min_{y\in\mathbb{R}^{n_y}, z\in\{0,1\}} E(y) \quad \text{s.t.} \quad G(y) \leq zb \tag{6}$$

with G such that $\{y \in \mathbb{R}^{n_y} | G(y) \leq 0\} = \{0\}$. It is closely related to problem (5) as $H_i(y)$ toggles the constraint on $H_i(y)\cdot G_i(y)$ in (5) like z does for the constraint on $G(y) - b$ in (6):

$$\begin{aligned} z = 1, G(y) - b \leq 0 \quad &\Longleftrightarrow \quad H_i(y) > 0, H_i(y)\cdot G_i(y) \leq 0, \\ z = 0, G(y) = 0 \quad &\Longleftrightarrow \quad H_i(y) = 0, H_i(y)\cdot G_i(y) = 0. \end{aligned}$$

In this article, we investigate different approaches to reformulate problem (1). Here, reformulations are optimal control problems that have the same feasible set and hence the same optimal solution if integrality is required, but have a possibly different and larger feasible set if the discrete decisions are relaxed. In Sect. 3 we discuss formulations of the dynamic equations with further references to the literature. With respect to formulations of the inequalities in the logical disjunctions in (1), not only tightness of the relaxation is crucial. Also, constraint qualifications may or may not hold and homotopies might be needed to obtain convergence to local solutions. We refer to this as "well-behavedness" of a relaxation for our benchmark problem.

There are several reformulations of logical relationships in the literature. *Generalized Disjunctive Programming* results directly from a logical modeling paradigm. It generalizes the disjunctive programming approach of [4]. Logical variables are usually either incorporated by means of big-M constraints or via a convex hull formulation, see [31, 50, 51, 67]. From a different point of view, disjunctive programming formulations can be interpreted as the result of *reformulation-linearization technique* (RLT) steps [64]. For both, the convex hull relaxation uses *perspective functions*. Based on this, the use of perspective cuts to strengthen convex MINLP relaxations has been proposed in various articles, for example [15, 22, 33]. We refer the reader to [8] for a recent survey on MINLP techniques.

Complementarity (4) *and Vanishing Constraints* (5) are another way to look at logical implications. The general concept of nonlinear optimization over nonconvex structures is discussed in [62, 63]. For the comparatively young problem class of MPVCs we refer to [2, 36]. Due to the lack of constraint qualification, various approaches for the computational solution of MPCCs and MPVCs have been devised and include regularizations [36, 54, 66], smoothing [16, 36], and combinations thereof; see [24] for an overview. Nonlinear programming for MPCCs is addressed in [3, 21, 45, 47]. Active set methods tailored to the nonconvex structure are discussed in [17, 39, 43]. Formulations of MPCCs and MPVCs in optimal control can be found in [7, 39, 43, 52, 53].

The remainder of this article is organized as follows. In Sect. 2 we describe a heavy-duty truck cruise control problem based on a dynamic vehicle model that includes gear shift decisions. This is a prototypical MIOCP with constraints based on logical implications. In Sect. 3 we explain the *partial outer convexification* approach for MIOCPs. We discuss extensions of this approach and different reformulations arising for the heavy-duty truck model, and numerically assess the advantage of partial outer convexification over an inner convexification. In Sect. 4 we investigate reformulations for logical implication constraints. We discuss merits and drawbacks as well as the consequences arising for underlying NLP solvers. Numerical results for the heavy-duty truck control problem are presented in Sect. 5.

2 A Cruise Control Problem for a Heavy-Duty Truck

In this section we describe a cruise control problem for a heavy-duty truck, and model it as a mixed-integer tracking control problem on a prediction horizon.

Typical heavy-duty trucks feature from $n_\mu = 8$ to $n_\mu = 24$ gears with different transmission ratios and efficiencies. The decision on energy-optimal gear shift strategies under real-time constraints usually requires extensive training on the driver's side, is a subject of intense scientific research, and bears considerable potential for savings especially in view of future hybrid engines, e.g., [35, 39, 68].

Controls and Dynamic System We describe a basic ODE truck model as introduced in [68] that is used for all computations. The ODE system of the truck model comprises three input controls: the indicated engine torque M_{ind}, the engine brakes torque M_{EB}, and the integer gear choice μ.

The vehicle model involves two differential states, velocity v and accumulated fuel consumption Q. Traveled distance s is chosen as the independent variable and we consider the interval $[0, s_\mathrm{f}]$. The first state $v(s)$ denotes velocity (in m/s) and is determined from the sum of directed torques,

$$mv(s)\dot{v}(s) = \big(M_\mathrm{acc}(s) - M_\mathrm{brk}(s)\big)i_\mathrm{A}/r_\mathrm{stat} - M_\mathrm{air}(s) - M_\mathrm{road}(s), \tag{7}$$

depending on the effective accelerating and braking torques M_acc, M_brk as well as on turbulent and static friction torques M_air, M_road (all in Nm). Further, i_A is the fixed axle transmission ratio and r_stat (in m) is the static tire radius. The second state, fuel consumption $Q(s)$ (in l/s), is computed from integration over a consumption map

$$v(s)\dot{Q}(s) = Q_\mathrm{fuel}\big(n_\mathrm{eng}\big(s, \mu(s)\big), M_\mathrm{ind}(s)\big), \tag{8}$$

depending on engine speed n_eng and torque M_ind. Several terms are computed from algebraic formulas. The transmitted engine speed n_eng (in 1/min) depends on the selected gear μ and is obtained from velocity,

$$n_\mathrm{eng}\big(s, \mu(s)\big) := v(s)i_\mathrm{A}i_\mathrm{T}\big(\mu(s)\big)60/(2\pi r_\mathrm{stat}).$$

The accelerating torque M_acc is computed from the ratio $i_\mathrm{T}(\mu)$ and efficiency $\eta_\mathrm{T}(\mu)$ associated with the selected gear μ. Braking torques M_brk are due to controlled engine brake torque M_EB and internal engine friction torque M_fric,

$$\begin{aligned} M_\mathrm{acc}(s) &:= i_\mathrm{T}\big(\mu(s)\big)\eta_\mathrm{T}\big(\mu(s)\big)M_\mathrm{ind}(s), \\ M_\mathrm{brk}(s) &:= M_\mathrm{EB}(s) + i_\mathrm{T}\big(\mu(s)\big)M_\mathrm{fric}\big(n_\mathrm{eng}\big(s, \mu(s)\big)\big). \end{aligned}$$

Additional braking torques due to turbulent friction M_air, and due to static road conditions M_road, are taken into account,

$$M_\mathrm{air}(s) := \frac{1}{2}c_\mathrm{w}A\rho_\mathrm{air}v^2(s), \qquad M_\mathrm{road}(s) := mg\big(\sin\gamma(s) + f_\mathrm{r}\cos\gamma(s)\big).$$

Here c_w is the shape coefficient, A denotes the flow surface (in m^2), and ρ_air the air density (in kg/m^3). Further, m is the vehicle's mass (in kg), gravity is denoted by g (in m/s^2), $\gamma(s)$ denotes the road's slope, and f_r is a rolling friction coefficient.

Objective The cost criterion to be minimized on the prediction horizon $[0, s_\mathrm{f}]$ is composed of a weighted sum of three different objectives. To simplify notation we transform the Lagrange terms into a Mayer term by introducing artificial differential states. First, the deviation of the truck's velocity from the desired one is penalized,

$$v(s)\dot{\Phi}_\mathrm{dev}(s) := \big(v(s) - v_\mathrm{des}(s)\big)^2 \tag{9}$$

Second, the fuel consumption is found from a fuel consumption rate map Q, see [39], and depends on the integer gear choice $\mu(s)$,

$$v(s)\dot{\Phi}_\mathrm{fuel} := Q\big(n_\mathrm{eng}\big(s, \mu(s)\big), M_\mathrm{ind}(s)\big). \tag{10}$$

Third, rapid changes of the engine and brake torques degrade driving comfort,

$$v(s)\dot{\Phi}_{\text{comf}} := \dot{M}^2_{\text{ind}}(s) + \dot{M}^2_{\text{brk}}(s), \tag{11}$$

where we approximate the torque derivatives by one-sided finite differences.

Constraints On the prediction horizon mechanical constraints and velocity limits need to be respected. Beside the bounds on the truck input controls and on the system's states for $s \in [0, s_{\text{f}}]$,

$$0 \le M_{\text{ind}}(s), \qquad 0 \le M_{\text{brk}}(s) \le M_{\text{brk,max}}, \tag{12}$$

the truck's velocity $v(s)$ is subject to velocity limits imposed by law,

$$v(s) \le v_{\text{law}}(s), \quad s \in [0, s_{\text{f}}]. \tag{13}$$

The indicated torque must respect state-dependent upper limits as specified by the engine characteristics for $s \in [0, s_{\text{f}}]$:

$$M_{\text{ind}}(s) \le M_{\text{ind,max}}\big(n_{\text{eng}}\big(s, \mu(s)\big)\big). \tag{14}$$

In addition, the transmitted engine speed $n_{\text{eng}}(s, \mu(s))$ must stay within prescribed limits according to the engine's specification,

$$n_{\text{eng,min}} \le n_{\text{eng}}\big(s, \mu(s)\big) \le n_{\text{eng,max}}, \quad s \in [0, s_{\text{f}}]. \tag{15}$$

Problem Formulation The MIOC problem formulation for the heavy-duty truck control problem on the prediction horizon $s \in [0, s_{\text{f}}]$ reads

$$\begin{aligned} \min_{x(\cdot),u(\cdot),\mu(\cdot)} \quad & \lambda_{\text{dev}}\Phi_{\text{dev}}(s_{\text{f}}) + \lambda_{\text{fuel}}\Phi_{\text{fuel}}(s_{\text{f}}) + \lambda_{\text{comf}}\Phi_{\text{comf}}(s_{\text{f}}) \\ \text{s.t.} \quad & \bigvee_{i \in \{1,\dots,n_\mu\}} \begin{bmatrix} \mu(s) = i \\ \text{ODE system (7), (8)} \\ \text{Constraints (14), (15)} \end{bmatrix} \quad \forall s \in [0, s_{\text{f}}], \\ & x(0) = x_0, \\ & \text{Constraints (12), (13)} \quad \forall s \in [0, s_{\text{f}}], \end{aligned} \tag{16}$$

with state vector $x(s) = (v, Q, \Phi_{\text{dev}}, \Phi_{\text{fuel}}, \Phi_{\text{comf}})(s)$, continuous control vector $u(s) = (M_{\text{ind}}, M_{\text{brk}})(s)$, integer controls $\mu(s)$, and initial state information x_0. Note that problem (16) is a MIOCP of form (1), with the gear choice $\mu(s)$ that causes switches in the right hand side as well as in the constraints and the associated vectors $v^i = (i_{\text{T}}(i), \eta_{\text{T}}(i))$ that contain gear-specific values for transmission ratio and efficiency.

Logical Implications The modeling and optimization with logical variables has gained increasing interest, [31, 50]. The main reason is that it provides a modeling paradigm that is both generic and intuitive, and allows for tailored relaxations and algorithms.

Logical literals $Y \in \{\text{false}, \text{true}\}$ can of course be identified with binary variables $z_i \in \{0, 1\}$, or can be used implicitly, as we did in formulation (16),

$$Y_i(t) = \text{true} \quad \Leftrightarrow \quad \mu(t) = i.$$

It is well known that logical relations and Boolean expressions can be represented as linear (in)equalities. For example, the implication is represented by

$$Y_1 \Rightarrow Y_2 \quad \Longleftrightarrow \quad \neg Y_1 \vee Y_2 \quad \Longleftrightarrow \quad 1 - z_1 + z_2 \geq 1.$$

For time-dependent logical literals $Y(\cdot)$, systems of differential equations, and implied constraints the representation is, however, neither straightforward nor unique.

In Sect. 3 we discuss two different formulations for the ODE system and review recent results. Constraints (14) and (15) are of particular interest in the context of this article. They represent the logical implication

When gear i is active, gear-specific constraints must hold.

In Sect. 4, we address different mathematical reformulations of this implication and investigate the tightness of the induced relaxations.

3 Inner and Outer Convexification in MIOC

In this section we investigate two different approaches to reformulate problem (1) when $n_c = 0$, i.e., when no mode-specific constraints are present and only the right hand side function of the dynamic system depends on the logical mode choice at time t. We denote the two approaches by *inner convexification* and *(partial) outer convexification*, depending on how the variables that represent the logical choice enter the right hand side.

The goal in both cases is identical: a formulation that allows the relaxation of discrete decisions. The relaxed MIOCPs are purely continuous control problems that can be solved using, for example, Bock's direct multiple shooting method or direct collocation.

Using the truck control problem from Sect. 2, we compare both approaches numerically and explain qualitatively why the outer convexification formulation performs so much better than the inner convexification. We close this section by referring to related work and extensions.

Inner Convexification For some control problems of type (1) it is possible to reformulate the time-dependent disjunctions by means of a function $g : [1, n_\omega] \to \mathbb{R}^{n_v}$ that can be inserted into the right hand side function $f(\cdot)$ and has the property

$g(i) = v^i$ for $i \in \{1, \dots, n_\omega\}$. Possibilities are a piecewise linear representation of the form

$$g(i + \xi_{i+1}) = \xi_i v^i + \xi_{i+1} v^{i+1} \tag{17}$$

with special ordered set type 2 (SOS-2) variables

$$\xi_i \in [0, 1], \qquad \sum_i \xi_i = 1, \qquad \xi_i \neq 0 \quad \Rightarrow \quad \xi_j = 0 \quad \forall j \neq i, i+1,$$

a convex combination

$$g\left(\sum_{i=1}^{n_\omega} \xi_i i\right) = \sum_{i=1}^{n_\omega} \xi_i v^i \tag{18}$$

with special ordered set type 1 (SOS-1) variables $\xi_i \in [0, 1]$, $\sum_i \xi_i = 1$, or fitted smooth convex functions $g(\cdot)$ as suggested in [25]. Either way, this approach allows a reformulation of (1) into

Definition 5 (MIOCP after IC Reformulation)

$$\begin{aligned} \min_{x(\cdot), u(\cdot), \psi(\cdot)} \quad & e\big(x(t_\mathrm{f})\big) \\ \text{s.t.} \quad & \dot{x}(t) = f\big(x(t), u(t), g(\psi)\big) \quad \forall t \in [0, t_\mathrm{f}], \\ & 0 \le d\big(x(t), u(t)\big) \quad \forall t \in [0, t_\mathrm{f}], \\ & x(0) = x_0, \\ & \psi(t) \in \{1, \dots, n_\omega\} \quad \forall t \in [0, t_\mathrm{f}]. \end{aligned} \tag{19}$$

By construction, problem (19) can be relaxed toward $\psi(t) \in [1, n_\omega]$.

Outer Convexification The outer convexification approach (or *partial* outer convexification, because the convexification applies to the integer controls only, and not to the rest of the control problem) has been investigated in the context of optimal control in [56, 60, 61]. It consists of an evaluation of all possible right hand sides, their multiplication with convex multipliers, and the summation of the products. We introduce control functions $\omega : [0, t_\mathrm{f}] \to \{0, 1\}^{n_\omega}$ as convex multipliers and obtain:

Definition 6 (MIOCP after OC Reformulation)

$$\begin{aligned} \min_{x(\cdot), u(\cdot), \omega(\cdot)} \quad & e\big(x(t_\mathrm{f})\big) \\ \text{s.t.} \quad & \dot{x}(t) = \sum_{i=1}^{n_\omega} \omega_i(t) f\big(x(t), u(t), v^i\big) \quad \forall t \in [0, t_\mathrm{f}], \\ & 0 \le d\big(x(t), u(t)\big) \quad \forall t \in [0, t_\mathrm{f}], \end{aligned} \tag{20}$$

$$x(0) = x_0$$

$$1 = \sum_{i=1}^{n_\omega} \omega_i(t), \quad \omega(t) \in \{0,1\}^{n_\omega} \ \forall t \in [0, t_\mathrm{f}].$$

Problem (20) can be relaxed toward $\omega(t) \in [0,1]^{n_\omega}$. We use the notation $\alpha(t) \in [0,1]^{n_\omega}$ to highlight the difference between the original problem and its continuous relaxation. In the following we need the notation of

Definition 7 (Fractionality of solutions) The *fractionality* of binary control functions $\omega(\cdot)$ on a time horizon $[0, t_\mathrm{f}]$ is given by

$$\frac{1}{n_\omega} \sum_{i=1}^{n_\omega} \int_0^{t_\mathrm{f}} \big(0.5 - \|\omega_i(\tau) - 0.5\|\big) \mathrm{d}\tau.$$

Discussion of Inner Versus Outer Convexification Relaxations of reformulation (19) may be faster to solve, as we only have one control function $\psi : [0, t_\mathrm{f}] \to [1, n_\omega]$ that enters the control problem instead of n_ω functions $\omega_i : [0, t_\mathrm{f}] \to [0,1]$ in (20). Hence, there are less derivatives to be computed and the subproblems arising in iterations of an interior point or SQP approach are cheaper to solve. Moreover in (20), the aggregated right hand side function $\sum_i f(\cdot, v^i)$ may become more expensive to evaluate. As n_ω reflects the number of possible combinations and switches, this number may get large. Note that the linear equality constraint may be used for elimination of one function at the cost of losing some sparsity.

However, depending on the separability properties of $f(\cdot)$, integer controls often decouple, leading to a reduced number n_ω of admissible choices, e.g., [30].

Example 1 (Outer Convexification and Separability) Assume we have

$$\dot{x}(t) = f_1\big(\cdot, v_1(t)\big) + f_2\big(\cdot, v_2(t)\big), \quad v_1(t) \in \Omega_1, v_2(t) \in \Omega_2.$$

Then an equivalent reformulation leading to $n_\omega = n_{\omega_1} + n_{\omega_2}$ controls instead of $n_\omega = n_{\omega_1} n_{\omega_2}$ is given for $t \in [t_0, t_f]$ by

$$\dot{x}(t) = \sum_{i=1}^{n_{\omega_1}} f_1\big(\cdot, v_1^i\big)\omega_{1,i}(t) + \sum_{i=1}^{n_{\omega_2}} f_2\big(\cdot, v_2^i\big)\omega_{2,i}(t),$$

$$\sum_{i=1}^{n_{\omega_1}} \omega_{1,i}(t) = 1, \quad \omega_1 \in \{0,1\}^{n_{\omega_1}}, \qquad \sum_{i=1}^{n_{\omega_2}} \omega_{2,i}(t) = 1, \quad \omega_2 \in \{0,1\}^{n_{\omega_2}}.$$

In most practical applications the binary control functions enter linearly (such as valve controls that indicate whether a certain flow term is present or not), or n_ω increases linearly with the number of choices (e.g., the gears), or integer controls decouple. Hence, one can expect a modest (linear) increase in the number of control functions. This increase is usually more than outweighed by important advantages of (20) over (19):

- Not in all cases can a meaningful inner convexification $g(\cdot)$ be found. An example are black-box simulators that can only be evaluated for certain modes (integer values), but not in between. In general, using $g(\cdot)$ for a relaxation may lead to problems such as divisions by zero, or index changes in the DAE case [6]. Evaluating the model only for vectors v^i avoids these problems;
- The integer gap between the optimal solutions of (19) and its relaxation may become arbitrarily large [56], whereas the integer gap between the optimal solutions of (20) and its relaxation is bounded by a multiple of the control discretization grid size Δt [61].

Finding a tight relaxation of (20) is vitally important for the computational solution of (1). It allows a decoupling of the MIOCP into a continuous OCP and a mixed-integer linear programming problem with a huge potential for computational savings and a posteriori bounds on the gap to the best possible MIOCP solution [37, 61]. Moreover, the relaxed control problems often have optimal integer solutions, which implies almost arbitrary computational savings when compared to an OCP-based branch&bound approach to solve (19) to optimality. This has first been compared in [40] for a benchmark problem posed in [25]. While identical solutions were obtained, a speedup of several orders of magnitude was observed for the outer convexification approach. Similar behavior can be observed when comparing to MINLP-based branch&bound. We use the truck control example from Sect. 2 to reproduce and explain this qualitative result.

Inner and Outer Convexification for Truck Control Optimal solutions by their very nature tend to exploit constraints as much as possible. In the special case of vehicle operation and for a fixed acceleration, it is natural that the gear is chosen that provides the largest torque when compared to all other gears.

In Fig. 1 we show the maximum indicated engine torque $M_{\text{ind,max}}$ depending on the velocity v for two adjacent gear choices $\mu(\cdot) = i$ and $\mu(\cdot) = i + 1$. Figure 1 (left) shows

$$M^{\text{IC}}_{\text{ind,max}}(v) = M_{\text{ind,max}}\big(n_{\text{eng}}\big(v(s), \xi i + (1-\xi)(i+1)\big)\big) \tag{21}$$

for $\xi = 0.0, 0.1, \ldots, 1.0$, while Fig. 1 (right) shows

$$M^{\text{OC}}_{\text{ind,max}}(v) = \alpha M_{\text{ind,max}}\big(n_{\text{eng}}\big(v(s), i\big)\big) + (1-\alpha) M_{\text{ind,max}}\big(n_{\text{eng}}\big(v(s), i+1\big)\big) \tag{22}$$

for $\alpha = 0.0, 0.1, \ldots, 1.0$. Apparently, in the particular case of truck control, the relaxation of (19) comprises combinations of the state variable $v(\cdot)$ and non-integral $\xi \in (0, 1)$ that yield an unphysical, larger value of $M^{\text{IC}}_{\text{ind,max}}$. Conversely, the relaxation of (20) is by construction maximal only for $\alpha \in \{0, 1\}$. Thus, one can expect that optimal solutions of the relaxed version of (20) are integral, whereas solutions of the relaxation of (19) may have non-integral solutions $\xi \in (0, 1)$. The two approaches would only coincide if both n_{eng} and $M_{\text{ind,max}}$ were linear functions

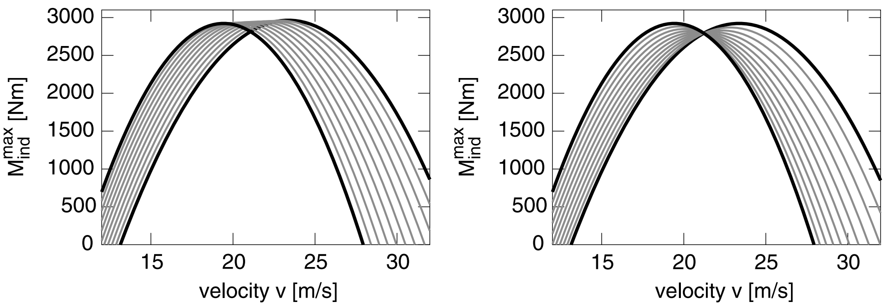


Fig. 1 Maximum indicated engine torque $M_{\text{ind,max}}$ depending on the velocity v for two adjacent gear choices $\mu(\cdot) = i$ and $\mu(\cdot) = i + 1$. *Left*: inner convexification (19). *Right*: outer convexification (20). Observe the maximal indicated engine torques for non-integral values of ξ (*gray*) on *the left hand side*, while non-integral values for α (*gray*) on *the right hand side* are never better than integral values (*black*)

Table 1 *Fractionality* as in Definition 7, *objective* function value, and *CPU* time obtained with SNOPT/IPOPT for the solution of relaxations of (19) and (20) with different control grids

# Control intervals	Inner convexification (19)			Outer convexification (20)		
	Fractionality	Objective	CPU [sec]	Fractionality	Objective	CPU [sec]
40	0.101895	1.01687	1/*	0.073163	1.03315	1/4
80	0.093632	0.99866	6/2	0.010639	1.01265	2/85
160	0.098872	0.98878	28/8	0.000409	1.00030	3/51
320	0.095983	0.98325	*/20	0.000820	0.99283	53/365

of v^i. The special case of truck control is prototypical for many (in particular time-optimal) MIOCPs, as often bang-bang solutions are optimal. However, this is not guaranteed in the general case.

To evaluate this effect, we apply both inner and outer convexification to the truck control problem (16), however for the time being without the constraints (14)–(15). For convenience, we use formulation (18) in the following. The effect is strongest for time- and energy-optimal driving, hence we consider the case where $\lambda_{\text{fuel}} = 1$, $\lambda_{\text{dev}} = \lambda_{\text{comf}} = 0$ in the uphill setting of Fig. 2.

Table 1 shows numerical results for the cruise control of the heavy-duty truck. Both the inner and the outer convexification formulations have been relaxed and solved to local optimality with the active-set solver SNOPT [28] and the interior point solver IPOPT [69]. The most important findings can be summarized as follows:

- For inner convexification, the fractionality of the optimal relaxed gear choices $\alpha_j(\cdot)$ does not improve for finer discretizations. For the outer convexification however, it goes to zero as the control discretization grid is equidistantly refined. The reason is that the optimal control $\alpha(\cdot)$ of (20) is of bang-bang type;
- There is a gap between the lower bound from the relaxed IC solutions and the best possible integer solution, which is almost attained by the OC formulation;

- The interior point code IPOPT seems to perform better on inner convexification, the SQP code SNOPT performs better on outer convexification;
- Both codes run into problems with inner convexification instances.

Although on single instances it may be faster to compute the relaxation of the inner convexification formulation, it is clear that outer convexification is the method of choice within a branch and bound framework. This is due to the tighter relaxation (better bound) in combination with the feature that relaxed solutions are usually already almost integer feasible if the control discretization grid is chosen adequately.

Extending the Outer Convexification Approach The advantages of outer convexification over inner convexification have stimulated additional research into the formulation (20). Here, we briefly review significant developments for the reader's convenience. First, to fully exploit the beneficial integrality property that has been exemplified above and that is related to bang–bang solutions in optimal control, direct methods need to be equipped with adaptivity in the control grid discretization and follow-up transformations into a switching time optimization, [56, 60]. Second, so-called sensitivity-seeking or path-constrained arcs (i.e., time periods during which $\alpha(\cdot)$ is not binary) can be treated with tailored *sum up rounding* strategies. This rounding strategy is a constructive part in the proof for the dependence of the integer gap on the control discretization grid size Δt. Furthermore, it is the optimal solution to a MILP that minimizes the deviation of a binary control $\omega(\cdot)$ from a relaxed one $\alpha(\cdot)$ with respect to

$$\max_{t\in[0,t_\mathrm{f}]}\left\| \int_0^t \omega(\tau) - \alpha(\tau)\mathrm{d}\tau \right\|.$$

If additional constraints like a maximum number of switches need to be fulfilled, tailored branching algorithms can be applied to efficiently solve the constrained MILP [37]. Third, the numerical algebra to cope with the specific structures induced by the control functions $\alpha(\cdot)$ can be exploited in the context of SQP approaches [41, 42]. Finally, an extension to more general MIOCPs is possible, e.g., to multi-objective problems [48], to differential-algebraic systems [27], and to certain partial differential equations [34]. For an overview see [57, 58].

4 Constraint Formulations

In this section, we study reformulations for logically implied constraints $c(\cdot)$, i.e., $n_c > 0$ in (1). Again, we consider reformulations that are equivalent for integer control functions and discuss properties of their relaxations. We do this both in general and for the special case of the constraints (14)–(15) of the heavy-duty truck model from Sect. 2 that result in

$$c^{\text{truck}}(v, M_{\text{ind}}, \mu) = \begin{pmatrix} M_{\text{ind,max}}(n_{\text{eng}}(v,\mu)) - M_{\text{ind}} \\ n_{\text{eng,max}} - n_{\text{eng}}(v,\mu) \\ n_{\text{eng}}(v,\mu) - n_{\text{eng,min}} \end{pmatrix}. \tag{23}$$

In the interest of notational simplicity we omit the lower linear engine speed bound $n_{\text{eng}}(v,\mu) - n_{\text{eng,min}}$ in the rest of this section. We note that the function $c^{\text{truck}}(\cdot)$ has components that are quadratic and linear in $i_{\text{T}}(\mu)$,

$$c^{\text{truck}}(v, M_{\text{ind}}, \mu) = \begin{pmatrix} 3000 - (3 i_{\text{A}} i_{\text{T}}(\mu) v/(\pi r_{\text{stat}}) - 125)^2 - M_{\text{ind}} \\ \pi r_{\text{stat}} n_{\text{eng,max}}/(30 i_{\text{A}} i_{\text{T}}(\mu)) - v \end{pmatrix}. \tag{24}$$

4.1 Inner Convexification of the Constraints

The inner convexification formulation (19) can be augmented in a straightforward way with constraints

$$0 \le c\big(x(t), u(t), g(\psi)\big)$$

that guarantee that integer feasible solutions $\psi \in \{1, \dots, n_\omega\}$ of (19) are feasible solutions of (1). Using (18) for $g(\cdot)$ we obtain the following reformulation of the constraints (14), (15),

$$0 \le 3000 - \left(3 i_{\text{A}} \left(\sum_{i=1}^{n_\omega} \alpha_i i_{\text{T}}(i)\right) v/(\pi r_{\text{stat}}) - 125\right)^2 - M_{\text{ind}}, \tag{25a}$$

$$0 \le \pi r_{\text{stat}} n_{\text{eng,max}} \Big/ \left(30 i_{\text{A}} \left(\sum_{i=1}^{n_\omega} \alpha_i i_{\text{T}}(i)\right)\right) - v. \tag{25b}$$

Just as in Sect. 3, the evaluation of convex combinations within a nonlinear function may give rise to optimality of feasible fractional values, while neighboring integer values may not be optimal. Hence, the inner convexification approach can be expected to potentially yield weak relaxations.

4.2 Outer Convexification/One Row Formulation of the Constraints

The outer convexification reformulation (20) can be applied to the constraint expression as well. Residuals are evaluated for all possible choices, and the constraint is imposed on the convex combination of residuals resulting in

$$0 \le \sum_{i=1}^{n_\omega} \alpha_i(t) c\big(x(t), u(t), v^i\big). \tag{26}$$

This reformulation avoids the problem of evaluation in fractional choices, and ensures that all feasible integer points are feasible points of the original MIOCP.

Remark 2 (LICQ for Outer Convexification) Let $(\bar{x}, \bar{u}, \bar{\alpha})$ be a feasible solution of the relaxation of problem (20), (26) and let the matrix

$$\begin{bmatrix} \sum_{i=1}^{n_\omega} \bar{\alpha}_i(t) \nabla c(\cdot, v^i) \\ \nabla d(\cdot) \end{bmatrix} := \frac{\partial}{\partial(x,u)} \begin{bmatrix} \sum_{i=1}^{n_\omega} \bar{\alpha}_i(t) c(\bar{x}, \bar{u}, v^i) \\ d(\bar{x}, \bar{u}) \end{bmatrix}$$

of active constraints have full row rank. Drop the upper bounds $\alpha_i(\cdot) \leq 1$, which are implicitly implied by $\alpha_i(\cdot) \geq 0$ and $\sum_i \alpha_i(\cdot) = 1$. Then LICQ is satisfied for the relaxation of problem (20), (26).

Proof We look at the constraint matrix of all (active) constraints in $(\cdot, \bar{\alpha})$, given by

$$\frac{\partial}{\partial(\alpha_1, \dots, \alpha_{n_\omega}, (x,u))} \begin{bmatrix} \sum_{i=1}^{n_\omega} \alpha_i(t) c(\cdot, v^i) \\ d(\cdot) \\ \sum_{i=1}^{n_\omega} \alpha_i(t) - 1 \\ \alpha_1(t) \\ \dots \\ \alpha_{n_\omega}(t) \end{bmatrix}$$

$$= \begin{bmatrix} c(\cdot, v^1) & \cdots & c(\cdot, v^{n_{n_\omega}}) & \sum_{i=1}^{n_{n_\omega}} \bar{\alpha}_i(t) \nabla c(\cdot, v^i) \\ 0 & \cdots & 0 & \nabla d(\cdot) \\ 1 & \cdots & 1 & 0 \\ 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \end{bmatrix}$$

Due to feasibility of $\bar{\alpha}$ and the SOS-1 constraint there exists at least one index $1 \leq i \leq n_\omega$ for which $\alpha_i(t) \geq 0$ is not active. Hence, the bottom n_ω rows are linearly independent. The first block of rows may contribute an entry in column i, but by assumption is linearly independent from the other rows due to the right-most block of columns, which concludes the proof. □

The same holds true if we eliminate one of the multipliers α_i using the SOS-1 constraint, which becomes an inequality constraint. For the heavy-duty truck model, the torque constraint (14) and the engine speed constraint (15) are reformulated to read

$$0 \leq \sum_{i=1}^{n_\mu} \alpha_i(t) M_{\text{ind,max}}\big(n_{\text{eng}}(v(s), i)\big) - M_{\text{ind}}(s), \tag{27a}$$

$$0 \leq n_{\text{eng,max}} - \sum_{i=1}^{n_\mu} \alpha_i(t) n_{\text{eng}}\big(v(s), i\big). \tag{27b}$$

Note that (25b) and (27b) are identical, as $i_{\text{T}}(\mu)$ enters linearly.

As the constraints are summed up, *compensatory effects* may lead to feasible residuals for fractional values of the convex multipliers in both cases, as observed in [39]. For example, a first gear choice leading to an engine speed violating the upper bound $n_{\text{eng,max}}$ and a second gear choice in violation of the lower bound $n_{\text{eng,min}}$ may form a feasible convex combination in (27b).

4.3 Complementarity Formulation

To address the problem of compensatory effects, [39] proposes to enforce feasibility individually for each possible choice of the integer control via

$$0 \le \alpha_i(t) c\big(x(t), u(t), v^i\big), \quad 1 \le i \le n_{n_\omega}. \tag{28}$$

For the heavy-duty truck model, torque and engine speed constraints are

$$0 \le \alpha_i(t)\big(M_{\text{ind,max}}\big(n_{\text{eng}}\big(v(s), i\big)\big) - M_{\text{ind}}(s)\big), \tag{29a}$$

$$0 \le \alpha_i(t)\big(n_{\text{eng,max}} - n_{\text{eng}}\big(v(s), i\big)\big) \tag{29b}$$

for $1 \le i \le n_\mu$. It is obvious that optimal solutions with nonzero convex multiplier $\alpha_i(t) > 0$ are now feasible for $\alpha_i(t) = 1$ as well. Compared to the outer convexification formulation, the number of constraints has increased from 4 to $4n_\mu$, though. More important, due to the structure of the constraints (29a)–(29b), the NLP obtained from discretization of the relaxed convexified MIOCP now is a MPVC in the form of (5). In the case of equality constraints, we obtain a MPCC, see (4).

4.4 Addressing the Complementarity Formulation

As mentioned in the introduction, MPCCs and MPVCs lose constraint qualification. This can also be seen directly.

Remark 3 (No LICQ for Complementarity) Let $(\bar{x}, \bar{u}, \bar{\alpha})$ be a feasible solution of the relaxation of problem (20), (28) with

$$\bar{\alpha}_i(t) = 0, \qquad c\big(\bar{x}, \bar{u}, v^i\big) = 0$$

for at least one $1 \le i \le n_\omega$. Then LICQ is not satisfied.

Proof We look at the constraint matrix of all constraints in $(\cdot, \bar{\alpha})$, given by

$$\frac{\partial}{\partial(\alpha_1, \ldots, \alpha_{n_\omega}, (x, u))} \begin{bmatrix} \alpha_1(t)c(\cdot, v^1) \\ \cdots \\ \alpha_{n_\omega}(t)c(\cdot, v^{n_\omega}) \\ d(\cdot) \\ \sum_{i=1}^{n_\omega} \alpha_i(t) - 1 \\ \alpha_1(t) \\ \cdots \\ \alpha_{n_\omega}(t) \end{bmatrix}$$

$$= \begin{bmatrix} c(\cdot, v^1) & \cdots & 0 & \bar{\alpha}_1(t)\nabla c(\cdot, v^1) \\ & \ddots & & \vdots \\ 0 & \cdots & c(\cdot, v^{n_{n_\omega}}) & \bar{\alpha}_{n_\omega}(t)\nabla c(\cdot, v^{n_\omega}) \\ 0 & \cdots & 0 & \nabla d(\cdot) \\ 1 & \cdots & 1 & 0 \\ 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \end{bmatrix}$$

and note that the rows that correspond to $\bar{\alpha}_i(t)c(\cdot, v^i) \geq 0$ contain only zeros, leading to a trivially linear dependent constraint system. $\square$

The constraint system also violates the weaker Mangasarian-Fromovitz constraint qualification. However, some cone-based constraint qualifications are satisfied, e.g., [2]. Thus, local minimizers are still KKT points.

Complementarity Pivoting Techniques Computational approaches for solving MPCCs and MPVCs directly are mentioned in the introduction, but generally rely on dedicated implementations of complementarity solvers. If one intends to use existing numerical software, then the complementarity reformulation (28) needs to be addressed by regularization or smoothing techniques that recover LICQ.

Regularization and Smoothing Techniques These techniques follow the principle of reformulation of the MPCC or MPVC in violation of LICQ using a parameter $\varepsilon > 0$ in a way that recovers constraint qualification for $\varepsilon \in (0, \overline{\varepsilon})$. Then, convergence of the sequence of local minimizers to a limit point is established for $\varepsilon \to 0$. The assessment of the type of stationarity obtained for the limit point is crucial. For MIOCP, we find it essential to ask for *Bouligand stationarity*. Lesser stationarity concepts allow for termination in *spurious stationary points* that possess trivial descent directions, thus giving rise to “missed” switches of the integer control.

In the following, an overview over different possible reformulations is given, and their general form is presented and applied to the particular truck problem.

Regularization Reformulations Conversely, regularization formulations relax the feasible set using $\varepsilon > 0$ by requiring

$$-\varepsilon \le \alpha_i(t) \cdot c\big(x(t), u(t), v^i\big), \quad 1 \le i \le n_{n_\omega} \quad \text{(MPVC)},$$

or, in what is sometimes referred to as the *lumped* formulation,

$$-\varepsilon \le \sum_{i=1}^{n_{n_\omega}} \alpha_i(t) \cdot c\big(x(t), u(t), v^i\big). \tag{30}$$

For MIOC and in particular for the truck model, the lumped formulation will satisfy LICQ but amounts to applying outer convexification to the original constraint, see Sect. 4.2. For the truck model, the first formulation reads

$$-\varepsilon \le \alpha_i(t) \cdot \big(M_{\text{ind,max}}\big(n_{\text{eng}}\big(v(s), i\big)\big) - M_{\text{ind}}(s)\big) \tag{31a}$$

$$-\varepsilon \le \alpha_i(t) \cdot \big(n_{\text{eng,max}} - n_{\text{eng}}\big(v(s), i\big)\big) \tag{31b}$$

NCP Function Reformulations A function $\phi : \mathbb{R}^2 \to \mathbb{R}$ is called *NCP*-function if

$$\phi(a,b) = 0 \quad \Rightarrow \quad a \ge 0, b \ge 0, ab = 0, \quad \text{(MPCC)},$$

$$\phi(a,b) = 0 \quad \Rightarrow \quad b \ge 0, ab \le 0, \quad \text{(MPVC)},$$

see, e.g., [18] for a survey. For MPCCs the Fischer-Burmeister function

$$\phi^{\text{FB}}(a,b) := a + b - \sqrt{a^2 + b^2}$$

may be used, [19]. For MPVC, [36] uses the NCP function

$$\phi^{\text{VC}}(a,b) = \frac{1}{2}\big(ab + \sqrt{a^2b^2} + \sqrt{b^2} - b\big). \tag{32}$$

Piecewise smooth complementarity functions to replace the complementarity constraint in a MPCC are developed in [46], and a one-to-one correspondence between strongly stationary points and KKT points of the reformulated MPCC is established. The non-smoothness is shown to not impede the convergence of SQP methods. The approach however cannot avoid convergence of SQP solvers to spurious stationary points in the degenerate case. The NCP formulation for the truck model is

$$0 \ge \phi^{\text{VC}}\big(M_{\text{ind}}(s) - M_{\text{ind,max}}\big(n_{\text{eng}}\big(v(s), i\big)\big), \alpha_i(t)\big), \tag{33a}$$

$$0 \ge \phi^{\text{VC}}\big(n_{\text{eng}}\big(v(s), i\big) - n_{\text{eng,max}}, \alpha_i(t)\big). \tag{33b}$$

Smoothing Reformulations Smoothing reformulations have first been developed for MPCCs and work by appropriately including a smoothing parameter $\varepsilon > 0$:

$$0 \geq \phi_\varepsilon\big(-c\big(x(t), u(t), v^i\big), \alpha_i(t)\big), \quad 1 \leq i \leq n_{n_\omega} \quad \text{(MPVC)}, \tag{34}$$

with NCP function ϕ, e.g., $\phi_\varepsilon^{\mathrm{FB}}(a, b) := a + b - \sqrt{a^2 + b^2 + \varepsilon}$ for MPCC and

$$0 \geq \phi_\varepsilon^{\mathrm{VC}}(a, b) := \frac{1}{2}\big(ab + \sqrt{a^2 b^2 + \varepsilon^2} + \sqrt{b^2 + \varepsilon^2} - b\big) \tag{35}$$

for MPVC. For MPCC, smoothing under an LICQ-type assumption and a second order condition has been shown to yield B-stationarity of the limit point. For MPVC, smoothing is shown in [36] to not yield a satisfactory framework for solving MPVCs. For the truck model, the smoothing reformulation reads

$$0 \geq \phi_\varepsilon^{\mathrm{VC}}\big(M_{\mathrm{ind}}(s) - M_{\mathrm{ind,max}}\big(n_{\mathrm{eng}}\big(v(s), i\big)\big), \alpha_i(t)\big), \tag{36a}$$

$$0 \geq \phi_\varepsilon^{\mathrm{VC}}\big(n_{\mathrm{eng}}\big(v(s), i\big) - n_{\mathrm{eng,max}}, \alpha_i(t)\big). \tag{36b}$$

Smoothing-Regularization Reformulations Smoothing-regularization approaches combine the concepts of smoothing and regularization to form

$$\varepsilon \geq \phi_\varepsilon\big(-c\big(x(t), u(t), v^i\big), \alpha_i(t)\big), \quad 1 \leq i \leq n_{n_\omega} \quad \text{(MPVC)}. \tag{37}$$

For MPVC, using (32), the smoothing-relaxation formulation reads $\varepsilon \geq \phi_\varepsilon^{\mathrm{VC}}(a, b)$. Convergence for $\varepsilon \to 0$ to a B-stationary limit point is established under a weak LICQ-type constraint qualification, assuming existence and asymptotic nondegeneracy of a sequence $\{x_\varepsilon\}_{\varepsilon \to 0}$ of feasible points for the sequence NLP(ε). LICQ is shown to be satisfied for all NLP(ε) with $\varepsilon \in (0, \overline{\varepsilon})$.

For the truck problem, the smoothing-regularization formulation reads

$$\varepsilon \geq \phi_\varepsilon^{\mathrm{VC}}\big(M_{\mathrm{ind}}(s) - M_{\mathrm{ind,max}}\big(n_{\mathrm{eng}}\big(v(s), i\big)\big), \alpha_i(t)\big), \tag{38a}$$

$$\varepsilon \geq \phi_\varepsilon^{\mathrm{VC}}\big(n_{\mathrm{eng}}\big(v(s), i\big) - n_{\mathrm{eng,max}}, \alpha_i(t)\big). \tag{38b}$$

4.5 Generalized Disjunctive Programming

Based on the work of Balas for integer linear programs, Grossmann and coworkers developed Generalized Disjunctive Programming (GDP) for mixed-integer nonlinear programs [31]. The problem formulation (1) uses disjunctive notation, hence it is natural to take the GDP point of view. It is motivated by disjunctions of the form

$$\begin{aligned}
&\min_{y,Y} e(y) + \sum_{i\in I} e_i \\
&\text{s.t.} \quad \begin{bmatrix} Y_i \\ g_i(y) \le 0 \\ e_i = \gamma_i \end{bmatrix} \vee \begin{bmatrix} \neg Y_i \\ B_i y = 0 \\ e_i = 0 \end{bmatrix} \quad \forall i \in I \\
&\qquad 0 \le y \le U, \qquad \Phi(Y) = \text{true}, \quad Y \in \{\text{false}, \text{true}\}^{|I|}
\end{aligned} \tag{39}$$

with application in process synthesis where Y_i represents presence or absence of units and y a vector of continuous variables. If a corresponding unit is not used, the equation $B_i y = 0$ eliminates variables and $e_i = 0$ sets the costs to zero.

As stated above, logical relations $\Phi(Y) = \text{true}$ can be translated into constraints with binary variables $z \in \{0, 1\}^{|I|}$. An interesting question, however, is how the disjunctions are formulated. We consider two approaches.

Big-M Using large enough constants M is a well-known technique to model logical relations in combinatorial optimization. For (39), this yields

$$g_i(y) \le M(1 - z_i), \qquad -Mz_i \le B_i y \le Mz_i, \qquad e_i = z_i \gamma_i.$$

Convex Hull Reformulation A possibly tighter relaxation can be obtained from the nonlinear convex hull reformulation. It makes use of the perspective of a function.

Definition 8 (Perspective function) The *perspective* of a function $f : \mathbb{R}^n \to \mathbb{R}$ is the function $\hat{f} : \mathbb{R}^{n+1} \to \mathbb{R}$ defined by

$$\hat{f}(\lambda, y) = \begin{cases} \lambda f(y/\lambda) & \text{if } \lambda > 0, \\ 0 & \text{if } \lambda = 0, y = 0, \\ \infty & \text{otherwise.} \end{cases}$$

An important property is that if $f(\cdot)$ is convex, then $\hat{f}(\cdot)$ is also convex. Perspectives have been used for strong formulations of MINLPs [15] and have been gaining increasing interest lately also for the derivation of *perspective cuts*, e.g., [33]. Perspectives can be used to obtain the nonlinear convex hull of a feasible set as follows:

Definition 9 (Nonlinear Convex Hull) Problem (6) can be equivalently restated as minimization over all (y, λ) of the nonconvex function

$$\Phi(y, \lambda) := \begin{cases} 0 & \text{if } \lambda = 0, y = 0, \\ F(y) & \text{if } \lambda = 1, G(y) \le b, \\ \infty & \text{else.} \end{cases} \tag{40}$$

wherein we have assumed w.l.o.g. that $\Phi(0, \lambda) = 0$. The closure $\overline{co}\Phi(y, \lambda)$ of the convex hull of Φ is given by

$$\overline{co}\Phi(y, \lambda) = \begin{cases} 0 & \text{if } \lambda = 0, y = 0, \\ \lambda F(y/\lambda) & \text{if } \lambda \in (0, 1], G(y) \leq \lambda b, \\ \infty & \text{else.} \end{cases} \tag{41}$$

The convex hull representation of the GDP (39) is then obtained by introduction of additional variables $0 \leq \lambda_{ij} \leq 1$, $\lambda_{i1} + \lambda_{i0} = 1$ and $0 \leq v_{ij} \leq \lambda_{ij} U$, $y = v_{i1} + v_{i0}$ and constraints $0 \geq \lambda_{i1} g_i(v_{i1}/\lambda_{i1})$, $B_i v_{i0} = 0$, $f_i = \lambda_{i1}\gamma_i$.

Numerical Difficulties For constraint formulations using perspectives usually LICQ holds. The reason is that unlike in Remark 3, the derivative $\lambda \nabla_v g(v/\lambda) \neq 0$ for $\lambda = 0$. E.g., for linear constraints $g(v) = av + b$ we have $\lambda \nabla_v g(v/\lambda) = a$.

However, for nonlinear $g(\cdot)$ there are computational challenges due to division by zero or near-zero values. For $\varepsilon > 0$ sufficiently small $\lambda f(y/\lambda)$ can be approximated by $(\lambda + \varepsilon) f(y/(\lambda + \varepsilon))$, [31], or by $(\lambda) f(y/(\lambda + \varepsilon))$, [50], or as proposed in the Ph.D. thesis of Nicolas Sawaya by $((1 - \varepsilon)\lambda + \varepsilon) f(y/((1 - \varepsilon)\lambda + \varepsilon))$. The results may depend heavily on the choice of ε and be computationally challenging.

4.6 Generalized Disjunctive Programming for MIOCP

First, we consider the Big-M formulation for GDP. For general constraints $c(\cdot)$ and SOS-1 variables $\alpha(\cdot)$ we obtain for $1 \leq i \leq n_{n_\omega}$

$$M\big(\alpha_i(t) - 1\big) \leq c\big(x(t), u(t), v^i\big) \tag{42}$$

and thus for the truck constraints (23) for $1 \leq i \leq n_\mu$

$$M\big(\alpha_i(t) - 1\big) \leq \big(M_{\text{ind,max}}\big(n_{\text{eng}}\big(v(s), i\big)\big) - M_{\text{ind}}(s)\big), \tag{43a}$$

$$M\big(\alpha_i(t) - 1\big) \leq \big(n_{\text{eng,max}} - n_{\text{eng}}\big(v(s), i\big)\big). \tag{43b}$$

This formulation does not cause LICQ problems and good values for M can be determined from bounds on $v(\cdot)$. Still, it is expected to give weak relaxations.

Second, we apply the GDP convex hull technique to the MIOCP (1). [50] proposed to lift all variables (controls, differential states, horizon lengths). We describe one of the many possibilities that stem from a modeler's freedom to lift only some of the variables and to use different formulations for the disjunctive constraints.

One first observation is that we can use the convex multiplier functions $\alpha_i(\cdot)$ from outer convexification in place of λ, and let $y(\cdot) := (x(\cdot), u(\cdot))$. The perspective formulation for the constraints $c(\cdot)$ then yields

$$0 \leq \alpha_i(t) c\big(x(t)/\alpha_i(t), u(t)/\alpha_i(t), v^i\big), \quad t \in [0, t_{\text{f}}], i = 1, \ldots, n_\omega. \tag{44}$$

To properly obtain the convex hull, we also apply this procedure to the dynamic constraint for $t \in [0, t_{\text{f}}]$, $i = 1, \ldots, n_\omega$:

$$0 = \alpha_i(t)\big(\dot{x}(t)/\alpha_i(t) - f\big(x(t)/\alpha_i(t), u(t)/\alpha_i(t), v^i\big)\big) \tag{45a}$$

$$= \dot{x}(t) - \alpha_i(t) f\big(x(t)/\alpha_i(t), u(t)/\alpha_i(t), v^i\big). \tag{45b}$$

We introduce disaggregated state derivatives $\dot{x}^i(\cdot)$, resulting states $x^i(\cdot)$, and disaggregated control decision variables $u^i(\cdot)$ for all nonlinearly entering controls:

$$\dot{x}(t) = \sum_{i=1}^{n_\omega} \dot{x}^i(t), \qquad x(t) = \sum_{i=1}^{n_\omega} x^i(t), \qquad u(t) = \sum_{i=1}^{n_\omega} u^i(t).$$

We consider a time discretization of the states of problem (1),

$$x(t_k) := s_k, \qquad \dot{x}(t) = f\big(x(t), u(t), v(t)\big), \quad t \in [t_k, t_{k+1}], \tag{46a}$$

$$s_{k+1} = x\big(t_{k+1}; t_k, s_k, u(\cdot), v(\cdot)\big), \quad 0 \le k \le N-1, \tag{46b}$$

which yields the MIOCP

$$\begin{aligned} &\min_{x(\cdot),u(\cdot),Y} e(s_N) \\ &\text{s.t.} \quad \bigvee_{i\in\{1,\dots,n_\omega\}} \begin{bmatrix} Y_{ik} \\ \dot{x}(t) = f(x(t), u(t), v^i) \\ x(t_k) = s_k \\ s_{k+1} = x(t_{k+1}; t_k, s_k, u(\cdot), v^i) \\ 0 \le c(x(t), u(t), v^i) \end{bmatrix} \quad \forall t \in [t_k, t_{k+1}] \\ &\qquad 0 \le d\big(x(t), u(t)\big) \quad \forall t \in [t_k, t_{k+1}] \end{aligned} \tag{47}$$

with $k = 0, \dots, N-1$, for which the convex hull formulation is, with piecewise constant controls $\alpha_i(t) = \alpha_{ik}$ and $t \in [t_k, t_{k+1}]$ for $k = 0, \dots, N-1$,

$$\begin{aligned} &\min_{s,u,\alpha} e(s_N) \\ &\text{s.t.} \quad \dot{x}^i(t) = \alpha_{ik} f\big(x^i(t)/\alpha_{ik}, u^i(t)/\alpha_{ik}, v^i\big) \\ &\qquad x^i(t_k) = \alpha_{ik} s_k \\ &\qquad s_{k+1} = \sum_{i=1}^{n_\omega} x^i\big(t_{k+1}; t_k, s_k, u^i(\cdot)/\alpha_{ik}, v^i\big) \\ &\qquad 0 \le \alpha_{ik} c\big(x^i(t)/\alpha_{ik}, u^i(t)/\alpha_{ik}, v^i\big) \\ &\qquad 0 \le d\left(\sum_{i=1}^{n_\omega} x^i(t), \sum_{i=1}^{n_\omega} u^i(t)\right) \\ &\qquad \sum_{i=1}^{n_\omega} \alpha_{ik} = 1, \qquad 0 \le x^i(t) \le \alpha_{ik} M^{\mathrm{s}}, \qquad 0 \le u^i(t) \le \alpha_{ik} M^{\mathrm{u}}. \end{aligned} \tag{48}$$

Note that for the purpose of the disjunctive term controlled by Y_{ik}, s_k is a constant when forming the perspective of the initial value constraint. Moreover, as the s_{k+1} enter linearly, we do not disaggregate them but instead aggregate the IVP results. Next, we enforce path constraints $0 \leq x^i(\cdot) \leq \alpha_{ik} M^{\mathrm{s}}$, $c(\cdot)$ and $d(\cdot)$ in the discretization points $t = t_k$ only, as is customary in direct optimal control. The respective constraints of (48) are replaced by

$$0 \leq s_k \leq M^{\mathrm{s}} \tag{49a}$$

$$0 \leq \alpha_{ik} c\big(s_k, u^i(t_k)/\alpha_{ik}, v^i\big) \tag{49b}$$

$$0 \leq d\left(s_k, \sum_{i=1}^{n_\omega} u^i(t_k)\right). \tag{49c}$$

We further substitute u by $\bar{u}_i(t) = u_i(t)/\alpha_{ik}$. This poses no problem for $\alpha_{ik} = 0$ due to the bound $u_i(t) \leq \alpha_{ik} M_k^{\mathrm{u}}$, and yields lifted controls $\bar{u}_i(t)$ with $0 \leq \bar{u}^i(t_k) \leq M^{\mathrm{u}}$:

$$\dot{x}^i(t) = \alpha_{ik} f\big(x^i(t)/\alpha_{ik}, \bar{u}^i(t_k), v^i\big) \tag{50a}$$

$$0 \leq \alpha_{ik} c\big(s_k, \bar{u}^i(t_k), v^i\big) \tag{50b}$$

$$0 \leq d\left(s_k, \sum_{i=1}^{n_\omega} \alpha_{ik} \bar{u}^i(t_k)\right) \tag{50c}$$

Since this formulation still poses numerical difficulties, we go one step further and modify the problem by aggregating the states over all time steps and not only during the shooting intervals. This idea comes from the observation that, except for the ODE constraint, the $x^i(\cdot)$ enter only via their convex combination. Hence, we replace $x^i(t)$ by $\alpha_{ik} x(t)$ and obtain $\sum_{i=1}^{n_\omega} x^i(t_{k+1}) = \sum_{i=1}^{n_\omega} \alpha_{ik} x(t_{k+1}) = x(t_{k+1})$ and thus (51)

$$\begin{aligned}
\min_{s,\bar{u},\alpha}\ & e(s_N) \\
\text{s.t.}\ \ & \dot{x}(t) = \sum_{i=1}^{n_\omega} \alpha_{ik} f\big(x(t), \bar{u}^i(t), v^i\big) \\
& x(t_k) = s_k \\
& s_{k+1} = x\big(t_{k+1}; t_k, s_k, \bar{u}(\cdot)\big) \\
& 0 \leq \alpha_{ik} c\big(s_k, \bar{u}^i(t_k), v^i\big) \\
& 0 \leq d\left(s_k, \sum_{i=1}^{n_\omega} \alpha_{ik} \bar{u}^i(t)\right) \\
& \sum_{i=1}^{n_\omega} \alpha_{ik} = 1, \qquad 0 \leq s_k \leq M^{\mathrm{s}}, \qquad 0 \leq \bar{u}^i(t) \leq M^{\mathrm{u}}
\end{aligned} \tag{51}$$

Table 2 Comparison of constraint reformulations for $\alpha_i = 1 \Rightarrow \gamma_i \leq v \leq \Gamma_i$ and their properties

Name	Formulation	Advantages	Drawbacks
Inner convexification Section 4.1, Eqs. (25a)–(25b)	$\gamma(\psi) \leq v \leq \Gamma(\psi)$ $\gamma(i) = \gamma_i$, $\Gamma(i) = \Gamma_i$	Easy to formulate very fast to solve	Integer infeasible, fractional evaluations, very weak relaxation
Outer convexification Section 4.2, Eqs. (27a)–(27b)	$\sum_i \alpha_i \gamma_i \leq v \leq \sum_i \alpha_i \Gamma_i$	Evaluation in integer points, fast to solve	Compensation effects
MPVC Section 4.3, Eqs. (29a)–(29b)	$\alpha_i \gamma_i \leq \alpha_i v \leq \alpha_i \Gamma_i$	Guarantees integer feasibility	CQs violated, needs tailored methods
MPVC reformulations Section 4.4, Eqs. (31a)–(31b), (36a)–(36b), (38a)–(38b)	Regularization, smoothing, combined	Use existing NLP solvers	Sequence of problems, ill-conditioned, ε-relaxed solution
GDP Big-M Section 4.5, Eqs. (43a)–(43b)	$\gamma_i - M(1-\alpha_i) \leq v$ $\leq \Gamma_i + M(1-\alpha_i)$	CQs, fast	Weak relaxation
GDP Convex Hull Section 4.6, Eq. (51) with Sect. 4.4, Eqs. (31a)–(31b)	Disaggregated controls constraints as MPVC (reformulation)	More degrees of freedom	More variables, in addition to the drawbacks of MPVC (reformulations)

is partial outer convexification of the ODE, uses the vanishing constraint formulation for the constraints, but differs in the disaggregation of the controls $\bar{u}^i(\cdot)$, which gives the system more freedom to potentially find a better solution. Note that this particular way of making use of a GDP formulation needs to be smoothened again, e.g., using (38a)–(38b). For the application at hand this approach was superior to a full lifting and the difficult numerical treatment of the constraints that are quadratic in $1/\alpha_i(\cdot)$.

Table 2 summarizes the reformulations for constraints directly depending on an integer control that have been discussed in this section.

5 Numerical Results

In this section we illustrate above concepts by applying the different formulations to the engine speed constraints of the heavy duty truck.

Reproducible Benchmark Implementations We aim to make the heavy-duty truck cruise control problem available to the community as a MIOCP benchmark problem. A complete description comprises scenario data for $\gamma(s)$ and $v_{\text{law}}(s)$ that characterizes the route to be traveled, with positioning information s assumed to be

Algorithm 1: Homotopy method for problems presented in Sect. 4.4 and Sect. 4.6

```
1 $\delta = 0.6, \varepsilon^* = 10^5$
2 while $\varepsilon^* > 1e - 3$ do
3     $\varepsilon = \delta \cdot \varepsilon^*$
4     Solve the problem corresponding to $\varepsilon$ starting from last solution $\sigma^*$
5     if terminal point infeasible, or cannot restore feasibility of the initial point
      then $\delta = 1.6 \cdot \delta$ else store solution as $\sigma^*, \varepsilon^* = \varepsilon, \delta = \delta/1.2$
6 end
```

available. It also comprises $M_{\text{ind,max}} := 3000 - (n_{\text{eng}}(s, \mu(s)) - 1250)^2/100$, M_{fric}, Q_{fuel}, $n_{\text{eng,min}}$, and $n_{\text{eng,max}}$ as vehicle- and engine-specific nonlinear data sets. This data as well as other parameter values, functions, and initial values is available on the web page [55] and in the thesis [39]. Moreover, this web page includes AMPL models for all reformulations we use throughout this article.

In the following calculations, we discretize all control problems using an implicit Euler method with 40 equidistant steps on the horizon [0, 1000].

Computational Setup All results were computed on a single core of an Intel Core i7-2600 CPU with 3.40 GHz and 8 GB memory. IPOPT 3.10.0 and SNOPT 7.2-8 were run with the standard solver options, invoked from AMPL version 20120505. For the homotopy methods, IPOPT warm start options were added. The homotopy method starts with a big value for ε to find a solution for the control problem without vanishing constraints. The last solution σ^* is stored together with the corresponding ε^*. ε^* is then adjusted by a factor δ to obtain a new ε as in Algorithm 1.

Note that also $\delta > 1$ is allowed, since sometimes the last solution cannot be restored even with the same ε due to numerical difficulties.

Discussion of Scenarios We present numerical results for two selected scenarios in Fig. 2 and Fig. 3. The track's height profile is shown on top, followed by eight plots of the relaxed (local) optimal solutions identified by IPOPT for the formulations listed in Table 2. From both scenarios, a clear picture emerges. Inner convexification solutions suffer from significant infeasibility. Outer convexification is computationally the fastest formulation, and yields reasonable approximations that in parts suffer from compensatory effects. In contrast, the three vanishing constraint formulations succeed in yielding feasible solutions for both scenarios. For scenario 2, these solutions are even integer feasible, while for scenario 1 there are at most two control intervals with fractional gear choices. For the feasible vanishing constraint solutions, we can also compare the resulting objective functions. Here, the simple regularized formulations performs best, with the smoothing-regularization and the plain vanishing constraint formulation tied in second place. The GDP Big-M formulation does not yield feasible solutions, while our convex hull variant of GDP performs slightly better than outer convexification also in terms of integer feasibility. It suffers from a high computational runtime, though. The combination of integer

Scenario 1
$\lambda_{\text{fuel}} = 25$
$\lambda_{\text{dev}} = 1$
$\lambda_{\text{comf}} = 1$

Unconstrained

Constraints	-
Objective	582.784
Fractionality	0.1
Infeasibility	$3e+03$
Runtime	2.82 sec

Inner Convexification

Constraints	(25a)-(25b)
Objective	54417.3
Fractionality	0.17
Infeasibility	$6e+02$
Runtime	0.852 sec

Outer Convexification

Constraints	(27a)-(27b)
Objective	58144.4
Fractionality	0.24
Infeasibility	1.3
Runtime	4.61 sec

Vanishing Constraints

Constraints	(29a)-(29b)
Objective	70210.7
Fractionality	0.041
Infeasibility	0
Runtime	35.8 sec

Relaxed VCs

Constraints	(31a)-(31b)
Objective	60222.7
Fractionality	0.0077
Infeasibility	$1.6e-05$
Runtime	18.8 sec

Smoothed VCs

Constraints	(36a)-(36b)
Objective	61364
Fractionality	0.022
Infeasibility	$1e-06$
Runtime	40.6 sec

GDP Big-*M*

Constraints	(43a)-(43b)
Objective	52976.1
Fractionality	0.62
Infeasibility	51
Runtime	8.35 sec

GDP Convex Hull

Constraints	(51)+(38a)-(38b)
Objective	58241.3
Fractionality	0.19
Infeasibility	0.96
Runtime	357 sec

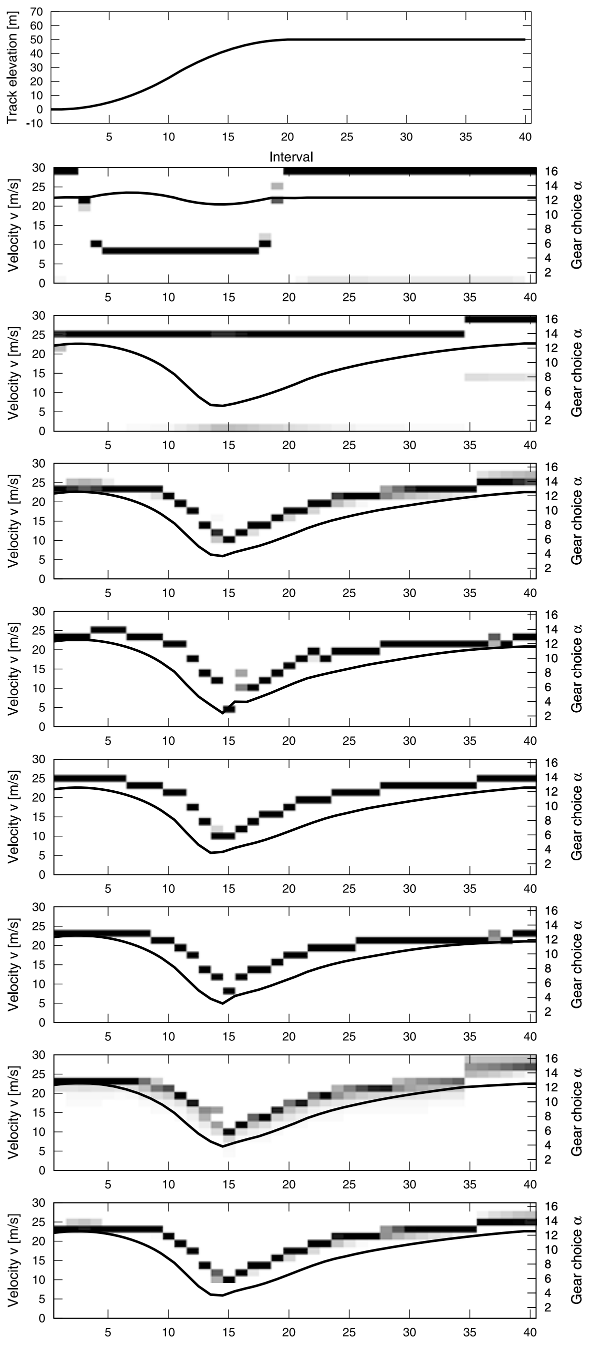


Fig. 2 Relaxed gear choices α_j (reflected by *the intensity of gray*) and corresponding velocities

Scenario 2
$\lambda_{\text{fuel}} = 10$
$\lambda_{\text{dev}} = 100$
$\lambda_{\text{comf}} = 1$

Unconstrained

Constraints	-
Objective	17352.1
Fractionality	0.31
Infeasibility	$5.8e+03$
Runtime	3.42 sec

Inner Convexification

Constraints	(25a)-(25b)
Objective	3748470
Fractionality	0.35
Infeasibility	$7e+02$
Runtime	1.34 sec

Outer Convexification

Constraints	(27a)-(27b)
Objective	3960800
Fractionality	0.33
Infeasibility	6
Runtime	5.15 sec

Vanishing Constraints

Constraints	(29a)-(29b)
Objective	774479
Fractionality	0.064
Infeasibility	0
Runtime	11.8 sec

Relaxed VCs

Constraints	(31a)-(31b)
Objective	4089720
Fractionality	0
Infeasibility	$1.4e-05$
Runtime	19.2 sec

Smoothed VCs

Constraints	(36a)-(36b)
Objective	4014800
Fractionality	$1e-06$
Infeasibility	$1e-06$
Runtime	32.4 sec

GDP Big-*M*

Constraints	(43a)-(43b)
Objective	3507270
Fractionality	0.69
Infeasibility	$2e+02$
Runtime	4.68 sec

GDP Convex Hull

Constraints	(51)+(38a)-(38b)
Objective	4002530
Fractionality	0.19
Infeasibility	0.62
Runtime	368 sec

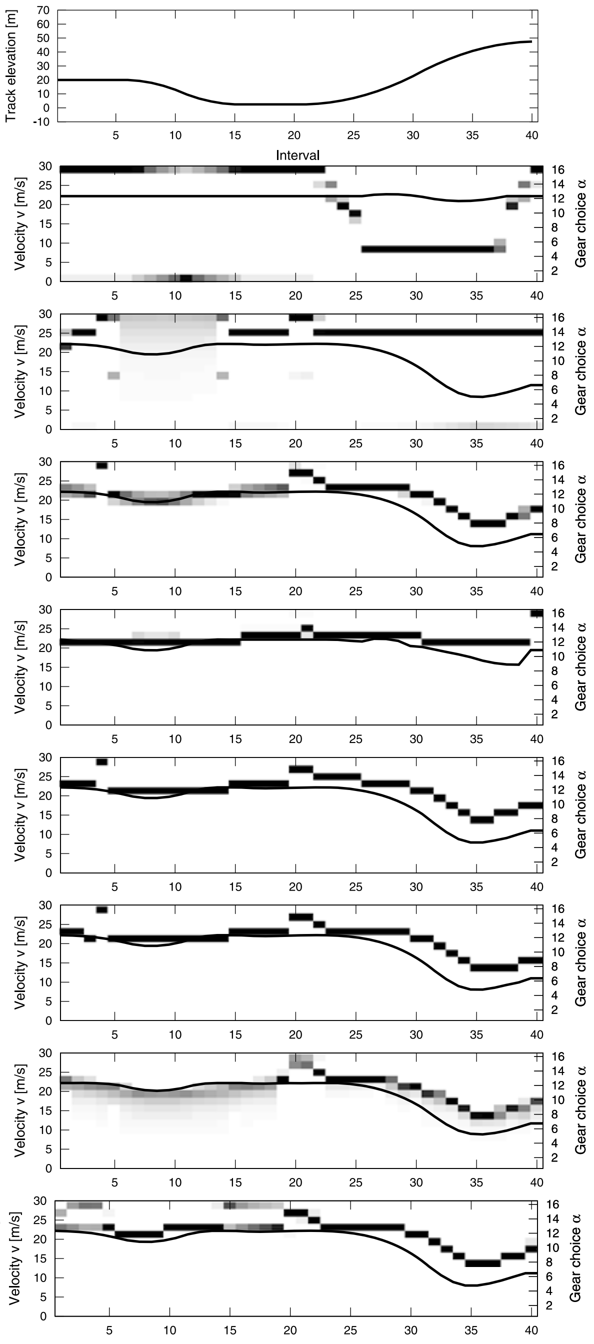


Fig. 3 Relaxed gear choices α_j (reflected by *the intensity of gray*) and corresponding velocities

control functions, nonlinear constraints, and nonlinear differential equations allows for a variety of different GDP implementations within a direct multiple shooting or direct collocation framework and may allow for significant better performance.

6 Future Developments

From the computational results, some evidence can be read in favor of vanishing constraint formulations. For the case of equality constraints depending on the integer control, complementarity constraints are obtained. In view of the computational inefficiency and the remaining infeasibility of homotopy based approaches, it will be desirable to treat MPCCs and MPVCs in their rigorous formulation of Sect. 4.3, Eqs. (29a)–(29b) in the future. SLP-EQP methods, first described by [20], can do so efficiently when the LP subproblem is replaced by the corresponding LPCC [17] or LPVC. Moreover, they can be shown to identify B-stationary points of such problems [47]. The efficiency improvements one would gain from such a computational setup would also make the application to closed-loop mixed-integer control [39] viable.

It will be interesting to study all constraints in the context of a Branch&Cut framework. Furthermore a comparison of global solutions would be helpful. As generic global solvers like COUENNE cannot yet solve the benchmark, a specific dynamic programming approach seems the best choice.

Overcoming the numerical problems associated with perspective functions and a better exploitation of the numerical features of the resulting optimal control problems are yet other challenges.

References

1. Abichandani, P., Benson, H., Kam, M.: Multi-vehicle path coordination under communication constraints. In: American Control Conference, pp. 650–656 (2008)
2. Achtziger, W., Kanzow, C.: Mathematical programs with vanishing constraints: optimality conditions and constraint qualifications. Math. Program., Ser. A **114**, 69–99 (2008)
3. Anitescu, M., Tseng, P., Wright, S.: Elastic-mode algorithms for mathematical programs with equilibrium constraints: global convergence and stationarity properties. Math. Program., Ser. A **110**, 337–371 (2007)
4. Balas, E.: Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. SIAM J. Algebr. Discrete Methods **6**, 466–486 (1985)
5. Bär, V.: Ein Kollokationsverfahren zur numerischen Lösung allgemeiner Mehrpunktrandwert aufgaben mit Schalt- und Sprungbedingungen mit Anwendungen in der optimalen Steuerung und der Parameteridentifizierung. Diploma thesis, Rheinische Friedrich-Wilhelms-Universität zu Bonn (1983)
6. Barton, P.: The modelling and simulation of combined discrete/continuous processes. Ph.D. thesis, Department of Chemical Engineering, Imperial College of Science, Technology and Medicine, London (1992)
7. Baumrucker, B., Biegler, L.: MPEC strategies for optimization of a class of hybrid dynamic systems. J. Process Control **19**(8), 1248–1256 (2009)

8. Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J., Mahajan, A.: Mixed-integer nonlinear optimization. In: Iserles, A. (ed.) Acta Numerica 2013, vol. 22. Cambridge University Press, Cambridge (2013). www.optimization-online.org/DB_HTML/2012/12/3698.html
9. Betts, J.: Practical Methods for Optimal Control Using Nonlinear Programming. SIAM, Philadelphia (2001)
10. Biegler, L.: Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. Comput. Chem. Eng. **8**, 243–248 (1984)
11. Biegler, L.: Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes. Series on Optimization. SIAM, Philadelphia (2010)
12. Bock, H., Longman, R.: Computation of optimal controls on disjoint control sets for minimum energy subway operation. Adv. Astronaut. Sci. **50**, 949–972 (1985)
13. Bock, H., Plitt, K.: A Multiple Shooting algorithm for direct solution of optimal control problems. In: Proceedings of the 9th IFAC World Congress, pp. 242–247. Pergamon Press, Budapest (1984)
14. Burgschweiger, J., Gnädig, B., Steinbach, M.: Nonlinear programming techniques for operative planning in large drinking water networks. Open Appl. Math. J. **3**, 1–16 (2009)
15. Ceria, S., Soares, J.: Convex programming for disjunctive optimization. Math. Program. **86**, 595–614 (1999)
16. Facchinei, F., Jiang, H., Qi, L.: A smoothing method for mathematical programs with equilibrium constraints. Math. Program. **85**, 107–134 (1999)
17. Fang, H., Leyffer, S., Munson, T.: A pivoting algorithm for linear programs with complementarity constraints. Optim. Methods Softw. **87**, 89–114 (2012)
18. Ferris, M., Kanzow, C.: Complementarity and related problems: a survey (1998)
19. Fischer, A.: A special Newton-type optimization method. Optimization **24**, 269–284 (1992)
20. Fletcher, R., de la Maza, E.S.: Nonlinear programming and nonsmooth optimization by successive linear programming. Math. Program. **43**(3), 235–256 (1989)
21. Fletcher, R., Leyffer, S.: Solving mathematical programs with complementarity constraints as nonlinear programs. Optim. Methods Softw. **19**(1), 15–40 (2004)
22. Frangioni, A., Gentile, C.: Perspective cuts for a class of convex 0–1 mixed integer programs. Math. Program., Ser. A **106**, 225–236 (2006)
23. Fügenschuh, A., Herty, M., Klar, A., Martin, A.: Combinatorial and continuous models for the optimization of traffic flows on networks. SIAM J. Optim. **16**(4), 1155–1176 (2006)
24. Fukushima, M., Qi, L. (eds.): Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods. Kluwer Academic, Dordrecht (1999)
25. Gerdts, M.: Solving mixed-integer optimal control problems by Branch&Bound: a case study from automobile test-driving with gear shift. Optim. Control Appl. Methods **26**, 1–18 (2005)
26. Gerdts, M.: A variable time transformation method for mixed-integer optimal control problems. Optim. Control Appl. Methods **27**(3), 169–182 (2006)
27. Gerdts, M., Sager, S.: Mixed-integer DAE optimal control problems: necessary conditions and bounds. In: Biegler, L., Campbell, S., Mehrmann, V. (eds.) Control and Optimization with Differential-Algebraic Constraints, pp. 189–212. SIAM, Philadelphia (2012)
28. Gill, P., Murray, W., Saunders, M.: SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM J. Optim. **12**, 979–1006 (2002)
29. Göttlich, S., Herty, M., Kirchner, C., Klar, A.: Optimal control for continuous supply network models. Netw. Heterog. Media **1**(4), 675–688 (2007)
30. Gräber, M., Kirches, C., Bock, H., Schlöder, J., Tegethoff, W., Köhler, J.: Determining the optimum cyclic operation of adsorption chillers by a direct method for periodic optimal control. Int. J. Refrig. **34**(4), 902–913 (2011)
31. Grossmann, I.: Review of nonlinear mixed-integer and disjunctive programming techniques. Optim. Eng. **3**, 227–252 (2002)
32. Gugat, M., Herty, M., Klar, A., Leugering, G.: Optimal control for traffic flow networks. J. Optim. Theory Appl. **126**(3), 589–616 (2005)
33. Günlük, O., Linderoth, J.: Perspective reformulations of mixed integer nonlinear programs with indicator variables. Math. Program. **124**(1–2), 183–205 (2010)

34. Hante, F., Sager, S.: Relaxation methods for mixed-integer optimal control of partial differential equations. Comput. Optim. Appl. **55**(1), 197–225 (2013)
35. Hellström, E., Ivarsson, M., Aslund, J., Nielsen, L.: Look-ahead control for heavy trucks to minimize trip time and fuel consumption. Control Eng. Pract. **17**, 245–254 (2009)
36. Hoheisel, T.: Mathematical programs with vanishing constraints. Ph.D. thesis, Julius-Maximilians-Universität Würzburg (2009)
37. Jung, M.N., Reinelt, G., Sager, S.: The Lagrangian relaxation for the combinatorial integral approximation problem. Optim. Methods Softw. (2012, submitted). www.optimization-online.org/DB_HTML/2012/02/3354.html
38. Kawajiri, Y., Biegler, L.: A nonlinear programming superstructure for optimal dynamic operations of simulated moving bed processes. Ind. Eng. Chem. Res. **45**(25), 8503–8513 (2006)
39. Kirches, C.: Fast Numerical Methods for Mixed-Integer Nonlinear Model-Predictive Control. Advances in Numerical Mathematics. Springer Vieweg, Wiesbaden (2011)
40. Kirches, C., Sager, S., Bock, H., Schlöder, J.: Time-optimal control of automobile test drives with gear shifts. Optim. Control Appl. Methods **31**(2), 137–153 (2010)
41. Kirches, C., Bock, H., Schlöder, J., Sager, S.: Block structured quadratic programming for the direct multiple shooting method for optimal control. Optim. Methods Softw. **26**(2), 239–257 (2011)
42. Kirches, C., Bock, H., Schlöder, J., Sager, S.: A factorization with update procedures for a KKT matrix arising in direct optimal control. Math. Program. Comput. **3**(4), 319–348 (2011)
43. Kirches, C., Wirsching, L., Bock, H., Schlöder, J.: Efficient direct multiple shooting for nonlinear model predictive control on long horizons. J. Process Control **22**(3), 540–550 (2012)
44. Leineweber, D., Bauer, I., Schäfer, A., Bock, H., Schlöder, J.: An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization (Parts I and II). Comput. Chem. Eng. **27**, 157–174 (2003)
45. Leyffer, S.: Complementarity constraints as nonlinear equations: theory and numerical experience. In: Optimization with Multivalued Mappings: Theory, Applications, and Algorithms, pp. 169–208. Springer, Berlin (2006)
46. Leyffer, S., López-Calva, G., Nocedal, J.: Interior methods for mathematical programs with complementarity constraints. SIAM J. Optim. **17**(1), 52–77 (2006)
47. Leyffer, S., Munson, T.: A globally convergent filter method for MPECs. Preprint ANL/MCS-P1457-0907, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA (2007)
48. Logist, F., Sager, S., Kirches, C., van Impe, J.: Efficient multiple objective optimal control of dynamic systems with integer controls. J. Process Control **20**(7), 810–822 (2010)
49. Martin, A., Möller, M., Moritz, S.: Mixed integer models for the stationary case of gas network optimization. Math. Program. **105**, 563–582 (2006)
50. Oldenburg, J., Marquardt, W.: Disjunctive modeling for optimal control of hybrid systems. Comput. Chem. Eng. **32**(10), 2346–2364 (2008)
51. Prata, A., Oldenburg, J., Kroll, A., Marquardt, W.: Integrated scheduling and dynamic optimization of grade transitions for a continuous polymerization reactor. Comput. Chem. Eng. **32**, 463–476 (2008)
52. Raghunathan, A., Biegler, L.: Mathematical programs with equilibrium constraints (MPECs) in process engineering. Comput. Chem. Eng. **27**, 1381–1392 (2003)
53. Raghunathan, A., Diaz, M., Biegler, L.: An MPEC formulation for dynamic optimization of distillation operations. Comput. Chem. Eng. **28**, 2037–2052 (2004)
54. Ralph, D., Wright, S.J.: Some properties of regularization and penalization schemes for MPECs. Optim. Methods Softw. **19**, 527–556 (2004)
55. Sager, S.: MIOCP benchmark site. mintoc.de
56. Sager, S.: Numerical Methods for Mixed-Integer Optimal Control Problems. Der Andere Verlag, Tönning (2005)
57. Sager, S.: Reformulations and algorithms for the optimization of switching decisions in nonlinear optimal control. J. Process Control **19**(8), 1238–1247 (2009)

58. Sager, S.: On the integration of optimization approaches for mixed-integer nonlinear optimal control. Habilitation, University of Heidelberg (2011)
59. Sager, S.: A benchmark library of mixed-integer optimal control problems. In: Lee, J., Leyffer, S. (eds.) Mixed Integer Nonlinear Programming, pp. 631–670. Springer, Berlin (2012)
60. Sager, S., Reinelt, G., Bock, H.: Direct methods with maximal lower bound for mixed-integer optimal control problems. Math. Program. **118**(1), 109–149 (2009)
61. Sager, S., Bock, H., Diehl, M.: The integer approximation error in mixed-integer optimal control. Math. Program., Ser. A **133**(1–2), 1–23 (2012)
62. Scholtes, S.: Convergence properties of a regularization scheme for mathematical programs with complementarity constraints. SIAM J. Optim. **11**, 918–936 (2001)
63. Scholtes, S.: Nonconvex structures in nonlinear programming. Oper. Res. **52**(3), 368–383 (2004)
64. Sherali, H.: RLT: a unified approach for discrete and continuous nonconvex optimization. Ann. Oper. Res. **149**, 185–193 (2007)
65. Sonntag, C., Stursberg, O., Engell, S.: Dynamic optimization of an industrial evaporator using graph search with embedded nonlinear programming. In: Proceedings of the 2nd IFAC Conference on Analysis and Design of Hybrid Systems (ADHS), pp. 211–216 (2006)
66. Stein, O., Oldenburg, J., Marquardt, W.: Continuous reformulations of discrete-continuous optimization problems. Comput. Chem. Eng. **28**(10), 3672–3684 (2004)
67. Stubbs, R., Mehrotra, S.: Generating convex polynomial inequalities for mixed 0–1 programs. J. Glob. Optim. **24**, 311–332 (2002)
68. Terwen, S., Back, M., Krebs, V.: Predictive powertrain control for heavy duty trucks. In: Proceedings of IFAC Symposium in Advances in Automotive Control, Salerno, Italy, pp. 451–457 (2004)
69. Wächter, A., Biegler, L.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)

Scheduling and Routing of Fly-in Safari Planes Using a Flow-over-Flow Model

Armin Fügenschuh, George Nemhauser, and Yulian Zeng

Abstract The scheduling and routing of small planes for fly-in safaris is a challenging planning problem. Given a fleet of planes and a set of flight requests with bounds on the earliest departure and latest arrival times, the planes must be scheduled and routed so that all demands are satisfied. Capacity restrictions on the load and fuel also must be satisfied. Moreover the refueling of the planes, which can only be done in certain locations, must be scheduled. We present a mixed-integer linear programming based formulation for this problem. For its solution we develop a primal heuristic based on randomized local search. We try to enhance the local search by using exact methods to solve subproblems that only involve a small number of planes. Using a branch-and-cut solver, the MILP formulation can be solved to proven optimality only for small instances. To achieve better dual bounds we present a set partitioning based formulation, where new columns are generated on demand by heuristics and exact methods. We also present a new formulation where the time windows are relaxed, and later reintroduced by incumbent branching. Numerical results on real-world instances show that this time-free approach gives the best results.

A. Fügenschuh (✉)
Fakultät für Maschinenbau, Helmut-Schmidt-Universität, Universität der Bundeswehr Hamburg, Holstenhofweg 85, 22043 Hamburg, Germany
e-mail: fuegenschuh@hsu-hh.de

G. Nemhauser · Y. Zeng
H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, 755 Ferst Drive NW, Atlanta, GA 30332-0205, USA

G. Nemhauser
e-mail: george.nemhauser@isye.gatech.edu

Y. Zeng
e-mail: yulian@gatech.edu

1 Introduction

Tourism is one of the largest components of the world's economy, having achieved astonishing growth rates over the last century. According to the World Tourism Organization (UNWTO) the number of international arrivals went up from 25 million in 1950 to 983 million in 2011. With an estimated US$ 1,030 billion, international tourism receipts had an approximate 6 per cent share of the worldwide exports of goods and services, and a 30 per cent share when considering the service export market only. It is also one of the largest employers in the world with an estimated 100 million employees. For lesser developed countries the income generated by tourism provides a significant contribution to the gross national product [27]. Our industrial project partner is a tourism company based in the southern part of Africa. They operate over 50 small planes, such as Beechcraft, Cessna, and Pilatus in five countries. Their main business is exclusive fly-in safaris. On such safaris their customers are transported between various remote locations and lodges where they spend a few hours or days before they are picked up and moved to the next landmark of interest. Long before the tourists arrive the company faces a scheduling and routing problem: when should the flight requests be served and by which plane. Scheduling planes and tourist requests is a challenging task that currently takes a human planner several days to do manually. He has to take several restrictions into account. Naturally the capacity limitations in terms of seats and payload weight of the planes must not be exceeded. Refueling is a crucial issue. Since most of the "airports" are in fact nothing more than small runways it must be ensured that the plane always has enough fuel to reach a destination where refueling is possible. The tourist requests impose further limitations, such as the number of intermediate stops and the maximum airtime, which also depend on the price the customers pay for their safari. Most importantly, earliest departure and latest arrival times have to be obeyed. Taking all of these factors under consideration, the planner tries to achieve a cheapest possible solution in terms of operational costs. Our goal is to support the human planner to achieve better solutions in less time. We accomplished this goal by providing a local search heuristic, a variation of which is being used by the company.

The air-travel routing and scheduling problem that we present here belongs to the large class of network design and logistics applications that have been a driving force behind the development of discrete and combinatorial solution methods since the early days of our discipline. As a special case it contains one of the most prominent combinatorial problems: the (symmetric) traveling salesman problem (TSP). Almost six decades ago Dantzig, Fulkerson, and Johnson [7] were able to solve, at that time, a large-scale instance with 42 cities. It took over 20 years until the 42 cities record was broken by a young German mathematician who had just left complex analysis [15] to undertake the solution of difficult real-world problems. Grötschel was able to solve a 120 cities instance using the valid inequalities he discovered for the TSP polytope within a branch-and-bound framework [16, 17]. Later Grötschel and Holland [18] improved this result by solving TSP instances with up to 1,000 cities to optimality. The asymmetric TSP with time-windows (see Ascheuer, Fischetti, and Grötschel [1, 2]) is a generalization of the TSP, which is another special case of our air-travel routing and scheduling problem.

The air-travel routing and scheduling problem is closely related to an on-demand air taxi problem considered by Espinoza et al. [9, 10]. Our problem, however, is more tightly constrained, and seemingly more difficult, primarily because of the limited number of fueling stations. We are, however, able to obtain feasible solutions by a fast heuristic that includes a greedy construction followed by simple local search. Then we enhance the local search by solving small subproblems including only two planes, exactly. Nevertheless those two-plane MILPs are still very difficult and the remainder of the paper deals with how to solve them. We consider several approaches. The most promising one removes the time structure of the problem to obtain a relaxation and then restores the time structure using a branch-and-bound process. Nevertheless solving these small MILPs only yields minor improvements and validates the quality of the basic heuristics used in practice.

The remainder of this paper is organized as follows. In Sect. 2 we review the relevant literature and give a formulation of the air-travel routing and scheduling problem as a mixed-integer linear program that uses a bi-level flow-over-flow structure of variables and a time-space network. This model is far too big to solve by standard mixed-integer programming techniques. In Sect. 3, we give a fast construction and local search heuristic for obtaining primal solutions. A variation of this heuristic is used by the company. Then to improve the local search capabilities we try to solve small sub-instances exactly. The remainder of the paper deals with methods for solving these instances. In Sect. 4 we describe aggregation of variables and valid inequalities to speed up the solution process using a branch-and-cut solver. We also tried a set partitioning formulation using column generation. Results on these approaches are reported in Sect. 5. As we will see, these classical approaches can solve only some of our test instances with two planes, ten airports and 20 flight requests. A new approach is presented in Sect. 6, which we call the *time-free formulation*. This relaxation is embedded in a branch-and-bound search procedure in Sect. 7. With this approach we obtain better results than with conventional branch-and-cut and column generation approaches and we believe that it also could be useful for other combinatorial optimization problems on time-space networks. Further conclusions are given in Sect. 8.

2 A Discrete-Time MILP Model

We give a formulation of the air-travel routing and scheduling problem as an MILP. Two facts are worth noting about our model. First, it has a bilevel flow-over-flow structure: There is a "master" flow of planes and a "sub" flow of requests. For both flows, the classical flow conservation constraints hold. Both flows are coupled by capacity constraints, so that a sub-flow can only exist on those arcs of the graph where a master flow of suitable capacity exists. Similar structures occur in several applications reported in the literature. Wieberneit [29] reviews a number of applications in service network design for freight transportation. These models typically have a bi-level structure where "sub"-level variables can only be activated on those arcs of a network where "master" level variables also take positive values. More specifically, Fügenschuh, Homfeld, and Schülldorf [13] describe a scheduling problem for cars in railway freight services where the trains are on the master-level and the cars on the sub-level. Helmberg and Röhl [19] report a case study of a joint online truck scheduling and inventory management problem for multiple warehouses. Here pallets can only be transported between warehouses if enough truck capacity is available. Erera et al. [8] use integer programming based local search for improving load planning designs in a less-than-truckload freight transportation application. Barnhart and Shen [3] utilize a bi-level variable structure for a network design problem for time-critical delivery of logistics services.

Second, our model uses a time-space network formulation, which to the best of our knowledge was first introduced by Ford and Fulkerson [12] in their analysis of maximal dynamic flows. Since then, several variants of this basic problem have been studied, for example, the quickest, minimum cost, or earliest arrival flows, among many other. For a survey we refer to Kotnyek [21] and the references therein. Many real-world problems are formulated as time-space network models, see Kennington and Nicholson [20] for a survey.

2.1 Instance Data

In this section we describe the instance data that is necessary to set up the model.

We denote by $\mathcal{P}$ the set of planes, by $\mathcal{V}$ the set of airports, and by $\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V}$ the set of direct trips between two airports. The set of pairs (i, θ), where $i \in \mathcal{V}$ and θ is a type of fuel, is denoted by $\mathcal{F}$. $\mathcal{R}$ denotes the set of passenger flight requests.

For each plane $p \in \mathcal{P}$, the number of seats for passengers is $\overline{s}_p$, the initial departure airport is D_p and the final arrival airport is A_p, where the plane starts and ends its tour, the minimum quantity of fuel on board the plane at departure is $\underline{\varphi}_p$, the maximum quantity of fuel at departure is $\overline{\varphi}_p$, the minimum quantity of fuel on board of the plane at arrival is $\underline{\psi}_p$, the maximum quantity of fuel at departure is $\overline{\psi}_p$, and the required fuel type is ρ_p.

For each tuple $(i, j) \in \mathcal{A}$ the air distance is denoted by $d_{i,j}$. It is assumed that the triangle inequality is satisfied. It is further assumed that all loops (from i to i) are included in the set $\mathcal{A}$; for such a tuple (i, i) the distance value is 0. For each tuple $(i, \theta) \in \mathcal{F}$ the fuel availability is denoted by $r_{i,\theta} \in \{0, 1\}$, where $r_{i,\theta} = 1$ if and only if fuel type θ is available at airport i. For each request $r \in \mathcal{R}$ the given data consists of the departure airport D_r, the arrival airport A_r, the number of requested passenger seats s_r, the total weight of the request (passengers and luggage) w_r, and the maximum number of allowed intermediate stops M_r. The maximum detour allowed for a request in relation to the air distance is denoted by $\overline{d}_r$.

For a plane $p \in \mathcal{P}$ on trip $(i, j) \in \mathcal{A}$ the travel cost is denoted by $c^p_{i,j}$. Travel cost depend on the distance between i and j and the flying cost of plane p per unit distance. The corresponding fuel consumption is denoted by $\gamma^p_{i,j}$. The maximum amount of fuel that can be taken during that trip is given by $\overline{f}^p_{i,j}$. It is defined as the tank's fuel capacity minus the fuel consumption for the trip from i to j, and the reserve fuel. The maximum trip payload $\overline{w}^p_{i,j}$ is defined as the minimum of the maximum takeoff payload of plane p at airport i plus the fuel that is consumed during the trip from i to j and the maximum landing payload for this plane at airport j. Here the maximum takeoff and landing payload of a plane take into account the takeoff or landing weight minus the empty weight minus reserve fuel and minus the weight of the pilot.

In order to set up a discrete time-space network of the scheduling and routing events we introduce a time discretization. The resolution of time we consider in our numerical computations is a 5 minute discretization. It is not necessary to consider smaller time step units because all input time data is aligned to 5 minutes. This discretization leads to a finite time horizon for each plane p, that is denoted by $\mathcal{T}_p$ and which represents the set of time steps used for this plane. This smallest and largest element in this set are chosen in such way that the plane is able to reach any request at its earliest departure time, even if it needs to fly to a refueling airport before, and that it is able to deliver any request at its latest arrival time, fly to a refueling airport after that, and finally reach the destination airport where the tour is finished. The number of travel unit time steps for plane p from i to j is denoted by $\delta^p_{i,j}$. It includes the turnover unit time steps at airport j that are necessary to turn the plane so that it is ready for the next takeoff. For each request r the earliest departure and latest arrival time steps are derived from the corresponding earliest departure and latest arrival time data using the same discretization of time. The requests also have a discrete time horizon, which for request r is defined as the set $\mathcal{T}_r$.

2.2 The Model Formulation

2.2.1 Variables

To represent the schedule of the planes we introduce variables

$$\forall (i,j) \in \mathcal{A}, p \in \mathcal{P}, t \in \mathcal{T}_p : y^{p,t}_{i,j} \in \{0,1\}, \tag{1a}$$

where $y^{p,t}_{i,j} = 1$ iff plane p flies from i to j and arrives at j at time step t. We introduce additional binary variables

$$\forall p \in \mathcal{P}, t \in \mathcal{T}_p : y^{p,t}_{\text{dep}}, y^{p,t}_{\text{arr}} \in \{0,1\} \tag{1b}$$

to model the departure resp. arrival of plane p at time step t.

The variables

$$\forall r \in \mathcal{R}, (i,j) \in \mathcal{A}_r, p \in \mathcal{P}, t \in \mathcal{T}_r : x^{r,p,t}_{i,j} \in \{0,1\} \tag{1c}$$

represent the schedule of the passengers, where $x_{i,j}^{r,p,t} = 1$ iff all passengers belonging to request r fly from i to j on plane p, arriving at j at time t. Further binary variables

$$\forall r \in \mathcal{R}, p \in \mathcal{P}, t \in \mathcal{T}_r : x_{\text{dep}}^{r,p,t}, x_{\text{arr}}^{r,p,t} \in \{0, 1\} \tag{1d}$$

are introduced to model the departure resp. arrival of request r at time step t using plane p.

The amount of fuel on plane $p \in \mathcal{P}$ on the trip (i, j) at the departure at airport i, arriving at j at time step t is modeled by the continuous variables

$$\forall (i, j) \in \mathcal{A}, p \in \mathcal{P}, t \in \mathcal{T}_p : f_{i,j}^{p,t} \in \mathbb{R}_+. \tag{1e}$$

The amount of fuel on plane p at the first departure and at the last arrival is modeled by the continuous variables

$$\forall p \in \mathcal{P}, t \in \mathcal{T}_p : f_{\text{dep}}^{p,t}, f_{\text{arr}}^{p,t} \in \mathbb{R}_+. \tag{1f}$$

The weight of plane p at time step t while flying from i to j is modeled by the continuous variables

$$\forall p \in \mathcal{P}, t \in \mathcal{T}_p, (i, j) \in \mathcal{A} : w_{i,j}^{p,t} \in \mathbb{R}_+. \tag{1g}$$

2.2.2 Objective

The objective is to minimize the operating costs of the planes given by:

$$\sum_{(i,j)\in\mathcal{A}} \sum_{t\in\mathcal{T}} \sum_{p\in\mathcal{P}} c_{i,j}^{p} \cdot y_{i,j}^{p,t} \to \text{min}. \tag{1h}$$

2.2.3 Constraints

Scheduling and Routing of Planes Each plane p departs in exactly one time step at the initial airport:

$$\forall p \in \mathcal{P} : \sum_{t\in\mathcal{T}_p} y_{\text{dep}}^{p,t} = 1. \tag{1i}$$

Plane p arrives in exactly one time step at the final airport:

$$\forall p \in \mathcal{P} : \sum_{t\in\mathcal{T}_p} y_{\text{arr}}^{p,t} = 1. \tag{1j}$$

For each airport j, plane p and at each time step t the following flow conservation constraints must hold. These constraints ensure that a plane at j that either comes from another airport i or is deployed there for the first time is either finishing its trip, waiting at airport j for one time step, or continues its journey to another airport k, where it will arrive at a future time step that corresponds to the turnover time and flight time:

$$\begin{aligned}
&\forall j \in \mathcal{V}, p \in \mathcal{P}, t \in \mathcal{T}_p : \\
&\sum_{i:(i,j)\in\mathcal{A}} y_{i,j}^{p,t} + \begin{cases} y_{\text{dep}}^{p,t}, & \text{if } j = D_p, \\ 0, & \text{else,} \end{cases} \\
&= \sum_{\substack{k:(j,k)\in\mathcal{A} \\ t+\delta_{j,k}^p \in \mathcal{T}_p}} y_{j,k}^{p,t+\delta_{j,k}^p} + \begin{cases} y_{\text{arr}}^{p,t}, & \text{if } j = A_p, \\ 0, & \text{else.} \end{cases}
\end{aligned} \tag{1k}$$

Scheduling and Routing of Requests Tourists of request r start their trips at or after the earliest starting time. For this trip, exactly one plane is selected:

$$\forall r \in \mathcal{R} : \sum_{p\in\mathcal{P}} \sum_{t\in\mathcal{T}_r} x_{\text{dep}}^{r,p,t} = 1. \tag{1l}$$

They finish their trip no later than the latest arrival time:

$$\forall r \in \mathcal{R} : \sum_{p\in\mathcal{P}} \sum_{t\in\mathcal{T}_r} x_{\text{arr}}^{r,p,t} = 1. \tag{1m}$$

Flow conservation constraints must hold for the requests:

$$\begin{aligned}
&\forall j \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{T}_r, p \in \mathcal{P} : \\
&\sum_{i:(i,j)\in\mathcal{A}_r} x_{i,j}^{r,p,t} + \begin{cases} x_{\text{dep}}^{r,p,t}, & \text{if } j = D_r, \\ 0, & \text{else,} \end{cases} \\
&= \sum_{\substack{k:(j,k)\in\mathcal{A}_r \\ t+\delta_{j,k}^p \in \mathcal{T}_r}} x_{j,k}^{r,p,t+\delta_{j,k}^p} + \begin{cases} x_{\text{arr}}^{r,p,t}, & \text{if } j = A_r, \\ 0, & \text{else.} \end{cases}
\end{aligned} \tag{1n}$$

Routing Restrictions for Requests The number of intermediate stops for the passengers is limited:

$$\forall r \in \mathcal{R} : \sum_{\substack{(i,j)\in\mathcal{A}_r \\ i\neq j}} \sum_{p\in\mathcal{P}} \sum_{t\in\mathcal{T}_r} x_{i,j}^{r,p,t} \leq M_r + 1. \tag{1o}$$

The maximal detour for passengers compared to a direct flight is bounded:

$$\forall r \in \mathcal{R} : \sum_{(i,j)\in\mathcal{A}_r} \sum_{p\in\mathcal{P}} \sum_{t\in\mathcal{T}_r} d_{i,j} \cdot x_{i,j}^{r,p,t} \leq \overline{d}_r. \tag{1p}$$

Coupling of Planes and Requests The plane flows and the request flows are coupled by seat capacity constraints:

$$\forall (i,j) \in \mathcal{A}, p \in \mathcal{P}, t \in \mathcal{T}_p : \sum_{r \in \mathcal{R}:(i,j)\in\mathcal{A}_r, t\in\mathcal{T}_r} s_r \cdot x_{i,j}^{r,p,t} \le \overline{s}_p \cdot y_{i,j}^{p,t}, \tag{1q}$$

hence if a subset of the requests uses a certain path at a certain time in a certain plane, that plane must use the same path at the very same time, and it must offer enough seats to carry all of the passengers.

Fuel Consumption and Refueling We introduce a flow conservation formulation to track the amount of fuel in a plane's tank. If a plane p does *not* use a certain trip (i, j) at time step t, then corresponding the fuel variable cannot take a positive value:

$$\forall (i,j) \in \mathcal{A}, p \in \mathcal{P}, t \in \mathcal{T}_p : f_{i,j}^{p,t} \le \overline{f}_{i,j}^{p} \cdot y_{i,j}^{p,t}. \tag{1r}$$

For airports that are non-refueling airports for the plane, the following flow conservation constraint must hold:

$$\begin{aligned} &\forall j \in \mathcal{V}, p \in \mathcal{P}, t \in \mathcal{T}_p, r_{j,\rho_p} = 0 : \\ &\sum_{i:(i,j)\in\mathcal{A}} f_{i,j}^{p,t} + \begin{cases} f_{\text{dep}}^{p,t}, & \text{if } j = D_p, \\ 0, & \text{else,} \end{cases} \\ &\quad = \sum_{\substack{k:(j,k)\in\mathcal{A} \\ t+\delta_{j,k}^p \in\mathcal{T}_p}} \left(f_{j,k}^{p,t+\delta_{j,k}^p} + \gamma_{j,k}^p \cdot y_{j,k}^{p,t+\delta_{j,k}^p}\right) + \begin{cases} f_{\text{arr}}^{p,t}, & \text{if } j = A_p, \\ 0, & \text{else.} \end{cases} \end{aligned} \tag{1s}$$

For airports that are refueling airports for the plane, we have a relaxation of the flow conservation constraints that allow for a refueling:

$$\begin{aligned} &\forall j \in \mathcal{V}, p \in \mathcal{P}, t \in \mathcal{T}, r_{j,\rho_p} = 1 : \\ &\sum_{i:(i,j)\in\mathcal{A}} f_{i,j}^{p,t} + \begin{cases} f_{\text{dep}}^{p,t}, & \text{if } j = D_p, \\ 0, & \text{else,} \end{cases} \\ &\quad \le \sum_{\substack{k:(j,k)\in\mathcal{A} \\ t+\delta_{j,k}^p \in\mathcal{T}_p}} \left(f_{j,k}^{p,t+\delta_{j,k}^p} + \gamma_{j,k}^p \cdot y_{j,k}^{p,t+\delta_{j,k}^p}\right) + \begin{cases} f_{\text{arr}}^{p,t}, & \text{if } j = A_p, \\ 0, & \text{else.} \end{cases} \end{aligned} \tag{1t}$$

The fuel limits at departure and arrival airports for each plane p must be obeyed:

$$\forall p \in \mathcal{P}, t \in \mathcal{T}_p : \begin{cases} \underline{\varphi}_p \cdot y_{\text{dep}}^{p,t} \leq f_{\text{dep}}^{p,t}, \\ f_{\text{dep}}^{p,t} \leq \overline{\varphi}_p \cdot y_{\text{dep}}^{p,t}, \\ \underline{\psi}_p \cdot y_{\text{arr}}^{p,t} \leq f_{\text{arr}}^{p,t}, \\ f_{\text{arr}}^{p,t} \leq \overline{\psi}_p \cdot y_{\text{arr}}^{p,t}. \end{cases} \tag{1u}$$

A formulation similar to the one above for fuel consumption was also used by Ascheuer, Fischetti, and Grötschel [1, 2] in a model for the single-vehicle traveling salesman problem with time-windows. Ascheuer et al. give credit to Maffioli and Sciomachen [22] resp. van Eijl [28]. In fact, it is much older since it was used by Gavish and Graves [14] in a model for the vehicle routing problem with time windows.

Weight Restrictions The maximum takeoff and landing weights for the plane depends on the performance of the plane and also on the airport (its runway length and the height of the trees in surrounding forests). The payload weight (passengers with baggage and fuel) is computed by the following constraints:

$$\forall (i,j) \in \mathcal{A}, p \in \mathcal{P}, t \in \mathcal{T}_p : w_{i,j}^{p,t} = \sum_{r \in \mathcal{R}:(i,j) \in \mathcal{A}_p, t \in \mathcal{T}_r} w_r \cdot x_{i,j}^{r,p,t} + f_{i,j}^{p,t}. \tag{1v}$$

We have a bound on the plane's maximum allowable takeoff and landing payload weight. This bound limits the payload weight at the departure airport and takes into account the fuel that is burned while traveling to the arrival airport:

$$\forall (i,j) \in \mathcal{A}, p \in \mathcal{P}, t \in \mathcal{T}_p : w_{i,j}^{p,t} \leq \overline{w}_{i,j}^{p} \cdot y_{i,j}^{p,t}. \tag{1w}$$

3 A Primal Heuristic

The MILP given in Sect. 2 cannot be solved even for obtaining good solutions for the size of instances needed which have 20 planes and 120 requests per day over a 7 day period. Thus for the practical application, it is necessary to design a primal heuristic that will hopefully yield good solutions.

We developed and implemented a primal heuristic that tries to generate feasible solutions. This heuristic mimics in part a human planner who constructs schedules for planes and requests from scratch, and it has a local search phase to improve the schedule afterward. The procedure consists of two parts: a construction first phase and a local search second phase. Both phases use a randomized greedy search procedure, that inserts a request in the locally best position of the current partial schedules. Below we describe the details of these heuristic algorithms. Since the algorithm has random components, it is repeated several times and then the solution with lowest cost is selected.

Insertion Heuristic The insertion heuristic is a randomized greedy-type algorithm. Given a request and the partially constructed schedules of a set of planes, it tries to insert the request by selecting two of its trips and inserting the new request between them. An insertion is valid only if the time window constraints, plane capacity constraints, intermediate stops constraints, and weight constraints are not violated. The algorithm enumerates all possible positions for all planes to insert the request and sets the valid insertions on a list. Finally one insertion from this list is randomly selected, where cheaper insertions (with respect to the increase of cost in the objective function) have a higher priority.

Construction Phase The basic idea of the construction phase is to build schedules sequentially by finding the locally best request-plane-pair with respect to some evaluation measure at each step. At the beginning each plane starts with an empty schedule. Then the following steps are repeated iteratively until all requests are scheduled. For each unscheduled request the insertion heuristic is called to identify the locally best plane to which the request can be assigned. In this way the locally best request-plane-pair is found. When inserting this request, the increase of the objective function is the least among all other potential insertions. Whenever the fuel level of a plane is too low then a trip to the nearest refueling location is inserted. Requests which then become infeasible because of too many intermediate stops are removed from the current schedule and have to be re-scheduled again. This might lead to cycling, so the heuristic terminates without a solution if a given iteration limit is reached. The pseudocode description of the construction phase is given in Algorithm 1. It uses Algorithm 2 and Algorithm 3 as subroutines.

Local Search Phase The reinsertion local search procedure starts with given feasible schedules for all requests. It iteratively removes and reinserts requests. In order to identify one or several requests that are suitable for removing and rescheduling, it applies different strategies, for example, removing a single request, all requests of one trip, or requests of several consecutive trips. Requests marked for removal are randomly selected, where preference is given to those requests that, by some measure, contribute high cost to the schedule. The removed requests are added to a list. Then they are re-added to the existing schedules using the same rules as in the construction phase. When fuel violations occur, a trip to the nearest refueling airport is scheduled. If this leads to further constraint violations for some requests, the respective requests are also removed from the schedule and added to the list of unscheduled requests. When all requests are rescheduled again, if the total cost has decreased we continue from the new schedule, if not we undo the most recent changes and continue with the previous schedule. Then we repeat this process from the beginning of the local search phase until a limit on the number of iterations is reached. The pseudocode description of the local search phase is given in Algorithm 4.

Algorithm 1: Construction Phase

```
Data: an instance of the air-travel scheduling problem
Initialize each plane with an empty schedule;
if the plane does not start at a refuel point then
    Add one flight to the nearest refuel point
end
while list of requests is not empty do
    CurrentRequest ← first element in list of requests;
    CandidatePlane ← getCandidatePlaneForRequest(CurrentRequest, Planes);
    if CandidatePlane is not Null then
        improvement ← true
    end
    else
        improvement ← false
    end
    ite ← 0;
    while improvement is true and ite < MaxIteration do
        ite++;
        CandidatePlaneCostIncrease ← increase in cost of CandidatePlane schedule after
        inserting CurrentRequest;
        RemainingRequests ← list of requests;
        Remove CurrentRequest from RemainingRequests;
        CandidateRequest ← getCandidateRequestForPlane(CandidatePlane,
        RemainingRequests);
        if CandidateRequest is not Null then
            CandidateRequestCostIncrease ← increase in cost of CandidatePlane
            schedule after inserting CandidateRequest
        end
        else
            improvement ← false
        end
        if improvement is true and
        LocallyBestRequestScheduleCostIncrease/LocallyBestPlaneScheduleCostIncrease
        < CriticalRatio then
            CurrentRequest ← CandidateRequest;
            CandidatePlane ← getCandidatePlaneForRequest(CurrentRequest, Planes);
            if CandidatePlane is not Null then
                improvement ← true
            end
            else
                improvement ← false
            end
        end
        else
            improvement ← false
        end
    end
    if CurrentRequest is not Null and CandidatePlane is not Null then
        Update schedules with CurrentRequest and CandidatePlane
    end
end
```

Algorithm 2: getCandidatePlaneForRequest

```
Data: a request and a list of planes
Tentatively insert request to the schedule of each plane;
if any feasible schedules are found then
    return plane of schedule with smallest increase in cost
end
else
    return Null
end
```

Algorithm 3: getCandidateRequestForPlane

```
Data: a plane and a list of requests
Tentatively insert each request to the schedule of plane;
if any feasible schedules are found then
    return request which leads to the smallest increase in cost
end
else
    return Null
end
```

3.1 Test Instances and Computational Results

We gathered 24 test instances, each of which is defined by a subset of requests flown on a single day by two planes, see Table 1. By reassigning these requests to the two planes we aim to re-optimize a given feasible schedule by local search. The two planes are chosen to have likely opportunities for cost improvements by switching some requests between them, for example, they operate over the course of the day in close proximity. The planes are of two different sizes: The Cessna C206 is a 5-seater with an operating range of about 1,000 km, and the Cessna C208 is a 12-seater with an operating range of about 1,500 km. The instances have between 8 and 13 airports and between 10 and 23 requests to be scheduled. The size of the respective time-space mixed-integer linear programs in terms of number of variables, constraints, and non-zeros are given in the next columns. On average, the MILPs have 223819 variables (approx. 80 % binary and 20 % continuous), 144678 constraints, and 1069674 non-zero entries in the constraint matrix (which is an average fill of less than 0.0033%). The objective function values of the primal solutions found by the heuristic are shown in the right-most column of Table 1.

The primal feasible solutions are used as initial solutions for the branch-and-bound process. We use IBM ILOG CPLEX 12.4 as our MILP solver on a quad core Intel Core i7 CPU 870 with 2.93 GHz and 8 MB cache running the Linux

Algorithm 4: Local Search Phase

```
Data: an instance of the air-travel scheduling problem and a feasible solution
ite ← 0;
improvement ← true;
while improvement is true and ite < MaxIteration do
    improvement ← false;
    ite++;
    RequestsGroup ← empty list;
    foreach Schedule in list of schedules do
        foreach consecutive three flights do
            Put all requests in the flights into a RequestGroup, and add the
            RequestGroup to RequestsGroup;
        end
    end
    foreach RequestGroup in RequestsGroup do
        Remove all requests in RequestGroup from the list of schedules;
        Handle fuel violations if there are any;
        foreach Request in RequestGroup do
            Reinsert Request into the locally best position;
        end
        Calculate the operation cost of new schedules;
        If there are any requests that failed to be reinserted, add a penalty cost;
        if new cost is smaller than old cost then
            Accept the new schedule;
            improvement ← true;
        else
            Reject the new schedule;
        end
    end
end
```

operating system. Results for this solver are presented in Table 2. It turns out that the MILP solver is barely able to handle these instances. On average it took almost 300 seconds to solve just the root LP relaxation. The gap between the LP lower bound and the heuristic upper bound is 22.44 %. For the subsequent branch-and-cut process a time limit of 3 hours (10800 seconds) was given. Only in five out of the 24 instances was the solver able to start branching. All the other instances got stuck in the cutting plane phase at the root node. However, the gap could be reduced to 13.04 % by adding these cutting planes, and in three cases optimality of the primal solution was proven. Moreover, a better primal solution was found for only one instance, specifically the only one in which more than a few nodes in the tree were evaluated.

Table 1 Test instances and primal solutions found by the heuristic

Instance	Planes (5s, 12s)	Airports	Requests	Variables	Constraints	Non-zeros	Primal solution
BUF-AIV	1, 1	10	11	158926	103350	756684	12614
BUF-ANT	1, 1	12	12	290998	216908	1354526	20724
BUF-BEE	1, 1	12	16	297356	176884	1447721	17633
BUF-BOK	1, 1	10	13	181882	114426	871538	12917
BUF-EGL	0, 2	12	17	351406	199744	1724375	22113
BUF-GNU	0, 2	12	18	299676	144700	1506505	17350
BUF-JKL	1, 1	12	14	296660	193292	1416232	20774
BUF-LEO	0, 2	13	23	505160	254866	2523918	24938
BUF-NAS	0, 2	10	11	161054	106248	767104	15931
BUF-OWL	1, 1	12	13	282794	195232	1334779	16898
BUF-ZEB	0, 2	10	11	157406	101280	752739	15925
EGL-BEE	1, 1	10	11	177190	132344	822825	16653
EGL-GNU	0, 2	9	13	151319	102283	713795	20238
EGL-LEO	0, 2	9	18	204772	123848	985122	19388
GNU-BEE	1, 1	9	12	125110	92942	580597	11311
GNU-JKL	1, 1	9	10	123152	99340	557441	11098
GNU-LEO	0, 2	12	19	358394	204280	1759808	18450
LEO-AIV	1, 1	8	12	129856	101800	589914	13615
LEO-ANT	1, 1	8	13	132508	102220	603641	17381
LEO-BEE	1, 1	11	17	291405	192755	1389432	18890
LEO-BOK	1, 1	9	14	186362	138914	862180	15372
LEO-JKL	1, 1	9	15	182533	128993	852243	17551
LEO-NAS	1, 1	9	12	155494	122560	708084	18231
LEO-OWL	1, 1	9	14	170231	123065	790973	15827
Average		10	14	223819	144678	1069674	17160

4 Improving the Dual Bound

We tried to improve the solution process by aggregating variables, which results in a more compact formulation (although it has a weaker LP relaxation), and by adding additional valid inequalities.

4.1 Aggregated Weight Variables

Aggregation of variables is a well-known presolve technique to speed-up the solution of MILPs (Marchand and Wolsey [23]). Here we apply it to the time and trip dependent weight variables $w_{i,j}^{p,t}$. As a first step we aggregate the trip index (i, j) to

Table 2 Computational results for the discrete time model using the disaggregated time-trip-dependent weight formulation

Instance	Root time	Obj. val.	Gap	b&c time	Nodes	Dual bd.	Obj. val.	Gap
BUF-AIV	267	10682	15.32 %	10800	0	11482	12614	8.97 %
BUF-ANT	444	15116	27.06 %	10800	0	16118	20724	22.22 %
BUF-BEE	636	10721	39.2 %	10800	0	12790	17633	27.47 %
BUF-BOK	257	10862	15.91 %	10800	0	12859	12917	0.45 %
BUF-EGL	455	16790	24.07 %	10800	0	17817	22113	19.42 %
BUF-GNU	499	12135	30.06 %	10800	0	12915	17350	25.56 %
BUF-JKL	1165	13544	34.8 %	10800	0	15371	20774	26.01 %
BUF-LEO	802	19384	22.27 %	10800	0	20057	24938	19.57 %
BUF-NAS	327	10462	34.33 %	10803	0	11098	15931	30.34 %
BUF-OWL	781	12599	25.44 %	10800	0	14050	16898	16.86 %
BUF-ZEB	430	11498	27.8 %	10800	0	12908	15925	18.95 %
EGL-BEE	44	16344	1.85 %	2227	0	16653	16653	0.00 %
EGL-GNU	76	11228	44.52 %	10800	0	16899	20238	16.50 %
EGL-LEO	112	18044	6.93 %	10800	0	19194	19388	1.00 %
GNU-BEE	127	7375	34.8 %	10800	0	10820	11311	4.34 %
GNU-JKL	54	8423	24.1 %	3145	0	11098	11098	0.00 %
GNU-LEO	172	13075	29.13 %	10800	0	14057	18450	23.81 %
LEO-AIV	33	12224	10.21 %	10800	1	13054	13615	4.12 %
LEO-ANT	38	14057	19.13 %	10800	5	16190	17381	6.85 %
LEO-BEE	206	16937	10.39 %	10800	0	17882	18890	5.39 %
LEO-BOK	72	13294	13.52 %	10800	5	14531	15372	5.48 %
LEO-JKL	66	15728	10.39 %	10800	0	17490	17551	0.35 %
LEO-NAS	9	12835	29.6 %	10815	174	12849	18192	29.37 %
LEO-OWL	29	14621	7.62 %	2241	5	15827	15827	0.00 %
Average	296	13249	22.44 %	9768	8	14750	17158	13.04 %

an airport index i introducing variables $w_i^{p,t}$, and as a second step we also omit the airport index to obtain a time dependent variable $w^{p,t}$ only.

4.1.1 Time-Airport-Dependent Weight Variables

Instead of variable $w_{i,j}^{p,t}$ we introduce variables that only depend on the airport i:

$$\forall i \in \mathcal{V}, p \in \mathcal{P}, t \in \mathcal{T}_p : w_i^{p,t} \in \mathbb{R}_+. \tag{2a}$$

Now the payload weight (passengers with baggage and fuel) is computed by

$$\forall i \in \mathcal{V}, p \in \mathcal{P}, t \in \mathcal{T}_p : w_i^{p,t} = \sum_{r \in \mathcal{R}: t \in \mathcal{T}_r} \sum_{j:(i,j) \in \mathcal{A}_p} w_r \cdot x_{i,j}^{r,p,t} + \sum_{j:(i,j) \in \mathcal{A}} f_{i,j}^{p,t}. \quad (2b)$$

And as bounds on the payload weight we then get

$$\forall i \in \mathcal{V}, p \in \mathcal{P}, t \in \mathcal{T}_p : w_i^{p,t} \leq \sum_{j:(i,j) \in \mathcal{A}} \overline{w}_{i,j}^{p} \cdot y_{i,j}^{p,t}. \quad (2c)$$

4.1.2 Time-Dependent Weight Variables

This is the highest aggregation level for the weight variables. Instead of variable $w_{i,j}^{p,t}$ we introduce variables

$$\forall p \in \mathcal{P}, t \in \mathcal{T}_p : w^{p,t} \in \mathbb{R}_+. \quad (3a)$$

Then the payload weight (passengers with baggage and fuel) is computed as

$$\forall p \in \mathcal{P}, t \in \mathcal{T}_p : w^{p,t} = \sum_{r \in \mathcal{R}: t \in \mathcal{T}_r} \sum_{(i,j) \in \mathcal{A}_p} w_r \cdot x_{i,j}^{r,p,t} + \sum_{(i,j) \in \mathcal{A}} f_{i,j}^{p,t}, \quad (3b)$$

and the weight is bounded by

$$\forall p \in \mathcal{P}, t \in \mathcal{T}_p : w^{p,t} \leq \sum_{(i,j) \in \mathcal{A}} \overline{w}_{i,j}^{p} \cdot y_{i,j}^{p,t}. \quad (3c)$$

4.2 Valid Inequalities

Valid inequalities, or cutting planes, are known to speed up the solving of MILPs (Marchand et al. [24], Cornuejols [6]). We describe additional valid inequalities for the air-travel routing and scheduling problem, and discuss how much they contribute to the solution process.

4.2.1 Minimum Fuel Cuts

When landing at airport i each plane p needs at least enough fuel to get from there to the nearest refueling location. Hence we define the minimum amount of fuel as

$$m_{i,p} := \min\{\gamma_{i,j}^{p} : (i,j) \in \mathcal{A}, r_{j,\rho_p} = 1\}. \quad (4)$$

This value is used as a lower bound for the amount of fuel:

$$\forall (i,j) \in \mathcal{A}, p \in \mathcal{P}, t \in \mathcal{T}_p : m_{i,p} \cdot y_{i,j}^{p,t} \leq f_{i,j}^{p,t}. \quad (5)$$

4.2.2 Minimum Number of Fuel-Stop Cuts

If the amount of fuel that is required at the arrival airport is higher than the amount at the departure airport, then the plane needs to visit a suitable refueling airport at least once during its schedule. Therefore it has to arrive at one of these airports,

$$\forall p \in \mathcal{P}, \overline{\varphi}_p - \gamma^p_{D_p,A_p} < \underline{\psi}_p : \sum_{t \in \mathcal{T}_p} \sum_{\substack{(i,j) \in \mathcal{A} \\ i \neq j, r_{j,\rho_p}=1}} y^{t,p}_{i,j} \geq 1 \tag{6}$$

and it has to leave these airports, unless it is the final destination airport for the plane:

$$\forall p \in \mathcal{P}, \overline{\varphi}_p - \gamma^p_{D_p,A_p} < \underline{\psi}_p :$$
$$\sum_{t \in \mathcal{T}_p} \sum_{\substack{(i,j) \in \mathcal{A} \\ i \neq j, r_{i,\rho_p}=1}} y^{t,p}_{i,j} + \begin{cases} \sum_{t \in \mathcal{T}_p} y^{p,t}_{\text{arr}}, & \text{if } r_{A_p,\rho_p} = 1 \\ 0, & \text{else,} \end{cases} \geq 1. \tag{7}$$

4.2.3 Maximum Number of Pickup and Delivery Cuts

The number of stops at non-refueling airports is bounded from above. For the pickup stops that is,

$$\forall i \in \mathcal{V}, \sum_{(i,\theta) \in \mathcal{F}} r_{i,\theta} = 0 :$$
$$\sum_{\substack{j:(i,j) \in \mathcal{A} \\ i \neq j}} \sum_{p \in \mathcal{P}} \sum_{t \in \mathcal{T}_p} y^{p,t}_{i,j} \leq \left|\{p \in \mathcal{P} : i = D_p\}\right| + \left|\{r \in \mathcal{R} : i \in \{D_r, A_r\}\right|, \tag{8}$$

and for the delivery stops that is

$$\forall j \in \mathcal{V}, \sum_{(j,\theta) \in \mathcal{F}} r_{j,\theta} = 0 :$$
$$\sum_{i:(i,j) \in \mathcal{A}} \sum_{p \in \mathcal{P}} \sum_{t \in \mathcal{T}_p} y^{p,t}_{i,j} \leq \left|\{p \in \mathcal{P} : j = A_p\}\right| + \left|\{r \in \mathcal{R} : j \in \{D_r, A_r\}\right|. \tag{9}$$

4.3 Computational Results

Computational results for the aggregated weight formulation and all valid inequalities together are shown in Table 3. The time for solving the root LP relaxation went

Table 3 Computational results for the discrete time model using the aggregated time-dependent weight formulation and all valid inequalities

Instance	Root time	Dual bd.	Gap	b&c time	Nodes	Dual bd.	Obj. val.	Gap
BUF-AIV	202	10902	13.57 %	10800	0	11727	12614	7.03 %
BUF-ANT	212	15229	26.51 %	10970	0	15893	20724	23.31 %
BUF-BEE	211	11702	33.64 %	10800	0	15051	17633	14.64 %
BUF-BOK	201	11044	14.5 %	10800	1	11914	12917	7.77 %
BUF-EGL	301	17091	22.71 %	10800	0	17091	22113	22.71 %
BUF-GNU	608	12925	25.5 %	10800	0	13505	17350	22.16 %
BUF-JKL	1187	13934	32.92 %	10800	0	16870	20774	18.79 %
BUF-LEO	771	19377	22.3 %	10800	0	20057	24938	19.57 %
BUF-NAS	153	10933	31.37 %	10875	0	11965	15931	24.90 %
BUF-OWL	880	12521	25.9 %	10800	0	12775	16898	24.40 %
BUF-ZEB	149	11697	26.55 %	10821	0	12597	15925	20.90 %
EGL-BEE	5	16519	0.8 %	422	0	16653	16653	0.00 %
EGL-GNU	82	11555	42.9 %	10818	1	16214	20238	19.88 %
EGL-LEO	43	18970	2.15 %	1744	0	19388	19388	0.00 %
GNU-BEE	56	7323	35.26 %	10800	0	10117	11311	10.56 %
GNU-JKL	58	8620	22.33 %	10800	0	10986	11098	1.01 %
GNU-LEO	307	13394	27.4 %	10899	0	13855	18450	24.91 %
LEO-AIV	24	13033	4.27 %	10800	8	13251	13615	2.67 %
LEO-ANT	88	14748	15.15 %	10800	3	15899	17381	8.53 %
LEO-BEE	184	17118	9.43 %	10800	3	17761	18890	6.03 %
LEO-BOK	91	13908	9.52 %	10800	0	13908	15372	9.52 %
LEO-JKL	99	15891	9.46 %	10870	3	16929	17551	3.55 %
LEO-NAS	4	18076	0.85 %	1005	1	18192	18192	0.00 %
LEO-OWL	16	15135	4.37 %	956	0	15827	15827	0.00 %
Average	247	13819	19.14 %	9191	0	14934	17158	12.20 %

down from 295 seconds to 247 seconds on average, and the root gap improved from 22.44 % to 19.14 %. Still, the solver was only able to finish the cutting plane phase at the root node (and started the branch-and-bound phase) for a few instances. For four (instead of three before) instances global optimality was proven, and on average the integrality gap when reaching the time limit improved from 13.04 % to 12.20 %. No better primal solutions were found for any instances. From these computational results we can conclude that solving a standard MILP for these problems does not yield good solutions in a reasonable amount of time. We next consider a column generation approach.

5 A Set Partitioning Formulation

Set partitioning formulations are a way to tackle network design problem because their LP relaxations generally give tighter bounds and frequently good primal solutions can be found efficiently. For example, Borndörfer, Grötschel, and Pfetsch [5] used a column generation approach for line planning in public transport. Usually these formulations lead to constraint matrices with a small number of rows but an exponential number of columns that cannot be generated explicitly. Instead, one uses a small subset of initial columns and generates more columns on demand by solving a pricing problem at each iteration. For a general introduction to column generation in integer programming we refer to Barnhart et al. [4] and for a historical survey to Nemhauser [25].

Set partitioning problems are integer programming problems of the general form

$$\begin{aligned} &\min c^T z \\ &\text{s.t.}\quad Az = 1, \\ &\qquad\; z \in \{0,1\}^n, \end{aligned} \tag{10}$$

where $A \in \{0,1\}^{m\times n}$ and $c \in \mathbb{R}^n_+$, see [26]. We can reformulate the air-travel scheduling problem as such a set partitioning problem. The columns of A represent feasible schedules for a single plane, and the rows of A represent either the index of the respective request or the index of the respective plane. For example, the column

$$\begin{array}{r} \text{request\#1 ..} \\ \text{request\#2 ..} \\ \text{request\#3 ..} \\ \text{request\#4 ..} \\ \text{plane\#1 ..} \\ \text{plane\#2 ..} \end{array} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \tag{11}$$

represents a schedule for the second plane which serves requests one and four. Note that in each column of A exactly one row corresponding to the planes has the entry 1, whereas the number of ones in the rows corresponding to the requests can vary between zero and the number of requests. Moreover, the detailed scheduling information, that is, the ordering of the requests and the sequence of airports, the time windows, and the fueling issues, are not stored directly in the column. Only columns that fulfill all these hard restrictions are allowed to be included in the matrix A. The variable $z_i \in \{0,1\}$ specifies if column i is chosen in a solution ($z_i = 1$) or not ($z_i = 0$). The overall goal of the set partitioning problem is to choose a subset of columns such that each plane is deployed once, each request is scheduled once, and the overall cost is minimized.

The number of possible schedules for each plane is too large to set up matrix A directly, although an optimal solution consists of just a very small number of columns, that is, only as many columns as planes are chosen. A column generation approach is used to overcome this difficulty. At the beginning only a few columns are used to set up an initial matrix A. In our case we transform the solution found by our primal heuristic into columns for A. Then the LP relaxation of (10) is solved. Denote by λ_r for $r \in \mathcal{R}$ the dual variables of the request rows, and by μ_p for $p \in \mathcal{P}$ the dual variables of the plane rows. The reduced cost rc_i of a column $a_{i,\cdot}$ of A representing a feasible schedule is defined as

$$rc_i := oc_i - \sum_{r \in \mathcal{R}} \lambda_r a_{i,r} - \sum_{p \in \mathcal{P}} \mu_p a_{i,p}, \tag{12}$$

where oc_i are the operational cost of the schedule as defined by (1h). We need to determine if there exists a new column representing a feasible schedule that still has negative reduced cost.

This column generation MILP is first solved by a modification of our primal heuristic. The previous objective function is replaced by (12). If the new schedules correspond to columns with negative reduced cost then these columns are added to the matrix A, and the LP relaxation is re-optimized. If no further column with negative reduced cost is identified then we start searching for columns with a modified version of the air-travel routing and scheduling model.

This MILP is solved once for each plane $p \in \mathcal{P}$. Its objective function for a fixed p is

$$\sum_{(i,j) \in \mathcal{A}} \sum_{t \in \mathcal{T}} c^p_{i,j} \cdot y^{p,t}_{i,j} - \sum_{r \in \mathcal{R}} \lambda_r \alpha_r - \mu_p \to \min. \tag{13}$$

Here $\alpha_r \in \{0, 1\}$ for each $r \in \mathcal{R}$ is a new binary variable, where $\alpha_r = 1$ if and only if request r is transported. The constraints are as before with two exceptions: The constraints (1l) and (1m) are replaced by

$$\forall r \in \mathcal{R} : \sum_{t \in \mathcal{T}_r} x^{r,p,t}_{\text{dep}} = \alpha_r, \tag{14}$$

and

$$\forall r \in \mathcal{R} : \sum_{t \in \mathcal{T}_r} x^{r,p,t}_{\text{arr}} = \alpha_r, \tag{15}$$

respectively. All of the other constraints the same, except that the summations or quantifiers over the set of $\mathcal{P}$ are removed, since this column generation MILP is set up for a fixed p (and has to be repeated for each p separately).

The MILP is then solved until the first feasible solution with a negative objective function value occurs, if such a solution exists. The vector $(\alpha_r)_{r \in \mathcal{R}}$ is then added to matrix A as a new column. If no such vector exists for any $p \in \mathcal{P}$ then all columns have non-negative reduced cost and no further "promising" column exists, hence the column generation method terminates. The objective function value of the set partitioning LP relaxation then is a lower bound on the objective function value of the plane scheduling problem. As a heuristic we can then solve the set partitioning problem to integer optimality which in some cases gives better primal solutions. However, one cannot expect that this integer solution is a global optimal solution. If

Table 4 Computational results for the column generation approach using heuristic and exact methods to generate new columns

Instance	Heur. col.	Heur. time	MIP col.	MIP time	Cols.	Time	Sp. LP val.	Sp. sol. val.	Gap
BUF-AIV	37	83	10	37964	47	38047	n.a.	12614	n.a.
BUF-ANT	55	47	0	20000	55	20047	n.a.	20615	n.a.
BUF-BEE	278	134	0	20000	278	20134	n.a.	17633	n.a.
BUF-BOK	49	84	9	34979	58	35062	n.a.	12917	n.a.
BUF-EGL	178	47	0	20000	178	20047	n.a.	21638	n.a.
BUF-GNU	305	166	0	20000	305	20166	n.a.	17350	n.a.
BUF-JKL	172	51	0	20000	172	20051	n.a.	20374	n.a.
BUF-LEO	436	180	0	20000	436	20180	n.a.	24938	n.a.
BUF-NAS	227	99	0	20000	227	20099	n.a.	15122	n.a.
BUF-OWL	77	51	4	35181	81	35233	n.a.	16649	n.a.
BUF-ZEB	155	42	1	27828	156	27870	n.a.	15925	n.a.
EGL-BEE	98	103	14	1589	112	1693	16653	16653	0.00 %
EGL-GNU	122	98	10	39210	132	39309	n.a.	19638	n.a.
EGL-LEO	616	274	8	2648	624	2922	19388	19388	0.00 %
GNU-BEE	103	75	4	25090	107	25164	n.a.	11165	n.a.
GNU-JKL	55	20	0	20000	55	20020	n.a.	11098	n.a.
GNU-LEO	472	178	0	20000	472	20178	n.a.	18450	n.a.
LEO-AIV	71	95	4	985	75	1080	13615	13615	0.00 %
LEO-ANT	149	196	11	26674	160	206870	n.a.	17381	n.a.
LEO-BEE	299	163	3	32867	302	33030	n.a.	18890	n.a.
LEO-BOK	177	95	4	21588	181	21683	n.a.	15372	n.a.
LEO-JKL	175	240	27	39606	202	39846	n.a.	17551	n.a.
LEO-NAS	80	77	1	4	81	81	18192	18192	0.00 %
LEO-OWL	68	54	6	21131	74	21186	n.a.	15827	n.a.

this is desired the column generation process needs to be integrated into a branch-and-bound process itself, which leads to a branch-and-price method.

The results of the column generation set partitioning reformulation are summarized in Table 4. The heuristic solution is used as the initial columns for matrix A. The next two columns show the results of the heuristic used for column generation: The number of columns generated over all iterations, and the CPU time used for the heuristic. When the heuristic does not find any new columns with negative reduced cost, the exact MILP is called. In the fourth and fifth column of the table we give the number of columns found by the MILP and the CPU used to solve the MILPs. The next two columns show the total number of generated columns and the total CPU time. If the method terminates and no more columns with negative reduced cost were found, then it is possible to obtain a meaningful lower bound from solving the LP relaxation of the set partitioning problem. This happens in four out of

24 cases, including one case for which the previous branch-and-cut approach could not prove optimality. In these cases the LP relaxation yields a bound that equals the optimal integer solution, hence it proves the global optimality of the solution. In three cases out of the four the heuristic primal solution is indeed a global optimal, and in one case it could be slightly improved. However, in 20 out of 24 cases the column generation process did not terminate properly, because in these cases the pricing mixed-integer linear problem could not be solved within a given time limit of 10,000 seconds (for each of the two planes). Thus there might exist columns with negative reduced cost, and therefore a lower bound could not be determined from the set partitioning LP relaxation. Still it is possible to solve this problem to integer optimality, which in seven of 20 cases gave a better primal feasible solution. The set partitioning/column generation approach gives slightly better results than trying to solve the MILP directly. However it still falls short of our goal.

6 A Time-Free MILP Model

A fundamental bottleneck of our air-travel scheduling model, and many other models based on time-expanded graphs, is the dependency of the model's size, both in the number of variables and constraints, on the granularity of the time discretization. If the application requires a fine time grid, the model size can be too large to solve by standard methods because of high running times for solving the node LP relaxations within the branch-and-bound tree and also a high memory requirement for storing the tree. One potential way to deal with this problem is to introduce a variable grid size based on an adaptive time discretization [9, 11]. For example, if it can be determined that there is no traffic at a certain time in a certain place then the grid size can be coarsened locally.

We use an extreme case of this approach by expanding the grid size maximally such that only a single time remains. We can also say that time is totally ignored. Everything happens simultaneously; there is no information about what happens earlier or later. With this aggregation we introduce a time-free model for the air-travel routing and scheduling problem.

Another way of looking at this transformation is projection: The time-free model is a projection of the time-indexed model onto a lower dimensional time-free subspace. Informally this is achieved by deleting the time index t from every variable and constraint of the model. In Table 5 we summarize the old discrete time and the new time-free variables.

Note that the y-variables representing the schedules of the planes have not only lost their time index t, but also either have a different domain or are replaced by a constant in the time-free model. The $y_{i,j}^{p,t}$ are binary decision variables. Any plane p can travel several times between airports i and j in a feasible solution. If we restrict the domain of their time-free counterparts $y_{i,j}^{p}$ to $\{0, 1\}$ then this model would have no feasible solution in case such multiple i-j-trips are necessary. To avoid this we enlarge the variables' domain to $\mathbb{Z}_+$, where the variable now represents the number

Table 5 Variables from the discrete time model and variables from the time-free model

Discrete time variable	Time-free variable
$y_{i,j}^{p,t} \in \{0, 1\}$	$y_{i,j}^{p} \in \mathbb{Z}_+$
$y_{\text{arr}}^{p,t} \in \{0, 1\}$	1 (constant)
$y_{\text{dep}}^{p,t} \in \{0, 1\}$	1 (constant)
$x_{i,j}^{r,p,t} \in \{0, 1\}$	$x_{i,j}^{r,p} \in \{0, 1\}$
$x_{\text{arr}}^{r,p,t} \in \{0, 1\}$	$x_{\text{arr}}^{r,p} \in \{0, 1\}$
$x_{\text{dep}}^{r,p,t} \in \{0, 1\}$	$x_{\text{dep}}^{r,p} \in \{0, 1\}$
$f_{i,j}^{p,t} \in \mathbb{R}_+$	$f_{i,j}^{p} \in \mathbb{R}_+$
$f_{\text{arr}}^{p,t} \in \mathbb{R}_+$	$f_{\text{arr}}^{p} \in \mathbb{R}_+$
$f_{\text{dep}}^{p,t} \in \mathbb{R}_+$	$f_{\text{dep}}^{p} \in \mathbb{R}_+$
$w_{i,j}^{p,t} \in \mathbb{R}_+$	$w_{i,j}^{p} \in \mathbb{R}_+$

of trips between i and j for plane p. The $y_{\text{arr}}^{p,t}$, $y_{\text{dep}}^{p,t}$ are binary decision variables. At first glance they are replaced by variables y_{arr}^{p}, y_{dep}^{p}. However, these variables are always equal to 1 because by constraints (1i) and (1j) it is guaranteed that each plane departs and arrives. Hence they can be replaced by the constant values 1.

The variables $x_{i,j}^{r,p,t}$ are binary. One could expect that the variables $x_{i,j}^{r,p}$ in the time-free model would be in $\mathbb{Z}_+$ for the same reason as for the $y_{i,j}^{p}$. However, we still restrict them to be binaries because a request is never transported on the same arc twice.

After projecting the variables we have to adjust the constraints accordingly. We reformulate the model by replacing the time-indexed variables by their time-free counterparts, and by discarding the time index quantifier. We refrain from writing down the entire model again, and give just one example.

Consider equations (1k). Their time-free counterpart is the family of equations:

$$\forall j \in \mathcal{V}, p \in \mathcal{P}:$$

$$\sum_{i:(i,j)\in\mathcal{A}} y_{i,j}^{p} + \begin{cases} 1, & \text{if } j = D_p, \\ 0, & \text{else,} \end{cases} = \sum_{k:(j,k)\in\mathcal{A}} y_{j,k}^{p} + \begin{cases} 1, & \text{if } j = A_p, \\ 0, & \text{else.} \end{cases} \tag{16}$$

Computational results for the time-free model are shown in Table 6. Because of the missing time index the model instances are much smaller. In all cases the root LP relaxation was solved within a fraction of a second. The gap between the LP relaxation as lower bound and the heuristic solution as upper bound is higher than the corresponding gap for the time-indexed model. All instances could be solved to proven integer optimality. For 17 instances this took less than one second, and fewer than 100 branch-and-bound nodes were necessary. On average, all 24 instances were solved in 6 seconds and needed 522 branch-and-bound nodes each. The average gap between the integer optimal solutions of the time-free model and the heuristic solution is 8.32 %. In two cases solving the time-free model proved global optimality of the heuristic primal solution for the original model with time.

Table 6 Computational results for the time-free model using the disaggregated weight formulation and all valid inequalities

Instance	Vars.	Cons.	Nzs.	Root time	Dual bd.	Gap	b&c time	Nodes	Dual bd.	Gap
BUF-AIV	2430	1104	12532	0	10820	14.22 %	0	82	12614	0.00 %
BUF-ANT	3820	1534	20150	0	12729	38.58 %	2	29	16317	21.26 %
BUF-BEE	4804	1648	25586	0	10621	39.76 %	17	970	15935	9.63 %
BUF-BOK	2762	1152	14332	0	10595	17.98 %	0	56	12917	0.00 %
BUF-EGL	5050	1674	26970	0	17274	21.88 %	4	470	19363	12.44 %
BUF-GNU	5296	1704	28328	0	12773	26.38 %	40	1467	16175	6.77 %
BUF-JKL	4312	1590	22868	0	13604	34.51 %	45	5115	18849	9.27 %
BUF-LEO	7734	2092	41926	0	18849	24.41 %	4	76	22425	10.08 %
BUF-NAS	2430	1104	12548	0	10927	31.41 %	0	0	15122	5.08 %
BUF-OWL	4066	1562	21510	0	12705	24.81 %	33	3585	16699	1.18 %
BUF-ZEB	2430	1104	12548	0	11807	25.86 %	3	303	14475	9.11 %
EGL-BEE	2430	1102	12532	0	13916	16.43 %	0	0	14842	10.87 %
EGL-GNU	2206	968	11310	0	14339	29.15 %	0	18	16338	19.27 %
EGL-LEO	2866	1078	14836	0	17129	11.65 %	0	9	18113	6.58 %
GNU-BEE	2074	948	10590	0	6407	43.36 %	2	149	9566	15.43 %
GNU-JKL	1810	902	9176	0	6915	37.69 %	1	13	9571	13.76 %
GNU-LEO	5542	1730	29686	0	13368	27.55 %	1	42	17513	5.08 %
LEO-AIV	1612	782	8090	0	12416	8.8 %	0	37	13327	2.11 %
LEO-ANT	1714	802	8626	0	14348	17.45 %	0	46	16391	5.70 %
LEO-BEE	4198	1452	22206	0	14599	22.76 %	1	35	16648	11.91 %
LEO-BOK	2338	990	12002	0	13323	13.33 %	0	14	14422	6.18 %
LEO-JKL	2470	1012	12706	0	13356	23.9 %	0	11	16914	3.63 %
LEO-NAS	1942	924	9898	0	14490	20.52 %	0	0	15929	12.62 %
LEO-OWL	2338	990	12002	0	13745	13.16 %	0	0	15540	1.82 %
Average	3278	1248	17207	0	12961	24.40 %	6	522	15667	8.32 %

We remark that the solution of the time-free model can be a non-simple path, i.e., it can have cycles. From such a solution it is not obvious how to obtain a corresponding solution in the time-space domain. For a given projected solution with nodes of out-degree greater than one there might exist a huge number of possible time-expanded simple path traversals. A simple example is shown in Fig. 1. Here the time-free solution allows for two different traversals.

The situation is even worse, since there exist projected feasible solutions not having a feasible non-projected counterpart. An example of this situation is given in Fig. 2. There is a request departing at A and arriving at C. We assume that at most two intermediate stops are allowed for this request. Then the depicted time-free solution is feasible, because it requires only one intermediate stop at B. However,

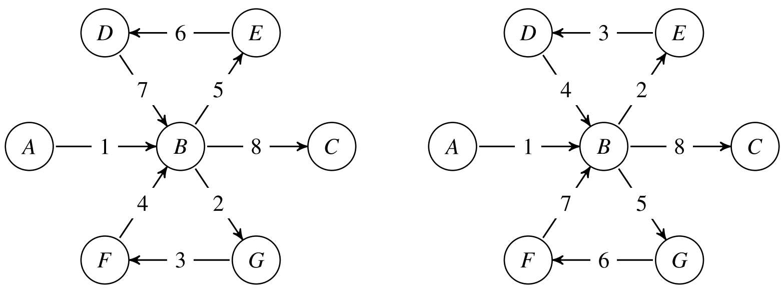


Fig. 1 A time-free solution with cycles having two different time-expanded traversals. *The arc label* indicates the ordering of the scheduled trips

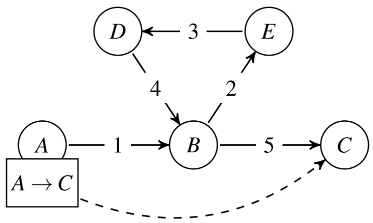


Fig. 2 One request from A to C with at most two intermediate stops. This is a feasible time-free solution that has no feasible time-expanded counterpart

there is only one time-expanded counterpart. After the plane arrives at B it has to fly the loop to E, D, and back to B, before it can arrive at C. Now the request would have four intermediate stops, hence it is not a feasible time-indexed solution. These examples illustrate that the time-free model indeed is only a relaxation of the time-indexed model. Getting feasible solutions for the time-indexed problem happens only rarely.

7 Embedding the Time-Free Model in Branch-and-Bound

In the previous section we showed that given a feasible time-free solution there could be no feasible time-indexed solutions or many. Our goal then is to find the least cost time-free solutions that has a corresponding time-indexed solution. We present a branch-and-bound algorithm that accomplishes this.

Given a time-free solution the inverse reconstruction problem is to decide if there exists a feasible time-indexed solution which, after projection, corresponds to the given time-free one. In order to answer this question for a given time-free solution we make use of the time-indexed model formulation. We set up the variables for the plane and request routes restricted to those variables that can be projected onto those variables that are non-zero in the time-free solution. Thus we make sure that the time-indexed solution (if it exists) is a pre-image of the time-free solution with respect to the projection function. As a positive side effect the instances are typically much smaller than the full problem where the trip set usually is the arc set of a complete graph on the airport nodes. We can solve the corresponding MILP problems within a few seconds. This either gives a proof of infeasibility or a feasible solution.

If the solution of the inverse problem for the integer optimal solution of the time-free model leads to a feasible solution with times then we are done, since the global optimal solution to the original problem with time was found. In general, unfortunately, this rarely happens, and the time-free solution turns out to be infeasible when time is included.

In order to proceed we embed this procedure within a branch-and-bound framework. The time-free reformulation is solved as the master-MILP. When a new incumbent has been found it is checked for time-feasibility by solving a sub-MILP problem. The solution status of this subproblem is returned to the master-MILP. If the subproblem is infeasible, the solution to the master-MILP is removed by branching, which works as follows. Assume that an incumbent solution z^* is given, where z is a vector of integer variables with bounds $\underline{z}$, $\overline{z}$. If for some index i the bounds are not tight, that is, $\underline{z}_i < z_i^*$ (or $z_i^* < \overline{z}_i$) then two branches are added to the branch-and-bound tree. In one branch we add the new bound constraints $z_i \leq z_i^* - 1$ and in the other we add $z_i \geq z_i^*$ (or $z_i \leq z_i^*$ and $z_i \geq z_i^* + 1$). Then we solve both child problems and this branching procedure is repeated unless all bounds are tight, that is, $\underline{z} = z^* = \overline{z}$. Then no further branchings are necessary and the node can be pruned. Note that in one of the two new child problems the same solution would be generated as in the parent node, since the bounds have not changed between parent and child.

This branching procedure is implemented in the solver CPLEX as an "incumbent callback". An incumbent callback is invoked if an incumbent solution is found at some node in the branch-and-bound process. Then it's up to the user to decide if the incumbent is feasible or not. In our case, we decide this by solving the time-indexed model as a subproblem until a feasible solution is found (optimality is not necessary) or a certificate of infeasibility is given.

Some implementation issues need to be resolved in order for this procedure to run fast. First, CPLEX invokes the callback again, even if the incumbent solution has already been rejected before. Checking feasibility by setting up and solving a sub-MILP problem is usually quite fast, but it's still undesired and time consuming to check solutions over and over again that already have been checked and rejected once. To avoid wasting time on chasing infeasible solutions, we implemented a look-up hash table where all previously found solutions are stored and quickly checked. Hence the sub-MILP is only computed for new solutions.

Second, the solver's own heuristics should be switched off. They only produce solutions for the time-free master-MILP, which are generally infeasible for the time-indexed problem. Moreover, the heuristics tend to find the same solutions over and over again, which need to be checked and stored. Heuristics are, on the other hand, a necessity in solving difficult MILPs fast. So one should have a good primal solution algorithm for the overall scheduling problem.

Our implementation takes these issues into account. The results are given in Table 7. The number of invokings of the incumbent callback is given in column two. The first check in our incumbent callback is whether the incumbent is new, or has it

Table 7 Computational results for the time-free model with branch-and-bound and incumbent callback

Instance	Inc. calls	New inc.	Inc. time	b&c nodes	b&c time	b&c dual bd.	Obj. val.	Gap
BUF-AIV	1	1	0	26	5	12614	12614	0.00 %
BUF-ANT	45181	6692	1588	77032	10800	16346	20724	21.12 %
BUF-BEE	63974	4739	1000	119151	10800	16251	17633	7.84 %
BUF-BOK	1	1	1	3	7	12917	12917	0.00 %
BUF-EGL	31320	4520	1192	54173	10800	19738	22113	10.74 %
BUF-GNU	72479	3472	484	139054	7577[a]	16763	17350	3.39 %
BUF-JKL	48329	6895	2127	85298	10800	19138	20774	7.87 %
BUF-LEO	19808	3437	660	32879	10800	22488	24938	9.82 %
BUF-NAS	971	62	13	1839	44	15549	15549	0.00 %
BUF-OWL	7410	339	125	19166	800	16797	16797	0.00 %
BUF-ZEB	2059	94	18	4093	142	14688	14688	0.00 %
EGL-BEE	51922	9514	1505	84959	10800	16447	16653	1.23 %
EGL-GNU	43409	13577	2557	59741	10800	16825	20238	16.86 %
EGL-LEO	51984	5904	1350	92819	8280	19388	19388	0.00 %
GNU-BEE	78156	9733	1849	142767	10800	10282	11311	9.09 %
GNU-JKL	70813	9334	1760	123016	10800	10244	11098	7.70 %
GNU-LEO	34611	3897	1699	61575	8928	17863	17863	0.00 %
LEO-AIV	146	10	4	273	12	13615	13615	0.00 %
LEO-ANT	529	57	18	980	63	16678	16678	0.00 %
LEO-BEE	35778	6122	2374	59502	10800	17306	18870	8.43 %
LEO-BOK	24765	3180	1694	43665	5664.07	15372	15372	0.00 %
LEO-JKL	5249	376	99	10243	440	17551	17551	0.00 %
LEO-NAS	1981	219	82	3528	183	18192	18192	0.00 %
LEO-OWL	81	5	1	153	6	15827	15827	0.00 %
Average	28790	3841	925	50664	5840	16203	17032	4.34 %

[a] 16 GB memory limit reached

been rejected before? The number of new incumbents is shown in the third column. On average only about 15 % of the incumbent calls stem from new incumbents. The incumbents are checked by a variant of the discrete-time MILP restricted to the subgraph of the incumbent. The time spent on these checkings is given in column four. On average it takes less than 1/3 second to solve these MILPs. The overall search procedure is a branch-and-bound approach, where the number of solved nodes and the CPU time is shown in the next two columns. We gave a time limit of 10800 seconds (3 hours). During this time limit 13 out of 24 instances could be solved to proven optimality, a substantial increase over the previous approaches. The overall integrality gap is 4.34 %, again a substantial improvement. It turned out that the heuristic's solutions could be improved only slightly on average (from 17160 down to 17032), but in some cases the improvements are quite significant (for example, the instances GNU-LEO was improved from 18450 to 17863).

8 Conclusions and Outlook

We introduced the time-free model with incumbent branching as a promising new method to solve scheduling and routing problems with time constraints. On our test set it gave better computational results compared to a direct solution of the MILP formulation using a time-space network approach and a set partitioning/column generation approach.

Much more can be done with the time-free approach. So far we provided only feasibility information to the time-free master model within the branch-and-bound process. If an instance has several planes, and in an incumbent all but one plane schedule is feasible, then the whole incumbent is declared infeasible. It might be more efficient to reschedule only those parts of the full schedule that are infeasible, which can be done by adding appropriate cutting planes.

Finally, we would like to apply the time-free approach to other applications that have time constraints including vehicle routing and job shop scheduling.

References

1. Ascheuer, N., Fischetti, M., Grötschel, M.: A polyhedral study of the asymmetric traveling salesman problem with time windows. Networks **36**(2), 69–79 (2000)
2. Ascheuer, N., Fischetti, M., Grötschel, M.: Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. Math. Program., Ser. A **90**(3), 475–506 (2001)
3. Barnhart, C., Shen, S.: Logistics service network design for time-critical delivery. Lect. Notes Comput. Sci. **3616**, 86–105 (2005)
4. Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., Vance, P.: Branch-and-price: column generation for solving huge integer programs. Oper. Res. **46**(3), 316–329 (1998)
5. Borndörfer, R., Grötschel, M., Pfetsch, M.: A column-generation approach to line planning in public transport. Transp. Sci. **41**(1), 123–132 (2007)
6. Cornuejols, G.: Valid inequalities for mixed integer linear programs. Math. Program., Ser. B **112**, 3–44 (2008)
7. Dantzig, G., Fulkerson, D., Johnson, S.: Solution of a large scale traveling salesman problem. Oper. Res. **2**(4), 393–410 (1954)
8. Erera, A., Hewitt, M., Savelsbergh, M., Zhang, Y.: Improved load plan design through integer programming based local search. Technical report 3357, Optimization Online (2012)
9. Espinoza, D., Garcia, R., Goycoolea, M., Nemhauser, G., Savelsbergh, M.: Per-seat, on-demand air transportation part I: problem description and an integer multicommodity flow model. Transp. Sci. **42**(3), 263–278 (2008)
10. Espinoza, D., Garcia, R., Goycoolea, M., Nemhauser, G., Savelsbergh, M.: Per-seat, on-demand air transportation part II: local search. Transp. Sci. **42**(3), 279–291 (2008)
11. Fischer, F., Helmberg, C.: Dynamic graph generation for large scale operational train timetabling. Technical report 2011-10, Fakultät für Mathematik, Technische Universität Chemnitz (2011)

12. Ford, L., Fulkerson, D.: Constructing maximal dynamic flows from static flows. Oper. Res. **6**(4), 419–433 (1958)
13. Fügenschuh, A., Homfeld, H., Schülldorf, H.: Single car routing in rail freight transport. In: Barnhart, C., Clausen, U., Lauther, U., Möhring, R. (eds.) Dagstuhl Seminar Proceedings 09261 (2009)
14. Gavish, B., Graves, S.: The traveling salesman problem and related problems. Technical report OR 078-78, Operations Research Center, Massachusetts Institute of Technology (1978)
15. Grötschel, M.: Der Satz von Frobenius für konvergente Potenzreihen. Master's thesis, Ruhr-Universität Bochum (1973)
16. Grötschel, M.: Polyedrische Charakterisierungen Kombinatorischer Optimierungsprobleme. Mathematical Systems in Economics, vol. 36. Verlag Anton Hain, Meisenheim am Glan (1977)
17. Grötschel, M.: On the symmetric travelling salesman problem: solution of a 120-city problem. Math. Program. Stud. **12**, 61–77 (1980)
18. Grötschel, M., Holland, O.: Solution of large-scale symmetric travelling salesman problems. Math. Program., Ser. A **51**(2), 141–202 (1991)
19. Helmberg, C., Röhl, S.: A case study of joint online truck scheduling and inventory management for multiple warehouses. Oper. Res. **55**(4), 733–752 (2007)
20. Kennington, J., Nicholson, C.: The uncapacitated time-space fixed-charge network flow problem: an empirical investigation of procedures for arc capacity assignment. ORSA J. Comput. **22**(2), 326–337 (2010)
21. Kotnyek, B.: An annotated overview of dynamic network flows. Technical report 4936, Institut National de Recherche en Informatique et en Automatique (INRIA) (2003)
22. Maffioli, F., Sciomachen, A.: A mixed-integer model for solving ordering problems with side constraints. Ann. Oper. Res. **69**, 277–297 (1997)
23. Marchand, H., Wolsey, L.: Aggregation and mixed integer rounding to solve MIPs. Oper. Res. **49**(3), 363–371 (2001)
24. Marchand, H., Martin, A., Weismantel, R., Wolsey, L.: Cutting planes in integer and mixed integer programming. Discrete Appl. Math. **123**, 397–446 (2002)
25. Nemhauser, G.: Column generation for linear and integer programming. In: Grötschel, M. (ed.) Optimization Stories, pp. 65–73. Deutschen Mathematiker Vereinigung, Bielefeld (2012)
26. Nemhauser, G., Wolsey, L.: Integer and Combinatorial Optimization. Wiley-Interscience, New York (1988)
27. UN World Tourism Organization: Tourism facts & figures (2012). www.unwto.org/facts/. Accessed 18 Sep 2012
28. van Eijl, C.: A polyhedral approach to the delivery man problem. Technical report 95-19, Department of Mathematics and Computer Science, Eindhoven University of Technology (1995). alexandria.tue.nl/repository/books/440440.pdf. Accessed 31 Jan 2013
29. Wieberneit, N.: Service network design for freight transportation: a review. OR Spektrum **30**, 77–112 (2008)

Mixed Integer Programming: Analyzing 12 Years of Progress

Tobias Achterberg and Roland Wunderling

Abstract Back in 2001, Bixby et al. (The Sharpest Cut: The Impact of Manfred Padberg and His Work, pp. 309–325, 2004) provided an analysis of the performance impact of the main mixed integer programming features and improvements up to CPLEX 8.0 for a workshop in honor of Manfred Padberg's 60th birthday, which was later published in a Festschrift edited by Martin Grötschel (The Sharpest Cut: The Impact of Manfred Padberg and His Work, 2004). Now, 12 years later, Grötschel's own 65th birthday celebration seems to be the ideal opportunity to provide an update on the state of affairs.

In this paper, we outline an unbiased way to analyze benchmark results and apply this scheme to assess the contribution of the main components in CPLEX 12.5 to the ability to solve MIPs. We highlight some of the more recent features, in particular the deterministic parallel optimizer.

1 Introduction

Mixed Integer Programming can look back to a grand past and forward to a promising future, see Bixby [14]. Indeed, Fig. 1 shows the progress of CPLEX versions from 6.0 to 12.5, which spans the years 1998 to 2012. The data has been computed using the geometric mean of solution times for the subset of models from our internal model library that can be solved by at least one of the versions and for which at least one of the versions takes more than 10 seconds, i.e., the [10,10k] model bracket described in detail in Sect. 2. The left scale applies to the bars in the chart, showing the number of problem instances that could not be solved within the time limit of 10000 seconds. This number decreased from 1152 to 55. The right scale

T. Achterberg (✉)
CPLEX Optimization, IBM, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, 14195 Berlin, Germany
e-mail: achterberg@de.ibm.com

R. Wunderling
CPLEX Optimization, IBM, Liebenauer Hauptstr. 2–6, 8041 Graz, Austria
e-mail: roland.wunderling@at.ibm.com

M. Jünger, G. Reinelt (eds.), *Facets of Combinatorial Optimization*, 449
DOI 10.1007/978-3-642-38189-8_18, © Springer-Verlag Berlin Heidelberg 2013

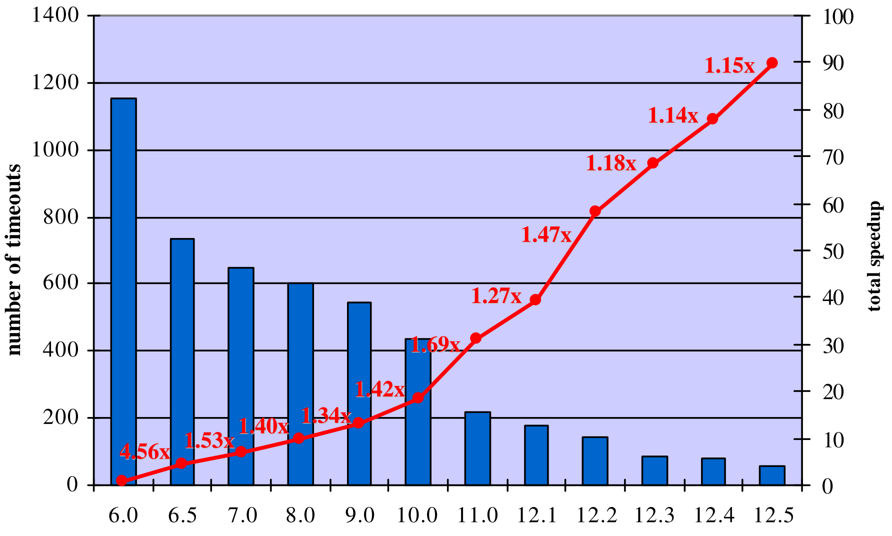


Fig. 1 Comparison of CPLEX versions 6.0 to 12.5 on 1753 problem instances

corresponds to the total speedup since version 6.0, indicated by the piecewise-linear curve. This shows a cumulative improvement by a factor of 89.6. The individual speedup factors listed in the chart reflect the version-to-version performance improvement on this test set. Of course, the computations have been conducted on identical machines, so that hardware speedup is not included in the numbers.

Where do these improvements come from? This is not a novel question, and answers have been provided before by Bixby et al. [15–17], using CPLEX 6.5 and CPLEX 8.0, respectively. Since MIP solvers consist of an arsenal of features, they measured the impact of each feature by comparing the performance of the default solver against turning off each feature individually. The observed performance degradation provides a reasonable measure for the importance of each feature.

More than 10 years have passed since CPLEX 8.0. The implementation of the existing ingredients has been improved and new components have been added. It is therefore interesting to analyze the performance of the current version CPLEX 12.5 using the same approach.

To do so, we first describe our benchmarking methodology. In particular, we identify a systematic bias that is often introduced when doing performance analysis and discuss our approach of avoiding it. This methodology is then applied to quantify the impact of the main algorithmic features of CPLEX 12.5. In particular, we wanted to investigate how their relative importance changed compared to CPLEX 8.0, and if so, why. Indeed, we found consistency as well as discrepancy in the results. While the measured relative importance of presolve and heuristics have not changed, we found large discrepancies in the evaluation of cuts. Interestingly, to a large degree this can be attributed to biases in how benchmarks were conducted in [17].

Finally, we analyze the impact of another development that occurred independently of MIP during the past decade: multithreaded architectures have turned mainstream. To continue to take advantage of the ongoing hardware evolution, MIP solvers had to adopt and did. Current state-of-the-art MIP solvers are parallel algorithms. Moreover, they are deterministic in that running the same solver with the same data on the same hardware is guaranteed to provide the same solution path and result. We will describe how determinism is achieved in CPLEX 12.5 and measure its performance impact.

Both authors of this paper were Ph.D. students of Martin Grötschel and owe him much regarding their scientific and commercial careers:

Tobias Achterberg: I started taking Martin Grötschel's classes on optimization because my father told me "Grötschel is the one", and indeed, my father was right. Soon after, I was hired at the Zuse Institute Berlin (ZIB) in Martin Grötschel's optimization group, where I prepared my master's theses and my Ph.D. thesis. But Martin Grötschel had a significant impact also on my commercial career: I do not know for sure, but I guess he was pulling some strings so that I was squeezed into the program of the MIP workshop 2005 at the IMA and could give a presentation [1]. Ed Rothberg and Zonghao Gu were in the audience and approached me after my talk; one year later I joined CPLEX.

Roland Wunderling: Martin Grötschel and I joined ZIB at roughly the same time; he as vice president and I as a physics master student working on computer graphics with no knowledge of mathematical programming. Also I had no idea who Martin Grötschel was, or I might have heeded his words of caution, when I approached him with the proposal to work on the simplex algorithm: "That's a tough topic now that CPLEX is around". Luckily I was naive enough to ignore him and set out working on SoPlex [59] while eagerly catching up on my learning by visiting all his fantastic courses. Eventually, I was ready to show Bob Bixby (was it just a coincidence that Martin Grötschel invited him at that time?) how SoPlex could beat CPLEX, and soon after I joined CPLEX.

2 Benchmarking

When conducting a quantitative computational performance analysis, the first step is to define a benchmarking methodology. Especially for MIP this is a challenging task that is frequently not solved adequately. What makes MIP performance analysis especially challenging is the great variability in computation time for solving an individual problem instance, see Koch et al. [36]. Seemingly performance-neutral changes in the benchmarking setup, such as changing the permutation of variables or constraints in the model or the starting random seed used by the algorithm, can turn a model from easily solvable to intractable or vice versa.

Of central importance is the selection of the test set and the number of problems therein. In particular, the size of the test set is important in order to average out the effects of performance variability, where geometric means should be used instead

Table 1 Comparison of CPLEX 12.5 and CPLEX 8.0

Bracket	Models	CPLEX 12.5	CPLEX 8.0					Affected		
		Tilim	Tilim	Faster	Slower	Time	Nodes	Models	Time	Nodes
All	2928	87	732	292	1777	4.71	8.04	2741	5.22	9.46
[0,10k]	2843	2	647	292	1777	4.93	8.73	2741	5.22	9.46
[1,10k]	1919	2	647	247	1585	10.30	19.79	1915	10.34	19.92
[10,10k]	1406	2	647	113	1265	21.91	37.29	1404	21.98	37.48
[100,10k]	1043	2	647	46	987	44.55	65.76	1043	44.55	65.76
[1k,10k]	783	2	647	14	767	78.60	106.35	783	78.60	106.35

of arithmetic means to limit the effect of an individual large number. The smaller the performance difference that should be analyzed, the larger a test set is needed.

The makeup of the test set can also affect the outcome of the benchmarks. It should provide a good representation of the type of models that the solver will be used for. This precludes the use of artificially generated models. Also it mandates to avoid over-representing one type of model in the set, e.g., by using multiple instances coming from the same application.

For pragmatic reasons, such as available computing resources and time, a large test set may need to be reduced. In this step it is important not to use a criterion based on data generated by a single solver, but to treat all solvers under consideration equally. Deviation from this rule can easily introduce a systematic bias of the results as we will describe in Sect. 2.2 in the context of partitioning the test set into subsets.

The base problem set used in this paper consists of problem instances coming from a mix of publicly available and commercial sources. We excluded all problem instances from our model collection that we have never been able to solve to optimality with any version between CPLEX 6.0 and CPLEX 12.5. This left a total of 3189 instances that we used for all MIP performance runs, 138 of which are infeasible. To keep the computation time under control, a time limit of 10000 seconds was used for all the tests. Additionally, we employed a tree memory limit of 6 GB. If this tree memory limit was hit, we treated the model as if a time limit was hit. We set the solve time to 10000 seconds and scaled the number of nodes processed for the problem instance accordingly.

All runs where conducted on a cluster of identical 12 core Intel Xeon CPU E5430 machines running at 2.66 GHz and being equipped with 24 GB of memory.

Table 1 compares the performance of CPLEX version 12.5 against version 8.0, the last version of CPLEX for which a performance analysis has been published, see Bixby et al. [17]. Note that CPLEX 12.5 runs in deterministic parallel mode, while CPLEX 8.0 uses opportunistic parallel mode (as it does not have a deterministic mode). Each row of the table lists comparative data for subsets of the test set. The table is organized as follows:

Column 1, “Bracket”, labels subsets of problem instances. The idea is to subdivide the test set into subsets of problem instances with different “hardness”, each row representing a different such subset. Subset “All” is the set of all models used for the first row of data. It excludes only models for which one of the solvers encountered a failure of some sort or where numerical difficulties lead to different optimal objective values for the two solvers (which could actually, due to the definition of feasibility tolerances, both be correct). The labels "[n,10k]" represent the subset of "All" models for which at least one of the solvers being compared took at least n seconds to solve and that were solved to optimality within the time limit by at least one of the solvers.

Column 2, "Models", shows the number of problem instances in each subset. Note that only 2928 rather than 3189 problem instances are listed in row "all" due to the exclusion rules explained above.

Column 3, "Tilim", gives the number of models in each subset for which a time (or memory) limit was hit by CPLEX 12.5. It is by design that the numbers in the last 5 rows match, since these models are always included in the corresponding subsets.

Similarly, column 4, gives the number of models in each subset for which CPLEX 8.0 hit a time limit. Comparing these numbers to the corresponding numbers in column 3 gives a good indication about how many models have been turned from unsolvable with CPLEX 8.0 to solvable with 12.5. Both versions hit the time limit on 85 problem instances, but that number could be arbitrarily increased by adding more intractable models to the base set. However, only two models can be solved within the time limit by CPLEX 8.0 and not by CPLEX 12.5, whereas CPLEX 12.5 solves 647 models that could not be solved with CPLEX 8.0.

Columns 5, "Faster", and 6, "Slower", show the number of problem instances in each subset that CPLEX 8.0 solved faster and slower than CPLEX 12.5, respectively. For this decision the CPLEX 8.0 solution time of an instance must be 10 % smaller or larger than the CPLEX 12.5 time, respectively. While CPLEX 12.5 outperforms version 8.0 most of the time, there remains a significant subset of models for which the opposite is true. As it turns out, this can in most part be attributed to performance variability. On certain models, however, some of the algorithms that have been added to the software do not help and only produce computational overhead. This is most likely the case for the models that could already be solved easily without such computations, which explains the fact that in the harder problem sets CPLEX 8.0 wins much less frequently.

Column 7, "Time" displays the shifted geometric mean of the ratios of solution times, see Achterberg [2], with a shift of $s = 1$ second. A value $t > 1$ in the table indicates that CPLEX 8.0 is by a factor of t slower (in shifted geometric mean) than CPLEX 12.5. Indeed, remarkable improvements have been achieved between the versions, and in this paper we will try to analyze where these come from. Note that time limit hits are accounted for with a value of 10000 seconds, which introduces a bias against the solver with fewer timeouts. In fact, using a time limit of 1000 seconds, we would only compute a speedup factor of 10.29 instead of 21.91 for the models that take at least 10 seconds to solve. Thus, the performance ratios need to be considered in conjunction with the number of timeouts of each solver.

Column 8, "Nodes", is similar to the previous column but shows the shifted geometric mean of the ratios of the number of branch-and-cut nodes needed for the problems by each solver, using a shift of $s = 10$ nodes. When a time limit is hit, we

Table 2 Comparison of CPLEX 12.5 using two different random seeds

Bracket	Models	Seed 1	Seed 2					Affected		
		Tilim	Tilim	Faster	Slower	Time	Nodes	Models	Time	Nodes
All	3159	94	112	588	588	1.00	0.99	2284	1.00	0.99
[0,10k]	3082	17	35	588	588	1.00	0.99	2284	1.00	0.99
[1,10k]	1879	17	35	565	558	1.00	0.98	1707	1.00	0.98
[10,10k]	1121	17	35	404	396	1.01	0.98	1067	1.01	0.98
[100,10k]	604	17	35	243	237	1.01	1.01	593	1.01	1.01
[1k,10k]	238	17	35	101	100	1.08	1.07	237	1.08	1.07

use the number of nodes at that point, which again introduces a bias. The numbers exceed those for the runtime by a factor of 1.5 to 2, indicating that reductions in search space came at a cost of additional computation.

The last three columns, under the heading "Affected", repeat some of the information for the subset of models in each bracket, where the changes between both versions had an effect on the solution path. Here, we assume that the solution path is identical if both the number of nodes and the number of simplex iterations are identical for the two solvers. Column 9, "Models" shows that almost all problems in the ≥ 1 second brackets are included, leaving essentially only models unaffected that could be solved by presolve alone already with CPLEX 8.0.

When you want to condense performance impact into one number such as in Fig. 1, you have to choose which model bracket to use. Throughout this paper we selected the [10,10k] bracket, since we wanted to eliminate models that are "easy" even without a specific feature and thus don't help assessing its impact. Also with a lower cutoff the results would be more dampened by the shift we use when computing geometric means. On the other hand we did not want to inflate the measured effect by using a higher cutoff value.

2.1 The Effect of Performance Variability

Performance variability in MIP is indeed an issue that needs to be considered in the analysis. One way to assess performance variability is to measure the performance impact of changing the initial random seed. This affects in particular the choice of the optimal basis of the initial LP relaxation and many decisions in primal heuristics. Clearly, this is conceptually a performance neutral change. However, Table 2 shows that some performance differences can be observed.

Over the entire set of 3159 models, there seems to be a 1 % difference in the node count. Note that in the benchmarking runs that we regularly conduct during the CPLEX development process we often see a 1 % difference in time as well. In other words, performance differences in CPLEX of about 1 % cannot be distinguished using our test set of 3000 problem instances. As was to be expected, the issue gets worse as the sample size decreases. Most notably, the last subset contains only 238 instances and shows what appears to be a substantial performance difference of 8 %. On the one hand, this means that problem sets of this small size are probably inadequate for reliably measuring performance differences of less than 10 %. On the other hand, this is also an artifact of our bracket definition, since a significant fraction of the models in the [1k,10k] bracket feature large variability so that one solver can solve them easily by luck, while the other solver hits the time limit. For this reason, we will mostly ignore the [1k,10k] bracket in the discussion of the results in this paper.

2.2 *Avoiding a Systematic Bias*

It is important to note that our definition of the subsets of the problem instances depends on the solvers being compared in each table, which implies that, e.g., bracket [10,10k] contains a slightly different set of models in each table displayed in this paper. This may seem confusing, but is of central importance to avoid a systematic bias in the analysis of the data. Given the large performance variability for MIP, using only data from, say, the reference solver CPLEX 12.5 to define the subsets would have produced results that can easily be misinterpreted. We will explain this with the following Gedanken-experiment:

Given a set of problem instances P to be used to compare a new solver B against a reference solver A, we classify the test set $P = E \cup H$ into easy models E and hard ones H, using the solution times from the reference solver A. Assume that the solution times of solver A for the test set P are uniformly distributed between 0 and 100 seconds such that the solution times for models in E are in $(0\text{ s}, 80\text{ s}]$ while for H they are in $(80\text{ s}, 100\text{ s}]$. Using arithmetic means for simplicity, the average solution time of solver A for set E is 40 seconds and 90 seconds for the hard set H. To model performance variability, we assume that the timing function just returns a uniformly distributed random number between 0 and 100 seconds for both solvers. Thus the expected value for the solution time of solver B is 50 seconds for both sets E and H. It is a common mistake to conclude that solver B outperforms solver A on the set of harder models by a factor of $1.8 = \frac{90}{50}$ at the cost of a slight degradation on the easier models by a factor of $0.8 = \frac{40}{50}$. The error is in the setup that uses only the reference solver for defining the subsets.

This is indeed an important issue for analyzing performance for MIP, as can be seen in Table 3. It shows the results of the same two experiments from Table 2, yet this time with the error of using only the data from seed 1 to define the subsets of models. The largest two subsets remain unchanged. The remaining rows, however, show a completely different situation from before. In fact, disregarding the bias, one would be tempted to claim seed 2 to be superior to seed 1 by up to 25 % "on harder models", even though we saw in the unbiased analysis that the opposite is actually the case. The biased subset selection puts problem instances for which the reference solver was unlucky into the "harder" subsets, giving the other solver a good chance

Table 3 Comparison of CPLEX 12.5 using two different random seeds (biased model grouping)

Bracket	Models	Seed 1	Seed 2					Affected		
		Tilim	Tilim	Faster	Slower	Time	Nodes	Models	Time	Nodes
All	3159	94	112	588	588	1.00	0.99	2284	1.00	0.99
[0,10k]	3082	17	35	588	588	1.00	0.99	2284	1.00	0.99
[1,10k]	1848	17	34	565	530	0.99	0.96	1677	0.99	0.95
[10,10k]	1074	17	33	404	351	0.95	0.90	1021	0.94	0.89
[100,10k]	552	17	31	243	188	0.87	0.85	541	0.87	0.85
[1k,10k]	207	17	27	101	70	0.76	0.71	206	0.76	0.71

of doing better on those. Thus, marketing messages about big speedups on "harder" problems should be taken with much care. The CPLEX marketing made exactly this mistake until version 11.

In our analysis, we therefore define all test sets and subsets by applying the filtering rules described above, using all runs to be compared. A consequence of this approach is that the test sets are not identical between comparisons. An extreme outcome from this can be observed by comparing the speedup of 9.17× between CPLEX version 8.0 and 12.5 in Fig. 1 to the value of 21.91× for the [10,10k] bracket in Table 1. What happened? Our [10,10k] filtering rule removes problems for which none of the versions being considered took less than 10 seconds. Since the versions prior to CPLEX 8.0 could not solve some of the models within this limit, the test set used in Fig. 1 contains several additional models that where excluded in Table 1, where they would have dampened the measured difference. Removing models that have become "easy" is important to be able to measure the impact of developments for the problems that matter. However, one needs to keep in mind that speedup factors between versions cannot be multiplied unless they are based on the same test set.

3 MIP Evolution

In Sect. 2 we laid out how to conduct and analyze benchmark runs in a sound and unbiased way. Now, we apply this methodology to assess the performance impact of the main ingredients of mixed integer programming solvers and some of the recent improvements to the state-of-the-art.

Most of current MIP solvers are based on the branch-and-bound method, which first appeared in Markowitz and Manne [43], Eastman [24] and Land and Doig [38] in the context of integer programming. See Cook [19] for an excellent review on the history of branch-and-bound.

An additional fundamental building block of modern MIP solvers are cutting planes, originating in the works of Dantzig, Fulkerson and Johnson [22, 23] and Gomory [32]. But even though the combination of branch-and-bound and cutting planes was already proposed by Markowitz and Manne [43] in 1957, it took more than 20 years until Crowder, Johnson and Padberg [20], Grötschel, Jünger and Reinelt [34], van Roy and Wolsey [58] and Padberg and Rinaldi [49, 50] came up with the first successful implementations of branch-and-cut to solve practical problems. See Bixby et al. [17] and Cook [19] for more details.

In addition to branch-and-cut, MIP solvers also include preprocessing techniques to reduce the size and to tighten the formulation of the problem instance at hand, see, e.g., Savelsbergh [55]. Finally, primal heuristics are employed to find feasible solutions without having to wait for them to appear as solutions to the LP relaxations of the branch-and-cut subproblems; see, e.g., Berthold [13] for an overview of MIP heuristics in SCIP [3, 56] and Raidl and Puchinger [53] and Fischetti and Lodi [27] for surveys on MIP heuristics and meta-heuristics. Heuristics are very important for practitioners, who are typically more interested in getting solutions of good quality as quickly as possible than in finding a provably optimal solution. We will see in Sect. 3.4 that heuristics are important but not as critical as clever branching, cutting and presolving for solving a MIP, i.e., for finding an optimal solution and proving its optimality.

3.1 Branching

Branching is at the core of the exponential worst-case complexity of LP based branch-and-cut algorithms. The choice how to split a given problem into two or more subproblems can have a very significant impact on the size of the resulting search tree and can thus be crucial for being able to solve a given problem instance in reasonable time.

Bixby et al. [17] report a speedup of 2.9× due to the improvements in the branching variable selection rule from CPLEX 5.0 to CPLEX 8.0. Depending on the relative number of integer variables in the problem formulation, CPLEX 5.0 was employing either *most infeasible branching* or *pseudo-reduced cost branching*. To our knowledge, the latter has never been published, but it is very similar to the active constraint based branching rules of Patel and Chinneck [51], using the magnitude of the coefficients and the dual solution value of the rows as weights. CPLEX 8.0 implements the *pseudo cost with strong branching initialization* rule of Linderoth and Savelsbergh [40].

Since the release of CPLEX 8.0 in 2002, a number of papers on improving the branching in MIP solvers have been published. Despite the interest in branching on hyperplanes, see for example Owen and Mehrotra [48], Mahajan and Ralphs [41], Karamanov and Cornuéjols [35], the current version 12.5 of CPLEX is still mostly branching on variables. Nevertheless, the selection of the branching variable has been improved. It is now a version of *hybrid branching*, see Achterberg and Berthold [5]. It uses *reliability branching* [7] as base line, adjusted by conflict scores [5] and using *pseudo-reduced cost branching* [51] and *inference branching* [2] as a tie-breaker. Finally, the strong branching initialization of pseudo costs has been

Table 4 Comparison of default (hybrid) branching and standard pseudo cost/strong branching

Bracket	Models	Default	Pseudo-cost branching with SB init					Affected		
		Tilim	Tilim	Faster	Slower	Time	Nodes	Models	Time	Nodes
All	3155	95	129	467	831	1.11	1.02	2046	1.17	1.04
[0,10k]	3083	23	57	467	831	1.11	1.03	2046	1.17	1.04
[1,10k]	1908	23	57	449	784	1.18	1.10	1621	1.22	1.11
[10,10k]	1160	23	57	314	555	1.29	1.26	1057	1.32	1.29
[100,10k]	633	23	57	193	333	1.42	1.44	608	1.44	1.46
[1k,10k]	283	23	57	89	163	1.84	1.83	280	1.85	1.84

Table 5 Impact of individual branching improvements in CPLEX 12.5 on the [10,10k] bracket

Feature	Models	Default	Modified code					Affected		
		Tilim	Tilim	Faster	Slower	Time	Nodes	Models	Time	Nodes
No reliability branching	1106	18	34	202	278	1.05	1.10	706	1.08	1.16
No pseudo-reduced costs	1142	20	32	371	472	1.06	1.00	1034	1.07	1.00
No non-chimerical branching	1089	12	24	181	208	1.03	1.03	658	1.04	1.04
Most infeasible branching	1309	9	475	121	996	8.36	12.61	1194	10.25	16.10

improved by incorporating ideas from *non-chimerical branching* of Fischetti and Monaci [28].

The performance improvements due to the additional refinements in the variable selection strategy can be seen in Table 4. For models in the [10,10k] bracket, using the plain *pseudo-cost branching with strong branching initialization* rule is 29 % slower than the more sophisticated variable selection of CPLEX 12.5. The contributions of the individual refinements are listed in the first three rows of Table 5, again using the [10,10k] bracket of the test set. Each of them yields a small performance improvement.

The results show that some progress has been achieved in branching variable selection since CPLEX 8.0, but certainly no break-through. The *pseudo cost with strong branching initialization* is still competitive with today's branching rules. Nevertheless, employing a clever branching rule is critical for LP based branch-and-bound MIP solvers. As can be seen in the last row of Table 5, using a naive branching rule like *most infeasible branching*, the performance degrades by a factor of more than 8 in the [10,10k] bracket, and many models become unsolvable within the time limit.

Table 6 Comparison of default and not using cutting plane separation

Bracket	Models	Default	No cuts					Affected		
		Tilim	Tilim	Faster	Slower	Time	Nodes	Models	Time	Nodes
All	3148	94	464	586	1229	2.22	7.16	2465	2.77	12.06
[0,10k]	3063	9	379	586	1229	2.27	7.42	2465	2.77	12.06
[1,10k]	1981	9	379	521	1155	3.54	14.86	1837	3.91	18.37
[10,10k]	1335	9	379	314	889	6.11	27.08	1289	6.51	30.46
[100,10k]	883	9	379	167	666	12.47	63.45	870	12.95	67.51
[1k,10k]	577	9	379	68	493	32.34	213.93	575	32.74	217.96

3.2 Cutting Planes

Cutting planes were introduced to CPLEX with version 3.0 in 1994, featuring clique cuts and knapsack cover cuts. Version 6.5.3, released in 1999, introduced flow covers, GUB covers and implied bound cuts. It also included a hidden disabled version of Gomory mixed-integer cuts, which were then made publicly available with version 6.6 in 2000. Version 7.0 was released in December 2000 and added disjunctive, mixed integer rounding and flow path cuts, which concluded the "systematic program … to include as many of these [mostly cutting plane] ideas as possible" (Bixby [14]).

The next addition to the cut arsenal happened in 2007, when zero-half cuts had been introduced in CPLEX 11.0 following the ideas of Koster, Zymolka and Kutschka [37]. Finally, CPLEX 12.1 included the multi-commodity network flow cuts of Achterberg and Raack [6] and incorporated cutting plane filtering as described by Achterberg [2]. The latter allowed the more aggressive use of existing cutting plane separators without cluttering up the LP relaxation too much.

Cutting planes are of major importance to solve mixed integer programs. Table 6 shows the performance degradation of CPLEX 12.5 when cutting planes are disabled. In the [10,10k] bracket of the test set, cutting planes yield a 6.1$\times$ speedup. But more importantly, 379 models cannot be solved without cutting planes within the time limit of 10000 seconds, while there are only 9 models for which CPLEX 12.5 hits the time limit but disabling cuts allows to solve them. This clearly shows that there is a substantial fraction of MIP models for which cutting planes are an indispensable solver ingredient.

Table 7 details the contribution of the individual cutting plane separators, by comparing defaults against turning off each cutting plane separator individually. It is interesting to note that mixed integer rounding (MIR) cuts are clearly the most useful cuts in CPLEX 12.5. This is in contrast to the results of Bixby et al. [17] for CPLEX 8.0, where Gomory mixed integer cuts were identified to contribute a speedup of 2.52$\times$ and MIR cuts only a speedup of 1.83$\times$. Most likely, this discrepancy has the following two sources. First, Bixby et al. use a different test set, namely one that consists of only 106 problem instances and that is clearly biased toward models for which cutting planes are critical, see also our discussion in Sect. 5.

Table 7 Impact of individual cutting plane separators in CPLEX 12.5 on the [10,10k] bracket

Feature	Models	Default	Modified code					Affected		
		Tilim	Tilim	Faster	Slower	Time	Nodes	Models	Time	Nodes
No MIR cuts	1118	9	78	212	341	1.48	1.83	693	1.88	2.64
No Gomory cuts	1130	20	62	358	417	1.28	1.50	956	1.34	1.61
No knapsack cover cuts	1108	19	47	279	325	1.14	1.27	903	1.18	1.35
No zero-half cuts	1116	17	35	274	276	1.08	1.19	720	1.12	1.31
No implied bound cuts	1098	11	25	236	288	1.07	1.10	785	1.10	1.15
No flow cover cuts	1100	14	23	214	266	1.06	1.06	626	1.10	1.11
No clique cuts	1104	16	21	184	193	1.04	1.11	490	1.08	1.27
No flow path cuts	1077	1	6	35	44	1.04	1.06	96	1.52	1.85
No MCF cuts	1079	1	6	33	60	1.03	1.03	124	1.25	1.29
No GUB cover cuts	1083	3	10	77	89	1.02	1.05	224	1.12	1.25
Cut-and-branch	1125	22	71	278	403	1.23	1.59	936	1.28	1.75

Second, in CPLEX 8.0 MIR cuts were only generated for constraints (or aggregations of constraints) that contain general integer variables. For binary or mixed binary constraints, only flow cover cuts had been separated. This strategy has been changed with the revision of the cut separation framework in CPLEX 12.1.

A second observation from Table 7 is that only MIR and Gomory cuts, and to a lesser degree knapsack cover cuts, are really essential for the ability to solve MIPs. Only these provide a significant difference in the number of time limit hits. The other cut separation procedures are also useful to reduce the solving time, but it seems that they do not make a big difference when it comes to the question whether a problem instance can be solved at all.

Even though most of the cutting plane procedures affect a large number of the problem instances, some of them turn out to be very specialized to certain classes of models. The flow path cuts, MCF cuts and GUB cover cuts are only used on a small fraction of the problem instances. On those subsets they do provide a significant improvement, most notably the 1.5$\times$ speedup of flow path cuts for the small set of 96 models on which they apply.

The last row of Table 7 shows the performance impact when CPLEX is run as a cut-and-branch solver, i.e., when cuts are only separated at the root node but not in local nodes of the search tree. As can be seen, there is a degradation of 23 % when node cuts are disabled. Even though there is also a moderate difference in the number of time limit hits, we conclude that root node cuts are more important.

Table 8 Comparison of default and not applying presolve

Bracket	Models	Default	No presolving					Affected		
		Tilim	Tilim	Faster	Slower	Time	Nodes	Models	Time	Nodes
All	3087	94	556	323	1841	3.50	3.97	2915	3.77	4.38
[0,10k]	3002	9	471	323	1841	3.63	4.20	2915	3.77	4.38
[1,10k]	2090	9	471	273	1703	6.27	7.18	2072	6.37	7.31
[10,10k]	1484	9	471	146	1295	11.40	13.19	1479	11.50	13.31
[100,10k]	1022	9	471	78	932	21.95	25.87	1019	22.17	26.12
[1k,10k]	687	9	471	33	648	43.28	49.00	687	43.28	49.00

3.3 Presolving

Presolving, or preprocessing, means to transform a given problem instance P into a different but equivalent problem instance P' that is hopefully easier to solve by the subsequently invoked solution algorithm.

In CPLEX, presolving capabilities were added with version 2.1 in 1993. It started with simple reductions like removing fixed variables and redundant constraints, and presolve was mainly viewed as a convenience feature to relieve the users from having to write their models in a most efficient way. The importance of presolve increased significantly with the introduction of matrix generators and modeling languages, see, e.g., Fourer [30], because a general algebraic model often produces fixed variables and redundant constraints if applied to a specific data set. Since its introduction, presolve has become more and more complex over the years and evolved into a very sophisticated and highly important solver component. Nowadays, it not only removes the garbage from the model formulation, but it also tightens the LP relaxation and extracts information that is exploited later during the solving process. See, e.g., Savelsbergh [55] and Achterberg [2] for an overview of the most common presolving techniques.

As can be seen in Table 8, presolving has a massive impact on CPLEX' ability to solve MIPs. When turning presolve off, 471 out of 3002 problem instances become unsolvable within the time limit, while only 9 can be solved for which default CPLEX fails. Except for the most trivial cases, presolving is almost always able to simplify or strengthen the problem instances, see the "affected models" column. Disabling presolve makes the solving process slower in the majority of cases, and it increases the average solving time for more challenging models (those that take more than 10 seconds to solve) by more than a factor of 11. Interestingly, this result is very much in line with the observations in Bixby et al. [17], even though we used a completely different test set and a much newer CPLEX version.

One can group the MIP presolving techniques into two classes, namely primal and dual presolve operations. Primal techniques are those that rely on feasibility arguments. They modify the problem instance in such a way that the set of feasible solutions is preserved, but either the size of the problem instance is reduced or the LP relaxation gets tighter. Examples are the removal of fixed variables and redundant

Table 9 Impact of individual presolving techniques in CPLEX 12.5 on the [10,10k] bracket

Feature	Models	Default	Modified code					Affected		
		Tilim	Tilim	Faster	Slower	Time	Nodes	Models	Time	Nodes
No primal reductions	1412	10	304	178	1161	6.06	5.26	1401	6.12	5.33
No dual reductions	1201	21	93	318	643	1.77	1.93	1171	1.79	1.96
No probing	1173	18	91	338	574	1.56	2.02	1114	1.61	2.09
No node presolving	1144	27	68	287	471	1.24	1.33	992	1.28	1.39
No root node restart	1134	23	35	332	441	1.10	0.95	985	1.12	0.94
No implied set cover probing	1088	14	22	167	217	1.04	1.06	634	1.07	1.10
No variable gcds	1087	8	7	46	47	1.04	1.08	133	1.39	1.94
No component analysis	1080	3	6	51	54	1.03	1.05	161	1.26	1.38
Regular Tarjan SCC	1075	0	0	5	34	1.02	1.00	–	–	–

constraints, bound strengthening, coefficient strengthening, lifting of constraints and aggregation of variables.

Dual techniques consider optimality. These reductions can eliminate feasible and even optimal solutions, as long as at least one optimal solution remains in the reduced problem, if it exists. Examples are dual fixing, fixings and aggregations based on symmetry and the removal of parallel and dominated columns.

Table 9 shows the impact of the individual groups of presolve reductions on the [10,10k] problem bracket. While the contribution of primal presolving techniques is clearly dominating, the dual reductions also yield a very significant speedup and have the same general applicability as primal reductions. Again, our results for node presolving (i.e., applying domain propagation at the each node of the search tree) is in line with the $1.3\times$ speedup reported by Bixby et al. [17] for CPLEX 8.0.

Root node restarts (see Achterberg [2]) have been added to CPLEX 11.0 and refined in the subsequent CPLEX 11 and 12 releases. This means to restart the solving process and apply another round of full presolve when, during the root cut loop, enough variables have been globally fixed. According to Table 9, root node restarts yield a $1.1\times$ improvement on average. Interestingly, this does not come from a reduction in the number of search tree nodes, which suggests that the main benefit is the smaller size of the LP relaxations due to the removal of fixed variables and redundant constraints.

Probing [55] is a very expensive presolving procedure in which one tentatively fixes binary variables to zero and one and propagates the fixings through the constraints. Analyzing the results of the propagation can detect fixed variables, aggregations, tighter bounds, implications, cliques and lifting opportunities to strengthen the LP relaxation. As can be seen in Table 9, probing contributes a significant performance boost, reducing the solving times by $1.56\times$ on average and the number of unsolved models by 73.

Even though probing is not challenging from a theoretical point of view, its implementation is a very serious exercise in high performance data structures and algorithms. It involves a great deal of computational experiments to find the right tuning of work limits and to identify special cases that have to be treated differently. For this reason, it is not surprising that probing saw a constant chain of improvements since its introduction in CPLEX 6.5, in particular throughout all of the CPLEX 11 and 12 releases.

One recent example of a probing extension is what we call *implied set cover probing* [10], which is new in CPLEX 12.5. This generalizes the idea of *implied literals detection* or *hyper-resolution* from the SAT community, see Bacchus [12] and Matsliah, Sabharwal and Samulowitz [45]. Translated to MIP, it basically means to apply probing on the members of the minimal covers implied by a knapsack constraint and to observe that at least one member of each cover must be set to zero. As with probing, the main challenge is to devise an efficient algorithm, which in particular aborts as early as possible if no reductions can be found for a given knapsack. Table 9 shows a small speedup of 1.07× on a bit more than half of the problem instances, which yields an overall speedup of 1.04×.

Another presolving operation for which the implementation can make a big difference is the strongly connected components analysis using Tarjan's algorithm [57]. This algorithm is applied on the implications between binary variables, stored implicitly in the clique table. It is used to detect equivalences of binary variables, which can then be aggregated. Tarjan's algorithm runs linearly in the size of the input graph, so it does not seem relevant to improve it for the use inside a MIP solver. But the issue is that the graph is only defined implicitly through the clique table. Since every clique in the clique table gives rise to quadratically many edges in the implication graph, just applying Tarjan's algorithm on the implicitly given graph yields a quadratic algorithm in the size of the largest clique. In particular, for large set partitioning models as they arise, for example, in airline crew scheduling applications, a quadratic run-time can be devastating. The algorithmic trick to get back to a linear algorithm is based on the observation that after a clique has been visited twice, all of the descendants of the clique members have already been visited. Thus we can backtrack in the depth first search process of Tarjan's algorithm.

Table 9 lists this improvement with a 1.02× overall speedup, which does not exceed the variability that one can expect from random noise. But algorithmic improvements like this are a special case: they do not change the solving path of the MIP solver, which means that the benchmark results do not suffer from random noise. Moreover, the overall speedup does not tell the full story here. Namely, for most of the problem instances, the quadratic overhead of the regular Tarjan algorithm applied to the clique table is marginal, as the cliques are typically short. But for some problem instances, the impact is significant. For example, the total solving time for *nw05*, which is a set partitioning model of an airline application, reduces from 151.5 to 4.2 seconds. Another example is *ivu06-big* from MIPLIB 2010 [36], which is a set partitioning model for duty scheduling in public transportation. Here, the regular Tarjan algorithm consumes 3543 seconds while our version specialized to clique-implied graphs finishes in 0.4 seconds.

The *variable gcds* presolve reduction, the effect of which is listed in Table 9, was introduced in CPLEX 12.5. It employs reasoning on the greatest common divisor of coefficients in (implied) equations and thus generalizes the implied integer variable detection, which has been in the code since CPLEX 6.5. An implied integer variable is a variable that is declared to be continuous, but which has to take integer values in any feasible solution. Therefore, such a variable can be treated as integer or continuous during the solving algorithm as deemed more convenient: for example, it could be used for branching or to strengthen cut coefficients, but is treated as continuous in a heuristic. The variable gcd detection in CPLEX 12.5 not only detects implied integer variables, it can also record the information that a variable x must always take values $x = pz + q$ with $p \in \mathbb{R}_{\geq 0}$, $q \in \mathbb{R}$ being constants and $z \in \mathbb{Z}$. Having extracted this information from the problem data, one could of course just replace x by $pz + q$ in the model, introducing a new variable $z \in \mathbb{Z}$ with appropriate bounds. But we observed that this can lead to significant numerical issues and a performance degradation. Instead, we keep x as is, but modify the algorithms in CPLEX to exploit the variable gcd relation explicitly. This includes getting tighter bounds in domain propagation and modifying the cutting plane formulas to derive stronger cut coefficients. Doing so affects about 12 % of the models in our test set and yields a 39 % performance improvement on those problem instances compared to the version that only detects implied integer variables.

The *component analysis*, which we have added in CPLEX 12.2, has a very similar performance impact to that of variable gcds. Here, the idea is that if the constraint matrix of a problem instance decomposes into several independent blocks, then one can solve them independently and combine the solution vectors. Table 9 shows that about 15 % of the problem instances are affected. This is pretty surprising, since one could think that a problem instance that decomposes into multiple blocks points to a modeling mistake; the problem should have been presented as multiple independent problems right from the beginning. One has to consider, however, that in many cases the block structure is only induced by the preceding presolve reductions.

Note that a feasible solution for the full problem is only available if we have a solution for each of the components. Thus, if one of the components is hard to solve, the simple approach of just solving each of the smaller components to optimality before starting to solve the largest component is impractical. One solution would be to solve the components in parallel and have a master process collect and merge the individual parts of the solution vector. But since this is, given the potential benefit, too complicated from the implementation point of view, we just decided to use a deterministic time limit (see Sect. 4.1.1 below) for the component solves and leave components that are too difficult for the main MIP solve.

3.4 Primal Heuristics

Branch-and-cut search is a complete procedure designed to find the optimal solution of a given problem instance or prove infeasibility thereof. In contrast, the goal of

Table 10 Comparison of default and not using heuristics

Bracket	Models	Default	No heuristics						Affected			
		Tilim	Tilim	Faster	Slower	Time	First	Nodes	Models	Time	First	Nodes
All	3154	94	225	779	924	1.28	2.30	2.32	2577	1.35	2.47	2.79
[0,10k]	3075	15	146	779	924	1.28	2.19	2.36	2577	1.35	2.47	2.79
[1,10k]	1920	15	146	677	867	1.50	3.42	2.79	1798	1.55	3.57	2.99
[10,10k]	1194	15	146	354	669	1.95	5.40	3.56	1163	1.99	5.48	3.69
[100,10k]	697	15	146	175	447	2.64	7.75	4.44	689	2.67	7.83	4.51
[1k,10k]	338	15	146	64	253	4.79	11.72	7.90	337	4.81	11.80	7.95

primal heuristics is to find good feasible solutions quickly. Even though heuristics are optional to solve MIPs, they are a standard ingredient in state-of-the-art MIP solvers [13, 17]. Embedded in the framework of branch-and-cut primal heuristics serve two goals:

algorithmic: The earlier good incumbent solutions are available during branch-and-bound search, the earlier subtrees can be pruned, thus reducing the size of the search tree.

pragmatic: It is often sufficient in practice to provide a good solution whereas a proof of optimality may not even be computationally tractable.

CPLEX incorporates a variety of different heuristics, which can be classified into two categories. *Starting heuristics* attempt to produce an initial feasible solution. In contrast, *improvement heuristics* take as input one or more feasible solutions and try to find better ones "close by". Examples for improvement heuristics include 1-opt, 2-opt, local branching [26], RINS [21] and polishing [54]. Starting and improvement heuristics mutually depend on one another. By definition, an improvement heuristic needs a solution to work on, which can be provided by a starting heuristic. Starting heuristics often produce solutions of poor quality and thus need improvement heuristics to turn them into better ones.

Table 10 compares default CPLEX against running without any heuristics. The first observation is that heuristics influence the optimization in most cases: compare the "models" and "affected models" columns. When it does not, this is often due to the simplicity of the model that can be solved by presolve. In fact, more than 93% of the models requiring more than 1 second to solve are affected by heuristics. Unfortunately, we still observe some cases in which none of our heuristics proves to be successful even though feasible solutions exist.

Heuristics contribute to the overall performance for solving problems to optimality by a moderate amount. The magnitude of the effect is in line with the results reported by Bixby et al. [17], even though since then several new heuristics have been added. One might be tempted to conclude that heuristics are less important and need not receive much further attention.

On the other hand, heuristics do play a role in being able to solve models at all. Disabling heuristics turn 146 models to be unsolvable, while it can only solve 15

Table 11 Impact of heuristic types on the [10,10k] model bracket

Feature	Models	Default	Modified code						Affected		
		Tilim	Tilim	Faster	Slower	Time	First	Nodes	Models	Time	Nodes
No starting	1179	24	81	407	591	1.48	5.47	2.37	1147	1.50	2.42
→ no before-LP	1112	18	32	295	308	1.07	1.78	1.09	779	1.10	1.12
No improvement	1124	8	68	322	421	1.28	1.00	1.51	958	1.32	1.62
→ no RINS	1110	16	64	303	367	1.21	1.00	1.34	920	1.26	1.43

models for which default CPLEX hits the time limit. This highlights the continued need for additional or improved heuristics that find solutions in cases in which the current arsenal fails. Doing so is unlikely to yield an overall speedup, but will expand the applicability of MIP technology.

The need for better heuristics is more obvious when considering the pragmatic aspect of finding feasible solutions quickly. Unfortunately, our testing methodology lends itself only to a limited extent to a careful analysis of this aspect. Instead, a different methodology would be needed such as the *primal integral* proposed by Achterberg, Berthold and Hendel [9]. To limit the scope, in this paper we restrict our analysis to observations that can be made within the given methodology. To this end, we included column "first" in Table 10 that shows the shifted geometric mean of the ratio of the time it took to find the first feasible solution in both runs. The impact of heuristics in this metric is much more pronounced.

The major part of this success can be attributed to a new class of starting heuristics, called *before-LP heuristics*, that have been first introduced in CPLEX 11.0. Unlike many starting heuristics the before-LP heuristics do not require a solution of the LP relaxation as input. Thus, they can be run before the root node LP relaxation has been solved or (since CPLEX 12.2) in parallel to solving the root LP. Moreover, before-LP heuristics should not solve linear programs using the full problem relaxation, or they would just duplicate the work for solving the LP relaxation.

CPLEX 12.5 contains the following types of before-LP heuristics:

- *Fix-and-propagate* successively fixes some variables and propagates their effects until a feasible solution is found or an incompatible fixing is detected.
- The *feasibility recovery* heuristic guesses an integer feasible solution, which is then likely to violate some constraints, and applies local search to recover feasibility.
- The *zero-objective* heuristic removes the objective function and applies presolve. If the model becomes small enough, the heuristic attempts to solve it as a sub-MIP.
- *Sub-MIP* heuristics fix enough variables such that presolve generates a small enough a problem to be solved as sub-MIP.

Considering that before-LP heuristics deliberately keep the work low, row 2 in Table 11 shows that they are remarkably successful at finding an initial feasible solution. About 70 % of the models are affected, and they account for a 78 % improvement in time to the first solution. Moreover, there is a small improvement in time to optimality due to before-LP heuristics, but they do not seem to contribute much to the ability to solve models at all. This is not surprising, since the same heuristics can also be applied later during the search so that the same or better solutions will typically be found after processing the root node.

Row 1 in Table 11 shows the impact of turning off all start heuristics, including the before-LP heuristics. Clearly, this accounts for the full impact of turning off heuristics for the time to first solution, since improvement heuristics can by definition not play a role in this regard. Moreover, general starting heuristics also affect the overall performance and, more importantly, the ability to solve certain models at all. They are thus more successful at finding useful feasible solutions than before-LP heuristics. Not only do they have access to stronger information with LP solutions, but they are also allotted more time to succeed.

A significant part of the success of starting heuristics for the overall performance comes from the fact that they produce starting points to be used by improvement heuristics. The effect of improvement heuristics is measured in row 3 of Table 11. In fact, turning off improvement heuristics reduces the performance impact of heuristics in general by a considerable margin. RINS proved to be the most successful among improvement heuristics in CPLEX.

Maybe more surprisingly, improvement heuristics play an important role for being able to solve models at all. 68 models become unsolvable if improvement heuristics are turned off. This amounts to almost half of the 146 cases from Table 10 for which heuristics are needed to solve a problem instance. It is thus often only due to the improvement heuristics that an optimal solution can be found at all.

3.5 Other Advances

Even though branching, cutting plane separation, presolving and primal heuristics are the main building blocks of a state-of-the-art MIP solver, there are of course other components that play a significant role. One of them is to deal with symmetry in the model. Symmetry detection was introduced to CPLEX with version 7.0 in 2000, but only as a non-default option that would detect symmetries using a heuristic and generate symmetry breaking constraints in advance on top of the presolved model. With CPLEX 10.0, released in 2006, we switched to the much more sophisticated symmetry detection algorithm AUTOM of Puget [52] and exploit symmetric patterns similarly to *0-setting* and *orbital branching*, see Margot [42] and Ostrowski et al. [47], respectively. As indicated in the "[10,10k]" bracket of Table 12, this yields a 91 % performance improvement on the 347 models to which it applies, leading to an overall speedup of 23 %. Moreover, there are some models in our test set that cannot be solved within the time limit if symmetry detection is disabled. Considering that only about 30 % of our models are affected by symmetry, these are important contributions.

A more recent advancement in CPLEX is the introduction of conflict analysis in CPLEX 12.3 in 2011, see Achterberg [1, 2]. In default settings, CPLEX analyzes

Table 12 Comparison of default and not exploiting symmetry

Bracket	Models	Default	No symmetry breaking					Affected		
		Tilim	Tilim	Faster	Slower	Time	Nodes	Models	Time	Nodes
All	3188	98	130	107	252	1.08	1.13	565	1.57	1.99
[0,10k]	3097	7	39	107	252	1.09	1.13	565	1.57	1.99
[1,10k]	1885	7	39	103	246	1.14	1.22	497	1.67	2.10
[10,10k]	1112	7	39	85	186	1.23	1.30	347	1.91	2.30
[100,10k]	587	7	39	53	124	1.36	1.42	219	2.28	2.57
[1k,10k]	238	7	39	25	72	1.78	1.90	115	3.27	3.79

Table 13 Comparison of default and not using conflict analysis

Bracket	Models	Default	No conflict analysis					Affected		
		Tilim	Tilim	Faster	Slower	Time	Nodes	Models	Time	Nodes
All	3176	97	114	189	300	1.05	1.08	991	1.16	1.28
[0,10k]	3090	11	28	189	300	1.05	1.08	991	1.16	1.28
[1,10k]	1868	11	28	183	297	1.08	1.13	879	1.18	1.30
[10,10k]	1101	11	28	149	241	1.13	1.18	630	1.24	1.33
[100,10k]	586	11	28	104	172	1.22	1.27	388	1.35	1.43
[1k,10k]	233	11	28	48	85	1.42	1.55	181	1.57	1.76

infeasible and bound-exceeding node LPs to derive conflict constraints, which are then used during node presolve for domain propagation. Long constraints are discarded if the conflict constraint table grows too large. Because CPLEX does not keep the structure of the search tree as it is done in SCIP, but instead just manages a flat list of open nodes, sophisticated reverse propagation rules [1] to strengthen the conflict constraints cannot be applied. Nevertheless, Table 13 shows that over half of the models in the [10,10k] bracket are affected by conflict analysis, on which a 24 % speedup is achieved. Unfortunately, in contrast to SAT [44], conflict analysis in CPLEX only helps to solve a few models that are otherwise intractable.

An interesting component that originated from the *feasibility pump* heuristic of Fischetti, Glover and Lodi [29] is called *pumpreduce*, which tries to reduce the number of fractional variables in the LP solution without sacrificing optimality. To achieve this, we first fix all structural variables and slack variables with non-zero reduced costs or dual solution values, respectively, to their current LP values, effectively forcing the solution of subsequent optimizations to stay on the optimal face of the LP polyhedron. Then, we iteratively modify the objective function and resolve with the primal simplex algorithm in order to drive the solution to a “more integral” basis. In its initial version in CPLEX 11.0, *pumpreduce* was mainly used to improve the LP solution as a starting point for primal heuristics. After extending the method in CPLEX 12.1, we discovered that *pumpreduce* can also be used for cut filtering, see Achterberg [4]. Namely, we now separate violated cutting planes for the initial

Table 14 Comparison of default and not using pumpreduce

Bracket	Models	Default	No pump-reduce					Affected		
		Tilim	Tilim	Faster	Slower	Time	Nodes	Models	Time	Nodes
All	3178	97	116	396	505	1.04	1.06	1517	1.09	1.14
[0,10k]	3101	20	39	396	505	1.04	1.07	1517	1.09	1.14
[1,10k]	1881	20	39	378	481	1.07	1.09	1244	1.12	1.14
[10,10k]	1135	20	39	262	364	1.11	1.15	836	1.16	1.21
[100,10k]	609	20	39	158	231	1.17	1.23	489	1.23	1.30
[1k,10k]	261	20	39	76	117	1.32	1.39	229	1.37	1.46

Table 15 Comparison of default (dynamic search) and traditional branch-and-cut

Bracket	Models	Default	Traditional branch-and-cut					Affected		
		Tilim	Tilim	Faster	Slower	Time	Nodes	Models	Time	Nodes
All	3159	96	226	493	945	1.30	1.33	2021	1.51	1.55
[0,10k]	3077	14	144	493	945	1.31	1.33	2021	1.51	1.55
[1,10k]	1905	14	144	472	907	1.54	1.69	1615	1.66	1.86
[10,10k]	1193	14	144	312	696	1.98	2.47	1086	2.12	2.70
[100,10k]	700	14	144	169	471	2.85	3.78	671	2.98	4.00
[1k,10k]	349	14	144	65	270	4.91	6.92	346	4.98	7.04

LP solution vector, then apply *pumpreduce* to find an alternative optimum and finally apply cut filtering based on this new LP solution. Consequently, cutting planes that are added to the LP relaxation will now cut off at least two optimal vertices of the LP polyhedron. The effect of *pumpreduce* in its current version is listed in Table 14. It yields a solid overall improvement of 11 % for the models in the [10,10k] bracket and also helps to solve a few more problems within the time limit. It affects about three quarters of the non-trivial problem instances.

Probably the most significant improvement in recent years of CPLEX development is the introduction of *dynamic search* in version 11.0. Dynamic search is not really a particular feature, but it mainly provides a way to drastically change how the features collaborate using information gained at runtime: it can restructure the search tree, modify the presolved model during the search, discard useless cutting planes or generate more of them, trigger expensive heuristics if needed and so on. Of course, this comes at a cost as some of those operations can be computationally expensive, but Table 15 shows that the net effect is a significant win. Not only does the performance improve by almost $2\times$ on the non-trivial models, it also helps to solve a large number of problem instances that are unsolvable with traditional branch-and-cut.

4 Parallelism

If you walked into a computer store in 2000 to buy a new computer you would probably have walked out with a Pentium 4 machine or equivalent. You would not have labeled it "single-core" since this was the default. Parallel shared memory computers were highly specialized equipment that required special motherboards that would support more than one CPU, significantly adding to their cost. The situation changed in 2004 as the first multi-core CPUs where introduced, first by SUN in their SPARC architecture and (2005) with the Athlon X2 processor by AMD. Today, single core processors have vanished from the general purpose computing market. Consequently, exploiting parallelism in state-of-the-art solvers has changed from being a premium feature to an indispensable requirement.

4.1 Deterministic Parallel Optimization

Branch-and-cut algorithms have the reputation of being embarrassingly parallel, simply by processing nodes of the search tree concurrently. Consequently, parallel MIP solvers have been around since the 1990s, see, e.g., Boehning, Butler and Gillett [18], Gendron and Crainic [31], Eckstein [25] and Linderoth [39]. In CPLEX, parallel MIP was introduced with version 4.0.3 in 1996.

However, using parallel algorithms for solving MIPs came at a price. Not only was the monetary cost for the hardware and software considerably higher, but the algorithm performed non-deterministically. This means that two executions of the same algorithm with the same data on the same machine are not guaranteed to follow the same solution path and thus can produce different solutions and (often very) different solving times.

Such non-determinism is caused by random timing differences when executing a concurrent algorithm. For instance, the decision to process or discard one node depends on the availability of an incumbent solution that may allow one to prune it. If such a new incumbent is found by a different processor at a different node, the exact timing determines if the decision to process a node comes out positive or negative.

With multi-core machines having become mainstream, such non-determinism is no longer acceptable in most situations. Having repeatable results is of paramount importance when developing and debugging applications. Technical support would become almost impossible if observed behavior cannot be reproduced and software demonstrations would become risky propositions. In 2007, CPLEX 11.0 was the first MIP solver to introduce deterministic parallel MIP, which is now standard among state-of-the art solvers.

4.1.1 Deterministic Clocks, Locks and Signaling

Central to the deterministic parallelization of the CPLEX solvers is the concept of a *deterministic clock*, which is characterized by the following features:

- a value that is monotonically increased with time,
- the increase happens at fine enough granularity,
- the value grows roughly linearly with time and
- the clock speed is roughly independent of the algorithm.

In CPLEX deterministic clocks are implemented as 64-bit integer counters that are incremented for each memory access [8].

Using the ideas in Olszewski, Ansel and Amarasinghe [46], deterministic clocks can be used to implement *deterministic locks*. Locks are used in concurrent computation to protect access to shared data so as to prevent threads to access data that is simultaneously modified by another thread. Before accessing such data, a lock must be acquired. The lock is only granted to one thread, while the other threads have to wait until the lock is released again. Clearly, without further attention, the order in which different threads will be granted the lock depends on the timing and is thus in general non-deterministic.

In order to make locks deterministic, we use a deterministic clock $d(t)$ for every thread $t \in 1, \dots, T$. When a thread t' is created by thread t we assign $d(t') = d(t)$. The determination whether or not to grant access to a lock to thread t is delayed until the following condition holds:

$$\left(\bigwedge_{t'<t} d\big(t'\big) > d(t) \right) \wedge \left(\bigwedge_{t'>t} d\big(t'\big) \geq d(t) \right).$$

In other words, a lock is only granted to the thread with the lowest deterministic time using the thread number as tie breaker. If a thread attempts to acquire a lock while this is not the case, it has to wait in order to give the other threads with lower deterministic time a chance to access the lock before. It is easy to see that this enforces determinism in the order in which threads will be granted locks. Furthermore, if all access to shared data is protected with such deterministic locks, the resulting concurrent algorithms will execute deterministically (as long as no other non-deterministic information, such as timing information is being used). This forms the basis for the CPLEX deterministic parallel solvers. They can be easily run in opportunistic (i.e., non-deterministic) mode, simply by switching to regular, non-deterministic locks instead.

A special class of parallel algorithms used in mathematical programming is called *concurrent optimization*. In this setting, different algorithms are available for solving a problem, yet there is no clear winner among them. Instead, each algorithm works best in some cases, but there is no known predictor for choosing it for a particular problem instance a priori. With concurrent optimization, a set of competing algorithms are executed concurrently. As soon as the first algorithm finishes, it terminates the others using a signal. On problems where the execution time of the algorithms is drastically different, such an algorithm will in practice yield deterministic results, but when this condition is violated, non-determinism becomes a problem. Fortunately, encapsulating the termination signal in a deterministic lock can overcome this issue. It is equivalent to using deterministic time rather than wall-clock time to declare a winner.

Table 16 Comparison of deterministic and opportunistic concurrent LP using 12 threads

Bracket	Models	Deterministic	Opportunistic concurrent LP				Affected	
		Tilim	Tilim	Faster	Slower	Time	Models	Time
All	2185	41	39	239	12	0.96	79	0.88
[0,10k]	2147	3	1	239	12	0.96	79	0.88
[1,10k]	658	3	1	225	12	0.89	42	0.82
[10,10k]	324	3	1	149	11	0.85	19	0.72
[100,10k]	135	3	1	78	7	0.78	12	0.66
[1k,10k]	41	3	1	20	2	0.77	8	0.58

Linear Programming is the most prominent example for concurrent optimization, with three competing algorithms: the primal and dual simplex algorithms and the barrier algorithm. While there is no known parallel implementation of the simplex algorithm that scales to many processors, the barrier algorithm lends itself very well to parallelism. By including the parallel barrier algorithm in the concurrent optimization, all available threads can be put to good use. As a consequence, the choice of which other algorithms to include influences the performance of the parallel barrier algorithm. In particular, with CPLEX we observed that including the primal simplex algorithm in the deterministic concurrent LP optimizer would lead to a deterioration of the overall performance. The primal simplex just does not win frequently enough to offset the overall performance reduction that barrier would suffer from using one less parallel thread.

4.1.2 Cost of Determinism

In order to make locks deterministic, we introduce additional potential wait times during synchronization. It should thus be expected that there is a performance penalty for achieving determinism, and the obvious question is how large this effect is.

The first algorithm we analyze is the concurrent LP solver. In Table 16 we compare the performance of the deterministic and opportunistic execution using our internal LP problem set consisting of 2185 models. We observe a clear performance degradation of up to 23 % due to determinism. Since the only difference between the deterministic and opportunistic concurrent LP optimizer is the synchronization for the termination signal at the end, the performance difference can be directly attributed to the deterministic lock. The deterministic lock, in turn, can only contribute additional waiting time if the deterministic clock of the dual algorithm and the barrier algorithm do not progress at the same pace.

For instance, if the barrier algorithm is the winner, but its deterministic clock ticks faster than the deterministic clock of the dual simplex algorithm, the deterministic lock will delay the signal until the deterministic clock of the dual simplex algorithm has caught up with the deterministic end time of the barrier algorithm.

Table 17 Comparison of deterministic (default) and opportunistic parallel MIP

Bracket	Models	Default	Opportunistic parallel MIP					Affected		
		Tilim	Tilim	Faster	Slower	Time	Nodes	Models	Time	Nodes
All	3162	93	101	1207	310	0.90	0.92	2130	0.86	0.88
[0,10k]	3098	29	37	1207	310	0.90	0.91	2130	0.86	0.88
[1,10k]	1868	29	37	1092	301	0.85	0.88	1666	0.84	0.86
[10,10k]	1107	29	37	635	258	0.86	0.87	1048	0.86	0.86
[100,10k]	603	29	37	329	188	0.88	0.89	589	0.88	0.88
[1k,10k]	249	29	37	128	92	0.98	0.99	247	0.98	0.99

During that time, it can happen that the dual simplex algorithm terminates as well and thus is declared the “deterministic winner”. Such situations are recognized as “affected models” in Table 16. This happens only for a small number of problem instances, which shows that in practice even the opportunistic version of the concurrent LP optimizer behaves deterministically on most of the problem instances, but there is no guarantee.

The measured degradation due to determinism increases with the runtime of the optimization. This can be explained by the observation that with more time, the inaccuracies of the deterministic clocks have more time to accumulate. Overall, the performance difference gives a good measure of the relative accuracy of the deterministic clocks for drastically different algorithms. In this view a difference of 23 % is surprisingly small, considering the dramatic difference in memory access patterns between barrier and simplex.

By construction it should be impossible that the deterministic algorithm outperforms the opportunistic one. Yet our table shows a hand full of cases where the opportunistic algorithm is reported to be slower (by at least 10 %). This is mostly due to wall clock time measurement errors. Only in one instance we observed a drastic loss of the opportunistic solver due to other reasons: In this case the dual simplex algorithm was the “deterministic winner”, whereas in opportunistic mode it was outperformed by barrier. But when the optimal basis from the barrier solution was copied to the master LP, in the master problem numerical errors triggered a long simplex run, which extended the 2702 seconds barrier solving time by an additional 6355 seconds.

The effect of determinism of the concurrent LP solver is also part of the “cost of determinism” for the CPLEX 12.5 MIP solver, which we evaluate in Table 17. The first observation is that indeed by giving up determinism, up to 15 % of performance can be gained. Opportunistic parallel is at least 10 % faster than deterministic parallel on more than a third of the models in the test set, while it is slower on less than 10 % of the problem instances. That there is so much variation in the winning algorithm comes from the performance variability of MIP. In contrast to deterministic concurrent LP optimization, where determinism needs to be enforced only once at the end, for MIP non-determinism can enter the algorithm at any point and thus dramatically change the solution path. In fact, most models are affected, with the

Table 18 Comparison of default and not using parallel root node computations

Bracket	Models	Default	No parallel root node						Affected			
		Tilim	Tilim	Faster	Slower	Time	First	Nodes	Models	Time	First	Nodes
All	3163	98	112	492	606	1.03	1.09	1.13	1922	1.07	1.14	1.22
[0,10k]	3084	19	33	492	606	1.04	1.08	1.13	1922	1.07	1.14	1.22
[1,10k]	1887	19	33	467	579	1.06	1.14	1.14	1472	1.08	1.18	1.18
[10,10k]	1119	19	33	325	421	1.09	1.21	1.18	964	1.11	1.25	1.22
[100,10k]	601	19	33	193	240	1.11	1.29	1.19	536	1.13	1.33	1.21
[1k,10k]	245	19	33	82	106	1.12	1.29	1.09	226	1.13	1.32	1.10

exception of those that can be solved by presolve or, within limits, those that can be solved at the root.

In recognition of the fact that deterministic locks are more expensive than regular ones, our MIP solver synchronizes less frequently in deterministic mode than in opportunistic mode. As a consequence, information is not shared as frequently between threads, which in turn adversely impacts the solve process. This can be observed as a decrease in number of nodes when using opportunistic parallelism.

While deterministic MIP is on average slower than opportunistic MIP, the number of timeouts does not significantly change between both options. This means that the performance gain one can obtain from opportunistic parallelism does not provide a qualitative boost that would allow one to solve problems that are otherwise unsolvable.

4.2 Performance Evaluation of Parallel MIP

Except for solving the initial root LP relaxation with the concurrent LP optimizer, the processing of the root node involves the cut loop, which is an inherently sequential algorithm. Thus computing resources are free to be used for other tasks. Since version 12.2, CPLEX not only exploits parallelism for processing the branch-and-cut tree, but is also applying more heuristics while the root node is being processed. Table 18 shows the impact of doing this. As expected, the main improvement that can be gained lies in the time to the first feasible solution. It also slightly helps overall performance, but not to a point that it would allow one to solve many more models.

The main exploitation of parallelism clearly lies in processing the tree concurrently. Table 19 shows the speedups that are achieved by increasing the number of threads to be used. These measurements include the impact of exploiting root node parallelism. We observe that the speedup improves with both, the number of threads as well as the "hardness" of the problem bracket.

However, the total speedup for a larger number of threads appears to be rather disappointing. While we are not able to fully explain this, we identified the following contributing factors.

Table 19 Scaling of deterministic parallel MIP using sequential mode as reference

Bracket	Models	2 threads		4 threads		6 threads		8 threads		10 threads		12 threads	
		Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
All	3143	0.81	1.03	0.69	1.10	0.65	1.14	0.62	1.17	0.61	1.21	0.60	1.22
[0,10k]	3095	0.81	1.02	0.69	1.08	0.64	1.12	0.61	1.15	0.60	1.18	0.59	1.20
[1,10k]	1990	0.72	1.06	0.56	1.17	0.50	1.23	0.47	1.28	0.46	1.33	0.44	1.35
[10,10k]	1353	0.65	1.05	0.47	1.19	0.41	1.25	0.36	1.31	0.35	1.37	0.34	1.39
[100,10k]	827	0.62	1.10	0.42	1.25	0.35	1.28	0.30	1.32	0.29	1.40	0.27	1.39
[1k,10k]	477	0.65	1.19	0.43	1.41	0.34	1.39	0.28	1.38	0.28	1.52	0.25	1.48

- The root node parallelization does not scale well. Thus, the possible parallel speedup is limited a priori through Amdahl's Law [11], which describes an upper bound of the achievable parallel speedup, if parts of an algorithm are not affected by the parallelization.
- The same is true for the ramp-up phase of the tree search, in which not enough nodes are available yet for parallel processing.
- We observe that with an increasing number of threads the number of nodes increases as well. The extra computational power from multiple threads is in part just used to process nodes that turn out to be discardable after the optimal solution is found.
- Since the sequential code is slower, it also produces more time limit hits. These are accounted for as 10000 seconds in the geometric mean which biases the results in favor of the sequential code.
- When 12 threads operate concurrently they all have to access data. This may yield memory bus contention which could slow down all threads.

In recognition of these effects, we attempted to compute the parallel speedup for the tree search in isolation with the following approach. First, we excluded all models for which a time limit was encountered in order to avoid the bias. We further excluded models that took less than 1000 nodes to solve for any of the solvers in order to reduce the effect of the ramp-up phase. Finally, we only measured the branch-and-cut time, i.e., we excluded the presolving and root node time. For the subset of models in the [10,10k] bracket, this yields a speedup of $6.31\times$ on 12 threads. Taking into account that the node count also increased by a factor of 1.11, this corresponds to an increase of a factor of 7 in node throughput, which is still considerably lower than linear. Given that we processed at least 1000 nodes, the ramp-up phase can only play a minor role in this.

In order to assess the slowdown due to memory contention we conducted two additional experiments. First, we ran the sequential solver alone on the machine. Second, we ran 12 identical copies of the solver concurrently on the same machine. In the second experiment, we observed a slow down of a factor of 1.31 in geometric mean, which varied between models in the range of 1 to 3. This variability does not allow us to estimate the true effect within one process, when data is shared and thus cache lines can get invalidated between threads. We only measured the effect for our machines, but in general hardware architecture is certain to also be a limiting factor in the achievable parallel speedup.

5 Conclusion

We have analyzed the performance impact of the main components of a MIP solver as implemented in CPLEX 12.5. Table 20 summarizes our key findings and compares the impact of the individual features. To make the comparison meaningful, we use the same model set for all rows by selecting the [10,10k] problem bracket w.r.t. all features. Thus, the evaluation is performed on those problem instances for which

Table 20 Impact of individual solver features on performance on the [10,10k] bracket

Feature	Models	Default	Modified code					Affected		
		Tilim	Tilim	Faster	Slower	Time	Nodes	Models	Time	Nodes
No presolving	1769	45	500	233	1394	7.57	8.58	1722	8.00	9.14
Most infeasible branching	1769	45	500	171	1071	4.63	6.53	1440	6.57	10.06
No cutting planes	1769	45	399	459	984	3.75	13.76	1611	4.26	17.64
No parallelism (12 threads)	1769	45	154	230	1216	2.33	0.73	1604	2.56	0.73
No dynamic search	1769	45	171	440	808	1.55	1.73	1466	1.70	1.94
No heuristics	1769	45	173	601	756	1.51	2.70	1640	1.57	2.91
No symmetry	1769	45	77	94	229	1.15	1.22	469	1.68	2.10
No conflict analysis	1769	45	63	173	271	1.09	1.12	824	1.19	1.28
No pumpreduce	1769	45	64	353	434	1.06	1.08	1150	1.11	1.13

at least one of the solvers listed in the table (including default CPLEX 12.5) was able to solve the model and for which at least one solver took at least 10 seconds. This problem set is larger than the individual [10,10k] sets used in the previous sections, which explains the differences in the data.

Note that the degradation value for most infeasible branching stands out, since branching cannot be turned off like the other features. Turning off branching would yield a pure cutting plane algorithm with little hope of solving larger models [60]. The degradation factor of 4.63 thus only represents a qualitative statement that the branching decisions are indeed of upmost importance.

The most important result is captured in the number of additional time limit hits due to disabling a certain feature. We like to view this as a proxy for differentiating features that are essential for the ability to solve certain models from those that merely provide a speedup. Among essential features, presolving and branching are undisputedly the most important, followed by cutting planes. No ranking can be determined between dynamic search and heuristics: both are equally important for solving models. Symmetry is a special case as it is critical only on a much smaller subset of the models. The additional time limit hits due to disabling symmetry detection are smaller by a factor of about 4 compared to dynamic search and heuristics. But this is in line with the difference in the number of affected models. Thus, if it applies, symmetry is as important for solving models as dynamic search and heuristics.

Parallelism needs to be considered separately. Using only 1 rather than 12 threads effectively reduces by a factor of 12 the allowed computation time. The comparison of 12 vs 1 thread is thus essentially imposing a $\frac{1}{12}10000$ second time limit, hence the increase in time limit hits. As we have shown in Sect. 4.2, the speedup due to parallelism is far from linear, which explains why the number of time limit hits for the sequential version is in fact surprisingly low. Thus indeed, exploitation of parallelism does not qualify as feature that helps solving more models. It cannot defeat the combinatorial nature of MIP, but only (moderately) accelerate the computation.

Conflict analysis and pumpreduce do not seem to be key ingredients for solving the models in our set. Primarily, they only help to improve performance on those models that can already be solved without the feature.

We also observe a division between the components in terms of their applicability. Presolving, branching, cuts, parallelism, dynamic search and heuristics are applicable throughout the test set, whereas the remaining features only affect a reduced set of models.

It is interesting to note the change in relative importance of cuts when comparing to the results in Bixby et al. [17], where cuts are reported to be five times more effective than presolve, the second most important factor. In our analysis, the effectiveness is reversed with presolve coming out as more than twice as effective as cuts for CPLEX 12.5. Clearly, cuts have not become so much weaker nor presolve that much stronger. The reason for this discrepancy can be found in the test set that was used in [17], which included only 106 instances. It was constructed by selecting problems that were unsolvable for CPLEX 5.0 but solved relatively easily with CPLEX 8.0. Therefore, the large degradation when turning off cuts only means that this is the main feature contributing to making more models solvable by CPLEX 8.0 (which was exactly the purpose of this particular experiment). In fact, cuts were the main source of improvement in CPLEX 6.5, see Bixby et al. [16]. Interestingly, despite this very special selection of the test set, the results for presolve and heuristics are roughly in line with our unbiased measurements.

Thus, comparing CPLEX 8.0 with CPLEX 12.5, the impact of heuristics and presolve apparently did not change much. For cuts let us estimate the improvement with $2\times$ and for branching with $1.29\times$ taken from Table 4. With these crude estimates, we can gauge the improvement between the two versions using the product of these contributions and the remaining features in Table 20 (dynamic search, symmetry, conflict analysis and pumpreduce). The resulting factor of about 5 to 6 is still far from the measured speedup of $21.91\times$ in Table 1 for the same model bracket. While this estimate cannot be expected to be numerically accurate, it clearly points out that something is missing from the full picture.

The authors claim that the missing link is the “black art of MIP”, or *soft features*. Soft features are characterized by exploiting existing features in better ways, rather than adding new concepts to the arsenal of weapons to attack MIPs. We have already discussed some soft features, namely root node restarts in Sect. 3.3, cut filtering in Sect. 3.2 and pumpreduce in Sect. 3.5.

Let us finish by presenting yet another soft feature, which unfortunately cannot be quantified in the same way as the other components that we discussed. As we pointed out, many features only affect a subset of models. It is thus important not to waste time on any feature that may turn out to be irrelevant for a particular model. More importantly, if a feature is helping to solve a particular model, it should not be disabled before it has provided its contribution, which may be crucial for solving the problem instance. The “black art of MIP” consists of dynamic self-tuning of the individual algorithms and the relative time allotment between algorithms during runtime. To do so, CPLEX attempts to measure the effectiveness of each algorithm to decide which to focus on more or less during the optimization. The deterministic clock described in Sect. 4.1 plays a central role in the self-tuning of CPLEX, since it allows one to base tuning decisions on time, without introducing non-determinism.

However, tuning alone will not qualitatively expand the applicability of MIP technology. Further essential advances are needed. Here "essential" is used to identify advances without which certain models cannot be solved. Where these are to be found is left for future research. Reviewing the ideas we analyzed in this paper, they can be found anywhere: among key features of general applicability, such as cuts or branching, among features that exploit special structure only available for a subset of models, such as symmetry, or even among soft features such as dynamic search.

References

1. Achterberg, T.: Conflict analysis in mixed integer programming. Discrete Optim. **4**(1), 4–20 (2007)
2. Achterberg, T.: Constraint integer programming. Ph.D. thesis, Technische Universität Berlin (2007)
3. Achterberg, T.: SCIP: solving constraint integer programs. Math. Program. Comput. **1**(1), 1–41 (2009)
4. Achterberg, T.: LP basis selection and cutting planes. In: Mixed Integer Programming Workshop (MIP 2010) (2010)
5. Achterberg, T., Berthold, T.: Hybrid branching. In: van Hoeve, W.J., Hooker, J. (eds.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2009). Lecture Notes in Computer Science, vol. 5547, pp. 309–311. Springer, Berlin (2009)
6. Achterberg, T., Raack, C.: The MCF-separator: detecting and exploiting multi-commodity flow structures in MIPs. Math. Program. Comput. **2**(2), 125–165 (2010)
7. Achterberg, T., Koch, T., Martin, A.: Branching rules revisited. Oper. Res. Lett. **33**, 42–54 (2005)
8. Achterberg, T., Junglas, D., Wunderling, R.: Deterministic parallelization through atomic task computation. US Patent US20120311604 A1 (2011)
9. Achterberg, T., Berthold, T., Hendel, G.: Rounding and propagation heuristics for mixed integer programming. In: Klatte, D., Lüthi, H.J., Schmedders, K. (eds.) Operations Research Proceedings 2011, pp. 71–76. Springer, Berlin (2012)
10. Achterberg, T., Sabharwal, A., Samulowitz, H.: Stronger inference through implied literals from conflicts and knapsack covers. In: Gomes, C., Sellmann, M. (eds.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2013). Lecture Notes in Computer Science, vol. 5547. Springer, Berlin (2013)
11. Amdahl, G.: Validity of the single processor approach to achieving large-scale computing capabilities. In: AFIPS Conference Proceedings, vol. 30, pp. 483–485 (1965)
12. Bacchus, F.: Enhancing Davis Putnam with extended binary clause reasoning. In: Eighteenth National Conference on Artificial Intelligence, pp. 613–619. American Association for Artificial Intelligence, Menlo Park (2002)
13. Berthold, T.: Primal heuristics for mixed integer programs. Master's thesis, Technische Universität Berlin (2006)
14. Bixby, R.E.: A brief history of linear and mixed-integer programming computation. In: Grötschel, M. (ed.) Optimization Stories, pp. 107–121. Deutsche Mathematiker-Vereinigung, Bielefeld (2012)

15. Bixby, R.E., Rothberg, E.: Progress in computational mixed integer programming—a look back from the other side of the tipping point. Ann. Oper. Res. **149**(1), 37–41 (2007)
16. Bixby, R.E., Fenelon, M., Gu, Z., Rothberg, E., Wunderling, R.: MIP: theory and practice—closing the gap. In: Powell, M., Scholtes, S. (eds.) Systems Modelling and Optimization: Methods, Theory, and Applications, pp. 19–49. Kluwer Academic, Norwel (2000)
17. Bixby, R.E., Fenelon, M., Gu, Z., Rothberg, E., Wunderling, R.: Mixed-integer programming: a progress report. In: Grötschel, M. (ed.) The Sharpest Cut: The Impact of Manfred Padberg and His Work. MPS-SIAM Series on Optimization, pp. 309–325. SIAM, Philadelphia (2004)
18. Boehning, R.L., Butler, R.M., Gillett, B.E.: A parallel integer linear programming algorithm. Eur. J. Oper. Res. **34**(3), 393–398 (1988)
19. Cook, W.: Markowitz and Manne + Eastman + Land and Doig = branch and bound. In: Grötschel, M. (ed.) Optimization Stories, pp. 227–238. Deutsche Mathematiker-Vereinigung, Bielefeld (2012)
20. Crowder, H., Johnson, E.L., Padberg, M.W.: Solving large scale zero-one linear programming problems. Oper. Res. **31**, 803–834 (1983)
21. Danna, E., Rothberg, E., Le Pape, C.: Exploring relaxation induced neighborhoods to improve MIP solutions. Math. Program. **102**(1), 71–90 (2005)
22. Danzig, G.B., Fulkerson, D.R., Johnson, S.M.: Solution of a large-scale traveling-salesman problem. Oper. Res. **2**, 393–410 (1954)
23. Danzig, G.B., Fulkerson, D.R., Johnson, S.M.: On a linear-programming, combinatorial approach to the traveling-salesman problem. Oper. Res. **7**, 58–66 (1959)
24. Eastman, W.: Linear programming with pattern constraints. Ph.D. thesis, Department of Economics, Harvard University, Cambridge, MA, USA (1958)
25. Eckstein, J.: Parallel branch and bound algorithms for general mixed integer programming on the CM-5. SIAM J. Optim. **4**(4), 794–814 (1994)
26. Fischetti, M., Lodi, A.: Local branching. Math. Program. **98**(1–3), 23–47 (2003)
27. Fischetti, M., Lodi, A.: Heuristics in mixed integer programming. In: Cochran, J.J. (ed.) Wiley Encyclopedia of Operations Research and Management Science, vol. 8, pp. 738–747. Wiley, New York (2011)
28. Fischetti, M., Monaci, M.: Branching on nonchimerical fractionalities. Oper. Res. Lett. **40**, 159–164 (2012)
29. Fischetti, M., Glover, F., Lodi, A.: The feasibility pump. Math. Program. **104**(1), 91–104 (2005)
30. Fourer, R.: On the evolution of optimization modeling systems. In: Grötschel, M. (ed.) Optimization Stories, pp. 377–388. Deutsche Mathematiker-Vereinigung, Bielefeld (2012)
31. Gendron, B., Crainic, T.G.: Parallel branch-and-bound algorithms: survey and synthesis. Oper. Res. **42**(6), 1042–1066 (1994)
32. Gomory, R.E.: Outline of an algorithm for integer solutions to linear programs. Bull. Am. Math. Soc. **64**, 275–278 (1958)
33. Grötschel, M. (ed.): The Sharpest Cut: The Impact of Manfred Padberg and His Work. MPS-SIAM Series on Optimization, vol. 4. SIAM, Philadelphia (2004)
34. Grötschel, M., Jünger, M., Reinelt, G.: A cutting plane algorithm for the linear ordering problem. Oper. Res. **32**, 1195–1220 (1984)
35. Karamanov, M., Cornuéjols, G.: Branching on general disjunctions. Math. Program. **128**, 403–436 (2011)
36. Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R.E., Danna, E., Gamrath, G., Gleixner, A.M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D.E., Wolter, K.: MIPLIB 2010. Math. Program. Comput. **3**, 103–163 (2011)
37. Koster, A., Zymolka, A., Kutschka, M.: Algorithms to separate $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts. Algorithmica **55**, 375–391 (2009)
38. Land, A., Doig, A.: An automatic method of solving discrete programming problems. Econometrica **28**, 497–520 (1960)
39. Linderoth, J.T.: Topics in parallel integer optimization. Ph.D. thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA (1998)

40. Linderoth, J.T., Savelsbergh, M.W.P.: A computational study of search strategies for mixed integer programming. INFORMS J. Comput. **11**, 173–187 (1999)
41. Mahajan, A., Ralphs, T.: Experiments with branching using general disjunctions. In: Chinneck, J.W., Kristjansson, B., Saltzman, M.J. (eds.) Operations Research and Cyber-Infrastructure. Operations Research/Computer Science Interfaces Series, vol. 47, pp. 101–118. Springer, Berlin (2009)
42. Margot, F.: Pruning by isomorphism in branch-and-cut. Math. Program. **94**(1), 71–90 (2002)
43. Markowitz, H.M., Manne, A.S.: On the solution of discrete programming problems. Econometrica **25**, 84–110 (1957)
44. Marques-Silva, J.P., Sakallah, K.A.: GRASP: a search algorithm for propositional satisfiability. IEEE Trans. Comput. **48**, 506–521 (1999)
45. Matsliah, A., Sabharwal, A., Samulowitz, H.: Augmenting clause learning with implied literals. In: Cimatti, A., Sebastiani, R. (eds.) SAT. Lecture Notes in Computer Science, vol. 7317, pp. 500–501. Springer, Berlin (2012)
46. Olszewski, M., Ansel, J., Amarasinghe, S.: Kendo: efficient deterministic multithreading in software. ACM SIGPLAN Not. **44**(3), 97–108 (2009)
47. Ostrowski, J., Linderoth, J.T., Rossi, F., Smriglio, S.: Orbital branching. Math. Program. **126**, 147–178 (2011)
48. Owen, J.H., Mehrotra, S.: Experimental results on using general disjunctions in branch-and-bound for general-integer linear programs. Comput. Optim. Appl. **20**(2), 159–170 (2001)
49. Padberg, M., Rinaldi, G.: Optimization of a 532-city symmetric traveling salesman problem by branch and cut. Oper. Res. Lett. **6**, 1–7 (1987)
50. Padberg, M., Rinaldi, G.: A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM Rev. **33**, 60–100 (1991)
51. Patel, J., Chinneck, J.W.: Active-constraint variable ordering for faster feasibility of mixed integer linear programs. Math. Program. **110**, 445–474 (2007)
52. Puget, J.F.: Automatic detection of variable and value symmetries. In: van Beek, P. (ed.) CP 2005. Lecture Notes in Computer Science, vol. 3709, pp. 475–489. Springer, Berlin (2005)
53. Raidl, G.R., Puchinger, J.: Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In: Blum, C., Aguilera, M., Roli, A., Sampels, M. (eds.) Hybrid Metaheuristics. Studies in Computational Intelligence, vol. 114, pp. 31–62. Springer, Berlin (2008)
54. Rothberg, E.: An evolutionary algorithm for polishing mixed integer programming solutions. INFORMS J. Comput. **19**, 534–541 (2007)
55. Savelsbergh, M.W.P.: Preprocessing and probing techniques for mixed integer programming problems. ORSA J. Comput. **6**, 445–454 (1994)
56. SCIP: Solving constraint integer programs. scip.zib.de
57. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972)
58. van Roy, T.J., Wolsey, L.A.: Solving mixed integer programming problems with automatic reformulation. Oper. Res. **35**(1), 45–57 (1987)
59. Wunderling, R.: Paralleler und objektorientierter Simplex-Algorithmus. Ph.D. thesis, Technische Universität Berlin (1996)
60. Zanette, A., Fischetti, M., Balas, E.: Lexicography and degeneracy: can a pure cutting plane algorithm work? Math. Program. **130**(1), 153–176 (2011)

Progress in Academic Computational Integer Programming

Thorsten Koch, Alexander Martin, and Marc E. Pfetsch

Abstract This paper discusses issues related to the progress in computational integer programming. The first part deals with the question to what extent computational experiments can be reproduced at all. Afterward the performance measurement of solvers and their comparison are investigated. Then academic progress in solving mixed-integer programming at the examples of the solver SIP and its successor SCIP is demonstrated. All arguments are supported by computational results. Finally, we discuss the pros and cons of developing academic software for solving mixed-integer programs.

1 Introduction

The field computational integer programming deals with the computational aspects of integer and combinatorial optimization. Ever since the paper of Dantzig, Fulkerson, and Johnson [33] it has been clear that one main goal would be to actually compute (optimal) solutions and that practical considerations would have a major influence on the evolution of this field.

This article discusses the developments that have been made in the last 20 years with and through academic research and software. The main point is to highlight important issues that arise when implementing, testing, and benchmarking mixed-integer programming (MIP) software. For concreteness, we use the MIP-solvers SIP

T. Koch (✉)
Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, 14195 Berlin, Germany
e-mail: koch@zib.de

A. Martin
Department Mathematik, Friedrich-Alexander-Universität Erlangen-Nürnberg, Cauerstr. 11, 91058 Erlangen, Germany
e-mail: alexander.martin@math.uni-erlangen.de

M.E. Pfetsch
Fachbereich Mathematik, Technische Universität Darmstadt, Dolivostr. 15, 64293 Darmstadt, Germany
e-mail: pfetsch@opt.tu-darmstadt.de

and SCIP as examples, see [67] and [3, 71]. Consequently, this article is written from the personal perspective and experience of the authors, which allows us to present more details compared to just a general overview. We believe that many parts will find their analogies with other solvers (both commercial and academic). All three authors were involved in the development of SCIP or its ancestor SIP and were or are still working at the group of Martin Grötschel at the University of Augsburg and at the Zuse Institute Berlin (ZIB). In fact, the authors are lucky to have lead the integer programming group for some time.

One main motivation for this article is the growing importance of computer driven experiments and how to draw scientifically meaningful conclusions from them in the area of computational integer programming in particular. Indeed, it seems that the publication standards of papers involving computations have not yet been fully fixed—in contrast to other physical sciences that are based on experiments. Actually, Hooker [49] called for a new paradigm to evaluate experimental results obtained on the computer, and Greenberg [42] already discussed standards of publication—apparently, both without much effect. Our article tries to bring (back) into focus several issues that one has to be aware of in the context of computational integer programming.

It seems to be important to note that most issues that we discuss are not primarily of a mathematical nature, but they are necessary to come to scientifically sound conclusions when evaluating mathematical ideas. Clearly, these topics are at the intersection of different fields like computer science, operations research, engineering, and mathematics. Consequently, researchers from all these fields—mathematicians, in particular—have to be aware of the loopholes, traps, and organizational issues that are related to such computations. We see our article as one contribution in this direction.

We explicitly mention mathematicians here, because the authors have heard several times throughout their careers that implementation would be an issue for engineers/computer scientists and should not be performed by mathematicians. We do not agree and believe in the interdisciplinary viewpoint stated above.

We begin with a brief historical review of the developments during the last two decades related to SIP and SCIP in order to put things into a perspective and—in a sense—to document its achievements. Then we discuss the question whether the computational experiments that have been performed in the past can be reproduced at all. This is a topic that is often neglected in the literature and that, we think, deserves more attention. The next section deals with our experiences of benchmarking MIP-solvers. In the final section, we discuss issues related to the development of academic software in this context.

2 Historical Overview

This section briefly reviews the last 20 to 30 years in computational integer programming; it focuses on academic developments, and, in particular, those at ZIB.

As mentioned above, we decided to largely neglect other (academic) software developments in this article. The following references give more balanced surveys on the available software: Atamtürk and Savelsbergh [17], as well as Linderoth and Lodi [61], give an overview of MIP-solvers, while Linderoth and Ralphs [62] focus on non-commercial MIP-solvers. For an overview of linear programming (LP) solvers, see Fourer [39]. More details and a discussion of possible future research topics can be found in the survey of Lodi [64].

2.1 General Developments Starting in the 1980s

Possibly the first major step for the development of *integer programming* solvers was the seminal paper by Crowder, Johnson, and Padberg [31], which introduces many concepts that are still part of modern MIP-solvers. In the context of combinatorial optimization, the development of *branch-and-cut* (B&C) algorithms turned out to be very influential, based, e.g., on the article by Grötschel, Jünger, and Reinelt [43] on the linear ordering problem. From a computational perspective, the paper of Padberg and Rinaldi [70] on the solution of the traveling salesman problem (TSP) was a major step. Apart from the introduction of many important components like the cut pool, it also coined the name “branch-and-cut”. At that time it became common to look for an $\mathcal{NP}$-hard combinatorial problem, investigate its facets, and then implement a specialized B&C algorithm to solve it. The algorithms consisted of a separation procedure for the identified problem-specific facets and valid inequalities plus some branching scheme.

At ZIB, a large number of specialized B&C solvers were implemented in the late 1990s, e.g., [28, 29, 35, 44, 45, 56, 75], to mention just a few.

People generally implemented their own branch-and-bound framework using one of the available out-of-the-box LP-solvers like OSL, MINOS, CPLEX, or XPRESS. General cutting planes like Gomory cuts were not considered to be useful in practice. For MIP-solving this only changed with the rediscovery of Gomory’s mixed integer cuts by Balas et al. [18], which made B&C approaches dominant also for MIP-solving.

The main focus of the B&C algorithms for combinatorial optimization was on separating, and the number of branch-and-bound nodes that had to be enumerated was relatively small (or the instance could not be solved anyway due to the computer standards of the time). This fits well with the statement that *branching is a sign of mathematical defeat*, which is attributed to Manfred Padberg.

Several general out-of-the-box MIP-solvers like OSL, XPRESS, or CPLEX existed, but apart from their versatility, these were inferior to the special implemented algorithms.

2.2 MIP-Solving at ZIB

The work of the integer programming group at ZIB started in the early 1990s, after Martin Grötschel had moved to Berlin. Beginning in 1992, Cray Research had funded a project to develop a general parallel B&C framework to solve large combinatorial optimization problems. This framework was targeted at the then state-of-the art distributed memory Cray T3D computer. The project was lead by Christian Hege and conducted by Roland Wunderling and Martin Grammel. Their assumption was that the central part of any B&C solver is the simplex algorithm, and therefore the first goal of the project was to develop a fast parallel distributed memory simplex algorithm.

When in 1996 Roland Wunderling finished his Ph.D. thesis [76] on the LP-solver library SoPlex (sequential object-oriented simplex), two things could be concluded:

1. It was possible to write a simplex-based LP-solver that matched the performance of the commercial implementations at that time.
2. Developing a (distributed memory) parallel simplex-based LP-solver is not an especially promising idea (see also [24]). The work on the distributed memory B&C framework ceased even some time before.

One of the main features of SoPlex was the exploitation of the sparsity of the constraint matrix. In most linear and integer programming problems, the matrix A is very sparse, see [65]. There are some exceptions in which specialized algorithms for dense linear algebra computations are needed, see, e.g., [34]; remarkably, this area never received much attention. Despite the enormous speed-up that linear programming achieved in practice as reported by Bixby [22], there is only a small number of articles that together seem to contain everything important on how to implement a simplex based LP-solver: [32, 37, 38, 47, 58, 60, 72, 73].

At the end of the 1990s, the performance of general out-of-the-box MIP-solvers tremendously improved. One main influencing action was *mining the literature*, as Bob Bixby, the founder of CPLEX and Gurobi, called it. At this point a significant amount of theoretical results, in particular, on cutting planes, had been published that were not utilized in the out-of-the box (commercial) MIP-solvers. By incorporating these insights, it was possible to enormously improve the performance of general MIP-solvers, and for the first time it became hard to beat them by special implementations. More and more (practical) problems could be modeled and solved directly without further programming. As the solvers evolved, it became common to use them as frameworks for specialized algorithms by only extending the basic solver, instead of implementing a complete B&C algorithm from scratch.

In 1994, Alexander Martin started to develop the general MIP-solver SIP (Solving Integer Programs). When he finished his habilitation treatise in 1998, two things could be concluded:

1. At that time it was possible to write a B&C based MIP-solver that matched the performance of commercial implementations: SIP was comparable in performance to CPLEX at the time, see [67], although a notable disadvantage was that it used CPLEX as embedded LP-solver.

2. It became clear that implementing (shared memory) parallel MIP-solvers had some potential, but proved to be difficult. The reason for this is that the backbone of MIP-solvers was (and still is) the (dual) simplex algorithm. Moreover, one particular problem was the insufficient memory bandwidth of the available machines.

When Alexander Martin moved to TU Darmstadt in 2000, Tobias Achterberg and Thorsten Koch continued the work on SIP at ZIB. In particular, it was interfaced with SoPlex, making SIP completely available in the source code for the first time.

A main component of MIP-solvers are rules to choose the branching variable in each node of the tree. One key idea is strong branching, which was invented in the 1990s, see [14, 15, 51]. Before actually branching on some variable, its value is tested by temporarily fixing the variable to its up and down value and comparing the resulting LP relaxations. This is obviously time-consuming, but currently the best choice in terms of the number of branch-and-bound nodes. In 2004, the state-of-the-art w.r.t. solving time was the so-called *pseudo cost branching*, an idea that was already developed in the 1970s, see [19, 63, 67], which tries to "learn" from previous branching decisions and constructs artificial costs of the variables that hopefully reflect their merit for branching. In 2005, a dynamic combination of both methods, the so-called *reliability branching*, was developed, see [7]. A variation of it is still state-of-the-art, see Achterberg and Berthold [4]. This was a major step forward, since the main handicap of *pseudo cost branching* is that, especially in the beginning when the branching decision is most influential, no additional information is available, and *most infeasible branching* was used; this missing information at the beginning is dynamically supplied by strong branching. As experiments revealed, most infeasible branching does not perform better than branching randomly.

Around 2003, MIPs were generalized by incorporating concepts from constraint programming leading to so-called *Constraint Integer Programming*, see [2] for an exact definition. This was motivated by an industry project with Infineon on the verification in chip design. Since it became clear that the basic infrastructure of SIP was very much tailored toward solving MIPs, Tobias Achterberg began to develop SCIP (Solving Constraint Integer Programs) as part of his Ph.D. thesis, see [2, 10]. He also extended SCIP by using SAT solving techniques, e.g., restarts and conflict analysis.

Since then, SCIP has been continuously developed and improved. Apart from the added functionality with respect to constraint programming, SCIP has been one of the fastest non-commercial MIP-solvers, see, e.g., Mittelmann [69], and used in numerous areas—often beyond classical MIP-solving; examples are:

- column generation and decomposition of MIPs [40],
- constraint programming and conflict analysis [1, 3, 9],
- counting solutions [11, 48],
- MIP-solver technology [4, 5, 21],
- mixed-integer nonlinear programming [27, 74],
- pseudo-Boolean optimization [20],
- semidefinite programming [16, 66],
- symmetries in integer programs [52, 53].

In 2007, the first version of the ZIB Optimization Suite was released, which integrated SCIP as CIP-solver, SoPlex as solver for the LP-relaxations, and Zimpl [54, 55] as modeling environment. The source codes are freely available for academic usage. Thus, a complete state-of-the-art solver environment is available in the source code for usage, improvement, and teaching.

In 2012, the third major version of the now called SCIP Optimization Suite has been released. It integrates, for example, a framework for distributed parallel computations and substantially extended the functionality to solve mixed-integer nonlinear programs.

However, the size of the suite (altogether about 800,000 lines of code) also demonstrates that computational integer programming has clearly evolved beyond the point where an individual can just sit down and implement a state-of-the-art solver as a Ph.D. thesis. A large number of complex and involved components are needed, and their integration is a major issue. We will discuss this further in Sect. 6.

When looking back over the last 25 years of work on the topic, the question that comes up is *How much progress has been made?* There is the well-known article by Bixby [22] that determines at least a million times speed-up for linear programming during 15 years and an article by Achterberg and Wunderling [6] investigating the improvements in MIP solving. Can we make similar conclusions regarding (academic) integer programming? To answer this question we have to compare the performance of different MIP solvers over time. But in order to compare results, they have to be available or at least reproducible, an issue addressed in the next section. The question on how to compare results is then discussed in Sect. 4.

3 Reproducibility of Computational Results

One of the foundations of scientific research is that experiments should be reproducible. The key question is what kind of reproducibility one actually requires. This generally varies over different natural sciences. The results of an experiment reproduced by independent researchers with the same or similar material and experimental set-up are generally accepted, where the degree of agreement of the results depends on the field. However, this is complicated in cases when the investigated material is destroyed during measurements as it may happen in biology, for instance. In areas that use computers to perform experiments reproducibility has been discussed for some time, see, e.g., [68, 77]. Clearly, this is highly relevant for computational integer programming. It seems, however, that we have not yet reached a generally accepted agreement on the kind of reproducibility that is needed or wanted. One guideline would be that at least the same/similar conclusions could be drawn from similar experiments—a statement that must be made more precise. In fact, one goal of this section is to provide an example from computational integer programming that illustrates the difficulties of reproducibility in this area.

There are four basic options for reproducing results of an algorithmic idea published in the literature:

- use the published results;
- run an old code on old machines;
- recompile and run an old code on new machines, possibly using new libraries;
- reimplement the published method.

Clearly, every option has severe drawbacks: Assuming that a new code is run on a new computer, it is obviously very difficult to compare running times across different machines. (We do not know of any computational experiment with a new algorithm that has been run on purpose on a much older machine.) While this might suffice to get a very crude estimation, there is one major obstacle to this approach: The sample sizes especially of the older articles are too small for today's standards. As it can be seen in [6], one needs several hundred and more instances to actually be able to measure smaller performance changes. Thus, we actually need to run the published algorithm on more instances than have been published in order to derive a sound comparison. Running an old code on old or new machines comes with complicated technical problems, as we shall illustrate below. This leaves us to reimplement the published method. But this last option is often practically too time consuming. One issue is that a reimplementation does not provide any scientific merit. Moreover, it moves the burden of supplying a good implementation of an algorithmic idea to the one that is performing the comparison. If the old idea performs badly, it might be due to a bad implementation.

We do not have good solutions for these issues, but in the following we will investigate how precisely we can reproduce old results and use the particular example of reproducing results from SIP to highlight the problems involved. Clearly, to provide a historic perspective like in this paper, only the first three options above are significant.

As one goal of this article we would like to show the progress achieved in the field and thus compare results of current solvers with previously published ones. As reference we will take Table 5.1 given on page 75 of [67], where the results of solving a set of instances using SIP 1.1 are listed, see the copy in Table 1. Is it possible to reproduce those results? Our observations concerning this question are:

- The test instances used in the article are still openly available, though there are no checksums (or similar) available to ensure that they are identical with the ones used in [67].
- The SIP source code is available. While not mentioned in [67], it seems clear which precise version of the code was used.
- The program was run on a SUN Ultra Enterprise 3000 with 4 UltraSparc processors with 167 MHz, 1 GB RAM using Solaris 7. Incidentally, this machine or a quite similar one is still available at ZIB, because it is planned to be exhibited in a museum. In a few years it will be very unlikely to find such a machine without major effort.
- As it turned out, this machine has no compiler installed. Furthermore, it is neither clear from the description in the article nor the source code which compiler was used. It seems that some version of the SUN SUNWspro C/C++ compiler had been applied. Probably it would be possible to locate an old CD with this software

Table 1 SIP with default settings; taken from Table 5.1 given on page 75 of [67]

Example	B&B	Cuts	Dual bound	Primal bound	Time	Gap %
lOteams	10370	0	922	924	3600.0	0.217
air03	8	0	340160	340160	6.7	0.000
air04	1220	0	56137	56137	1532.5	0.000
air05	3588	0	26374	26374	1696.8	0.000
arkiOOl	100776	4	7579808.299	7646059.57	3600.1	0.874
bell3a	25146	0	878430.316	878430.316	45.3	0.000
bell5	337394	1	8966406.491	8966406.491	536.7	0.000
blend2	15055	5	7.598985	7.598985	122.4	0.000
cap6000	4323	2578	−2451418.742	−1236924	3604.0	49.543
dano3mip	1	0	576.2316203	–	3710.3	–
danoint	12655	0	62.94058146	70	3600.3	11.216
dcmulti	2637	0	188182	188182	14.6	0.000
dsbmip	867	0	−305.198175	−305.198175	42.7	0.000
egout	222	0	568.1007	568.1007	0.2	0.000
enigma	8002	524	0	0	24.2	0.000
fast0507	234	0	172.2530211	177	3604.8	2.756
fiber	783	372	405935.18	405935.18	16.9	0.000
fixnet6	1669	0	3983	3983	14.6	0.000
flugpl	7976	25	1201500	1201500	4.4	0.000
gen	11	20	112313.3627	112313.3627	0.3	0.000
gesa2	209525	33	25771445.96	25783761.56	3600.0	0.048
gesa2_o	264243	0	25711931.57	25823063.47	3600.0	0.432
gesa3	5297	0	27991042.65	27991042.65	97.1	0.000
gesa3_o	74472	0	27991042.65	27991042.65	1144.7	0.000
gt2	2215	5	21166	21166	3.2	0.000
harp2	23990	15966	−73944202.17	−70801289	3600.1	4.250
khb05250	2637	0	106940226	106940226	16.3	0.000
l1521av	3209	269	4722	4722	93.8	0.000
lseu	303	164	1120	1120	1.1	0.000
misc03	699	14	3360	3360	4.1	0.000
misc06	308	0	12850.86074	12850.86074	4.2	0.000
misc07	35585	0	2810	2810	378.8	0.000
mitre	1286	3865	115155	115155	1125.8	0.000
mod008	884	371	307	307	9.8	0.000
modO1O	237	3	6548	6548	5.6	0.000

Table 1 (*Continued*)

Example	B&B	Cuts	Dual bound	Primal bound	Time	Gap %
mod011	6108	0	−54558535.01	−54558535.01	2791.4	0.000
modglob	1000000	0	20652263.27	20763655.71	3495.8	0.539
noswot	1000000	179	−43	−41	2270.7	4.651
nw04	1827	0	16862	16862	732.9	0.000
p0033	77	53	3089	3089	0.1	0.000
p0201	507	136	7615	7615	5.0	0.000
p0282	1345	2308	258411	258411	38.3	0.000
p0548	1610	902	8691	8691	25.3	0.000
p2756	23151	6923	3113.257351	3141	3600.2	0.891
pk1	501934	0	11	11	1581.8	0.000
pp08a	1000000	0	5446.190476	8620	2092.7	58.276
pp08aCUTS	624198	0	6970.027419	7650	3600.0	9.756
qiu	17378	0	−132.873137	−132.873137	2326.5	0.000
qnet1	17694	12	16029.69268	16029.69268	1229.9	0.000
qnet1_o	3806	0	16029.69268	16029.69268	158.6	0.000
rentacar	105	0	30356760.98	30356760.98	53.2	0.000
rgn	2505	315	82.19999924	82.19999924	9.6	0.000
rout	200371	316	1048.991823	1079.19	3600.0	2.879
set1ch	841033	0	39920.71098	67819.5	3600.0	69.886
seymour	1947	0	406.4218572	438	3601.8	7.770
stein27	4666	0	18	18	8.0	0.000
stein45	54077	0	30	30	277.7	0.000
vpm1	1000000	0	19.5	20	1892.6	2.564
vpm2	555712	0	13.75	13.75	1368.7	0.000
Total (59)	8017878	35366			77823.6	226.547

and try to install it. Nevertheless, it is unlikely that we would be able to reproduce the exact binary.

- SIP used CPLEX 5.0 as its LP-solver, i.e., it needs the CPLEX callable library. Finding an old CD will not help, since any available license would have expired for years. In the meantime ILOG, the owner of CPLEX at that time, has been acquired by IBM. IBM is still in the possession of the source code, but is not willing to release the code, even in binary form, due to the legal effort required.

We conclude that while it might be technically possible to reproduce the binary (or something very similar) and run it on a machine similar to the original one, it would require a lot of effort to do so. Furthermore, there is no way to check whether we actually succeeded in producing the precise environment under which the tests where performed in the paper. It seems to be the only possibility to compare the new results with the old log files—which are still available—and those printed in the article. If there is any mismatch, this could be due to a hundred different reasons, like a different compiler switch for some subsystem, a different version of a system library, etc.

Another point is determinism and reproducibility regarding computer hardware. Computer hardware is not free of errors, e.g., Intel is listing several hundred errors for their CPUs in so-called *Specification Updates* on arki.intel.com. It should be noted though that most of these errors are very rare and need extremely complicated conditions to appear. Computers and in particular RAM are also prone to errors due to cosmic rays. While all this might be an issue on large scale super-computers, we can neglect it on workstation level. Another issue is that modern systems have features like NUMA, Hyperthreading, Turbo-Boost, etc. that require careful attention to ensure reproducible results.

Since we are interested in the algorithmic advances and less in the question whether the computing machinery got faster, it should suffice to compile the code on a modern machine, using a modern compiler and get comparable results apart from the timings. Tobias Achterberg, now at IBM, kindly agreed to do this. He compiled the original SIP code together with CPLEX 5.0.1 using gcc 4.0.1 on a modern Intel Xenon X5260 powered Linux computer. As it turned out, we achieved identical results regarding the number of branch-and-bound nodes for all but three instances. Trying to solve the instance *mitre* the program crashed due to a numerical error from CPLEX. The runs for the instances *lseu* and *p0033* differ. The most likely explanation is that the LP-solver returned a different optimal basis. Indeed, Intel x86 architecture CPUs do double precision (64 bit) floating point calculations internally with 80 bit precision. This can lead to different results compared to SPARC CPUs which use 64 bits throughout. Even though both conform with the IEEE 754 standard. There are further reasons which might lead to different results. These differences can be large enough to let the simplex algorithm terminate at a different vertex of the optimal face. As a consequence, SIP then generated different cuts, which in turn led to a different number of nodes. The returned objective function value was the same in all solved cases. Since the original solutions are not available anymore, we cannot check whether the solutions returned are also the same.

Different to the original runs, the time limit given to SIP was in wall-clock time. Due to changes in the operating system API, CPU time measurement seemed not to work correctly (anymore). We also conducted limited experiments concerning the CPLEX 5.0.1 MIP-solver. Here we experienced that any change of the compiler options would also change the number of branch-and-bound nodes. And again, there were problems regarding the time measurement.

Moreover, we tried to link the newly compiled SIP code to more recent versions of CPLEX. This succeeded only half-way because of changes in the CPLEX API and behaviors that could not easily be translated back to match the functionality expected by the SIP code. We found the Version 10.1 (current is 12.5) to be the best suitable one (later versions had some used functions removed). But also in 10.1 the behavior of some functions has changed. Between Versions 7 and 8, CPLEX changed the return codes given for the optimization functions. We tried to fix this in the old code, but there seems to be at least one issue for which we do not know a solution.

SIP could also run using multiple threads in a non-deterministic parallel mode. Again, we were not able to reliably reproduce the old results. The major reason is that especially in the area of multi-threaded computations the underlying computational environment changed significantly. Furthermore, in the case of non-deterministic codes where decisions are taken depending on timings, the reproduction of results on different hardware is, by construction, hardly possible at all.

One main observation is that the use of closed-source (commercial) libraries is a major roadblock for reproducibility. If a company goes out of business, the source code or binary might be lost forever. In any case it is usually difficult to get the old code. Moreover, licensing problems are critical.

We observe that it is important to save all relevant information of the experiments. The source code used for experiments should be available including all needed libraries, all necessary data and log files. It seems to be useful to also save the binaries. Table 3 below, for example, was produced using a binary made in 2003. The current trend to use dynamically linked libraries is disastrous for reproducibility. While a statically linked program might run still decades later, trying to run a dynamically linked program later on is often hopeless and even might not produce the same result due to possibly changed system libraries.

Finally, one might also question whether the originally published results were correct to begin with. In the current academic system, the review process conducted by journals usually does not include looking at the code or the experimental data. Questions similar to the ones above led to the founding of the journal *Mathematical Programming Computation* in 2008 published by the Mathematical Optimization Society, see [30]. A notable feature of the journal is that authors are encouraged to submit the code and all the data necessary to reproduce the results of the article. A special group of *Technical Editors* then recompiles the code and reproduces the results. Furthermore, the code is reviewed to assess whether it resembles the description in the paper. Initially, questions were raised whether this will always be possible. From the experience made in the last years the answer is yes. It was even possible to rerun and review a code that was designed for a BlueGene supercomputer. And in this case, as in many others, the review process helped to substantially improve the quality and usability of the code submitted. Also the review process forces the authors to explicitly state the details of all third party codes needed.

The solving of publicly available (benchmark) instances can lead to a certain competitiveness. As announced on June 28th, 2011 on the website ilk.uvt.nl/icga of the International Computer Games Association (ICGA), the four times world chess championship winner program Rybka was banned and disqualified because an investigation by [59] found the programmer guilty of plagiarizing two other programs. This also can only happen if the code is not available for reviewing.

Assume you have the source code for the program in question. Furthermore, there is a script that runs the test and so documents the settings used and the complete log files for the runs used for the publication. They have to be detailed enough to decide whether another run was similar/identical. The best situation is if the environment

Table 2 List of MIPLIB versions

Version	Name	Date	Who	Reference
1	MIPLIB	1991	Bixby, Boyd, Indovina	[25]
2	MIPLIB 2.0	1992	Bixby, Boyd, Indovina	[25]
3	MIPLIB 3.0	1996	Bixby, Ceria, McZeal, Savelsbergh	[26]
4	MIPLIB 2003	2003	Achterberg, Koch, Martin	[8]
5	MIPLIB 2010	2010	Many	[57]

that has been used for the publication is still available, i.e., the computer/operating system and the binary. Given that a static binary was used, it should be possible to reproduce the experiments easily. Therefore, a static linked binary of the program should be kept. Once the computer is not available anymore, but at least the architecture/operating system is still available, chances are relatively good that the binary will run on a more modern environment. If the architecture/operating system is no longer available because it is too old, there is a high probability that emulators are available.

If the binary is not available anymore, it gets more complicated. If the source code to all used third party libraries is available, it is possible to completely compile the program again, although maybe not by the same compiler. If the library source code is not available, there might be (new) versions of the library for the new target architecture. But APIs change over time and it may be necessary to adjust the source code of the program. Here it is important that the log files are available and detailed enough to allow a check whether the program produces similar results.

To summarize here is what you want to keep:

1. source code and makefiles,
2. run scripts and log files,
3. source code for all used non-system libraries, and
4. a static linked binary of the program or as close as you can get to it.

4 How to Measure Performance of Integer Programming Solvers

An obvious question that arises when trying to find better ways to solve integer programs is how to measure progress. The common solution is to have a publicly available and accessible library of test instances to compare algorithms and implementations against. This was started for LPs with the NETLIB by Gay [41] and then for integer programming with the MIPLIB by Bixby, Boyd, and Indovina [25]. Due to the permanent advances in algorithms and the increase in computer speed, instances that are difficult in the beginning become easy to be solved over time. Consequently, the test instance libraries have to evolve, dropping "too easy" instances and adding harder ones. For the MIPLIB this has been done five times by now, as shown in Table 2.

The first two authors have been involved in these updates: first in 2003, see [8], and in the current version in 2010, see [57]. For the first time a consensus of all major solver developers in industry and academia on the selection of the instances could be achieved. It was agreed to substantially extend the test set. The effort and investigations to produce the 5th incarnation of the MIPLIB were substantial. The reward was a comprehensive instance library that is accepted by researchers worldwide. Furthermore, it was possible to give a true snapshot of the state-of-the art in MIP-solvers.

We report on some of the issues related to benchmarking, based on the experiences from the current MIPLIB. A key issue is that people tend to like condensed information. For measuring performances, be it the speed of a computer or the publication performance of a researcher, one would like to have a single number to describe it, since single numbers are easy to compare. Sometimes this is impossible. Regarding the computation of citation scores for researchers, we refer to [12, 13] for a discussion why this is not catching the truth. For integer programs it is slightly easier, both to compute a single number and to show that this number has to be interpreted very carefully.

It is the central problem to decide which measures to compare and on which instances. One way is to select a large number of instances by some high-level argument, for instance, they should be real-world instances or from a mix of applications. A more detailed description of this can be found in [57]. The selection of the instances seemed to work quite well, since incidentally the geometric mean performance of the three top solvers on the benchmark set were nearly equal, while the maximum difference on a particular instance was a factor greater than 1,000. This answers the question what such a mean number tells you in case that you have to solve a particular class of instances: nothing. It reveals that by careful (or sloppy) selection of the instances it is quite easy to come up with a test set where one solver is 1,000 times faster than the other. Note the above characteristic makes it easy for the marketing departments to produce funny comparison numbers. Basically, all three top commercial solver vendors claim to be faster than the competition, see [36, 46, 50].

It is even more complicated to compare solvers on instances that one solver can solve in a given time and the other solver cannot. In many cases it is not possible to wait until both solvers have solved the instance, e.g., if we have a one-hour time limit and solver B would need 10,000 times as long, we would have to wait more than a year. How should these instances be counted? One possibility is to only take those instances that both solvers can solve. This is quite biased toward the weaker solver: Imagine a solver that only checks whether the zero solution is feasible on instances that have no objective function. It will be at least as fast as any other solver and there is no way to beat it. Another option is to use a time limit and then report this limit as the solution time. Due to the large differences in solution time the ratio between two solvers then becomes just a lower bound in favor of the weaker solver. By increasing the time limit, the ratio will also continue to increase, as long as one solver solves one instance more. One could also simply not use averaging times and just count the number of instances that can be solved by a particular solver in a given amount of

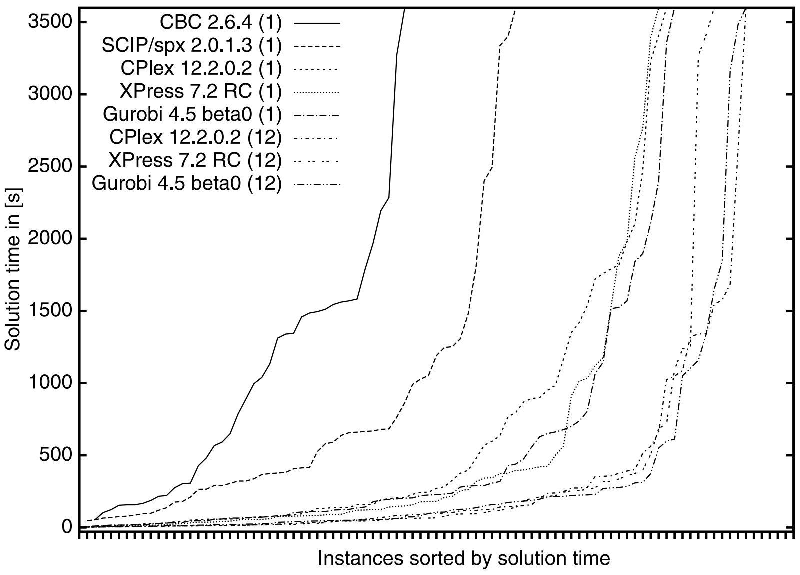


Fig. 1 MIPLIB 2010 benchmark set, distribution of solving time for 1 and 12 threads

time. The obvious problem is that again the result depends heavily on the particular time limit and very much on the selected test set.

Figure 1 shows the distribution of solution times for the 89 instances of the *benchmark* test set MIPLIB 2010. For each solver the times have been sorted in ascending order, therefore a specific position of the x-axis does not necessarily correspond to a particular instance. It can be observed that the instances fall into basically three categories: easy instances that can be solved within a few minutes, those instances that cannot be solved at all, and those in between. The criteria for the benchmark set were, among others, that at least two solvers where able to solve a particular instance within two hours and that the instance was not too easy. Therefore, compared to a larger more random set of instances, the amount of easy and unsolved instances is reduced. But we still can see that the *in-between* category is small. By speeding up the solvers using 12 threads this phenomenon becomes more pronounced, i.e., the in-between group actually shrinks. This is a phenomenon that can be observed in general. Making the computer faster will just solve those problems faster that could be solved before, but the number of instances that could not be solved at all will stay mostly the same.

5 Measuring Advances in Computational Integer Programming

As described in the introduction, the work on the MIP-solvers SIP and SCIP has now spanned more than 16 years from 1996 to 2012. In the following we give an impression on the advances of the field during this time.

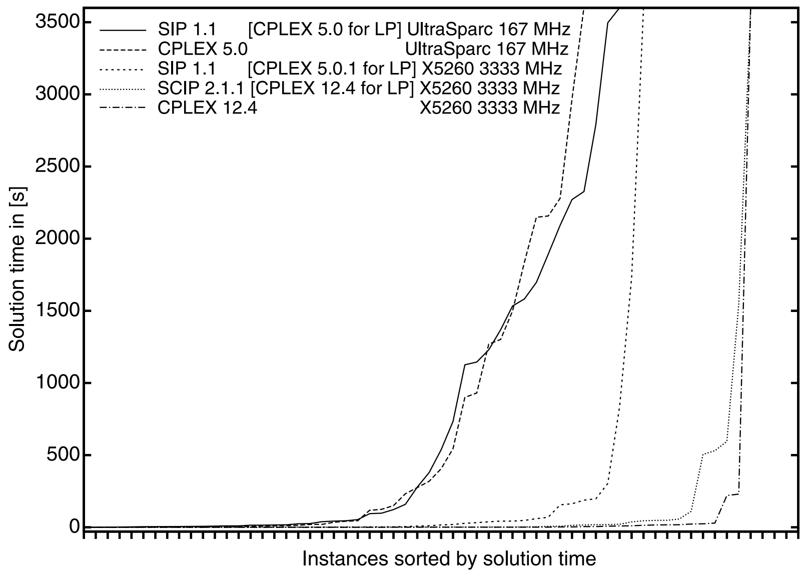


Fig. 2 1998 benchmark set, distribution of solving time for different computers and codes

First, Fig. 2 shows the time distribution for the test set used in [67] in 1998. The results on the UltraSparc CPU were taken from [67] (see Table 1). The results on the 3333 MHz Intel Xenon X5260 CPU were obtained by recompiling the original SIP code and linking to CPLEX 5.0.1 using gcc 4.0.1. The picture shows the comparison between the old and the new codes. In 1998, SIP could solve 41 out of 59 instances. Instance *mitre* that could be solved on the UltraSparc now stopped with a numerical error and was counted as a timeout with 3,600 seconds. For the X5260 we did not impose the 1,000,000 node limit used in the paper (this limit was probably set in order to control memory consumption). This affects the results for instances *10teams*, *2756*, *rout*, *pp08aCUTS*, *vmp1*, *gesa2_o*—compare Table 1.

The geometric mean of the running time using SIP on the UltraSparc was 40.2 seconds. This is now down to about 1.2 seconds, which gives us a speed-up factor of roughly 30 between the 167 MHz and the 3333 MHz computer. The clock speed ratio is about 20, but one should keep in mind that these are two totally different architectures: UltraSparc is a pure RISC architecture with in-order execution, while the Xenon is a much more complex out-of-order execution CISC CPU. Moreover, note that since reading times are included in the time measurement and due to the slight variations that are always present, fractions of seconds are not measured accurately enough to draw conclusions from it.

It should be noted that in case of instance *dano3mip*, the optimum is still unknown, and in case of instance *seymour*, while it has been solved, none of the commercial solvers is able to solve it within one hour even on 12 threads.

What can be concluded from the comparison of the curves of SIP 1.1 on the UltraSparc and the X5260 is that basically the same number of instances can be

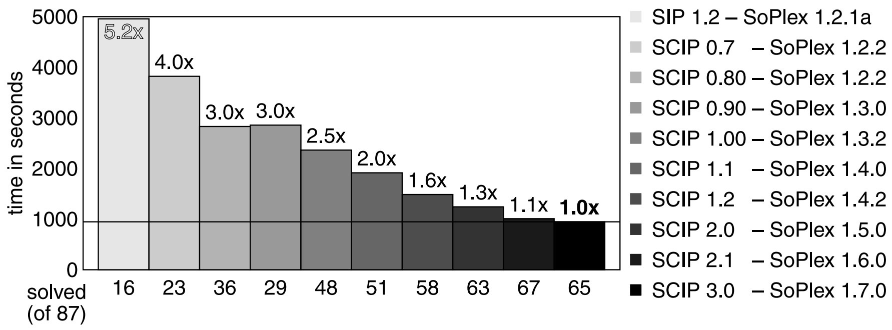


Fig. 3 MIPLIB 2010 benchmark set, performance of different SIP/SCIP versions on the same computer

solved, but it now only takes 1/30th of the time. This speed-up results in a nearly rectangular shaped curve for the faster computer for the MIPLIB 2010 case. SIP 1.1 either solves an instance in two minutes, or not at all. The same is essentially true for the new codes: They are able to solve about ten more instances, but do this very fast, while a few instances remain which have a solving time close to the time limit.

Figure 3 shows the performance over time relative to the current version of SCIP. All results were computed on an Intel Xeon X5672 CPU at 3200 MHz. Note that due to the time limit of two hours, the maximum slow-down factor compared to the latest version is five. The time for SIP 1.2 was computed using an old binary. It was able to solve only 16 out of 87 instances within the time limit and consequently was rated 4.9 times slower than the current version.

Note that it is unlikely that SIP will solve much more instances when increasing the time limit. This can be seen by Fig. 2, where SIP on a roughly 30 times faster machine can only solve 5 instances more.

As mentioned before, the above factor is a lower bound on the real slow-down, which is likely to be arbitrarily large, because there will be instances which can be solved with SCIP 2.1 and which might take practically forever using SIP 1.2. In this sense, the number of solved instances is much more important than the slow-down factor.

Figure 4 depicts a comparison between contemporary solvers conducted by Hans Mittelmann (see plato.asu.edu/ftp/milpc.html for the latest results). There are several interesting facts to note:

- Because of the one-hour time limit compared to the two-hours in Fig. 3, SCIP/SoPlex solves 10 instances less. As Fig. 1 shows, the bend in the curve is more pronounced in the commercial solvers, i.e., the number of *in-between* instances is bigger for SCIP. Therefore SCIP would benefit more from an increased time limit as compared to the commercial solvers.
- The speed-up from SCIP using CPLEX as LP-solver instead of SoPlex is just about 30 %. Given that CPLEX on pure LP benchmarks is much faster than So-

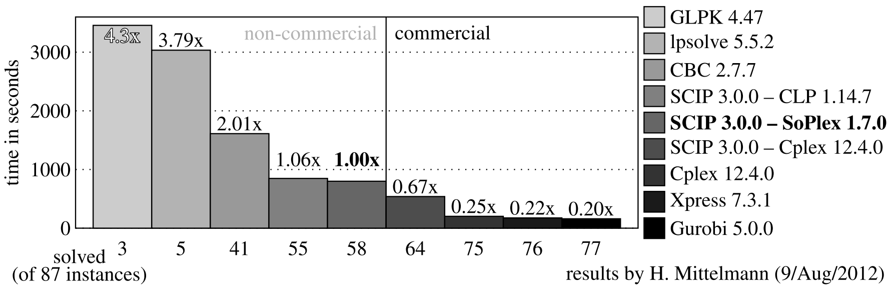


Fig. 4 MIPLIB 2010 benchmark set, comparison of MIP-solver performance relative to SCIP/SoPlex

Plex and that solving the LPs takes a considerable amount of the total running time of a MIP-solver, this is a surprisingly small difference. The reasons for this are manifold, and we are currently investigating this phenomenon which was also observed by others, e.g., Bixby [23].

- If two solvers are compared that solve a substantially different number of instances to optimality, the speed-factor is underestimated in favor of the code that solves less instances. For LPSOLVE and GLPK the factor given in the picture is meaningless as they solve only 3 or 5 instances, respectively.

We can now make an estimation on the progress in MIP-solving by SIP/SCIP since 1998. As computed above, the hardware speed-up factor is about 30. The latest version of SCIP/SoPlex is at least five times faster than SIP 1.2/SoPlex. We assume SIP 1.2 was at least as fast as SIP 1.1. Since SIP 1.1 was run with CPLEX as LP-solver, we now must also compare relatively to SCIP/CPLEX, which is about 3.5 times slower than Gurobi. This gives us an estimated lower bound on the speed-up from SIP 1.1/CPLEX on an UltraSparc to Gurobi on a modern PC of about $30 \times 5 \times 3.5 \approx 525$. Multiplying this number with the average speed-up from multithreading (approximately a factor of 3) gives an average speed-up of 1.63 times per year over 15 years or a practical doubling of the MIP-solver performance every 18 months. Remember that this is only a lower bound and that the speed-up is distributed extremely unevenly on the instances.

A more general picture can be drawn from the MIPLIB. In MIPLIB we classify an instance as *easy*, if a commercial solver on a high-end PC can solve the instance within an hour. It is classified as *hard* if it can be solved by some solver, but not by every solver, and as *unsolved* otherwise. People often report or publish if they are able to solve an instance for the first time. Table 3 lists the number of *easy*, *hard*, and *unsolved* instances in MIPLIB 2003 and 2010 over time. Note that this includes also the progress through faster computers.

One has to be cautious regarding the interpretation of these numbers, because part of the progress results from the library instances used to tune the solver algorithms. Therefore the progress is probably overstated.

Table 3 Number of instances in each class of MIPLIB 2003/2010 over time

Date	Easy	Hard	Unsolved
MIPLIB 2003			
Start 2003	22	3	35
2004	27	12	21
2005	28	13	19
2006	28	13	19
2007	31	22	7
2008	34	20	6
2009	35	19	6
2010	35	19	6
2011	41	15	4
2012	44	12	4
MIPLIB 2010			
05.2011	185	42	134
07.2011	196	33	132
08.2011	202	29	130
01.2012	202	30	129
02.2012	203	40	118
03.2012	204	41	116
04.2012	204	43	114
05.2012	206	42	113
06.2012	208	45	108
07.2012	208	47	106
08.2012	208	50	103

6 Developing Academic Integer Programming Codes

Last but not least, we want to discuss issues related to the development of academic integer programming codes. We think that such a discussion is important, since we have the impression that currently researchers may not be aware of several of these issues or might even disagree over the consequences. Our main question is

> Does it still make sense to develop integer programming codes in academia?

Before addressing this question, we point to several organizational obstacles that have to be dealt with. We dispense with licensing issues here, because this is a longer topic of its own, and rather focus on code development and publications.

Concerning *code development* we mentioned above that solver development has more and more become a team effort. As an example, a new release of SCIP requires a tremendous amount of work that we briefly mention in the following, as it might give an example for other projects. SCIP alone contains more than 400,000 lines of code and surely contains (possibly many) errors. Thus, a big part of the work concerns debugging. Bugs are either reported by the users through a web interface or are found by a significant amount of tests. These bugs might become visible, because the computed result differs from the known optimum or from a previously computed value. Bugs can also be found because one of the checkpoints (asserts) or unit-tests in SCIP is triggered. Some bugs may be due to numerical issues and thus require longer time to debug. The infrastructure of SCIP helps debugging, but in total the preparation of a release is spread over the time span of three months and the work of about four full-time developers.

There is one further aspect that seems to be relevant in this context. It is indispensable to further advance the field teamwork, and this has repercussions on how research is conducted. Like in physics and other areas, research in computational integer programming is more and more becoming team-work. Moreover, students are able to learn how a solver works and are possibly able to join commercial MIP-solver teams. (This seems to be the employee recruitment strategy in the field.) We think that all this is only possible through academic research.

With respect to *publications*, we all know that it is still hard to publish papers on computational optimization in first class journals. Basically, there is no credit for the code development work that is involved. It is a fact that still almost all codes used for publications are not publicly available. This is, actually, the biggest obstacle for reproducibility. The introduction of the journal *Mathematical Programming Computation* is a little step in the right direction, but a different way of handling papers by editors of journals is needed. It is clear that the goal should be that every code used for computations in a paper should be available for possible reproduction or even improvement. Moreover, the effort needed for code development should be taken into account.

We now come back to our initial question: Is there (still) room for academic solvers in times when the commercial solvers seem to be computationally ahead?

There are several commercial codes available that either serve their purpose as a stand-alone program or can be used as a B&C framework for individual applications through callbacks. Clearly, for many applications it suffices to just use a stand-alone MIP-solver, since this already “finishes the job” in many cases.

It is also a fact that many publications in the field use these commercial solvers as a B&C framework through callbacks. To quantify this claim, we conducted the following literature investigation. We checked all articles in the journals Mathematical Programming A and B (MPA and MPB, resp.) and Mathematical Computation (MPC) in the years from 2003 (for MPA/MPB) and 2009 (for MPC) to 2012 that perform computations and use integer programming techniques. Table 4 shows the results.

Here, we have not counted articles on mixed integer nonlinear programming. Several articles simply apply a MIP-solver, but have been counted if there is additional coding involved. Furthermore, note that double counts in the number of frameworks are possible, e.g., article [57] on MIPLIB 2010, which compares several MIP-solvers. Articles that use own implementations to handle subproblems, bounds, primal heuristics, etc. are listed in “own”. These articles often also use MIP-solvers, e.g., for solving MIP-subproblems; such articles are counted twice.

Table 4 Statistics on articles in Mathematical Programming A and B (MPA/MPB) and Mathematical Programming Computation (MPC) that use MIP-solvers/frameworks (2003–2012)

Journal	# articles
MPA	61
MPB	33
MPC	12
Total	106

Framework	# articles
CPLEX	51
XPRESS	9
COIN-OR	14
SCIP	6
ABACUS	4
MINTO	2
CONCORDE	2
Unkown	2
Own	27

It should be clear that these numbers have to be treated with care. However, it seems to be clear from these numbers that CPLEX is clearly the framework that has been used the most. Moreover, most of the articles use closed-code frameworks; for instance, we have: CPLEX + XPRESS = 60 vs. COIN-OR + SCIP + ABACUS = 24 (MINTO/CONCORDE should probably be counted as a closed code, and ABACUS was a closed code at the time of the publication of some of the articles).

Of course, this dominance of commercial codes has reasons. Some of the arguments for using commercial MIP-solvers are the following:

1. Commercial solvers are highly tuned and, thus, also promise the best performance for individual applications.
2. These solvers have to be used as a black-box. Thus, there is no possibility to tune the implementation of the framework. Consequently, researchers can concentrate on their own code, which reduces the amount of work needed.
3. Many researchers in the field have developed a code over the years, which is often based on a particular solver. Thus, changing the framework would require significant reimplementation effort.
4. One main argument against using such solvers was that licensing was problematic. However, most commercial solvers offer academic licenses today, so this is currently not an issue.

Ironically, all of these arguments can be turned around and used against using commercial MIP-solvers:

1. The promise of best performance might be wrong, and it is (almost) impossible to check whether small changes to the system or implementation might lead to

a still better performance. Moreover, using a black-box solver does not help to understand why a code is fast.
2. We do not know (completely) what happens in a black-box solver. Thus, it is scientifically questionable to have significant parts of the code in which we are not able to determine exactly what happens. More severely, the resulting code might produce wrong results, since some effects inside the black-box could not be taken into account.
3. The code basis is not really a scientific issue. Possibly, researchers would be willing to switch their code bias, if this would promise a significantly improved performance.
4. Licensing might change (see the comments in Sect. 3).

An additional argument for academic MIP-solvers is that some functionality of a black-box solver might not be available through its API. Moreover, the usage of API functions might incur unintended effects—sometimes even if no action should actually be taken; examples are refactorizations of the basis in the LP-solver or even removing old basis information or the automatic deactivation of certain algorithmic components when using callbacks.

All these arguments support the development of academic MIP-solvers or, more generally, B&C frameworks. However, it is unclear whether academic implementations for MIP-solving will be able to keep up with the performance of commercial solvers. Currently, the difference seems to be still acceptable as we have seen in this paper. Of course, this might change in the future, but predictions are always difficult. It is our belief that new ideas in the field have to be developed both in academia and industry—this worked very well in the past. Otherwise, the performance of MIP-solvers, academic and commercial, will stall.

Acknowledgements We thank all developers of SCIP/SoPlex (for a complete list, see [71]); in particular, we thank Timo Berthold and Stefan Heinz for preparing Fig. 3 and Fig. 4. Thanks to Madeline Lips for her help with the literature investigation in Sect. 6. We also thank a referee for helpful and insightful comments that improved this paper. Last but not least, we are most thankful to Martin Grötschel. Investing in a long-term project as the development of a general MIP solver is only possible if you get the time and the trust to do it. Both has been given to all three of us by Martin Grötschel in an excellent environment at ZIB.

References

1. Achterberg, T.: Conflict analysis in mixed integer programming. Discrete Optim. **4**(1), 4–20 (2007)
2. Achterberg, T.: Constraint integer programming. Ph.D. thesis, Technische Universität Berlin (2007)
3. Achterberg, T.: SCIP: solving constraint integer programs. Math. Program. Comput. **1**(1), 1–41 (2009)
4. Achterberg, T., Berthold, T.: Hybrid branching. In: van Hoeve, W.J., Hooker, J.N. (eds.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Lecture Notes in Computer Science, vol. 5547, pp. 309–311. Springer, Berlin (2009)

5. Achterberg, T., Raack, C.: The MCF-separator—detecting and exploiting multi-commodity flows in MIPs. Math. Program. Comput. **2**(2), 125–165 (2010)
6. Achterberg, T., Wunderling, R.: Mixed integer programming: analyzing 12 years of progress. In: Facets of Combinatorial Optimization: Festschrift for Martin Grötschel. Springer, Berlin (2013)
7. Achterberg, T., Koch, T., Martin, A.: Branching rules revisited. Oper. Res. Lett. **33**, 42–54 (2005)
8. Achterberg, T., Koch, T., Martin, A.: MIPLIB 2003. Oper. Res. Lett. **34**(4), 361–372 (2006)
9. Achterberg, T., Brinkmann, R., Wedler, M.: Property checking with constraint integer programming. Technical report 07-37, ZIB, Berlin (2007)
10. Achterberg, T., Berthold, T., Koch, T., Wolter, K.: Constraint integer programming: a new approach to integrate CP and MIP. In: Perron, L., Trick, M.A. (eds.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Lecture Notes in Computer Science, vol. 5015, pp. 6–20. Springer, Berlin (2008)
11. Achterberg, T., Heinz, S., Koch, T.: Counting solutions of integer programs using unrestricted subtree detection. In: Perron, L., Trick, M.A. (eds.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Lecture Notes in Computer Science, vol. 5015, pp. 278–282. Springer, Berlin (2008)
12. Adler, R., Ewing, J., Taylor, P.: Citation statistics. Technical report, IMU, ICIAM, IMS (2008). www.mathunion.org/fileadmin/IMU/Report/CitationStatistics.pdf
13. Adler, R., Ewing, J., Taylor, P.: Citation statistics. Not. Am. Math. Soc. **55**(8), 968–969 (2008)
14. Applegate, D., Bixby, R.E., Chvátal, V., Cook, W.: Finding cuts in the TSP. Technical report 95-05, DIMACS (1995)
15. Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton University Press, Princeton (2006)
16. Armbruster, M., Fügenschuh, M., Helmberg, C., Martin, A.: LP and SDP branch-and-cut algorithms for the minimum graph bisection problem: a computational comparison. Math. Program. Comput. **4**(3), 275–306 (2012)
17. Atamtürk, A., Savelsbergh, M.: Integer-programming software systems. Ann. Oper. Res. **140**, 67–124 (2005)
18. Balas, E., Ceria, S., Cornuéjols, G., Natraj, N.: Gomory cuts revisited. Oper. Res. Lett. **19**, 1–9 (1996)
19. Benichou, M., Gauthier, J.M., Girodet, P., Hentges, G., Ribiere, G., Vincent, O.: Experiments in mixed-integer programming. Math. Program. **1**, 76–94 (1971)
20. Berthold, T., Heinz, S., Pfetsch, M.E.: Nonlinear pseudo-boolean optimization: relaxation or propagation? In: Kullmann, O. (ed.) Theory and Applications of Satisfiability Testing—SAT 2009. Lecture Notes in Computer Science, vol. 5584, pp. 441–446. Springer, Berlin (2009)
21. Berthold, T., Heinz, S., Pfetsch, M.E., Vigerske, S.: Large neighborhood search beyond MIP. In: Gaspero, L.D., Schaerf, A., Stützle, T. (eds.) Proceedings of the 9th Metaheuristics International Conference (MIC 2011), pp. 51–60 (2011)
22. Bixby, R.E.: Solving real-world linear programs: a decade and more of progress. Oper. Res. **50**(1), 3–15 (2002)
23. Bixby, R.E.: Personal communication (2012)
24. Bixby, R.E., Martin, A.: Parallelizing the dual simplex method. INFORMS J. Comput. **12**, 45–56 (2000)
25. Bixby, R.E., Boyd, E.A., Indovina, R.R.: MIPLIB: a test set of mixed integer programming problems. SIAM News **25**, 16 (1992)
26. Bixby, R.E., Ceria, S., McZeal, C., Savelsbergh, M.: An updated mixed integer programming library: MIPLIB 3.0. Optima **58**, 12–15 (1998)
27. Bley, A., Gleixner, A., Koch, T., Vigerske, S.: Comparing MIQCP solvers to a specialised algorithm for mine production scheduling. In: Bock, H.G., Phu, H.X., Rannacher, R., Schlöder, J.P. (eds.) Modeling, Simulation and Optimization of Complex Processes: Proceedings of the Fourth International Conference on High Performance Scientific Computing, March 2–6, 2009, Hanoi, Vietnam pp. 25–40. Springer, Berlin (2009)

28. Borndörfer, R.: Aspects of set packing, partitioning, and covering. Ph.D. thesis, Technische Universität Berlin (1998)
29. Borndörfer, R., Ferreira, C.E., Martin, A.: Decomposing matrices into blocks. SIAM J. Optim. **9**, 236–269 (1998)
30. Cook, W., Koch, T.: Mathematical programming computation: a new MPS journal. Optima **78**, 1, 7, 8, 11 (2008)
31. Crowder, H., Johnson, E.L., Padberg, M.W.: Solving large-scale zero-one linear programming problems. Oper. Res. **31**(5), 803–834 (1983)
32. Dantzig, G.: Linear Programming and Extensions. Princeton University Press, Princeton (1963)
33. Dantzig, G., Fulkerson, R., Johnson, S.: Solution of a large-scale traveling-salesman problem. Oper. Res. **2**, 393–410 (1954)
34. Eckstein, J., Boduroğlu, I.I., Polymenakos, L.C., Goldfarb, D.: Data-parallel implementations of dense simplex methods on the connection machine CM-2. ORSA J. Comput. **7**(4), 402–416 (1995)
35. Ferreira, C.E., Martin, A., Weismantel, R.: Solving multiple knapsack problems by cutting planes. SIAM J. Optim. **6**(3), 858–877 (1996)
36. FICO: www.fico.com/en/FIResourcesLibrary/Xpress_7.2_Benchmarking_2773FS.pdf (2012). Accessed May 2012
37. Forrest, J.J., Goldfarb, D.: Steepest-edge simplex algorithms for linear programming. Math. Program. **57**, 341–374 (1992)
38. Forrest, J., Tomlin, J.: Updated triangular factors of the basis of maintain sparsity in the product form simplex method. Math. Program. **2**, 263–278 (1972)
39. Fourer, R.: Linear programming—software survey. OR/MS Today **38**(3) (2011)
40. Gamrath, G., Lübbecke, M.: Experiments with a generic Dantzig-Wolfe decomposition for integer programs. In: Festa, P. (ed.) Experimental Algorithms. Lecture Notes in Computer Science, vol. 6049, pp. 239–252. Springer, Berlin (2010)
41. Gay, M.: Electronic mail distribution of linear programming test problems. Math. Program. Soc. COAL Bull. **13**, 10–12 (1985). www.netlib.org/netlib/lp
42. Greenberg, H.J.: Computational testing: why, how and how much. ORSA J. Comput. **2**(1), 94–96 (1990)
43. Grötschel, M., Jünger, M., Reinelt, G.: A cutting plane algorithm for the linear ordering problem. Oper. Res. **32**(6), 1195–1220 (1984)
44. Grötschel, M., Monma, C.L., Stoer, M.: Polyhedral and computational investigations for designing communication networks with high survivability requirements. Oper. Res. **43**(6), 1012–1024 (1995)
45. Grötschel, M., Martin, A., Weismantel, R.: Packing Steiner trees: a cutting plane algorithm and computational results. Math. Program., Ser. A **72**(2), 125–145 (1996)
46. Gurobi Inc.: www.gurobi.com/products/gurobi-optimizer/prior-versions (2012). Accessed May 2012
47. Harris, P.M.J.: Pivot selection methods of the DEVEX LP code. Math. Program. **5**, 1–28 (1973)
48. Heinz, S., Sachenbacher, M.: Using model counting to find optimal distinguishing tests. In: van Hoeve, W.J., Hooker, J.N. (eds.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Lecture Notes in Computer Science, vol. 5547, pp. 117–131. Springer, Berlin (2009)
49. Hooker, J.N.: Needed: an empirical science of algorithms. Oper. Res. **42**, 201–212 (1994)
50. IBM—CPLEX: www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/cplex-optimizer/cplex-performance/ (2012). Accessed May 2012
51. ILOG CPLEX: Reference Manual (1997). www.cplex.com
52. Januschowski, T., Pfetsch, M.E.: Branch-cut-and-propagate for the maximum k-colorable subgraph problem with symmetry. In: Achterberg, T., Beck, J.C. (eds.) Proceedings of the 8th International Conference, CPAIOR 2011. Lecture Notes in Computer Science, vol. 6697, pp. 99–116. Springer, Berlin (2011)

53. Kaibel, V., Peinhardt, M., Pfetsch, M.E.: Orbitopal fixing. Discrete Optim. **8**(4), 595–610 (2011)
54. Koch, T.: ZIMPL user guide. Technical report 01-20, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Berlin (2001)
55. Koch, T.: Rapid mathematical programming. Ph.D. thesis, Technische Universität Berlin (2004)
56. Koch, T., Martin, A.: Solving Steiner tree problems in graphs to optimality. Networks **32**, 207–232 (1998)
57. Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R.E., Danna, E., Gamrath, G., Gleixner, A.M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D.E., Wolter, K.: MIPLIB 2010. Math. Program. Comput. **3**, 103–163 (2011)
58. Kostina, E.: The long step rule in the bounded-variable dual simplex method: numerical experiments. Math. Methods Oper. Res. **55**, 413–429 (2002)
59. Lefler, M., Hyatt, R., Williamson, H.: ICGA Panel Members: Rybka investigation and summary of findings for the ICGA. Technical report, International Computer Games Association (2011). ilk.uvt.nl/icga/investigation/Rybka_disqualified_and_banned_by_ICGA.rar
60. Lembke, C.E.: The dual method of solving the linear programming problem. Nav. Res. Logist. Q. **1**, 36–47 (1954)
61. Linderoth, J.T., Lodi, A.: MILP software. In: Cochran, J. (ed.) Wiley Encyclopedia of Operations Research and Management Science, vol. 5, pp. 3239–3248. Wiley, New York (2011)
62. Linderoth, J.T., Ralphs, T.K.: Noncommercial software for mixed-integer linear programming. In: Karlof, J. (ed.) Integer Programming: Theory and Practice. Operations Research Series, pp. 253–303. CRC Press, Boca Raton (2005)
63. Linderoth, J.T., Savelsbergh, M.W.P.: A computational study of search strategies for mixed integer programming. INFORMS J. Comput. **11**, 173–187 (1999)
64. Lodi, A.: MIP computation. In: Jünger, M., Liebling, T., Naddef, D., Nemhauser, G., Pulleyblank, W., Reinelt, G., Rinaldi, G., Wolsey, L. (eds.) 50 Years of Integer Programming 1958–2008, pp. 619–645. Springer, Berlin (2009)
65. Luce, R., Tebbens, J.D., Liesen, J., Nabben, R., Grötschel, M., Koch, T., Schenk, O.: On the factorization of simplex basis matrices. Technical report 09-24, Zuse Institute Berlin, Berlin (2009)
66. Mars, S., Schewe, L.: SDP-package for SCIP. Technical report, TU Darmstadt (2012)
67. Martin, A.: Integer programs with block structure. Habilitation thesis, Technische Universität Berlin (1998)
68. McGeoch, C.C.: A Guide to Experimental Algorithmics. Cambridge University Press, Cambridge (2012)
69. Mittelmann, H.: Decision tree for optimization software: benchmarks for optimization software (2003). plato.asu.edu/bench.html
70. Padberg, M., Rinaldi, G.: A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM Rev. **33**, 60–100 (1991)
71. SCIP: Solving constraint integer programs. scip.zib.de
72. Suhl, L.M., Suhl, U.H.: Computing sparse LU factorizations for large-scale linear programming bases. ORSA J. Comput. **2**(4), 325–335 (1990)
73. Suhl, L.M., Suhl, U.H.: A fast LU update for linear programming. Ann. Oper. Res. **43**(1–4), 33–47 (1993)
74. Vigerske, S.: Decomposition of multistage stochastic programs and a constraint integer programming approach to mixed-integer nonlinear programming. Ph.D. thesis, Humboldt-Universität zu Berlin (2012)
75. Wessäly, R.: Dimensioning survivable capacitated networks. Ph.D. thesis, Technische Universität Berlin (2000)
76. Wunderling, R.: Paralleler und objektorientierter Simplex-Algorithmus. Ph.D. thesis, Technische Universität Berlin (1996)
77. Yale Law School Roundtable on Data and Code Sharing: School Roundtable on Data and Code Sharing: Reproducible research. Comput. Sci. Eng. **12**, 8–13 (2010)