COMBINATORIAL OPTIMIZATION

NATO Science for Peace and Security Series

This Series presents the results of scientific meetings supported under the NATO Programme: Science for Peace and Security (SPS).

The NATO SPS Programme supports meetings in the following Key Priority areas: (1) Defence Against Terrorism; (2) Countering other Threats to Security and (3) NATO, Partner and Mediterranean Dialogue Country Priorities. The types of meeting supported are generally "Advanced Study Institutes" and "Advanced Research Workshops". The NATO SPS Series collects together the results of these meetings. The meetings are co-organized by scientists from NATO countries and scientists from NATO's "Partner" or "Mediterranean Dialogue" countries. The observations and recommendations made at the meetings, as well as the contents of the volumes in the Series, reflect those of participants and contributors only; they should not necessarily be regarded as reflecting NATO views or policy.

Advanced Study Institutes (ASI) are high-level tutorial courses to convey the latest developments in a subject to an advanced-level audience.

Advanced Research Workshops (ARW) are expert meetings where an intense but informal exchange of views at the frontiers of a subject aims at identifying directions for future action.

Following a transformation of the programme in 2006 the Series has been re-named and re-organised. Recent volumes on topics not related to security, which result from meetings supported under the programme earlier, may be found in the NATO Science Series.

The Series is published by IOS Press, Amsterdam, and Springer Science and Business Media, Dordrecht, in conjunction with the NATO Emerging Security Challenges Division.

Sub-Series

A.	Chemistry and Biology	Springer Science and Business Media
B.	Physics and Biophysics	Springer Science and Business Media
C.	Environmental Security	Springer Science and Business Media
D.	Information and Communication Security	IOS Press
E.	Human and Societal Dynamics	IOS Press

http://www.nato.int/science
http://www.springer.com
http://www.iospress.nl

Sub-Series D: Information and Communication Security – Vol. 31

ISSN 1874-6268 (print)
ISSN 1879-8292 (online)

Combinatorial Optimization

Methods and Applications

Edited by

Vašek Chvátal
Canada Research Chair in Combinatorial Optimization
Concordia University
Montreal, Canada

IOS Press
Amsterdam • Berlin • Tokyo • Washington, DC

Published in cooperation with NATO Emerging Security Challenges Division

Proceedings of the NATO Advanced Study Institute Meeting on
Combinatorial Optimization: Methods and Applications
Montreal, Quebec, Canada
19-30 June 2006

ISBN 978-1-60750-717-8 (print)
ISBN 978-1-60750-718-5 (online)
Library of Congress Control Number: 2011925297

Publisher
IOS Press BV
Nieuwe Hemweg 6B
1013 BG Amsterdam
Netherlands
fax: +31 20 687 0019
e-mail: order@iospress.nl

Distributor in the USA and Canada
IOS Press, Inc.
4502 Rachael Manor Drive
Fairfax, VA 22032
USA
fax: +1 703 323 3668
e-mail: iosbooks@iospress.com

Combinatorial Optimization: Methods and Applications
V. Chvátal (Ed.)
IOS Press, 2011

Preface

This book consists of six articles that came out of the following eight lecture series given at the NATO Advanced Study Institute (ASI) "Combinatorial Optimization: Methods and Applications" held at Université de Montréal on June 19 – 30, 2006:

Gérard Cornuéjols
(Carnegie Mellon University, USA)
Mixed integer programming

Sanjeeb Dash
(IBM Thomas J. Watson Research Center, USA)
Mixed integer rounding cuts and cyclic group polyhedra

Yury Kochetov
(Sobolev Institute of Mathematics, Russia)
Facility location problems. Discrete models and local search methods

Bernhard Korte; substituted by Stephan Held after the first lecture
(Forschungsinstitut für Diskrete Mathematik, Germany)
Making chips faster

Gleb Koshevoy
(Russian Academy of Sciences, Russia)
Discrete convexity and its applications in combinatorics

Shmuel Onn
(Technion - Israel Institute of Technology, Israel)
Convex discrete optimization

Dieter Rautenbach
(Institut für Optimierung und Operations Research Universität Ulm, Ulm, Germany)
Optimization and timing in VLSI design

Jens Vygen
(Forschungsinstitut für Diskrete Mathematik, Germany)
Combinatorial optimization in VLSI placement and routing

The six articles are ordered alphabetically by the last name of their first author. The article by Nannicini et al. has been written in 2010 as a follow-up to the lectures given by Cornuéjols at the ASI. The article by Onn is a reprint of his monograph published by Springer in *Encyclopedia of Optimization 2009*, which follows the outline of his lectures given at the ASI. The remaining four articles also follow the lectures given at the ASI, but they have been updated in 2010.

This page intentionally left blank

Acknowledgments

The NATO Advanced Study Institute from which this book originates was co-directed by my long-time friend and collaborator Najiba Sbihi of Ecole Mohammadia d'Ingénieurs in Rabat. It was a part of the Theme Semester on Combinatorial Optimization at the Centre de recherches mathématiques in Montreal. François Lalonde, the Centre's Director, guided our first steps in his energizing and amiable way. We were fortunate that he assigned the task of coordinating the logistics of the Institute to Diane Bélanger from the Département de mathématiques et de statistique: on her own initiative, she did much more than her job description demanded and she did it extremely well. She became the third co-director in every respect except for her title. Dania El-Khechen and Perouz Taslakian created a superb guide to Montreal for the participants. After the Institute had come to its end, Sakina Benhima took charge of the closing administrative duties.

Odile Marcotte, a Deputy Director of the Centre, helped me with editing one of the contributions to this book and my graduate student Mark Goldsmith helped me with two. André Montpetit, the Technical Editor at the Centre, prepared the entire manuscript for transfer to IOS Press. His efficient, thoughtful, and always prompt assistance was a delight.

I thank all the lecturers and all the other people named here for their help and for the pleasure I had working with them. I also thank NATO's *Security Through Science Programme* of and the Centre de recherches mathématiques for their generous support of the Institute.

Vašek Chvátal

This page intentionally left blank

The NATO Advanced Study Institute

The NATO Advanced Study Institute "Combinatorial Optimization: Methods and Applications" was held at Université de Montréal as its forty-fifth Séminaire de Mathématiques Supérieures on June 19 – 30, 2006. It was co-directed by Vašek Chvátal (Concordia University, Montreal, Canada) and Najiba Sbihi (Ecole Mohammadia d'Ingénieurs, Rabat, Morocco) with the help of its coordinator Diane Bélanger (Université de Montréal). Its lectures were divided into the following twelve series:

The week of Monday June 19 to Friday June 23:

Fritz Eisenbrand
(Max-Planck-Institut für Informatik, Germany)
My favorite open problems in integer programming

Shmuel Onn
(Technion - Israel Institute of Technology, Israel)
Convex discrete optimization

Sanjeeb Dash
(IBM Thomas J. Watson Research Center, USA)
Mixed integer rounding cuts and cyclic group polyhedra

Dieter Rautenbach
(Institut für Optimierung und Operations Research Universität Ulm, Ulm, Germany)
Optimization and timing in VLSI design

Najiba Sbihi
(Ecole Mohammadia d'Ingénieurs, Morocco)
Mathematics of supply chain management

Gleb Koshevoy
(Russian Academy of Sciences, Russia)
Discrete convexity and its applications in combinatorics

The week of Monday June 26 to Friday June 30:

Yury Kochetov
(Sobolev Institute of Mathematics, Russia)
Facility location problems. Discrete models and local search methods

Michel Goemans
(Massachusetts Institute of Technology, USA)
Approximation Algorithms

Gérard Cornuéjols
(Carnegie Mellon University, USA)
Mixed integer programming

Bernhard Korte; substituted by Stephan Held after the first lecture
(Forschungsinstitut für Diskrete Mathematik, Germany)
Making chips faster

Jens Vygen
(Forschungsinstitut für Diskrete Mathematik, Germany)
Combinatorial optimization in VLSI placement and routing

Lisa Fleischer
(IBM Thomas J. Watson Research Center, USA)
Generalized congestion games

Each of these series consisted of five one-hour lectures (with the exception of Sbihi's, who followed the tradition of a co-director giving only four one-hour lectures). They were attended by some sixty students, at the graduate student and postdoctoral level, from Algeria, Armenia, Belgium, Bulgaria, Canada, Czech Republic, France, Germany, Israel, Morocco, Russian Federation, Slovak Republic, Turkey, Ukraine, and United States.

Contents

This page intentionally left blank

Combinatorial Optimization: Methods and Applications
V. Chvátal (Ed.)
IOS Press, 2011

doi:10.3233/978-1-60750-718-5-1

Mixed Integer Rounding Cuts and Master Group Polyhedra

Sanjeeb DASH [1]
IBM T.J.Watson Research Center, New York, USA

Abstract. We survey recent research on mixed-integer rounding (MIR) inequalities and a generalization, namely the two-step MIR inequalities defined by Dash and Günlük (2006). We discuss the master cyclic group polyhedron of Gomory (1969) and discuss how other subadditive inequalities, similar to MIR inequalities, can be derived from this polyhedron. Recent numerical experiments have shed much light on the strength of MIR inequalities and the closely related Gomory mixed-integer cuts, especially for the MIP instances in the MIPLIB 3.0 library, and we discuss these experiments and their outcomes. Balas and Saxena (2007), and independently, Dash, Günlük and Lodi (2007), study the strength of the MIR closure of MIPLIB instances, and we explain their approach and results here. We also give a short proof of the well-known fact that the MIR closure of a polyhedral set is a polyhedron. Finally, we conclude with a survey of the complexity of cutting-plane proofs which use MIR inequalities.

This survey is based on a series of 5 lectures presented at the Séminaire de mathématiques supérieures, of the NATO Advanced Studies Institute, held in the Université de Montréal, from June 19 – 30, 2006.

Keywords. Integer programming, cutting planes, subadditive inequalities, group polyhedra, proof complexity

1. Introduction

Over the last 10 – 15 years, cutting planes have emerged as a vital tool in mixed-integer programming. We call a linear inequality satisfied by all integer points in a polyhedron P a *cutting plane* (or *cut*) for P. Most commercial software which solve mixed-integer programs, such as ILOG-CPLEX [1] or XPRESS-MP [2], use sophisticated algorithms to find cutting planes and combine them with linear programming based branch-and-bound in a *branch-and-cut* system. There is a lot of literature on problem-specific cutting planes; in the context of the traveling salesman problem (TSP) for example, comb inequalities are very useful in solving TSP instances to optimality. In this survey, we will mainly discuss cutting planes for general (mixed) integer programs. That is, we will not assume any underlying combinatorial structure. In such cases, the *mixed-integer rounding* (MIR) inequalities (or MIR cut) and the closely related *Gomory mixed-integer* (GMI) cuts form the most important class of cutting planes.

[1] IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA; E-mail: sanjeebd@us.ibm.com.

The GMI cut was derived by Gomory in 1960 [3]. After some initial limited experimentation, these cuts were hardly used to solve mixed-integer programs for a long time. One exception is the paper of Geoffrion and Graves [4] in 1974 who described a successful combination of GMI cuts with LP based branch-and-bound to solve MIPs. In particular, they used a "hybrid branch-and-bound/cutting-plane approach" where "the cuts employed are the original mixed integer cuts proposed by Gomory in 1960, and are applied to each node problem in order to strengthen the LP bounds." They used the above ideas to solve a pure 0-1 integer program with several hundred binary variables. Some of the implementation ideas in this paper anticipate the (independent) work of Balas, Ceria, Cornuéjols and Natraj [5], who performed a systematic study of GMI cuts and popularized them as a tool for general mixed-integer programs in the 1990s. See [6] for additional historical information. Subsequent computational studies [7] confirmed the usefulness of the GMI cut for practical mixed-integer programs.

Nemhauser and Wolsey [8, p. 244] introduced mixed-integer rounding inequalities, or cutting planes that can be produced by what they call the MIR *procedure*. These authors later [9] strengthened and redefined the MIR procedure and the resulting inequality; see [10] for a discussion on the development of MIR inequalities. They also showed that the MIR inequality, the GMI cut, and the *split cut* were equivalent. Split cuts were defined by Cook, Kannan and Schrijver [11], and are a special case of the *disjunctive cuts* introduced by Balas [12]. Marchand and Wolsey [13] later showed that many cuts in the literature are special cases of the MIR cut. They also computationally established the usefulness of the MIR cut; the MIR cuts in their experiments are different from the GMI cuts used in [5, 7]. The definition of the MIR cut we use in this paper is equivalent to the one in [9], though our presentation is based on [14]. By the late 1990s, the importance of the GMI cut and the MIR cut in solving mixed-integer programs was clear, and these cuts have become a standard feature of major commercial MIP solvers.

Gomory [15] introduced group relaxations of integer programs. Some computational studies of group relaxations can be found in White [16], Shapiro [17], and Gorry, Northup and Shapiro [18]. The latter paper contains a study of a group relaxation based branch-and-bound algorithm. After some initial work on this topic, by the mid-1970s, group relaxations were not viewed as a central technique to solve integer programs (see [19]). In his important work on polyhedral properties of group relaxations, Gomory [20] studied *corner polyhedra* (the convex hull of solutions of group relaxations) and the related cyclic group polyhedra. He highlighted the role of subadditive functions in obtaining cutting planes for corner polyhedra and for MIPs, and derived the GMI cut for pure integer programs from a facet of the master cyclic group polyhedron (MCGP). GMI cuts for mixed-integer programs can similarly be derived from the mixed-integer extension of the MCGP; see Gomory and Johnson [21]. The MCGP has a rich polyhedral structure and it is natural to ask if it has other facets that would lead to useful cutting planes for mixed-integer programs. We call such cuts *group cuts*. Following recent work on this topic by Gomory, Johnson and coauthors in [22] and [23], cyclic group polyhedra and group cuts are an active area of study; see [10, 24–29].

An active research topic is the computational study of the strength of *closures* (or *elementary closures*) of different families of cutting planes. For a family of cutting planes $\mathcal{F}$, and a polyhedron P, the closure is the set of all points in P satisfying the cutting planes in $\mathcal{F}$. Chvátal [30] rediscovered Gomory cuts (we call them Gomory – Chvátal cuts (GC) in this paper) and initiated the study of the *Chvátal closure*, i.e., the closure

with respect to Gomory – Chvátal cuts. A computational study of the Chvátal closure can be found in a recent paper by Fischetti and Lodi [31]. They framed the problem of separating a point from the Chvátal closure, or equivalently, of finding a violated GC cut, as an MIP (such an MIP can also be found in Bockmayr and Eisenbrand [32]). See [33] for related work on verifying that an inequality has Chvátal rank 1 or 2. Fischetti and Lodi used a standard MIP solver to find violated GC cuts and approximately optimized over the Chvátal closure of MIP instances in the MIPLIB 3.0 problem set. This procedure is computationally intensive, but yields strong bounds for many of the pure integer instances in MIPLIB, in contrast to using Gomory cuts derived from optimal simplex tableau rows, as attempted in earlier papers on this topic. Independently, Bonami and Minoux [34] approximately optimized over the closure with respect to lift-and-project cuts for 0-1 MIP instances from MIPLIB 3.0. Recently Bonami et al. [35] optimized over the closure of *projected GC cuts* for the mixed-integer instances from MIPLIB using an MIP solver to find violated cuts. GC cuts and lift-and-project cuts are both special cases of split/MIR cuts. Motivated by the above work, Dash, Günlük and Lodi [10] and independently, Balas and Saxena [36] approximately optimize over the MIR closure of MIP instances, and show that a large fraction of the integrality gap can be closed in this way for MIPLIB 3.0 instances. However, it takes (in these papers) much more time to optimize over the closures above than to solve the original MIP (with few exceptions, as noted in [31, 36]). This is not surprising as it is NP-hard to separate an arbitrary point from the Chvátal closure [37], or from the split closure [38] (though one can solve an LP to find a violated lift-and-project cut).

Besides their usefulness in practice, cutting planes have some very interesting theoretical properties. Many statements with a combinatorial flavor, such as "the maximum size clique in a specific graph G has fewer than 10 nodes," can be proved by a *Gomory – Chvátal cutting-plane proof*, or a sequence of GC cuts, where each one is obtained from some original constraints modeling the combinatorial problem and the previous GC cuts. This is a consequence of Gomory's result [39] on the finite termination of his cutting plane algorithm for solving integer programs. Pudlák [40] showed that for 0-1 integer programs without integer solutions, GC cutting-plane proofs certifying their infeasibility have exponentially many cuts in the worst case. A similar result for MIR cutting plane proofs was recently proved in [41].

In this paper, we survey recent work on MIR cuts and cyclic group polyhedra. We start off with a simple extension of MIR cuts, namely the *two-step* MIR cuts in Dash and Günlük [25], a parametric family of group cuts. We then discuss fundamental properties of Gomory's cyclic group polyhedra, including Gomory's characterization of the convex hull of nontrivial facets of these polyhedra in Section 3. We further discuss some important facets of these polyhedra, and their relationship to subadditive functions, and to valid inequalities for mixed-integer programs. We move on (in Section 4) to a discussion of the MIR closure of a polyhedral set, and present a short proof that the MIR closure is a polyhedron, based on (but different from) a recent proof in Dash, Günlük, and Lodi [10]. This result follows from the result of Cook, Kannan and Schrijver [11] that the split closure of a polyhedral set is a polyhedron. We also present the MIP model from [10] to find violated MIR cuts. In Section 5, we discuss computational work on using cyclic group polyhedra to solve integer programs. We discuss recent studies on the computational effectiveness of two-step MIR cuts in Dash, Günlük and Goycoolea [24], and *interpolated group cuts* in Fischetti and Saturni [42]. We also discuss computational studies of the

strength of the MIR closure. Finally, in Section 6, we discuss a recent result from [41] that shows that an MIR cutting plane proof certifying that an infeasible integer program has no integral solutions can have exponentially many cuts in the worst case.

2. The MIR Inequality and Extensions

Consider the set

$$P = \{v \in \mathbb{R}^I, x \in \mathbb{Z}^n : Cv + Ax = d,\ v, x \geq 0,\ x \text{ integer}\}.$$

By a mixed-integer program, we mean the problem of minimizing a linear function $g^T v + h^T x$ subject to $(v, x) \in P$, where $g \in \mathbb{R}^I$, $h \in \mathbb{R}^n$. Throughout the paper, we represent matrices by A, B, C, G or H, and polyhedral sets by P or Q (with superscripts when appropriate). We denote the continuous relaxation of P by P^{LP}, and the convex hull of P by conv(P). We will study valid inequalities for P, or (equivalently) cutting planes for P^{LP}. We assume that all numerical data is rational. Let

$$Q = \left\{v \in \mathbb{R}^{|J|}, x \in \mathbb{Z}^{|I|} : \sum_{j \in J} c_j v_j + \sum_{i \in I} a_i x_i = b,\ v, x \geq 0\right\},$$

where the equation defining Q is obtained as a linear combination of the equations defining P, and I and J are index sets of the integer/noninteger variables. In other words, $a = \lambda^T A$, $c = \lambda^T C$ and $b = \lambda^T d$ for some real vector λ of appropriate dimension. As $P \subseteq Q$, valid inequalities for Q yield valid inequalities for P. For a valid inequality derived in this way, we will refer to $\sum_{j \in J} c_j v_j + \sum_{i \in I} a_i x_i = b$ as its *base* equation. Finally, for a number v, let $\hat{v}$ stand for $v - \lfloor v \rfloor$, the fractional part of v.

2.1. *The* MIR *Inequality*

There are many equivalent ways of defining the MIR inequality; see [10]. The presentation here is based on the presentation in the book [14] by Wolsey. Define

$$Q^1 = \{v \in \mathbb{R}, z \in \mathbb{Z} : v + z \geq b,\ v \geq 0\}.$$

The *basic mixed-integer* inequality, defined in [14] as

$$v + \hat{b} z \geq \hat{b} \lceil b \rceil, \tag{1}$$

is valid and facet-defining for Q^1.

Lemma 1. $\text{conv}(Q^1) = \{v, z \in \mathbb{R} : v + z \geq b,\ v + \hat{b} z \geq \hat{b} \lceil b \rceil,\ v \geq 0\}$.

Proof. The result is trivial if $\hat{b} = 0$; so assume $\hat{b} \neq 0$. Let Q' be the set on the right-hand side of the equation in Lemma 1. Let $(v, z) \in Q^1$. If $z \geq \lceil b \rceil$, then $v \geq 0$ implies that $v + \hat{b} z \geq \hat{b} \lceil b \rceil$. If $z \leq \lfloor b \rfloor$, then $v + z \geq b$ implies that

$$v \geq \hat{b} + \lfloor b \rfloor - z \geq \hat{b} + \hat{b}(\lfloor b \rfloor - z) = \hat{b}(\lceil b \rceil - z).$$

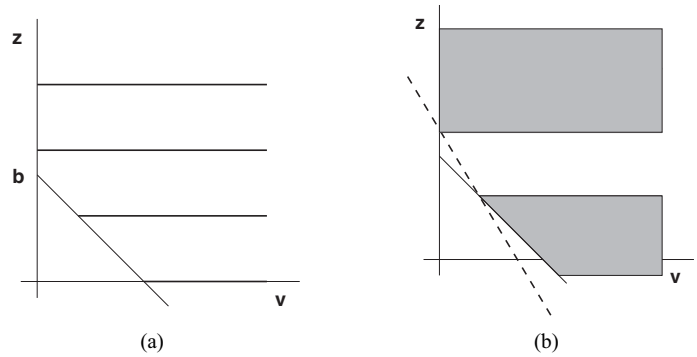

Figure 1. The basic mixed-integer inequality

Therefore, Eq. (1) is a valid inequality for Q^1 and $\text{conv}(Q^1) \subseteq Q'$. On the other hand, the extreme points of Q' are $(0, \lceil b \rceil)$ and $(\hat{b}, \lfloor b \rfloor)$, given by the intersections of the first and third inequalities with the second inequality. Both these points lie in Q^1, and thus $Q' \subseteq \text{conv}(Q^1)$. □

In Figure 1(a), we depict the points in Q^1 by horizontal lines. In Figure 1(b), the half-plane above the dashed line represents Eq. (1), and contains the shaded regions which are $Q^1 \cap \{z \leq \lfloor b \rfloor\}$ and $Q^1 \cap \{z \geq \lceil b \rceil\}$. If b is not integral (i.e., $\hat{b} \neq 0$), then the point $(\overline{v}, \overline{z}) = (0, b)$ violates the basic mixed-integer inequality.

An approach to deriving valid inequalities for more general sets defined by a single inequality is to combine variables to get a structure resembling Q^1. For example, consider Q, and a set $S \subseteq I$. We can relax the equation defining Q by rounding up the coefficients of x_i for $i \in I \setminus S$, and dropping continuous variables with negative coefficients to obtain

$$\sum_{c_j>0} c_j v_j + \sum_{i \in S} a_i x_i + \sum_{i \in I \setminus S} \lceil a_i \rceil x_i \geq b, \tag{2}$$

as a valid inequality for Q. Writing $a_i = \lfloor a_i \rfloor + \hat{a}_i$ for $i \in S$, and re-arranging terms, we get

$$\left(\sum_{c_j>0} c_j v_j + \sum_{i \in S} \hat{a}_i x_i\right) + \left(\sum_{i \in S} \lfloor a_i \rfloor x_i + \sum_{i \in I \setminus S} \lceil a_i \rceil x_i\right) \geq b. \tag{3}$$

The first part of this inequality is nonnegative, and the second part is integral for all $(v, x) \in Q$. Therefore, the basic mixed-integer inequality implies that

$$\left(\sum_{c_j>0} c_j v_j + \sum_{i \in S} \hat{a}_i x_i\right) + \hat{b}\left(\sum_{i \in S} \lfloor a_i \rfloor x_i + \sum_{i \in I \setminus S} \lceil a_i \rceil x_i\right) \geq \hat{b} \lceil b \rceil. \tag{4}$$

is valid for Q. The coefficients of x_i in this inequality are $\hat{b}\lfloor a_i \rfloor + \hat{a}$ if $i \in S$, and $\hat{b}\lceil a_i \rceil$ if $i \in I \setminus S$. Therefore, $S = \{i \in I : \hat{a}_i \leq \hat{b}\}$ gives the strongest inequality of this form, which is

$$\sum_{c_j>0} c_j v_j + \sum_{i \in I : \hat{a}_i \leq \hat{b}} (\hat{b}\lfloor a_i \rfloor + \hat{a}_i) x_i + \sum_{i \in I : \hat{a}_i > \hat{b}} \hat{b} \lceil a_i \rceil x_i \geq \hat{b} \lceil b \rceil. \tag{5}$$

We call Eq. (5) the *mixed-integer rounding* inequality for Q. Note that the coefficients of x_i can be written as $\hat{b}\lfloor a_i \rfloor + \min\{\hat{a}_i, \hat{b}\}$. Finally, as observed by Cornuéjols, Li, and Vandenbussche [43], scaling the the equation defining Q by a rational number t before writing the MIR inequality, can be useful in some circumstances. We call such an inequality a *t-scaled* MIR inequality.

Assume $\hat{b} \neq 0$. Subtracting $\hat{b}$ times $\sum_{j\in J} c_j v_j + \sum_{i\in I} a_i x_i = b$ from the MIR inequality, and dividing by $(1-\hat{b})$, we get the equivalent inequality:

$$\sum_{j\in J: c_j>0} c_j v_j - \sum_{j\in J: c_j<0} \frac{\hat{b}c_j}{1-\hat{b}} v_j + \sum_{i\in I: \hat{a}_i \leq \hat{b}} \hat{a}_i x_i + \sum_{i\in I: \hat{a}_i > \hat{b}} \frac{\hat{b}(1-\hat{a}_i)}{1-\hat{b}} x_i \geq \hat{b}, \tag{6}$$

This is the *Gomory mixed-integer cut*, as originally defined in [3].

To define the MIR inequality for P, we start off with some multiplier vector λ and define $c = \lambda^T C$, $a = \lambda^T A$, and $b = \lambda^T d$. Then $cv + ax = b$ is a valid inequality for P, and we call Eq. (5) an MIR inequality for P. Define the MIR *rank* of an inequality valid for P^{LP} to be 0. For an inequality valid for P, define the MIR rank to be a positive number t if it does not have MIR rank $t-1$ or less, but is a nonnegative linear combination of MIR inequalities derived from inequalities with MIR rank $t-1$. All cutting planes for a pure integer program have finite MIR rank, but this is not true in the case of mixed-integer programs [11].

A linear inequality $g^T v + h^T x \geq q$ is called a *split cut* for P if it is valid for both $P^{\mathrm{LP}} \cap \{\pi^T x \leq \gamma\}$ and $P^{\mathrm{LP}} \cap \{\pi^T x \geq \gamma + 1\}$, where π and γ are integral. The inequality $g^T v + h^T x \geq q$ is said to be derived from the *disjunction* $\pi^T x \leq \gamma \vee \pi^T x \geq \gamma + 1$. All points in P satisfy any split cut for P. The basic mixed-integer inequality is a split cut for Q^1 derived from the disjunction $z \leq \lfloor b \rfloor$ and $z \geq \lceil b \rceil$, when $\hat{b} \neq 0$. Therefore, the MIR inequality (5) is a split cut for P. Nemhauser and Wolsey [9] showed that every split cut for P is also an MIR inequality; more precisely, given a split cut, there is an MIR cut which is equivalent to it in the sense that it differs from the split cut by some multiple of the equations $Cv + Ax = d$.

The derivation of the MIR inequality can be viewed as a special case of the following approach. Consider a linear transformation

$$\mathcal{T}: Q \to Q' = \left\{ v' \in \mathbb{R}, z' \in \mathbb{Z} : \begin{pmatrix} v' \\ z' \end{pmatrix} = U \begin{pmatrix} v \\ x \end{pmatrix} + w \right\} \subseteq Q^1,$$

where U is a matrix, w is a vector. If $\alpha^T \binom{v'}{z'} \geq \beta$ is valid for Q', then $\alpha^T U \binom{v}{x} + \alpha^T w \geq \beta$ is valid for Q. In the case of the MIR inequality, $w = 0$, and the coefficients in the first and second rows of U are defined, respectively, by the coefficients in the first and second bracketed expressions in Eq. (3). This approach is used in *local cuts* for the TSP by Applegate et al. [44] with varying linear transformations into varying sets, and not a fixed set such as Q^1. The transformations in [44] are defined by the operation of shrinking sets of nodes to single nodes, and thus have a combinatorial nature, unlike the somewhat abstract transformations in the case of the MIR inequality.

Subsequent to the work of Wolsey [14], and Marchand and Wolsey [13], deriving (or explaining) valid inequalities for mixed-integer programs in the above manner, starting from valid inequalities for very simple polyhedral sets became an active research topic. For example, let $\sum_{i\in I} \alpha_i x_i \geq \beta$ be a valid inequality for P. Then

$\sum_{i\in I}\lceil\alpha_i\rceil x_i \geq \lceil\beta\rceil$ is valid for P, and is called a *Gomory – Chvátal* cut if P has no continuous variables, and a *projected Chvátal – Gomory* cut [35] otherwise. The above inequality can be derived by mapping P into the set $Q^0 = \{z \in \mathbb{Z} : z \geq \beta\}$ via the mapping $z = \sum_{i\in I}\lceil\alpha_i\rceil x_i$, and then using the only facet-defining inequality of Q^0, namely $z \geq \lceil\beta\rceil$. Günlük and Pochet [45] define the *mixing-mir* inequalities from facets of the set $\{(x,z) \in \mathbb{R}\times\mathbb{Z}^n : x + z_i \geq b_i, \text{ for } i = 1,\ldots,n,\ x \geq 0\}$. The underlying motivation behind such research in the context of mixed-integer programming and in local cuts for the TSP is that strong (facet-defining) inequalities for simple sets defined on few variables yield useful inequalities for problems with many variables. The hard part is to find the appropriate small sets and linear transformations. In recent work, Espinoza [46] generated linear transformations dynamically, in a manner similar to the work in [44], to find useful cutting planes for Q, though with moderate success.

2.2. *Two-Step* MIR *Inequalities*

In the relaxation of Q in Eq. (2), the coefficient a_i of an integer variable x_i is either unchanged or rounded up to $\lceil a_i\rceil$. One can get different cuts for Q by increasing a_i to a number less than $\lceil a_i\rceil$, say $\lfloor a_i\rfloor + \alpha$ where $0 < \alpha < 1$, thereby obtaining a stronger relaxation than Eq. (2).

Dash and Günlük [25] obtained such cuts from a simple mixed-integer set with three variables:

$$Q^2 = \{v \in \mathbb{R},\ y,z \in \mathbb{Z} : v + \alpha y + z \geq \beta,\ v,y \geq 0\},$$

where $\alpha,\beta \in \mathbb{R}$ are parameters that satisfy $1 > \beta > \alpha > 0$, and $\lceil\beta/\alpha\rceil > \beta/\alpha$. Though β is required to be less than 1 in Q^2, the fact that z can take on negative values makes the set fairly general. We do not know of an explicit description of the convex hull of points in Q^2; however one can obtain the inequalities describing the convex hull in polynomial time using results in [47] and [48]. Dash and Günlük showed that the following inequalities are valid for Q^2, and facet-defining under some conditions:

$$v + \alpha y + \beta z \geq \beta, \tag{7}$$

$$(1/(\beta - \alpha\lfloor\beta/\alpha\rfloor))v + y + \lceil\beta/\alpha\rceil z \geq \lceil\beta/\alpha\rceil. \tag{8}$$

Lemma 2 ([25])**.** *The inequality* (7) *is valid and facet-defining for* Q^2*. If* $1/\alpha \geq \lceil\beta/\alpha\rceil$*, then the inequality* (8) *is valid and facet-defining for* Q^2*.*

Proof. The inequality (7) can be obtained by treating $v + \alpha y$ as a continuous variable and applying the basic mixed-integer inequality (1) to $(v + \alpha y) + z \geq \beta$. To see that inequality (8) is valid, notice that the inequalities

$$(1/\alpha)v + y + (\beta/\alpha)z \geq \beta/\alpha,$$
$$(1/\alpha)v + y + (1/\alpha)z \geq \beta/\alpha,$$

are valid for Q^2. Therefore, for any $\gamma \in R$ satisfying $1/\alpha \geq \gamma \geq \beta/\alpha$, the inequality

$$(1/\alpha)v + y + \gamma z \geq \beta/\alpha$$

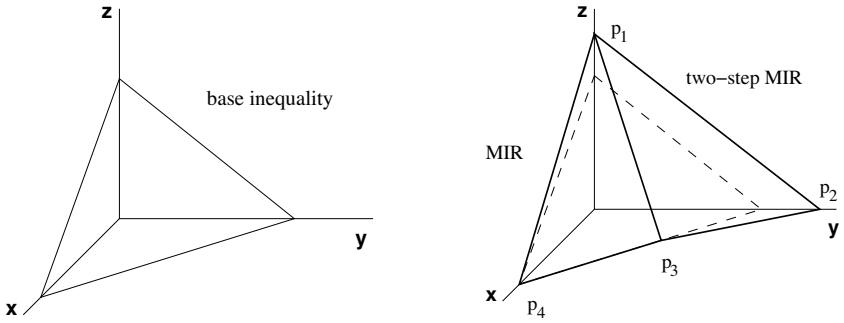

Figure 2. Some facets of Q^2

is also valid as it can be obtained as a convex combination of valid inequalities. If $1/\alpha \geq \lceil \beta/\alpha \rceil$, then

$$v/\alpha + y + \lceil \beta/\alpha \rceil z \geq \beta/\alpha \tag{9}$$

is valid for Q^2. Applying (1) to the previous inequality with v/α treated as a continuous variable and $y + \lceil \beta/\alpha \rceil z$ treated as an integer variable gives inequality (8). Consider the following points in Q^2:

$$p_1 = (0, 0, 1), \quad p_2 = (0, \lceil \beta/\alpha \rceil, 0), \quad p_3 = (\beta - \alpha \lfloor \beta/\alpha \rfloor, \lfloor \beta/\alpha \rfloor, 0), \quad p_4 = (\beta, 0, 0).$$

As depicted in Figure 2, the points p_1, p_3 and p_4 are affinely independent and tight for Eq. (7). Also, the affinely independent points p_1, p_2 and p_3 are tight for Eq. (8), and thus these two inequalities are facet-defining for Q^2. □

We call Eq. (8) the *two-step* MIR inequality for Q^2; it is obtained by applying (1) twice, and has MIR rank at most two. The inequalities (7) and (8) are not necessarily sufficient to describe the convex hull of Q^2. However, let $Q^{2+} = Q^2 \cap \{z \geq 0\}$; Eqs. (7) and (8), along with the inequalities $v, x, z \geq 0$, define the convex hull of integer solutions of Q^{2+} [25]. The additional restriction $1/\alpha \geq \lceil \beta/\alpha \rceil$ is not required. Conforti and Wolsey [49] characterized the convex hull of a generalization of Q^{2+} (and of Q^{2+}) which they call $X^{2\text{DIV}}$. They assume all variables are nonnegative and thus their results do not imply Lemma 2. We note that if $1/\alpha \leq \lceil \beta/\alpha \rceil$ ($1/\alpha = \lceil \beta/\alpha \rceil$), inequality (8) is an $(1/\alpha)$-scaled MIR inequality for Q^{2+} (Q^2). Thus, it is only when $1/\alpha > \lceil \beta/\alpha \rceil$ that the two-step MIR inequality can have MIR rank 2.

We illustrate the use of inequality (8) in an example. Consider the equation $3.35x_1 + 2.5x_2 - 1.2x_3 = 4.7$ with all variables nonnegative and integral. This equals $.35x_1 + .5x_2 + .8x_3 + w = .7$, where $w = 3x_1 + 2x_2 - 2x_3 - 4$. Here $\beta = 0.7$; let $\alpha = 0.4$. Then $1/\alpha = 2.5 > \lceil \beta/\alpha \rceil = 2$. As $x_1, x_3 \geq 0$, the inequality $(.1x_2) + .4(x_1 + x_2) + (x_3 + w) \geq .7$ is a relaxation of the previous equation. Of the three terms in brackets, the first two are nonnegative, and the last two are integral; we can thus apply inequality (8) to obtain the valid inequality: $\frac{1}{2}x_1 + \frac{2}{3}x_2 + x_3 + w \geq 1$.

We now formalize this procedure to obtain valid inequalities for Q. Let $\beta = \hat{b}$, and choose $\alpha \in (0, \hat{b})$ such that $1/\alpha \geq \lceil \hat{b}/\alpha \rceil$. Define $\tau = \lceil \hat{b}/\alpha \rceil \geq 2$, and define $\rho = \hat{b} - \alpha \lfloor \hat{b}/\alpha \rfloor$. Let k_i, l_i be integers such that $k_i \leq \lfloor \hat{a}_i/\alpha \rfloor$, and $l_i \geq \lceil \hat{a}/\alpha \rceil$, for $i \in I$. Let I_0, I_1 and I_2 be

sets which form a partition of I, and let $w = \sum_{i\in I}\lfloor a_i\rfloor x_i - \lfloor b\rfloor$. We can relax the equation in Q to obtain

$$\sum_{c_j>0} c_j v_j + \sum_{i\in I_1}(k_i\alpha + (\hat{a}_i - k_i\alpha))x_i + \sum_{i\in I_2} l_i\alpha\, x_i + \sum_{i\in I_0} x_i + w \geq \hat{b},$$

which can be rewritten as

$$\sum_{c_j>0} c_j v_j + \sum_{i\in I_1}(\hat{a}_i - k_i\alpha)x_i + \alpha\Big(\sum_{i\in I_1} k_i x_i + \sum_{i\in I_2} l_i x_i\Big) + \Big(\sum_{i\in I_0} x_i + w\Big) \geq \hat{b},$$

We can map points in Q to points in Q^2 by equating the first variable in Q^2 with the sum of the first two terms in the equation above, and equating the second and third variables with the first and second bracketed expressions, respectively. Applying inequality (8) and substituting for w leads to the inequality

$$\sum_{c_j>0} c_j v_j + \sum_{i\in I} \gamma_i x_i + \rho\,\tau\, z \geq \rho\,\tau\lceil b\rceil, \tag{10}$$

where

$$\gamma_i = \rho\tau\lfloor a_i\rfloor + \begin{cases} \rho\tau & \text{if } i \in I_0, \\ k_i\rho + \hat{a}_i - k_i\alpha & \text{if } i \in I_1, \\ l_i\rho & \text{if } i \in I_2. \end{cases}$$

By inspection, the strongest inequality of this form is obtained by setting $k_i = k_i^* = \lfloor \hat{a}_i/\alpha\rfloor$ and $l_i = l_i^* = \lceil \hat{a}/\alpha\rceil$ for $i \in I$, and letting

$$I_0 = \{i \in I : \hat{a}_i \geq \hat{b}\},$$
$$I_1 = \{i \in I \setminus I_0 : \hat{a}_i - k_i^*\alpha < \rho\}, \qquad I_2 = \{i \in I \setminus I_0 : \hat{a}_i - k_i^*\alpha \geq \rho\}.$$

In other words,

$$\gamma_i = \rho\tau\lfloor a_i\rfloor + \min\{\rho\tau,\ k_i^*\rho + \hat{a}_i - k_i^*\alpha,\ l_i^*\rho\}.$$

As shown in [25], these inequalities imply the *strong fractional cuts* of Letchford and Lodi [50]. They can be separated in polynomial time, under some restrictions.

Theorem 3 (Dash, Goycoolea, Günlük [24])**.** *Given a point* $(v^*, x^*) \in Q^{\mathrm{LP}}$, *the most violated two-step* MIR *inequality can be found in polynomial time*, *assuming* $\tau \leq k$ *for some fixed* k.

Here, the violation of Eq. (10) with respect to (v^*, x^*) is given by its right-hand side minus the left-hand side evaluated at (v^*, x^*). The algorithm given in [24] is very simple; just write down the inequality (10) for all feasible choices of α from the set $\{\hat{a}_i/t : i \in I, t \in \mathbb{N}, \hat{a}_i/t \geq \hat{b}/k\} \cup \{1/t : t \in \mathbb{N}, t\hat{b} \leq k\}$, and compute the violation.

Kianfar and Fathi [51] generalize the two-step MIR inequalities and generate *n-step* MIR inequalities for Q. These inequalities have MIR rank at most n.

3. Master Polyhedra

Gomory [20] developed the concepts of corner polyhedra and master polyhedra as tools to generate cutting planes for general integer programs, and to solve asymptotic integer programs. We do not discuss the latter aspect here. The *corner polyhedron* (*strengthened* corner polyhedron) for a vertex of P^{LP} is the convex hull of integer points satisfying only the linearly independent constraints which define the vertex (all constraints tight at the vertex). Clearly, valid inequalities for a corner polyhedron of P yield valid inequalities for P. The central element of Gomory's approach is the fact that many different corner polyhedra can be viewed as faces of a much smaller number of *master polyhedra*, which are much more 'regular' and amenable to analysis. Gomory, and later Gomory and Johnson [21] showed how to obtain facets of master polyhedra, which yield valid inequalities for corner polyhedra and thereby for P. In this section, we describe in detail master cyclic group polyhedra, or master polyhedra associated with corner polyhedra for single constraint (plus nonnegativity of variables) systems, such as the one defining Q. Many results are available for such polyhedra, but master polyhedra for multiple constraint systems are less well-understood, beyond the initial results of Gomory. We briefly discuss some recent research on this topic later.

For the set Q, assume that $\hat{a}_i$ ($i \in I$) and $\hat{b}$ are rational numbers with a common denominator, say n, and let $\hat{b} = r/n$, where $0 < r < n$. Rewrite Q as

$$Q = \left\{ v \in \mathbb{R}^{|J|},\ x \in \mathbb{Z}^{|I|} : \left(\sum_{i \in I} \lfloor a_i \rfloor x_i - \lfloor b \rfloor \right) + \sum_{j \in J} c_j v_j + \sum_{i \in I} \hat{a}_i x_i = \hat{b},\ v, x \geq 0 \right\}$$

Let $I_k = \{i \in I : \hat{a}_i = k/n\}$ and define the mapping

$$\begin{aligned} w_k &= \sum_{i \in I_k} x_i, & z &= -\left(\sum_{i \in I} \lfloor a_i \rfloor x_i - \lfloor b \rfloor \right) \\ v_+ &= \sum_{c_j > 0} c_j v_j, & v_- &= -\sum_{c_j < 0} c_j v_j, \end{aligned} \tag{11}$$

that maps each point (v, x) in Q to a point (v_+, v_-, w) in the polyhedron

$$P'(n, r) = \mathrm{conv}\left\{ v_+, v_- \in \mathbb{R},\ w \in \mathbb{Z}^{n-1} : v_+ - v_- + \sum_{i=1}^{n-1} \frac{i}{n} w_i - z = \frac{r}{n},\ v_+, v_-, w \geq 0,\ z \in \mathbb{Z} \right\}.$$

(If I_k is empty for some k, set w_k to zero.) If Q has no continuous variables, then (11) maps points in Q to points in the *master cyclic group polyhedron* of Gomory:

$$P(n, r) = \mathrm{conv}\left\{ w \in \mathbb{Z}^{n-1} : \sum_{i=1}^{n-1} \frac{i}{n} w_i - z = \frac{r}{n},\ w \geq 0,\ z \in \mathbb{Z} \right\}. \tag{12}$$

We view $P'(n, r)$ as the mixed-integer extension of $P(n, r)$. For $P(n, r)$, z can be assumed to be nonnegative, but not for $P'(n, r)$. Note that the constraint defining $P(n, r)$ can be written as $\sum_{i=1}^{n-1} i w_i \equiv r \pmod{n}$. In [3], Gomory first derived the GMI cut as a valid inequality for points satisfying a similar equation.

Recently, Dash, Fukasawa and Günlük [52] studied the polyhedron

$$K(n,r) = \text{conv}\left\{(x,y) \in \mathbb{Z}^n \times \mathbb{Z}^n : \sum_{i=1}^{n} ix_i - \sum_{i=1}^{n} iy_i = r,\ x,y \geq 0\right\} \tag{13}$$

where $n, r \in \mathbb{Z}$ and $0 < r \leq n$, and characterized the convex hull of its nontrivial facets. $K(n,r)$ was first defined by Uchoa [53] in a slightly different form. Replacing z in (12) by y_n, and multiplying the defining constraint by n, we see that $P(n,r)$ defines a face of $K(n,r)$. Facets of $P(n,r)$ can be lifted to obtain facets of $K(n,r)$, but not all facets can be obtained this way [52].

3.1. Basic properties

Both $P(n,r)$ and $P'(n,r)$ are full-dimensional, unbounded polyhedra, and their characteristic (recession) cones are $\mathbb{R}^{n-1}_+$ and $\mathbb{R}^{n+1}_+$ respectively. Also, the inequalities $x_i \geq 0$ for $i = 1, \ldots, n-1$ define facets for both $P(n,r)$ and $P'(n,r)$, and the inequalities $v_- \geq 0$ and $v_+ \geq 0$ are facet-defining for $P'(n,r)$. Therefore, if $\eta^T w \geq \eta_o$ is a facet defining inequality of $P(n,r)$, then $\eta_o \geq 0$, $\eta \geq \mathbf{0} \in \mathbb{R}^{n-1}$. Further, for any *nontrivial* facet (i.e., not defined by the nonnegativity inequalities), $\eta_o > 0$. We will assume in what follows that η_o is scaled to be 1. Finally, any nontrivial facet of $P(n,r)$ has the following property: there is a set of $n-1$ linearly independent integer points χ^i in $P(n,r)$ satisfying $\chi^i_i \geq 1$, for $i = 1, \ldots, n-1$. Gomory characterized the convex hull of nontrivial facets of $P(n,r)$.

Theorem 4 ([20]). *If $r \neq 0$, then $\sum_{i=1}^{n-1} \eta_i w_i \geq 1$ is a nontrivial facet of $P(n,r)$ if and only if $\eta = (\eta_i)$ is an extreme point of the inequalities*

$$\eta_i + \eta_j \geq \eta_{(i+j) \bmod n} \qquad \forall i, j \in \{1, \ldots, n-1\}, \tag{14}$$

$$\eta_i + \eta_j = \eta_r \qquad \forall i, j \text{ such that } r = (i+j) \bmod n, \tag{15}$$

$$\eta_j \geq 0 \qquad \forall j \in \{1, \ldots, n-1\}, \tag{16}$$

$$\eta_r = 1. \tag{17}$$

The property (14) is called *subadditivity*.

Gomory actually proved a more general result. Consider an integer $k > 0$, and let $r, n \in \mathbb{Z}^k_+$, with each component of n greater than 1, and $0 < r_l < n_l$, for $l = 1, \ldots, k$. Let $G \subseteq \mathbb{Z}^k_+$ consist of all vectors with the lth component contained in $\{0, \ldots, n_l - 1\}$, and let $G_+ = G \setminus \mathbf{0}$. In the defining constraint for $P(n,r)$, replace $\sum_{i=1}^{n-1} iw_i \equiv r \pmod n$ by $\sum_{i \in G_+} iw_i \equiv r \pmod n$, where a vector equals another modulo n if and only if the component-wise modular equations hold. Then Theorem 4 is true if r, n are defined as above, the indices i, j are elements of G_+, and we replace $\{1, \ldots, n-1\}$ by G_+. The elements of G, along with the operation of addition modulo n, form the abelian group $\mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$, which is cyclic when $k = 1$. When $k > 1$, some valid inequalities (e.g., scaled MIR inequalities) when applied to the individual constraints of $P(n,r)$ define facets of $P(n,r)$. For $k = 2$ (equivalently, $G = \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$), in limited shooting experiments (see Section 3.3) we observe that taking integral linear combinations of the first and second constraints (with small multipliers, e.g., 1,1) and writing a scaled MIR (or two-step MIR) inequality on the resulting constraint also yields facets. Dey and Richard studied facet-defining inequalities in the case $k \geq 2$ (though for an "infinite group" [54] variant of $P(n,r)$); see [28, 29]. When $k >= 2$, the relaxation of $P(n,r)$ obtained by removing the integrality of w is studied in [55] and [56] (in a different form).

Optimizing a linear function over $P(n,r)$ is similar to solving a knapsack problem and can be done in pseudo-polynomial time (in n) via dynamic programming; therefore, one can also separate a point from $P(n,r)$ in pseudo-polynomial time. The theorem above implies that separation can in fact be done by obtaining a basic optimal solution of the constraints in the theorem. It also yields a way of showing that a particular inequality defines a facet of $P(n,r)$.

Gomory and Johnson [21] showed that the polyhedral structure of $P'(n,r)$ can be analyzed completely by just studying $P(n,r)$.

Theorem 5 ([21]). *An inequality defines a nontrivial facet of $P'(n,r)$ if and only if it has the form*

$$n\eta_1 v_+ + n\eta_{n-1} v_- + \sum_{i=1}^{n-1} \eta_i w_i \geq 1, \tag{18}$$

where $(\eta_1, \ldots, \eta_{n-1})$ defines a facet of $P(n,r)$.

Gomory and Johnson also described the convex hull of nontrivial facets of $P'(n,r)$, as in Theorem 4, when r is nonintegral, but for our purpose Theorem 5 suffices. Valid inequalities for $P'(n,r)$ yield valid inequalities for Q. Given a facet (18) of $P'(n,r)$,

$$n\eta_1\Big(\sum_{c_j\geq 0} c_j v_j\Big) + n\eta_{n-1}\Big(\sum_{c_j<0} c_j v_j\Big) + \sum_{i\in I} f(\hat{a}_i)x_i \geq 1 \tag{19}$$

is a valid inequality for Q, where $f(\hat{a}_i) = \eta_k$ if $\hat{a}_i = k/n$. We call such inequalities *group cuts* for Q. We will see that the GMI cut can be derived as a group cut. Let $(v', x') \in Q^{\mathrm{LP}} \setminus \mathrm{conv}(Q)$, and let (v', x') be mapped to (v'_+, v'_-, w') via (11). Then (v', x') satisfies all group cuts (and $(v'_+, v'_-, w') \in P'(n,r)$) if and only if the *separation LP*

$$\min\Big\{(nv'_+)\eta_1 + (nv'_-)\eta_{n-1} + \sum_i w'_i\eta_i : \eta \text{ satisfies Eqs. (14)}-\text{(17)}\Big\}.$$

has optimum value at least 1. If some group cut is violated by (v', x'), then the optimum solution of the separation LP yields a most violated group cut with objective value less than 1.

For $P(n,r)$, let e_i $(i = 1, \ldots, n-1)$ be the unit vectors in $\mathbb{R}^{n-1}$ with ones in the ith component and zeros elsewhere.

Lemma 6. *If $\eta \in \mathbb{R}^{n-1}$ satisfies the constraints* (15)$-$(17) *and $\eta^T x \geq 1$ is a valid inequality for $P(n,r)$, then η satisfies the subadditivity constraints* (14)*.*

Proof. Let $j,k \in \{1,\ldots,n-1\}$, and let $i = j + k \bmod n$. Let $\chi = e_j + e_k + e_{r-i}$. Then $\chi \in P(n,r)$, and $\eta^T\chi = \eta_j + \eta_k + \eta_{r-i} \geq 1$. As $\eta_i + \eta_{r-i} = 1$, it follows that $\eta_j + \eta_k \geq \eta_i$. □

Proof of Theorem 4. Let $\gamma^T w \geq 1$ define a nontrivial facet of $P(n,r)$. We argued earlier that $\gamma \geq \mathbf{0}$. We next show that γ satisfies Eqs. (15) and (17). Observe that $e_r, e_i + e_{r-i} \in P(n,r)$ and thus $\gamma_r \geq 1$, and $\gamma_i + \gamma_{r-i} \geq 1$. Further, there are integral points χ^1, χ^2 and χ' in $P(n,r)$ lying on this facet such that $\chi^1_i \geq 1$ and $\chi^2_{r-i} \geq 1$ and $\chi'_r \geq 1$. Then $\gamma^T(\chi^1 + \chi^2) = 2$. But

$$\chi = \chi^1 + \chi^2 - e_i - e_{r-i} \in P(n,r) \implies \gamma^T \chi = 2 - \gamma_i - \gamma_{r-i} \geq 1.$$

Therefore, $\gamma_i + \gamma_{r-i} \leq 1$. Similarly $\gamma^T \chi' = 1$, and

$$2\chi' - e_r \in P(n,r) \implies \gamma^T(2\chi' - e_r) = 2 - \gamma_r \geq 1 \implies \gamma_r \leq 1.$$

Therefore γ satisfies Eqs. (15)–(17). By Lemma 6, it also satisfies the subadditivity constraints.

Let η satisfy Eqs. (14)–(17). For any integral point $w^* \in P(n,r)$

$$\sum_i \eta_i w_i^* \geq \sum_i \eta_{(iw_i^*)} \geq \eta_{(\sum_i iw_i^*)} = \eta_r = 1,$$

where the subscripts inside the brackets are computed modulo n. Therefore, $\eta^T w \geq 1$ defines a valid inequality for $P(n,r)$. Therefore, a nontrivial facet of $P(n,r)$ is an extreme point of Eqs. (14)–(17); otherwise it would be a convex combination of solutions of this system, each of which defines a valid inequality for $P(n,r)$. Let $P(n,r) = \{w \in \mathbb{R}^{n-1} : Aw \geq \mathbf{1},\ w \geq 0\}$, where $Aw \geq \mathbf{1}$ represents the nontrivial facets of $P(n,r)$ and $\mathbf{1}$ is a vector with all components 1. Therefore, if A_j stands for the jth column of A, then for any index $i \neq r$, $A_i + A_{r-i} = A_r = \mathbf{1}$. Observe that $e_r \in P(n,r)$ and $\eta^T e_r = \eta_r = 1$. Therefore

$$\min\{\eta^T w : Aw \geq \mathbf{1},\ w \geq 0\} = 1,$$

and an optimal dual vector y of the above LP satisfies

$$y^T A \leq \eta, \qquad y^T \mathbf{1} = 1, \qquad y \geq 0.$$

If $y^T A_i < \eta_i$ for any index $i \neq r$, then

$$1 = \eta_i + \eta_{r-i} > y^T A_i + y^T A_{r-i} = y^T \mathbf{1} = 1.$$

Therefore $y^T A_i = \eta_i$ for $i = 1, \ldots, n-1$. This implies that η is a convex combination of nontrivial facets $\{\eta^j\}$ of $P(n,r)$, which in turn are solutions of Eqs. (14)–(17). Thus, if η is an extreme point (14)–(17), $\eta^T x \geq 1$ defines a facet of $P(n,r)$, and the vector y is a unit vector. □

As the defining equation for $P(n,r)$ has the same form as Q, we can apply the MIR or two-step MIR inequalities to $P(n,r)$. We define the t-scaled (two-step) MIR inequality for $P(n,r)$ for rational $t > 0$ as the inequality obtained by applying the t-scaled (two-step) MIR inequality to $Q_p = \{w \in \mathbb{Z}^{n-1}, z \in \mathbb{Z} : \sum_{i=1}^{n-1}(i/n)w_i - z = r/n,\ w \geq 0\}$ and then substituting out z. We focus on t-scaled inequalities for integral t. For an integer $n > 0$ and integers t and i, define $(ti)_n = ti \bmod n$, where $k \bmod n$ stands for $k - n\lfloor k/n \rfloor$. For an integer t, the t-scaled MIR inequality for $P(n,r)$ becomes

$$\sum_{(ti)_n < (tr)_n} \frac{(ti)_n}{(tr)_n} x_i + \sum_{(ti)_n \geq (tr)_n} \frac{n - (ti)_n}{n - (tr)_n} x_i \geq 1. \tag{20}$$

Given an integer $t \neq n$, it is shown in [26] that the $(-t)$-scaled MIR inequality is the same as the t-scaled MIR inequality and also the $(n-t)$-scaled MIR inequality.

Definition 7. An inequality $\sum_{i=1}^{n-1} \eta_i w_i \geq 1$ is a *two-slope* inequality if $\eta_i - \eta_{i-1}$ equals either η_1 or $-\eta_{n-1}$ for $i = 2, \ldots, n-1$.

Theorem 8 (Gomory, Johnson [21])**.** *Every two-slope inequality for $P(n, r)$ which satisfies the constraints* (14)–(17) *defines a facet for $P(n, r)$.*

Proof. By definition, for $i = 2, \ldots, n-1$ either $\eta_1 + \eta_{i-1} = \eta_i$ or $\eta_{n-1} + \eta_i = \eta_{i-1}$. The constraints above are just the subadditivity constraints (14) (satisfied as equalities) when $i \neq r, r+1$, and constraints of the form (15) otherwise. These, along with the constraint $\eta_r = 1$, define $n-1$ linearly independent constraints from Eqs. (14)–(17). □

Let $\eta^T w \geq 1$ stand for the MIR inequality for $P(n, r)$. By definition $\eta_i = i/r$ if $i < r$, and $\eta_i = (n-i)/(n-r)$ otherwise. It clearly is a two-slope inequality, and satisfies Eqs. (15)–(17). As it is a valid inequality for $P(n, r)$, Lemma 6 implies that it satisfies the subadditivity constraints, and therefore the requirements of Theorem 8. This implies the following result of Gomory.

Corollary 9 ([20])**.** *The* MIR *inequality for $P(n, r)$ defines a facet of $P(n, r)$.*

The following result can be proved in a similar manner.

Corollary 10. *If an integer $t > 0$ is a divisor of n and tr is not a multiple of n, then the t-scaled* MIR *inequality defines a facet of $P(n, r)$.*

We see later that for every nonzero integer t, such that tr is not a multiple of n, the t-scaled MIR inequality (20) defines a facet of $P(n, r)$ [25]. We also note that t-scaled MIR inequalities for $P(n, r)$ for some nonintegral $t > 0$ define facets of $P(n, r)$. The two-step MIR inequalities also yield facets of $P(n, r)$.

Theorem 11 (Dash and Günlük [25])**.** *Let $\Delta \in \mathbb{Z}^+$ be such that $r > \Delta > 0$, and $n > \Delta\lceil r/\Delta \rceil > r$. The two-step* MIR *inequality for $P(n, r)$, obtained by applying Eq.* (10) *to Q_p with $\alpha = \Delta/n$ and $b = r/n$ defines a facet of $P(n, r)$.*

We call a facet in the above result a 1-scaled two-step MIR facet of $P(n, r)$ with parameter Δ. Note that $\alpha = \Delta/n$ satisfies the conditions of Lemma 2: (i) $r > \Delta > 0 \Rightarrow \hat{b} = r/n > \alpha > 0$, and (ii) $n > \Delta\lceil r/\Delta \rceil > r \Rightarrow n/\Delta > \lceil r/\Delta \rceil > r/\Delta \Rightarrow 1/\alpha > \lceil \hat{b}/\alpha \rceil > \hat{b}/\alpha$. Therefore the parameter α is assigned valid values. The proof of Theorem 11 is identical to the proof of Corollary 9. Further, if tr is not a multiple of n, then for appropriate choices of α, the t-scaled two-step MIR inequalities define facets of $P(n, r)$ [25]. If t is a divisor of n, the proof of this result is similar to that of Corollary 9. The two-step MIR facets contain the *2slope* facets in Araoz et al. [22]. Other facet classes can be found in [22] (*3slope* facets), and also in [57].

An automorphism ϕ is a bijection from $\{0, 1, \ldots, n-1\}$ to itself such that

$$\phi((a+b) \bmod n) = (\phi(a) + \phi(b)) \bmod n.$$

A bijection ϕ is an automorphism if and only if $\phi(i) = (ti)_n$ where t is coprime with n (n and t have no common divisors). The inverse of an automorphism is also an automorphism; if $\phi(i) = (ti)_n$, then $\phi^{-1}(i) = (ui)_n$ where u satisfies $tu \equiv 1 \pmod{n}$ (such a u exists as t and n are coprime).

Theorem 12 (Gomory [20])**.** *Let r be an integer such that $0 < r < n$. Let ϕ be an automorphism defined by $\phi(i) = (ti)_n$, and let $s = \phi(r)$. If $\sum_i \eta_i w_i \geq 1$ is a nontrivial facet of $P(n, r)$, then $\sum_i \eta_i w_{\phi(i)} \geq 1$ is a nontrivial facet of $P(n, s)$. Equivalently, $\sum_i \eta_{\phi^{-1}(i)} w_i \geq 1$ is a nontrivial facet of $P(n, s)$.*

Theorem 12 says that for an automorphism ϕ, the facets of $P(n, r)$ are identical to facets of $P(n, \phi(r))$ after permuting facet coefficients, i.e., $P(n, r)$ and $P(n, \phi(r))$ have the same polyhedral structure (are isomorphic polyhedra). Therefore, for a given n, we need only study $P(n, r)$ where r is a divisor of n in order to understand $P(n, r)$ for all r.

Example 13. Let $w^* = (w_1^*, w_2^*, w_3^*, w_4^*)$ be a nonnegative integer vector satisfying

$$1w_1 + 2w_2 + 3w_3 + 4w_4 \equiv 3 \bmod 5.$$

Note that 2 is coprime with 5, and $2 \times 3 \equiv 1 \bmod 5$. Clearly w^* also satisfies $2(1w_1 + 2w_2 + 3w_3 + 4w_4) \equiv 2 \times 3 \equiv 1 \bmod 5$, or $2w_1 + 4w_2 + 1w_3 + 3w_4 \equiv 1 \bmod 5$. Note that the above equation is precisely the defining equation of $P(5, 1)$, except that the variable indices are different from their coefficients. Now any solution w' of the last equation satisfies $3(2w_1 + 4w_2 + 1w_3 + 3w_4) \equiv 3 \bmod 5$, which is the same as the first modular equation. Therefore, $P(5, 3)$ and $P(5, 1)$ have the same polyhedral structure.

3.2. Subadditivity

A function $f\colon \mathbb{R} \to \mathbb{R}$ is subadditive if for all $u, v \in \mathbb{R}$, $f(u) + f(v) \geq f(u + v)$. Given a subadditive function f, it is easy to see that a nonnegative integral solution x of $\sum_{i \in I} a_i x_i = b$ satisfies $\sum_{i \in I} f(a_i) x_i \geq f(b)$. More generally, if the coefficients of Q are multiples of $1/n$, then

$$nf(1/n) \sum_{c_j > 0} c_j v_j - nf(1 - 1/n) \sum_{c_j < 0} c_j v_j + \sum_{i \in I} f(a_i) x_i \geq f(b)$$

is a valid inequality for Q. If we don't know n, we can replace $nf(1/n)$ by $\lim_{v \to 0+} f(v)/v$ and $nf(1-1/n)$ by $\lim_{v \to 0+} f(1-v)/v$. Thus, subadditive functions yield valid inequalities for integer programs and for $P(n, r)$.

Gomory and Johnson [21] showed how to obtain subadditive functions from facets of $P(n, r)$ using the property (14). Let $\sum_{i=1}^{n-1} \eta_i w_i \geq 1$ define a nontrivial facet of $P(n, r)$. Define a function $f(v)$ over the domain $[0, 1]$ as follows:

$$f(0) := 0, \qquad f(i/n) := \eta_i, \quad \forall i = 1, \ldots, n-1, \tag{21}$$

$$f((i + \delta)/n) := (1 - \delta) f(i/n) + \delta f((i + 1)/n), \quad \forall i = 0, \ldots, n-1 \text{ and } \delta \in (0, 1). \tag{22}$$

Then define $f(v)$ for all $v \in \mathbb{R}$ by $f(v) := f(\hat{v})$. Gomory and Johnson proved that f is subadditive. We call f a *facet-interpolated* function (FIF); such a function is piecewise linear and continuous. In Figure 3, we plot the coefficients of two facet-defining inequalities for $P(10, 7)$, and depict the corresponding facet-interpolated functions. Note that the coefficients of the GMI cut (6), when divided by the right-hand side $\hat{b}$, are given by the function in Figure 3(a). If f stands for this function, and a_i is the coefficient of x_i in Q,

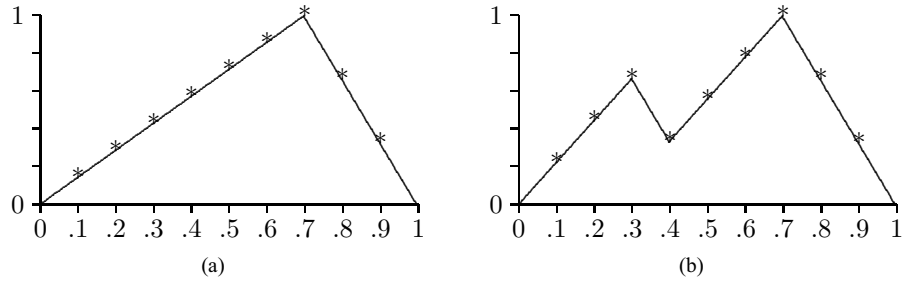

Figure 3. (a) The MIR facet for $P(10, 7)$; (b) A two-step MIR facet for $P(10, 7)$

then the coefficient of x_i in the GMI cut is given by $\hat{b}$ times $f(\hat{a}_i)$; if a continuous variable v_j has a positive (negative) coefficient c_j, then $\hat{b}c_j$ times the slope of f at the origin (at 1) gives its cut coefficient.

An interesting aspect of FIFs is that one can use a facet of $P(n', r')$ to obtain a valid inequality for $P(n, r)$, where n, r are completely unrelated to n', r'. For FIFs derived from some simple facets of $P(n', r')$, such as t-scaled MIR facets where t is a divisor of n', the corresponding valid inequality for $P(n, r)$ is dominated by the t-scaled MIR inequality for $P(n, r)$ [26]. A similar statement is true for two-step MIR FIFs, when Δ in Theorem 11 satisfies some additional conditions.

FIFs form a somewhat restricted subclass of all subadditive functions; they only take nonnegative values and they are periodic. Dash, Fukasawa and Günlük [52] show how to obtain more general subadditive functions by interpolating coefficients of facets of $K(n, r)$, and give examples of such functions which take on nonnegative values.

3.3. Shooting Experiments

Gomory and Johnson [21] showed that $P'(n, r)$ has exponentially many facets (in n). Is there a way of determining which facets are more "important" and yield more important group cuts?

Gomory proposed using the solid angle subtended at the origin by a facet as a measure of the importance of the facet. Gomory, Johnson and Evans [23] estimate the solid angle subtended at the origin by a facet of $P(n, r)$ (in a 'shooting' experiment) by generating vectors uniformly distributed over the unit sphere and computing the frequency with which different facets are hit by these directions. Given a direction $d \geq 0$ (assume d has norm 1), we say that d hits a nontrivial facet $\eta^T w \geq 1$ of $P(n, r)$ if it is the last facet intersected by the ray $\{td : t \geq 0\}$. In Figure 4, we depict d by the arrow. The ray $\{td : t \geq 0\}$ intersects an inequality $\eta^T w \geq 1$ when $\eta^T(td) = 1$. Therefore the facet hit by d is given by

$$\max\{t : \eta^T(td) = 1,\ \eta \text{ is a facet}\} \equiv \max\left\{\frac{1}{\eta^T d} : \eta \text{ is a facet}\right\} \equiv \min\{\eta^T d : \eta \text{ is a facet}\}.$$

Therefore, for any $d \in R^{n-1}_+$, a basic optimal solution of the linear program

$$\min\{d^T\eta : \eta \text{ satisfies Eqs. (14)}-\text{(17)}\}$$

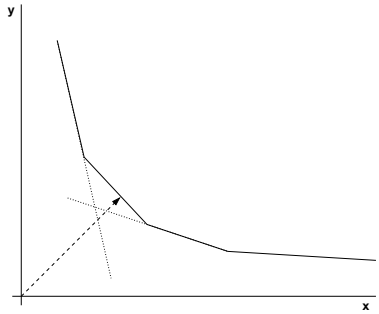

Figure 4. Facet hit by a direction vector

gives the facet hit by d. The facet hit by d can be interpreted as the one most violated by d. Similar shooting experiments were first performed by Kuhn in the 1950s in the context of the TSP, see [58, 59].

An important question about the shooting experiment is whether its results depend on the r in $P(n, r)$? We next give a result from [26] which says that the classes of facets discussed earlier are invariant under automorphisms. For example, the scaled MIR facets of $P(10, 1)$ are isomorphic to the scaled MIR facets of $P(10, 3)$. In this section, we only discuss integer scaling factors.

Theorem 14 ([26])**.** *Let n, r be integers with $0 < r < n$. Let k be an integer coprime with n, and let $\phi(i) = ki \bmod n$. Then the scaled* MIR *and two-step* MIR *facets of $P(n, r)$ are isomorphic, respectively, to the scaled* MIR *facets and scaled two-step* MIR *facets of $P(n, \phi(r))$.*

Proof. We only prove the result for scaled MIR facets. It is not difficult to see that the t-scaled MIR inequality in Eq. (20) can be written as

$$\sum_{i=1}^{n-1} h^{tr/n}(ti/n)w_i \geq 1,$$

where $h^b(v) = \hat{v}/\hat{b}$ if $\hat{v} < \hat{b}$, and $h^b(v) = (1 - \hat{v})/(1 - \hat{b})$ otherwise (the function in Figure 3(a)). The values of $h^b(v)$ depend only on $\hat{b}$ and $\hat{v}$, the fractional parts of b and v, respectively. Thus if b' and v' are numbers such that $b - b'$ and $v - v'$ are integral, then $h^b(v) = h^{b'}(v')$. Let k and ϕ be defined as in the theorem. From Eq. (20), the t-scaled MIR inequality for $P(n, r)$ is isomorphic to the following inequality of $P(n, \phi(r))$:

$$\sum_{i=1}^{n-1} h^{tr/n}(ti/n)\, w_{\phi(i)} \geq 1. \tag{23}$$

Let s be the unique integer such that $sk \bmod n = t$. We will show that Eq. (23) is the s-scaled MIR inequality for $P(n, \phi(r))$. If j is any integer between 1 and $n - 1$, then $s\phi(j) \bmod n = skj \bmod n = tj \bmod n \Rightarrow s\phi(j)/n - tj/n$ is integral. Therefore Eq. (23) is the same as

$$\sum_{i=1}^{n-1} h^{s\phi(r)/n}(s\phi(i)/n)w_{\phi(i)} \geq 1,$$

which is the s-scaled MIR inequality for $P(n, \phi(r))$. We conclude that the t-scaled MIR inequality defines a facet of $P(n, r)$ if and only if the s-scaled MIR inequality defines a facet of $P(n, \phi(r))$. □

One can refine the previous theorem and show that the family of t-scaled MIR facets of $P(n, r)$ with $\gcd(n, t) = l$ is isomorphic to the family of s-scaled MIR facets of $P(n, \phi(r))$, where $\gcd(n, s) = l$. One can then conclude that if n is a multiple of 2 (or 3), then the $(n/2)$-scaled ($(n/3)$-scaled) MIR facet of $P(n, r)$ is isomorphic to the $(n/2)$-scaled ($(n/3)$-scaled) MIR facet of $P(n, \phi(r))$. Thus the $n/2$-scaled and $n/3$-scaled MIR facets remain invariant over isomorphic master polyhedra, as long as n is a multiple of 2 or 3. It is incorrectly stated in [26, Theorem 5] that this invariance holds for the t-scaled MIR facet, where t is the largest divisor of n.

Corollary 15 ([25]). *If an integer $t > 0$ is not a divisor of n and tr is not a multiple of n, then the t-scaled* MIR *inequality defines a facet of $P(n, r)$.*

Proof. Assume s is the largest common divisor of t and n. Then $k = t/s$ and n have no common divisors. Define ϕ using k as in Theorem 14; from its proof we know that the s-scaled MIR inequality for $P(n, \phi(r))$ is isomorphic to the t-scaled MIR inequality for $P(n, r)$. This is because $sk \bmod n = t$. Now s is a divisor of n, and sr is not a multiple of n (otherwise tr would also be a multiple of n). Corollary 10 implies that the s-scaled MIR inequality for $P(n, \phi(r))$ defines a facet, and so does the t-scaled MIR inequality of $P(n, r)$. □

An argument similar to the one above can be used to show that t-scaled two-step MIR inequalities define facets of $P(n, r)$.

Gomory, Johnson and Evans [23] observe that in a shooting experiment, a relatively small number of facets of $P(n, r)$ absorb most of the hits and the most important facets of $P(n, r)$ are related to the MIR inequality (they are t-scaled MIR facets [25]). Evans [60] reports that the 2slope facets [22] constitute another important class of facets. The experiments in [23] and [60] are performed on $P(n, r)$ with $n \leq 30$. Dash and Günlük [26] extend these experiments to $P(n, r)$ for n up to 200, and measure the importance of additional facet classes.

Table 1 contains results for selected master polyhedra from [26]. The second and third columns give the number of (integrally) scaled MIR facets and two-step MIR facets. The fourth column gives the number of *distinct* facets hit in 100,000 shots. The fifth, sixth, and seventh columns give, respectively, the hit frequencies for the scaled MIR facets, the scaled two-step MIR facets, and most frequently hit facet from these classes. For example, for $P(50, 1)$ we see that the 378 scaled MIR and two-step MIR facets (out of 65,346 facets) absorb almost 20% of all hits, and that a single facet from this class absorbs 8.1% of all hits. Thus, a very few facets absorb a large fraction of all hits, and are mostly scaled MIR and two-step MIR facets. In addition, neither class is uniformly more important than the other. For example, for $P(42, 1)$ the scaled MIR facets are more important, whereas the scaled two-step MIR facets are more important for $P(42, 21)$. For the examples in Table 1, the facet in column 7 turns out to be a t-scaled MIR facet with t the largest divisor of n for which the t-scaled MIR inequality is valid and facet-defining for $P(n, r)$. For $P(50, 1)$, the facet in column 7 is the 25-scaled MIR facet, absorbing 8.1% of all hits whereas all remaining scaled MIR facets combined absorb only 6.5% of the

Table 1. Shots absorbed by MIR based facets for different $P(n, r)$

	Number of facets			% of shots hit		
The group	MIRs	2-step MIRs	total hit	MIRs	2-step MIRs	top facet
$P(42, 1)$	21	186	46,407	23.5	6.9	9.4
$P(42, 21)$	11	236	53,754	11.8	15.4	9.6
$P(50, 1)$	25	353	65,346	14.6	5.1	8.1
$P(50, 25)$	13	452	68,202	8.9	11.1	8.1
$P(72, 1)$	36	673		11.6	2.2	5.0
$P(72, 18)$	27	616		2.3	2.2	0.9
$P(100, 1)$	50	1534		4.8	0.5	3.2
$P(200, 1)$	100	6584		0.5	0.0	0.4

shots. For $P(72, 18)$, the 36-scaled MIR does not define a facet as 36×18 is a multiple of 72. Instead, the most important facet is the 18-scaled MIR.

We believe that if n is even and r is odd, then the $(n/2)$-scaled MIR facet is the most important facet of $P(n, r)$. This is the case for all $P(n, r)$ studied by Dash and Günlük. However, we do not expect such a role for other divisors, independent of r. In the discussion after Theorem 14 we point out the invariance of the $(n/2)$-scaled MIR — when it exists (n is even and r is odd) — over isomorphic master polyhedra. For example, for $P(100, 4)$, the most important MIR facet is the 10-scaled MIR facet (neither 25 nor 50 are valid scaling parameters). This facet of $P(100, 4)$ is isomorphic to the 30-scaled MIR of $P(100, 28)$, which is therefore the most important MIR facet of $P(100, 28)$. See Cornuéjols, Li and Vandenbusche [43], who first suggested the possibility that the 1-scaled MIR inequality need not be uniformly superior to other scaled MIR inequalities.

The practical implications of the shooting results are not clear. First, most practical MIPs have upper and lower bounds on variables; at least one of these is ignored when deriving group cuts from corner polyhedra relaxations. Secondly, continuous variables are often present; then the 1-scaled MIR seems to have a special role. Recall from Eq. (18) that in any facet η of $P'(n, r)$, the coefficients of v_+ and v_- are $n\eta_1$ and $n\eta_{n-1}$, respectively. From Eq. (14), it follows that for any index i between 1 and $n-1$ $i\eta_1 \geq \eta_i$ and $i\eta_{n-1} \geq \eta_{n-i}$. Using $\eta_r = 1$, this implies that

$$\eta_1 \geq \frac{1}{r} \quad \text{and} \quad \eta_{n-1} \geq \frac{1}{n-r}. \tag{24}$$

For the 1-scaled MIR facet, $\eta_1 = 1/r$ and $\eta_{n-1} = 1/(n-r)$; therefore, the 1-scaled MIR inequality has the smallest possible coefficients for the continuous variables among all group cuts for Q.

Gomory, Johnson and Evans [23] proposed using FIFs based on important (as measured by shooting) facets of $P(n, r)$ with small n as valid inequalities for Q. Dash and Günlük [26] argue that one can trivially find inequalities dominating the ones generated by the approach above. The reason is that the important facets of $P(n, r)$ are scaled MIR and two-step MIR facets, and the valid inequalities for Q obtained from many of the corresponding FIFs are dominated by the scaled MIR and two-step MIR inequalities for Q. See Section 3.2.

4. MIR Closure

In this section, we discuss properties of the MIR closure of a polyhedral set $P = \{v \in \mathbb{R}^l,\ x \in \mathbb{Z}^n : Cv + Ax = b,\ v, x \geq 0\}$ with m constraints. We define the MIR *closure* of P as the set of points in P^{LP} which satisfy all MIR cuts for P, and denote it by P^{MIR}. Nemhauser and Wolsey's result [9] showing the equivalence of split cuts and MIR cuts for P implies that the *split closure* of P — defined as the set of points in P^{LP} satisfying all split cuts for P — equals its MIR closure. Cook, Kannan and Schrijver [11] showed that the split closure of P is a polyhedron. Andersen, Cornuéjols and Li [61], Vielma [62], and Dash, Günlük and Lodi [10] give alternative proofs that the split closure of a polyhedral set is a polyhedron. The latter proof is in terms of the MIR closure of P, and we discuss it below. Caprara and Letchford [38] studied the separation problem for split cuts, i.e., the problem of finding a violated split cut given a point $(v^*, x^*) \in P^{\text{LP}}$ or proving that no such cut exists. They proved that this separation problem is NP-hard.

For a vector w, let w^+ stand for $\max\{w, \mathbf{0}\}$, where the maximum is taken componentwise. In this section, we assume that A, C and b have integral components (this is without loss of generality, as we assumed earlier that they were rational matrices). Let

$$\begin{aligned}\Pi = \{(\lambda, c^+, \breve{\alpha}, \bar{\alpha}, \breve{\beta}, \bar{\beta}) \in \mathbb{R}^m \times \mathbb{R}^l \times \mathbb{R}^n \times \mathbb{Z}^n \times \mathbb{R} \times \mathbb{Z} : \quad & c^+ \geq \lambda C, \quad & c^+ \geq \mathbf{0}, \\ & \breve{\alpha} + \bar{\alpha} \geq \lambda A, \quad & \mathbf{1} \geq \breve{\alpha} \geq \mathbf{0}, \\ & \breve{\beta} + \bar{\beta} \leq \lambda b, \quad & 1 \geq \breve{\beta} \geq 0\},\end{aligned}$$

here $c^+, \breve{\alpha}, \bar{\alpha}$ are row vectors. Note that for any $(\lambda, c^+, \breve{\alpha}, \bar{\alpha}, \breve{\beta}, \bar{\beta}) \in \Pi$,

$$c^+ v + (\breve{\alpha} + \bar{\alpha})x \geq \breve{\beta} + \bar{\beta} \tag{25}$$

is valid for P^{LP} as it is a relaxation of $(\lambda C)v + (\lambda A)x = \lambda b$. Furthermore, using the basic mixed-integer inequality (1), we infer that

$$c^+ v + \breve{\alpha} x + \breve{\beta}\bar{\alpha} x \geq \breve{\beta}(\bar{\beta} + 1) \tag{26}$$

is a valid inequality for P. We call Eq. (26) a *relaxed* MIR *inequality* derived from the *base inequality* (25). (We use the notation $\breve{a}$ and $\breve{\beta}$ because for a fixed λ, the best choice for $\breve{\beta}$ equals $\widetilde{\lambda b}$, and the best choices for components of $\breve{a}$ are 0 or fractional parts of components of λA, see the next paragraph.) If $\breve{\beta} = 0$, then the relaxed MIR inequality is trivially satisfied by all points in P^{LP}. If $\breve{\beta} = 1$, then Eq. (26) is identical to its base inequality (25) and is satisfied by all points in P^{LP}. Further, Eq. (26) is a split cut for P derived from the disjunction $\bar{\alpha}x \leq \bar{\beta} \vee \bar{\alpha}x \geq \bar{\beta} + 1$, and is therefore violated by $(v^*, x^*) \in P^{\text{LP}}$ only if $\bar{\beta} < \bar{\alpha}x^* < \bar{\beta} + 1$. This implies the following lemma.

Lemma 16. *A relaxed* MIR *inequality* (26) *violated by* $(v^*, x^*) \in P^{\text{LP}}$ *satisfies* (i) $0 < \breve{\beta} < 1$, (ii) $0 < \Delta < 1$, *where* $\Delta = \bar{\beta} + 1 - \bar{\alpha}x^*$.

It is easy to see that the MIR inequality (5) for P is also a relaxed-MIR inequality. For a given multiplier vector λ, let α denote λA. Further, set $c^+ = (\lambda C)^+$, $\bar{\beta} = \lfloor \lambda b \rfloor$ and $\breve{\beta} = \lambda b - \lfloor \lambda b \rfloor$. Also, define $\breve{\alpha}$ and $\bar{\alpha}$ as follows: if $\alpha_i - \lfloor \alpha_i \rfloor < \breve{\beta}$ then $\breve{\alpha}_i = \alpha_i - \lfloor \alpha_i \rfloor$ and $\bar{\alpha}_i = \lfloor \alpha_i \rfloor$, otherwise $\breve{\alpha}_i = 0$ and $\bar{\alpha}_i = \lceil \alpha_i \rceil$. Clearly, $(\lambda, c^+, \breve{\alpha}, \bar{\alpha}, \breve{\beta}, \bar{\beta}) \in \Pi$ and the corresponding

relaxed MIR inequality (26) is the same as the MIR inequality (5). Therefore, every split cut is also a relaxed MIR inequality.

Recall the discussion in Section 2, where we observe that the MIR inequality is the strongest inequality of the form (3). This property and Lemma 16 are used by Dash, Günlük and Lodi [10, Lemma 6] to show that a point in P^{LP} satisfies all MIR inequalities, if and only if it satisfies all relaxed MIR inequalities. Therefore the MIR closure of P can be defined as

$$P^{\mathrm{MIR}} = \{(v, x) \in P^{\mathrm{LP}} : c^+ v + \breve{\alpha} x + \breve{\beta}\bar{\alpha} x \geq \breve{\beta}(\bar{\beta} + 1) \text{ for all } (\lambda, c^+, \breve{\alpha}, \bar{\alpha}, \breve{\beta}, \bar{\beta}) \in \Pi\}.$$

Therefore, for a given point $(v^*, x^*) \in P^{\mathrm{LP}}$, one can test if $(v^*, x^*) \in P^{\mathrm{MIR}}$ by solving the nonlinear integer program (MIR-SEP):

$$\begin{array}{ll} \max & \breve{\beta}(\bar{\beta} + 1) - (c^+ v^* + \breve{\alpha} x^* + \breve{\beta}\bar{\alpha} x^*) \\ \text{s.t.} & (\lambda^T, c^+, \breve{\alpha}, \bar{\alpha}, \breve{\beta}, \bar{\beta}) \in \Pi \end{array}$$

If every solution of MIR-SEP has objective value zero or less, then $(v^*, x^*) \in P^{\mathrm{MIR}}$. On the other hand, if some solution has positive objective value, it gives a violated relaxed MIR inequality. For a point (v^*, x^*), we define the violation of a relaxed MIR inequality to be its right-hand side minus its left-hand side evaluated at (v^*, x^*). The violation is bounded above by $\breve{\beta}(\bar{\beta} + 1 - \bar{\alpha} x^*)$ which is strictly less than 1 for a violated MIR inequality.

Observe that MIR-SEP becomes an LP if $\bar{\alpha}$ and $\bar{\beta}$ are fixed to some integral values. Let ϕ be a solution of MIR-SEP with objective value μ. Fix the values of $\bar{\alpha}$ and $\bar{\beta}$ in MIR-SEP to the corresponding values in ϕ, say $\bar{\alpha}_\phi$ and $\bar{\beta}_\phi$, respectively. The resulting LP has a basic optimal solution ϕ' with objective value $\geq \mu$. The LP constraints (other than the variable bounds) can be written as

$$\begin{bmatrix} A^T & -I & \\ C^T & & -I \\ b^T & & -1 \end{bmatrix} \begin{pmatrix} \lambda^T \\ \breve{\alpha}^T \\ c^{+T} \\ \breve{\beta} \end{pmatrix} \begin{matrix} \leq \\ \leq \\ \geq \end{matrix} \begin{pmatrix} \bar{\alpha}_\phi^T \\ 0 \\ \bar{\beta}_\phi \end{pmatrix}.$$

This implies the following result.

Theorem 17 ([10]). *If there is an* MIR *inequality violated by the point* (v^*, x^*), *then there is another* MIR *inequality violated by* (v^*, x^*) *for which* $\breve{\beta}$ *and the components of* $\lambda, \breve{\alpha}$ *are rational numbers with denominator equal to a subdeterminant of* $[A\ C\ b]$.

We will now argue that Theorem 17 implies that the MIR closure of P is a polyhedron. This argument is different from the one in [10]; there Theorem 17 is used in a different manner. Consider a polyhedron in $\mathbb{R}^k$ defined by $Gx + Hy \leq b$, let Φ be the maximum absolute value of subdeterminants of $[G\ H\ b]$. The convex hull of its x-integral points is a polyhedron, with extreme points whose x-coefficients have magnitude bounded by $(k+1)\Phi$, and other coefficients have encoding size bounded by a polynomial function of k and Φ. See Theorems 16.1 and 17.1 in [63]. By Lemma 16, every violated relaxed-MIR inequality satisfies $0 < \breve{\beta} < 1$. Therefore, Theorem 17 implies that $\breve{\beta}$ can be assumed to lie in the set $\mathcal{B} = \{r/s \in (0,1) : r, s \in \mathbb{Z},\ s \text{ is a subdeterminant of } [A\ C\ b]\}$. For any fixed $\breve{\beta}$, MIR-SEP becomes a mixed-integer program; call this MIR-SEP($\breve{\beta}$). The convex

hull of its $(\bar{\alpha}, \bar{\beta})$-integral solutions is a polyhedron, say $P'(\check{\beta})$. The set $\mathcal{M}$ formed by the union of the extreme points of $P'(\check{\beta})$ for $\check{\beta} \in \mathcal{B}$ is clearly finite. Further, the relaxed-MIR inequalities defined by this set define the MIR closure of P. After all, if a point (v^*, x^*) is not contained in the MIR closure of P, by Theorem 17, it is violated by some inequality with $\check{\beta}$ in $\mathcal{B}$ and therefore by some inequality in $\mathcal{M}$.

The above argument also implies that the relaxed-MIR cuts defining the MIR closure of P can be derived from disjunctions contained in a bounded set; more precisely $(\bar{\alpha}, \bar{\beta})$ are contained in the set $D = [-(n + l + 1)\Phi^2, (n + l + 1)\Phi^2]^{n+1}$, where Φ is the maximum absolute value of subdeterminants of $[A\ C\ b]$. Therefore, one can derive using the equivalence of split cuts and MIR cuts that the split closure of P equals

$$\bigcap_{(c,d)\in D} \text{conv}(P^{\text{LP}} \cap \{cx \le d\} \cup P^{\text{LP}} \cap \{cx \ge d + 1\}).$$

Further, the vector of multipliers λ also has bounded coefficients. This result is similar to [10, Lemma 21], where a bound on the magnitude of λ needed for nonredundant MIR cuts is given; the latter bound is sharper as the proof in [10] does not use such general arguments.

Lemma 18 ([10, Lemma 21])**.** *Assume that the coefficients in $Cv + Ax = b$ are integers. If there is an* MIR *inequality violated by the point (v^*, x^*), then there is another* MIR *inequality violated by (v^*, x^*) with $\lambda_i \in (-m\Psi, m\Psi)$, where m is the number of rows in $Cv + Ax = b$, and Ψ is the largest absolute value of subdeterminants of C.*

If P has no continuous variables, it is easier to show that the MIR closure of P is a polyhedron; one can show that $\lambda \in (0, 1)^m$. Caprara and Letchford [38, Lemma 1] show (in a different form) that $\lambda \in (-1, 1)^m$, when P has no continuous variables; this yields a short proof that the split closure of P is a polyhedron.

Assume (in this paragraph) that P has no continuous variables, i.e., $C = 0$. The problem of obtaining a violated Gomory – Chvátal cut be framed as a mixed-integer program, see Bockmayr and Eisenbrand [32] and Fischetti and Lodi [31]. One can then infer, as in the discussion on the MIR closure above, that the Chvátal closure of P is a polyhedron. Bockmayr and Eisenbrand study this mixed-integer program and use the fact that its integral hull has polynomially many extreme points in fixed dimension to show that the Chvátal closure of P in fixed dimension has polynomially many facets. In fixed dimension, the number of extreme points of $P'(\check{\beta})$ can be shown to be bounded by a polynomial function of the encoding size of $Cv + Ax = b$. If one can show that $\mathcal{M}$ is the union of $P'(\check{\beta})$ for polynomially many choices of $\check{\beta}$, then one could give a positive answer to Eisenbrand's question [64]: Does the MIR (or split) closure of P have polynomially many facets in fixed dimension? Dash, Günlük and Lodi show that the separation problem for MIR cuts can be framed as a mixed-integer program (see the discussion below). Unfortunately, the number of variables in their separation MIP depends on the encoding size of $[A\ C\ b]$ and one cannot use the technique of Bockmayr and Eisenbrand to give a positive answer to Eisenbrand's question.

We next describe the approximate MIR separation model in [10] obtained by approximately linearizing the product $\check{\beta}(\bar{\beta} + 1 - \bar{\alpha}x^*)$ in the objective function of MIR-SEP. Define a new variable Δ that stands for $(\bar{\beta} + 1 - \bar{\alpha}x)$. Let $\tilde{\beta} \le \check{\beta}$ be an approximation of $\check{\beta}$ which is representable over some $\mathcal{E} = \{\epsilon_k : k \in K\}$. Let a number δ be *representable* over

$\mathcal{E}$ if $\delta = \sum_{k\in\bar{K}} \epsilon_k$ for some $\bar{K} \subseteq K$. One can write $\tilde{\beta}$ as $\sum_{k\in K} \epsilon_k \pi_k$ using binary variables π_k, and approximate $\check{\beta}\Delta$ by $\tilde{\beta}\Delta$. This last term can be written as $\sum_{k\in K} \epsilon_k \pi_k \Delta$. This results in the following approximate MIP model APPX-MIR-SEP for the separation of the most violated MIR inequality:

$$\max \quad \sum_{k\in K} \epsilon_k \Delta_k - (c^+ v^* + \check{\alpha} x^*) \tag{27}$$

$$\text{s.t.} \quad (\lambda, c^+, \check{\alpha}, \bar{\alpha}, \check{\beta}, \bar{\beta}) \in \Pi \tag{28}$$

$$\check{\beta} \geq \sum_{k\in K} \epsilon_k \pi_k \tag{29}$$

$$\Delta = (\bar{\beta} + 1) - \bar{\alpha} x^* \tag{30}$$

$$\Delta_k \leq \Delta \qquad \forall k \in K \tag{31}$$

$$\Delta_k \leq \pi_k \qquad \forall k \in K \tag{32}$$

$$\pi \in \{0,1\}^{|K|} \tag{33}$$

Let z^{sep} and $z^{\text{apx-sep}}$ denote the optimal values of MIR-SEP and APPX-MIR-SEP, respectively. For any integral solution of APPX-MIR-SEP, $(\lambda, c^+, \check{\alpha}, \bar{\alpha}, \check{\beta}, \bar{\beta}) \in \Pi$ and

$$\sum_{k\in K} \epsilon_k \Delta_k \leq \sum_{k\in K} \epsilon_k \Delta \pi_k \leq \check{\beta}\Delta,$$

implying that $z^{\text{sep}} \geq z^{\text{apx-sep}}$. In other words, if the approximate separation problem finds a solution with objective function value $z^{\text{apx-sep}} > 0$, the corresponding MIR cut is violated by at least as much. In the computational experiments of Dash, Günlük and Lodi with APPX-MIR-SEP, they use $\mathcal{E} = \{2^{-k} : k = 1, \ldots, \bar{k}\}$ for some small number $\bar{k}$ (between 5 and 7). They prove that with this choice of $\mathcal{E}$, APPX-MIR-SEP yields a violated MIR cut provided that there is an MIR cut with a "large enough" violation. More precisely, they prove [10, Theorem 8] that $z^{\text{apx-sep}} > z^{\text{sep}} - 2^{-\bar{k}}$.

If the relaxed-MIR inequality $\mathcal{I}$ with maximum violation has a value of $\check{\beta}$ which is representable over $\mathcal{E}$, one can choose $\pi \in \{0,1\}^{\bar{k}}$ such that $\check{\beta} = \sum_{k\in K} \epsilon_k \pi_k$. Set $\Delta = \bar{\beta} + 1 - \bar{\alpha} x^*$. Set $\Delta_k = 0$ if $\pi_k = 0$, and $\Delta_k = \Delta$ if $\pi_k = 1$. Then $\Delta_k = \pi_k \Delta$ for all $k \in K$, and $\check{\beta}\Delta = \sum_{k\in K} \epsilon_k \Delta_k$. Therefore, the relaxed-MIR inequality $\mathcal{I}$ yields an optimal solution of APPX-MIR-SEP whose objective function value equals the violation of the $\mathcal{I}$. In other words, $z^{\text{sep}} = z^{\text{apx-sep}}$. Theorem 17 implies that $\check{\beta}$ in a violated MIR cut can be assumed to be a rational number with a denominator equal to a subdeterminant of $[A\ C\ b]$. This implies the following result.

Theorem 19 ([10])**.** *Let* Φ *be the least common multiple of all subdeterminants of* $[A\ C\ b]$*,* $K = \{1, \ldots, \log \Phi\}$*, and* $\mathcal{E} = \{\epsilon_k = 2^k/\Phi, \forall k \in K\}$*. Then* APPX-MIR-SEP *is an exact model for finding violated* MIR *cuts.*

Caprara and Letchford [38], and more recently, Balas and Saxena [36], present optimization models for finding a violated split cut for P. In both papers, the authors use two sets of multipliers that guarantee that the split cut is valid for both sides of the disjunction; see Eqs. (8)–(13) in [38] and Eq. (SP) in [36]. It is argued in [10] that the separation model in Caprara and Letchford (Eqs. (8)–(13)) actually finds the most violated

MIR cut (the objective function equals four times the objective function of MIR-SEP). In other words, an optimal solution of MIR-SEP is also optimal for the Caprara – Letchford model and vice-versa. Similarly, the Balas – Saxena model (Eq. (2.1) or (PMILP) in [36]) is equivalent to MIR-SEP, and has the same objective function. Consequently, the models in [38] and [36] are also equivalent to each other.

Caprara and Letchford do not perform any computational tests with their model. As for Balas and Saxena, instead of bounding $\check{\beta}$ by $\tilde{\beta}$ and linearizing $\tilde{\beta}\Delta$ as in [10], they fix the term corresponding to $1 - \check{\beta}$ in their model to specific values between 0 and 1/2, and for each value, solve an MIP to obtain a violated split cut. We will discuss some of the computational results in [10] and [36] in the next section.

5. Computational Issues

Balas, Ceria, Cornuéjols and Natraj [5] showed that GMI cuts (MIR cuts with simplex tableau rows as base inequalities), when added in *rounds* (all violated GMI cuts for the current optimal tableau are added simultaneously), are very useful in solving the general mixed-integer programs in MIPLIB 3.0. Bixby et al. [7] extended this observation to larger problem sets, and performed additional experiments confirming the usefulness of GMI cuts relative to other cuts in the CPLEX solver. Marchand and Wolsey [13] proposed a different, effective way of generating MIR inequalities; their heuristic aggregates constraints of the original formulation to obtain base inequalities which are different from simplex tableau rows. Despite their effectiveness, GMI cuts often cause numerical difficulties, and this aspect limits their use. In some cases, simple implementations yield invalid cuts [65]; in other cases, adding too many GMI cuts makes the resulting LP hard to solve. The issue of invalid cuts is addressed in a recent paper by Cook et. al. [66] who generate *provably valid* GMI cuts with negligible reduction in performance (as measured by computing time and quality of bounds).

After Gomory [15] introduced group relaxations in 1965, White [16] and Shapiro [17] showed that for a number of small IPs, group relaxations yielded strong bounds on optimal values. For 12 of the 14 problems in the latter paper where the IP optimal is known and is different from the LP relaxation value, the group relaxation bound equals the optimal value. Gorry, Northup and Shapiro [18] performed a detailed study of a group relaxation based branch-and-bound algorithm, and solved problems with up to 176 rows and 2385 columns. In the above papers, group relaxations were solved via dynamic programming algorithms. See Salkin [67, Chapter 9] for a discussion on early computational work on this topic. In general it is NP-hard to solve the group relaxation problem or, equivalently, to optimize over an arbitrary corner polyhedron [68] (or strengthened corner polyhedra [69]). In a recent computational study, Fischetti and Monaci [69] optimize over the corner polyhedra (also strengthened corner polyhedra) associated with optimal vertices of LP relaxations of some MIPLIB 3.0 and MIPLIB 2003 instances by solving MIPs and show that the average integrality gap closed is 23.61% (34.48%) as opposed to 25.32% with GMI cuts. The gap closed using corner polyhedra relaxations is worse than that with GMI cuts; this is because throwing away the nonactive bounds on variables often results in weak bounds on optimum values for MIPLIB instances, especially those which have binary variables.

Two interesting directions of research extending the above work consist of (a) generating valid inequalities other than MIR cuts — especially group cuts — from the same

base inequalities, and (b) aggregating constraints of the original formulation to obtain base inequalities different from simplex tableau rows or those generated by Marchand and Wolsey. The difference between recent work on (a) and earlier work in the previous paragraph is that the corner polyhedron is used to generate cutting planes, rather than as a relaxation by itself.

For the mixed-integer knapsack polyhedron, Atamtürk [70] developed valid inequalities (via lifting) which use upper and lower bounds on variables, unlike the MIR cut in Eq. (5) which uses only one of the bounds. For a collection of randomly generated multiple knapsack instances [71], his inequalities along with MIR cuts close a significantly larger fraction of the integrality gap than MIR cuts alone. He uses scaled constraints of the original formulation as base inequalities. Fischetti and Saturni [42] and Dash, Goycoolea and Günlük [24] study the effectiveness of group cuts derived from simplex tableau rows relative to GMI cuts. The first paper contains a study of group cuts derived via interpolation; violated cuts of this type are obtained by solving an LP. The second paper presents a heuristic to generate violated two-step MIR inequalities, and shows that they are useful for the randomly generated instances of Atamtürk. For the unbounded instances (variables are nonnegative and not bounded above) in Atamtürk's data set, two-step MIR inequalities derived from rows of the initial optimal simplex tableau combined with GMI cuts (derived from the same tableau rows) close 78.65% of the integrality gap, whereas GMI cuts alone close only 56.25% of the integrality gap. For these instances, two-step MIR inequalities seem to be as effective as the lifted knapsack cuts of Atamtürk, and more effective than K-cuts [43]. Fischetti and Saturni show that K-cuts for $K = 1, \ldots, 50$ (or $1-50$ scaled GMI cuts) close 75.99% of the integrality gap. Interestingly, the integrality gap closed via (strengthened or otherwise) corner polyhedra relaxations [69] is 79.43%, which is only slightly better than the gap closed using two-step MIR inequalities. These experiments suggest that the shooting experiments discussed earlier yield useful information for problems which resemble master cyclic group polyhedra in that the integer variables can take values from a large interval, and the constraints have general (not 0-1) coefficients. For the bounded Atamtürk instances, we note that the lifted knapsack cuts are more (about $5-6\%$) effective in closing the integrality gap than two-step MIR inequalities.

The authors in [24] observe that for the problems in MIPLIB 3.0, the gap closed by one round of two-step MIR cuts + GMI cuts is essentially the same as that closed by one round of GMI cuts alone. Interpolated group cuts, and $1-50$ scaled GMI cuts also seem to behave similarly [42]. Motivated by this observation, Dash and Günlük [27] demonstrate that for a collection of practical instances (from MIPLIB 3.0, MIPLIB 2003, MILPLib [72], and instances from [73]), after GMI cuts derived from the initial optimal tableau rows are added, the solution of the resulting relaxation satisfies all non-GMI group cuts derived from the initial tableau rows for 35% of the instances. In other words, additional group cuts beyond the GMI cut derived from the initial optimal tableau rows are not useful at all for these instances. On the other hand, for 82% of the remaining instances (which potentially have violated group cuts), one can find violated two-step MIR inequalities. Thus, unlike the Atamtürk instances, group cuts from single tableau rows do not seem to be very useful for the above instances. However, two-step MIR inequalities seem to be important relative to other group cuts. Fukasawa and Goycoolea [74] extend the above experiments for MIPLIB instances to show that no other valid inequalities derived from the mixed-integer knapsacks defined by initial optimal tableau rows

and bounds on variables improve the integrality gap by a significant margin. The above experiments suggest that for MIPLIB instances, in order to obtain cutting planes which improve the integrality gap closed by GMI cuts, it is important to use information from multiple constraints simultaneously. Some recent computational work in this direction can be found in Espinoza [75].

Bonami and Minoux [34] approximately optimize over the lift-and-project closure of 0-1 mixed integer programs via the equivalence of optimization and separation; in this context the separation problem can be framed as a linear program. Lift-and-project cuts are split cuts derived from disjunctions of the form $x_i \leq 0 \vee x_i \geq 1$ for some integral variable x_i. Balas and Perregard [76] proposed a method to generate *strengthened lift-and-project* cuts from simplex tableau rows, and their method and some variants were implemented by Balas and Bonami [77] with encouraging results. Independently, Fischetti and Lodi [31] show that for many practical MIPs, one can separate points from the Chvátal closure of pure integer programs in reasonable time by formulating the separation problem as an MIP and solving it with a general MIP solver. They apply their separation algorithm to approximately optimize over the Chvátal closures of MIPLIB instances and obtain tight bounds on optimal solution values for many instances. Bonami et al. [35] extend the definition of Gomory – Chvátal cuts to mixed integer programs (*projected Gomory – Chvátal cuts*) and use a similar MIP based separation procedure to obtain bounds for mixed-integer programs in MIPLIB 3.0.

Optimizing over the split closure of MIPs should lead to stronger bounds than in the papers above. Balas and Saxena [36] approximately optimize over the split closure of MIPLIB instances to obtain strong bounds on their optimal values, and so do Dash, Günlük and Lodi [10], who combine APPX-MIR-SEP with some heuristics to find violated MIR cuts. In MIPLIB 3.0, 62 of the 65 instances have a nonzero integrality gap, other than *dsbmip*, *enigma* and *noswot*. Balas and Saxena show that for these instances (other than *arki001* where they use cuts which potentially have MIR rank 2), at least 71.3% of the integrality gap can be closed by optimizing over the split closure (the computation time is quite high though). For these 62 instances, Dash, Günlük and Lodi show that at least 59.3% of the integrality gap can be closed in one hour of computation time; a round of GMI cuts closes only 28.4% of the integrality gap. For the 21 pure IPs, the gaps closed by Dash, Günlük and Lodi, Fischetti and Lodi (GC cuts), Balas and Saxena, and GMI cuts are on the average 59.2%, 56.5%, 76.0% and 29.8%, respectively. For the remaining 41 MIPs, the corresponding numbers (with projected-GC cuts instead of GC cuts) are, respectively, 59.3, 28.8, 68.9, and 27.7. Note that in Bonami et. al. [35], the bounds for projected-GC cuts are obtained with only 20 minutes of computation, whereas the bounds with GC cuts in [31] are obtained after 3 hours of computation, on the average.

6. Proof Complexity

If NP $\neq$ coNP, then it cannot be true that for arbitrary $Ax \leq b$ without 0-1 solutions, there is a polynomial-size (in the encoding size of A, b) certificate of the absence of 0-1 solutions. We next discuss a recent result in [41] which proves the existence of a family of inequality systems without 0-1 solutions for which MIR cutting-plane proofs of 0-1 infeasibility have exponential length.

A *boolean circuit* can be viewed as a description of the elementary steps in an algorithm via a directed acyclic graph with three types of nodes: input nodes — nodes with no

incoming arcs, a single output node — the only node with no outgoing arcs, and computation nodes (also called gates), each of which is labelled by one of the Boolean functions $\wedge, \vee$, and $\neg$. For nodes i and j, an arc ij means that the value computed at i is used as an input to the gate at node j. A computation is represented by placing 0-1 values on the input gates, and then recursively applying the gates to inputs on incoming arcs, till the function at the output node is evaluated. A function $f: \mathbb{R}^n \to \mathbb{R}$ is *monotone* if for x, y in $\mathbb{R}^n$, $x \leq y$ implies $f(x) \leq f(y)$. *Monotone operations* are monotone functions with one or two inputs; some examples are

$$tx, \qquad r + x, \qquad x + y, \qquad \lfloor x \rfloor, \qquad \text{thr}(x, 0)$$

where t is a nonnegative constant, x and y are real variables, and r is a real constant; $\text{thr}(x, 0)$ is a *threshold function* which returns 0, if $x < 0$, and 1 otherwise. The functions $\wedge$ and $\vee$ are monotone operations over the domain $\{0, 1\}$. The function $f(x, y) = x - y$, where $x, y \in \mathbb{R}$, is not monotone. A *monotone boolean circuit* uses only $\wedge$ gates and $\vee$ gates; a *monotone real circuit* is one with arbitrary monotone operations as gates.

Consider $\text{CLIQUE}_{k,n}$ (say k is a fixed function of n), the function which takes as input n-node graphs (represented by incidence vectors of their edges) and returns 1 if the graph has a clique of size k or more, and 0 otherwise. This function is monotone, as adding edges to a graph (changing some zeros to ones in the incidence vector) causes the maximum clique size to increase. Every monotone boolean function can be computed by a monotone boolean circuit. Razborov [78] showed that any monotone boolean circuit solving $CLIQUE_{k,n}$ (for appropriate k) has a super-polynomial number of gates, and Alon and Boppana [79] strengthened his bound to an exponential lower bound. Pudlák [40], and independently, Cook and Haken [80], proved that the above bounds hold for monotone real circuits.

Theorem 20 ([40]). *Let C_n be a monotone real circuit which takes as input graphs on n nodes* (*given as incidence vectors of edges*), *and returns* 1 *if the input graph contains a clique of size $k = \lfloor n^{2/3} \rfloor$, and* 0 *if the graph contains a coloring of size $k - 1$* (*and returns 0 or 1 for all other graphs*). *Then $|C_n| \geq 2^{\Omega((n/\log n)^{1/3})}$.*

Pudlák [40] presented a set of linear inequalities $\mathcal{I}$ related to the problem of Theorem 20 such that if $\mathcal{I}$ has a 0-1 solution, then there is a graph on n nodes which has both a clique of size k and a coloring of size $k - 1$. He proved that given a Gomory – Chvátal cutting plane proof $\mathcal{P}$ with L cuts which proves that $\mathcal{I}$ has no 0-1 solution, one can construct a monotone real circuit with $O(\text{poly}(\text{size}(\mathcal{P})))$ gates solving $\text{CLIQUE}_{k,n}$. Therefore L is exponential in n. Pudlák used the following properties of Gomory – Chvátal cuts in mapping short cutting-plane proofs to small monotone real circuits (here g, h are row vectors, r, s, t are numbers, x, y, z, c, e, f are column vectors):

1. If $gx + hy \leq t$ is a Gomory – Chvátal cut for $Ax + By \leq c$, then for any 0-1 vector $\overline{y}$ with the same dimension as y, $gx \leq t - h\overline{y}$ is a Gomory-Chvátal cut for $Ax \leq c - B\overline{y}$;
2. if $gx + hy \leq t$ is a Gomory – Chvátal cut for $Ax \leq e$, $By \leq f$, then there are numbers r and s such that $gx \leq r$ is a Gomory – Chvátal cut for $Ax \leq e$, and $hy \leq s$ is a Gomory – Chvátal cut for $By \leq f$, and $r + s \leq t$.
3. The number r (or s) can be computed from A, e (or B, f) with polynomially many monotone operations.

Property (1) is easy to prove for all integer vectors $\bar{y}$. Consider property (2). As $gx + hy \leq t$ is a Gomory – Chvátal cut for $Ax \leq e$, $By \leq f$, we can assume that g, h, t are integral and there are nonnegative multiplier vectors λ, μ such that $g = \lambda A$, $h = \mu B$, $t = \lfloor \lambda e + \mu f \rfloor$. Clearly, $gx \leq \lfloor \lambda e \rfloor$ ($hy \leq \mu f$) is a Gomory – Chvátal cut for $Ax \leq e$ ($By \leq f$), and so is $hy \leq \lfloor \mu f \rfloor$, and $\lfloor \lambda e \rfloor + \lfloor \mu f \rfloor \leq \lfloor \lambda e + \mu f \rfloor$. Property (3) also follows from this; the number $\lfloor \lambda e \rfloor$ can be computed from e via polynomially many monotone operations (the coefficients of λ are treated as nonnegative constants).

In general, for a class of cutting planes C, if we can prove properties (1) – (3) in the previous paragraph for C, then given a C-proof of the fact that Pudlák's inequality system $\mathcal{I}$ has no 0-1 solution, we can construct a monotone real circuit solving $CLIQUE_{k,n}$ with polynomially many gates (in the size of the C-proof). This would yield an exponential worst-case lower bound on the size of C-proofs certifying that $\mathcal{I}$ has no 0-1 solution (of course, additional details have to be verified). In general, for many classes of cutting planes properties (1) and (2) hold, e.g., the matrix cuts of Lovász and Schrijver (cuts based on the N and N_+ operators; see [81, 82]. Property (3) is often hard to prove, and is not known to hold for matrix cuts. We prove in [41] that slight variants of properties (2) and (3) hold for MIR cuts, and thereby obtain an exponential worst-case lower bound on the complexity of MIR cuts. We state this result below. For completeness, we explicitly give the inequality system $\mathcal{I}$.

Let $k = \lfloor n^{2/3} \rfloor$. Let z be a vector of $n(n-1)/2$ 0-1 variables, such that every 0-1 assignment to z corresponds to the incidence vector of a graph on n nodes (assume nodes are numbered from $1, \ldots, n$). Let x be the 0-1 vector of variables $(x_i \mid i = 1, \ldots, n)$ and let y be the 0-1 vector of variables $(y_{ij} \mid i = 1, \ldots, n, j = 1, \ldots, k-1)$. Consider the inequalities

$$\sum_{i=1}^{n} x_i \geq k, \tag{34}$$

$$x_i + x_j \leq 1 + z_{ij}, \qquad \forall i, j \in N, \text{ with } i < j, \tag{35}$$

$$\sum_{j=1}^{k-1} y_{ij} = 1, \qquad \forall i \in N, \tag{36}$$

$$y_{is} + y_{js} \leq 2 - z_{ij}, \qquad \forall i, j \in N \text{ with } i < j, \text{ and } \forall s \in \{1, \ldots, k-1\}. \tag{37}$$

Then, in any 0-1 solution of the above inequalities, the set of nodes $\{i \mid x_i = 1\}$ forms a clique of size k or more, and for all $j \in \{1, \ldots, k-1\}$, the set $\{i \mid y_{ij} = 1\}$ is a stable set. Thus, the variables y_{ij} define a mapping of nodes in a graph to $k-1$ colors in a proper coloring. Let $Ax + Cz \leq e$ stand for the inequalities (34) and (35), along with the bounds $\mathbf{0} \leq x \leq \mathbf{1}$. Let $By + Dz \leq f$ stand for the inequalities (36) and (37), along with the bounds $\mathbf{0} \leq y \leq \mathbf{1}$ and $\mathbf{0} \leq z \leq \mathbf{1}$. Then any 0-1 solution of $Ax + Cz \leq e$ and $By + Dz \leq f$ corresponds to a graph which has both a clique of size k, and a coloring of size $k-1$. Clearly, no such 0-1 solution exists. Note that the above inequalities have $O(n^3)$ variables and constraints.

Theorem 21 ([41]). *Every* MIR *cutting-plane proof of* $\mathbf{0}^T x + \mathbf{0}^T y + \mathbf{0}^T z \leq -1$ *from* $Ax + Cz \leq e$ *and* $By + Dz \leq f$ *has exponential length.*

Many families of inequalities are either special cases of MIR cuts (e.g., Gomory–Chvátal cuts, lift-and-project cuts), or can be obtained as rank k MIR cuts for some fixed number k. For example, the two-step MIR inequalities have MIR rank 2 or less. Therefore, one trivially obtains exponential worst-case lower bounds for the complexity of cutting plane proofs for all such families of cutting planes.

The technique of deriving a polynomial size monotone circuit from a proof of infeasibility is called *monotone interpolation*, and was proposed by Krajíček [83, 84] to establish lower bounds on the lengths of proofs in different proof systems. Razborov [85], and Bonet, Pitassi and Raz [86], first used this idea to prove exponential lower bounds for some proof systems.

Acknowledgements

I would like to thank Vašek Chvátal for inviting to me present the lectures at the NATO Summer School on which this survey is based. I would also like to thank Oktay Günlük for numerous productive discussions on corner polyhedra.

References

[1] ILOG (2003) *ILOG CPLEX 9.0 User's Manual*.
[2] Dash Associates, Blisworth (2006) *XPRESS-MP Reference Manual Release 17*.
[3] Gomory, R. E. (1960) An algorithm for the mixed integer problem. Tech. Rep. RM-2597, RAND Corporation, Santa Monica, CA.
[4] Geoffrion, A. M. and Graves, G. W. (1974) Multicommodity distribution system design by Benders Decomposition. *Management Science*, **20**, 822–844.
[5] Balas, E., Ceria, S., Cornuéjols, G., and Natraj, N. (1996) Gomory cuts revisited. *Operations Research Letters*, **19**, 1–9.
[6] Cornuéjols, G. (2007) Revival of the Gomory cuts in the 1990's. *Annals of Operations Research*, **149**, 63–66.
[7] Bixby, R. E., Fenelon, M., Gu, Z., Rothberg, E., and Wunderling, R. (2000) MIP: theory and practice—closing the gap. Powell, M. J. D. and Scholtes, S. (eds.), *System Modelling and Optimization*, (Cambridge, 1999), pp. 19–49, Kluwer, Boston, MA.
[8] Nemhauser, G. L. and Wolsey, L. A. (1988) *Integer and Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, New York, NY.
[9] Nemhauser, G. L. and Wolsey, L. A. (1990) A recursive procedure to generate all cuts for 0-1 mixed integer programs. *Mathematical Programming*, **46**, 379–390.
[10] Dash, S., Günlük, O., and Lodi, A. (2010) MIR closures of polyhedral sets. *Mathematical Programming*, **121**, 33–60.
[11] Cook, W., Kannan, R., and Schrijver, A. (1990) Chvátal closures for mixed integer programming problems. *Mathematical Programming, Series A*, **47**, 155–174.
[12] Balas, E. (1979) Disjunctive programming. *Annals of Discrete Mathematics*, **5**, 3–51.
[13] Marchand, H. and Wolsey, L. A. (2001) Aggregation and mixed integer rounding to solve MIPs. *Operations Research*, **49**, 363–371.
[14] Wolsey, L. A. (1998) *Integer Programming*. Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, New York, NY.
[15] Gomory, R. E. (1965) On the relation between integer and noninteger solutions to linear programs. *Proceedings of the National Academy of Sciences of the United States of America*, **53**, 260–265.
[16] White, W. W. (1966) On a group theoretic approach to linear integer programming. Tech. Rep. 66-27, Operations Research Center, University of California, Berkeley.
[17] Shapiro, J. F. (1968) Group theoretic algorithms for the integer programming problem. II. Extension to a general algorithm. *Operations Research*, **16**, 928–947.

[18] Gorry, G. A., Northup, W. D., and Shapiro, J. F. (1973) Computational experience with a group theoretic integer programming algorithm. *Mathematical Programming*, **4**, 171–192.
[19] Geoffrion, A. M. (1976) A guided tour of recent practical advances in integer linear programming. *Omega*, **4**, 49–57.
[20] Gomory, R. E. (1969) Some polyhedra related to combinatorial problems. *Linear Algebra and its Applications*, **2**, 451–558.
[21] Gomory, R. E. and Johnson, E. L. (1972) Some continuous functions related to corner polyhedra. *Mathematical Programming*, **3**, 23–85.
[22] Aráoz, J., Evans, L., Gomory, R. E., and Johnson, E. L. (2003) Cyclic groups and knapsack facets. *Mathematical Programming*, **96**, 377–408.
[23] Gomory, R. E., Johnson, E. L., and Evans, L. (2003) Corner polyhedra and their connection with cutting planes. *Mathematical Programming, Series B*, **96**, 321–339.
[24] Dash, S., Goycoolea, M., and Gunluk, O. (2010) Two-step MIR inequalities for mixed integer programs. *INFORMS Journal on Computing*, **22**, 236–249.
[25] Dash, S. and Günlük, O. (2006) Valid inequalities based on simple mixed-integer sets. *Mathematical Programming, Series A*, **105**, 29–53.
[26] Dash, S. and Günlük, O. (2006) Valid inequalities based on the interpolation procedure. *Mathematical Programming*, **106**, 111–136.
[27] Dash, S. and Günlük, O. (2008) On the strength of Gomory mixed-integer cuts as group cuts. *Mathematical Programming*, **115**, 387–407.
[28] Dey, S. S. and Richard, J.-P. P. (2008) Facets of two-dimensional infinite group problems. *Mathematics of Operations Research*, **33**, 140–166.
[29] Dey, S. S. and Richard, J.-P. P. (2010) Relations between facets of low- and high-dimensional group problems. *Mathematical Programming*, **123**, 285–313.
[30] Chvátal, V. (1973) Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, **4**, 305–337.
[31] Fischetti, M. and Lodi, A. (2007) Optimizing over the first Chvátal closure. *Mathematical Programming*, **110**, 3–20.
[32] Bockmayr, A. and Eisenbrand, F. (2001) Cutting planes and the elementary closure in fixed dimension. *Mathematics of Operations Research*, **26**, 304–312.
[33] Hunsaker, B. and Tovey, C. A. Separation of rank 1 Chvátal–Gomory inequalities. *Ninth INFORMS Computing Society Conference*, (Annapolis, MD, 2005).
[34] Bonami, P. and Minoux, M. (2005) Using rank-1 lift-and-project closures to generate cuts for 0-1 MIPs, a computational investigation. *Discrete Optimization*, **2**, 288–307.
[35] Bonami, P., Cornuéjols, G., Dash, S., Fischetti, M., and Lodi, A. (2008) Projected Chvátal–Gomory cuts for mixed integer linear programs. *Mathematical Programming, Series A*, **113**, 241–257.
[36] Balas, E. and Saxena, A. (2008) Optimizing over the split closure. *Mathematical Programming, Series A*, **113**, 219–240.
[37] Eisenbrand, F. (1999) On the membership problem for the elementary closure of a polyhedron. *Combinatorica*, **19**, 297–300.
[38] Caprara, A. and Letchford, A. N. (2003) On the separation of split cuts and related inequalities. *Mathematical Programming*, **94**, 279–294.
[39] Gomory, R. E. (1958) Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, **64**, 275–278.
[40] Pudlák, P. (1997) Lower bounds for resolution and cutting plane proofs and monotone computations. *The Journal of Symbolic Logic*, **62**, 981–998.
[41] Dash, S. (2010) On the complexity of cutting-plane proofs using split cuts. *Operations Research Letters*, **38**, 109–114.
[42] Fischetti, M. and Saturni, C. (2007) Mixed-integer cuts from cyclic groups. *Mathematical Programming*, **109**, 27–53.
[43] Cornuéjols, G., Li, Y., and Vandenbussche, D. (2003) *k*-cuts: a variation of Gomory mixed integer cuts from the LP tableau. *INFORMS Journal on Computing*, **15**, 385–396.
[44] Applegate, D., Bixby, R., Chvátal, V., and Cook, W. (2001) TSP cuts which do not conform to the template paradigm. Jünger, M. and Naddef, D. (eds.), *Computational Combinatorial Optimization*, (Schloß Dagstuhl, 2000), vol. 2241 of *Lecture Notes in Computer Science*, pp. 261–303, Springer, Berlin.
[45] Günlük, O. and Pochet, Y. (2001) Mixing mixed-integer inequalities. *Mathematical Programming, Series*

A, **90**, 429–457.
[46] Espinoza, D. G. (2006) *On Linear Programming, Integer Programming and Cutting Planes*. Ph.D. thesis, Georgia Institute of Technology.
[47] Atamtürk, A. and Rajan, D. (2004) alid inequalities for mixed-integer knapsack from two-integer variable restrictions. Tech. Rep. BCOL.04.02, Department of Industrial Engineering and Operations Research, University of California, Berkeley.
[48] Agra, A. and Constantino, M. (2006) Description of 2-integer continuous knapsack polyhedra. *Discrete Optimization*, **3**, 95–110.
[49] Conforti, M. and Wolsey, L. A. (2008) Compact formulations as a union of polyhedra. *Mathematical Programming*, **114**, 277–289.
[50] Letchford, A. N. and Lodi, A. (2002) Strengthening Chvátal–Gomory cuts and Gomory fractional cuts. *Operations Research Letters*, **30**, 74–82.
[51] Kianfar, K. and Fathi, Y. (2009) Generalized mixed integer rounding inequalities: facets for infinite group polyhedra. *Mathematical Programming, Series A*, **120**, 313–346.
[52] Dash, S., Fukasawa, R., and Günlük, O. (2008) On a generalization of the master cyclic group polyhedron. *Mathematical Programming*, to appear.
[53] Uchoa, E. Robust branch-and-cut-and-price for the CMST problem and extended capacity cuts. *IMA Hot Topics Workshop: Mixed-Integer Programming*, (Minneapolis, MN, 2005).
[54] Gomory, R. E. and Johnson, E. L. (1972) Some continuous functions related to corner polyhedra. II. *Mathematical Programming*, **3**, 359–389.
[55] Andersen, K., Louveaux, Q., Weismantel, R., and Wolsey, L. A. (2007) Inequalities from two rows of a simplex tableau. Fischetti, M. and Williamson, D. P. (eds.), *Integer Programming and Combinatorial Optimization*, (Ithaca, NY, 2007), vol. 4513 of *Lecture Notes in Computer Science*, pp. 1–15, Springer, Berlin.
[56] Cornuéjols, G. and Margot, F. (2009) On the facets of mixed integer programs with two integer variables and two constraints. *Mathematical Programming, Series A*, **120**, 429–456, 10.1007/s10107-008-0221-1.
[57] Miller, L. A., Li, Y., and Richard, J.-P. P. (2008) New inequalities for finite and infinite group problems from approximate lifting. *Naval Research Logistics*, **55**, 172–191.
[58] Kuhn, H. W. (1966) Discussion. Robinson, L. (ed.), *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems*, (Yorktown Heights, NY, 1964), pp. 118–121, IBM, White Plains, NY.
[59] Kuhn, H. W. (1991) Nonlinear programming: a historical note. Lenstra, J. K., Rinnooy Kan, A. H. G., and Schrijver, A. (eds.), *History of Mathematical Programming*, pp. 82–96, North-Holland.
[60] Evans, L. (2002) *Cyclic Groups and Knapsack Facets with Applications to Cutting Planes*. Ph.D. thesis, Georgia Institute of Technology.
[61] Andersen, K., Cornuéjols, G., and Li, Y. (2005) Split closure and intersection cuts. *Mathematical Programming*, **102**, 457–493.
[62] Vielma, J. P. (2007) A constructive characterization of the split closure of a mixed integer linear program. *Operations Research Letters*, **35**, 29–35.
[63] Schrijver, A. (1986) *Theory of Linear and Integer Programming*. Wiley-Interscience Series in Discrete Mathematics, Wiley, Chichester.
[64] Eisenbrand, F. Split cuts and the stable set polytope of quasi-line graphs. *IMA Hot Topics Workshop: Mixed-Integer Programming*, (Minneapolis, MN, 2005).
[65] Margot, F. (2009) Testing cut generators for mixed-integer linear programming. *Mathematical Programming Computation*, **1**, 69–95.
[66] Cook, W., Dash, S., Fukasawa, R., and Goycoolea, M. (2009) Numerically safe Gomory mixed-integer cuts. *INFORMS Journal on Computing*, **21**, 641–649.
[67] Salkin, H. M. (1975) *Integer Programming*. Addison-Wesley, Reading, MA.
[68] Letchford, A. N. (2003) Binary clutter inequalities for integer programs. *Mathematical Programming, Series B*, **98**, 201–221.
[69] Fischetti, M. and Monaci, M. (2008) How tight is the corner relaxation? *Discrete Optimization*, **5**, 262–269.
[70] Atamtürk, A. (2003) On the facets of the mixed-integer knapsack polyhedron. *Mathematical Programming, Series B*, **98**, 145–175.
[71] http://ieor.berkeley.edu/ atamturk.data.
[72] Mittelmann, H. D., MILPLib. http://plato.asu.edu/sub/testcases.html.

[73] http://www.or.deis.unibo.it/research_pages/ORinstances/MIPs.html.
[74] Fukasawa, R. and Goycoolea, M. (2007) On the exact separation of mixed integer knapsack cuts. Fischetti, M. and Williamson, D. P. (eds.), *Integer Programming and Combinatorial Optimization*, (Ithaca, NY, 2007), vol. 4513 of *Lecture Notes in Computer Science*, pp. 225–239, Springer, Berlin.
[75] Espinoza, D. G. (2008) Computing with multi-row Gomory cuts. Lodi, A., Panconesi, A., and Rinaldi, G. (eds.), *Integer Programming and Combinatorial Optimization*, (Bertinoro, 2008), vol. 5035 of *Lecture Notes in Computer Science*, pp. 214–224, Springer, Berlin.
[76] Balas, E. and Perregaard, M. (2003) A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer Gomory cuts for 0-1 programming. *Mathematical Programming*, **94**, 221–245.
[77] Balas, E. and Bonami, P. (2009) Generating lift-and-project cuts from the LP simplex tableau: open source implementation and testing of new variants. *Mathematical Programming Computation*, **1**, 165–199.
[78] Razborov, A. A. (1985) Lower bounds on the monotone complexity of some Boolean functions. *Doklady Akademii Nauk SSSR*, **281**, 798–801 (Russian), English translation: *Soviet Mathematics. Doklady*, **31**, 354–357.
[79] Alon, N. and Boppana, R. B. (1987) The monotone circuit complexity of Boolean functions. *Combinatorica*, **7**, 1–22.
[80] Haken, A. and Cook, S. A. (1999) An exponential lower bound for the size of monotone real circuits. *Journal of Computer and System Sciences*, **58**, 326–335.
[81] Pudlák, P. (1999) On the complexity of the propositional calculus. Cooper, S. B. and Truss, J. K. (eds.), *Sets and Proofs*, (Leeds, 1997), vol. 258 of *London Mathematical Society Lecture Note Series*, pp. 197–218, Cambridge Univ. Press, Cambridge.
[82] Dash, S. (2005) Exponential lower bounds on the lengths of some classes of branch-and-cut proofs. *Mathematics of Operations Research*, **30**, 678–700.
[83] Krajíček, J. (1994) Lower bounds to the size of constant-depth propositional proofs. *The Journal of Symbolic Logic*, **59**, 73–86.
[84] Krajíček, J. (1997) Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *The Journal of Symbolic Logic*, **62**, 457–486.
[85] Razborov, A. A. (1995) Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. *Rossiĭskaya Akademiya Nauk. Izvestiya. Seriya Matematicheskaya*, **59**, 201–224 (Russian), English translation: *Izvestiya. Mathematics*, **59**, 205–227.
[86] Bonet, M., Pitassi, T., and Raz, R. (1997) Lower bounds for cutting planes proofs with small coefficients. *The Journal of Symbolic Logic*, **62**, 708–728.

Combinatorial Optimization: Methods and Applications
V. Chvátal (Ed.)
IOS Press, 2011

doi:10.3233/978-1-60750-718-5-33

Combinatorial Optimization in VLSI Design

Stephan HELD [a,1], Bernhard KORTE [a], Dieter RAUTENBACH [b] and Jens VYGEN [a]
[a] *Forschungsinstitut für Diskrete Mathematik, Universität Bonn, Germany*
[b] *Institut für Optimierung und Operations Research, Universität Ulm, Germany*

Abstract. VLSI design is probably the most fascinating application area of combinatorial optimization. Virtually all classical combinatorial optimization problems, and many new ones, occur naturally as subtasks. Due to the rapid technological development and major theoretical advances the mathematics of VLSI design has changed significantly over the last ten to twenty years. This survey paper gives an up-to-date account on the key problems in layout and timing closure. It also presents the main mathematical ideas used in a set of algorithms called BonnTools, which are used to design many of the most complex integrated circuits in industry.

Keywords. combinatorial optimization, VLSI, physical design, layout, placement, routing, timing optimization, clock tree synthesis

1. Introduction

The ever increasing abundance, role and importance of computers in every aspect of our lives is clearly a proof of a tremendous scientific and cultural development — if not revolution. When thinking about the conditions which made this development possible most people will probably first think mainly of technological aspects such as the invention and perfection of transistor technology, the possibility to fabricate smaller and smaller physical structures consisting of only a few atoms by now, and the extremely delicate, expensive yet profitable manufacturing processes delivering to the markets new generations of chips in huge quantities every couple of months. From this point of view the increase of complexity might be credited mainly to the skills of the involved engineering sciences and to the verve of the associated economic interests.

It is hardly conceived how important mathematics and especially mathematical optimization is for all parts of VLSI technology. Clearly, everybody will acknowledge that the physics of semiconductor material relies on mathematics and that, considered from a very abstract level, computer chips are nothing but intricate machines for the calculation of complex Boolean functions. Nevertheless, the role of mathematics is far from being fully described with these comments. Especially the steps of the design of a VLSI chip preceding its actual physical realization involve more and more mathematics. Many of the involved tasks which were done by the hands of experienced engineers until one

[1]Corresponding Author: Stephan Held, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, Lennéstr. 2, 53113 Bonn, Germany; E-mail: held@or.uni-bonn.de

or two decades ago have become so complicated and challenging that they can only be solved with highly sophisticated algorithms using specialized mathematics.

While the costs of these design and planning issues are minor compared to the investments necessary to migrate to a new technology or even to build a single new chip factory, they offer large potentials for improvement and optimization. This and the fact that the arising optimization problems, their constraints and objectives, can be captured far more exactly in mathematical terms than many other problems arising in practical applications, make VLSI design one of the most appealing, fruitful and successful application areas of mathematics.

The Research Institute for Discrete Mathematics at the University of Bonn has been working on problems arising in VLSI design for more than twenty years. Since 1987 there exists an intensive and growing cooperation with IBM, in the course of which more than one thousand chips of IBM and its customers (microprocessor series, application specific integrated circuits (ASICs), complex system-on-a-chip designs (SoC)) have been designed with the so-called BonnTools. In 2005 the cooperation was extended to include Magma Design Automation. Some BonnTools are now also part of Magma's products and are used by its customers.

The term BonnTools [1] refers to complete software solutions which have been developed at the institute in Bonn and are being used in many design centers all over the world. The distinguishing feature of BonnTools is their innovative mathematics. With its expertise in combinatorial optimization [2,3,4] the institute was able to develop some of the best algorithms for the main VLSI design tasks: placement, timing optimization, distribution of the clocking signals, and routing. Almost all classical combinatorial optimization problems such as shortest paths, minimum spanning trees, maximum flows, minimum cost flows, facility location and so forth arise at some stage in VLSI design, and the efficient algorithms known in the literature for these problems can be used to solve various subproblems in the design flow. Nevertheless, many problems do not fit into these standard patterns and need new customized algorithms. Many such algorithms have been developed by our group in Bonn and are now part of the IBM design flow.

In this paper we survey the main mathematical components of BonnTools. It is a common feature of these components that they try to restrict the optimization space, i.e., the set of feasible solutions which the algorithms can generate, as little as possible. This corresponds to what is typically called a flat design style in contrast to a hierarchical design style. The latter simplifies a problem by splitting it into several smaller problems and restricting the solution space by additional constraints which make sure that the partial solutions of the smaller problems properly combine to a solution of the entire problem. Clearly, this can seriously deteriorate the quality of the generated solution.

While imposing as few unnecessary restrictions to the problems as possible, the BonnTools algorithms are always considered with respect to their theoretical as well as practical performance. Wherever possible, theoretical performance guarantees and rigorous mathematical proofs are established. The running time and practical behavior of the implemented algorithms is always a main concern, because the code is used for real practical applications.

The beauty and curse of applying mathematics to VLSI design is that problems are never solved once for good. By new technological challenges, new orders of magnitude in instance sizes, and new foci on objectives like the reduction of power consumption for

portable devices or the increase of the productivity of the factories, new mathematical problems arise constantly and classical problems require new solutions. This makes this field most interesting not only for engineers, but also for mathematicians. See [5] for some interesting open problems.

The paper is organized as follows. In the rest of this introduction we explain some basic terminology of VLSI technology and design. Then, in Section 2, we describe our placement tool BonnPlace and its key algorithmic ingredients. The placement problem is solved in two phases: global and detailed placement. Global placement uses continuous quadratic optimization and a new combinatorial partition algorithm (multisection). Detailed placement is based on a sophisticated minimum cost flow formulation.

In Section 3 we proceed to timing optimization, where we concentrate on the three most important topics: repeater trees, logic restructuring, and choosing physical realizations of gates (sizing and V_t-assignment). These are the main components of BonnOpt, and each uses new mathematical theory.

As described in Section 4, BonnCycleOpt further optimizes the timing and robustness by enhanced clock skew scheduling. It computes a time interval for each clock input of a memory element. BonnClock, our tool for clock tree synthesis, constructs clock trees meeting these time constraints and minimizing power consumption.

Finally, Section 5 is devoted to routing. Our router, BonnRoute, contains the first global router that directly considers timing, power consumption, and manufacturing yield, and is provably close to optimal. It is based on a new, faster algorithm for the min-max resource sharing problem. The unique feature of our detailed router is an extremely fast implementation of Dijkstra's shortest path algorithm, allowing us to find millions of shortest paths even for long-distance connections in very reasonable time.

1.1. A Brief Guided Tour through VLSI Technology

VLSI — very large-scale integrated — chips are by far the most complex structures invented and designed by man. They can be classified into two categories: memory chips and logic chips. In a memory chip transistors are packed into a rectangular array. For the design of such chips no advanced mathematics is needed since the individual storage elements (transistors) have to be arranged like a matrix. Logic chips have a very individual design where mathematical methods — as explained below — are essential.

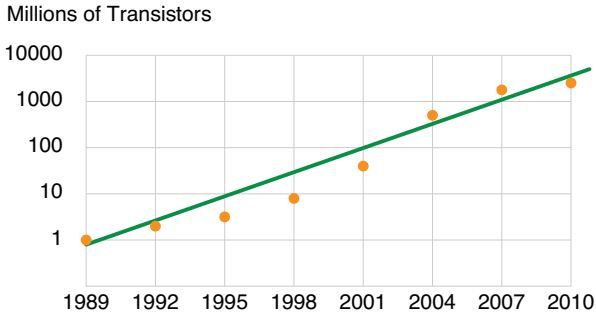

Figure 1. Number of transistors per logic chip

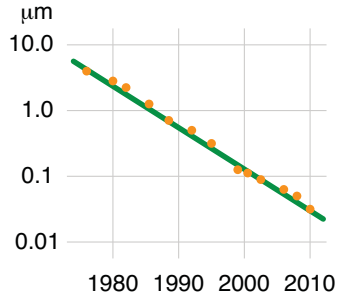

Figure 2. Feature sizes in micron

Integrated circuits are around since 1959. Jack Kilby of Texas Instruments was one of its inventors. Since then the degree of integration grew exponentially. While the first integrated circuits had only a few transistors on a silicon chip, modern chips can have up to one million transistors per mm^2, i.e., a chip of 2 cm^2 total size can carry up to 2 billion transistors. The famous Moore's law, a rule of thumb proposed by Gordon Moore in 1965 [6] and updated in 1975, states that the number of transistors per chip doubles every 24 months (see Figure 1).

This empirical observation is true ever since. As the size of a chip remains almost constant (between 1 and 4 cm^2), the minimum feature size on a chip has to halve about every 4 years. See Figure 2 for the development of feature sizes on leading-edge computer chips.

It is frequently asked how this extremely rapid development of chip technology will continue. Technological as well as physical limitations have to be considered. However, technological limitations could be overruled so far by more sophisticated manufacturing approaches. Thus, the quite often predicted end of silicon technology is not yet in sight. Certainly, there are genuine physical limitations. Today less than 100,000 electrons are used to represent one bit, the absolute minimum is one. The switching energy of a single transistor amounts nowadays to 10,000 attojoule (atto = 10^{-18}). The lower bound derived from quantum physics is 0.000,001 attojoule. Some experts believe that the limit of feature size is around 5 nanometers (today 32 nanometers) and they predict that such dimensions will be possible between 2020 and 2025. In any case, silicon technology will be alive for some further decades. There is some interesting (theoretical) research on quantum computing. However, nobody knows when such ideas can be used in hardware and for mass production.

The extreme dynamics of chip technology can be demonstrated by cost reduction over time. In Figure 3 the trend of cost reduction is shown from 1960 on. Green dots

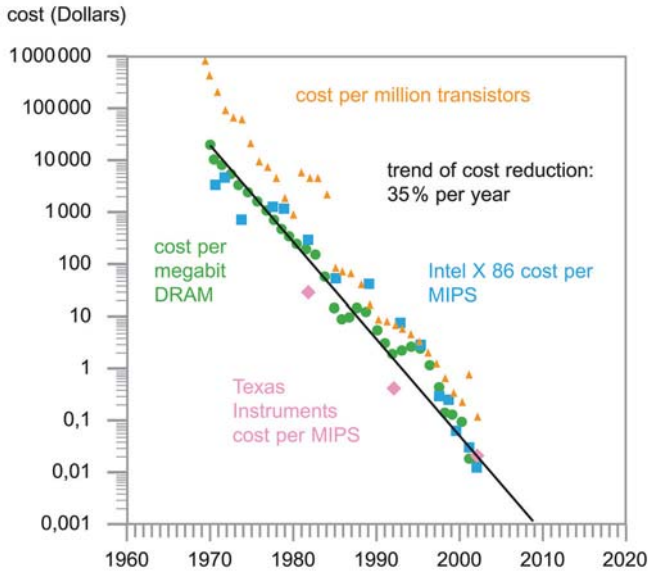

Figure 3. Trend of cost reduction in microelectronics

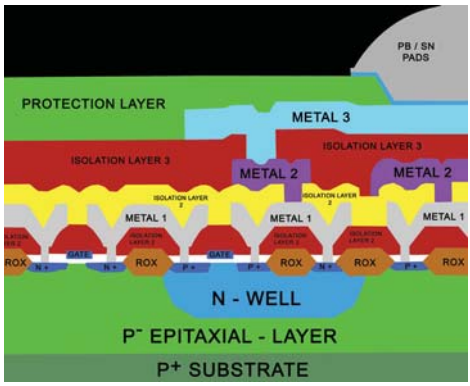

Figure 4. Schematic cross-sectional view of a chip

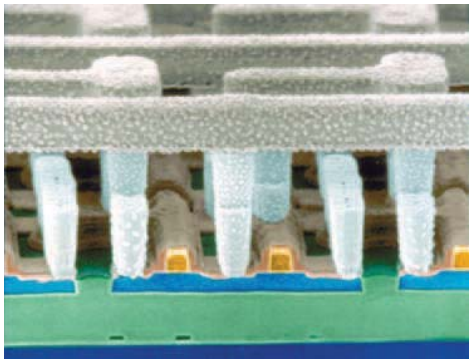

Figure 5. Microscopic view of a chip with aluminum wiring

demonstrate the cost of 1 megabit of memory (DRAM), red triangles the cost of 1 million transistors in silicon technology, blue squares the cost of 1 MIPS (million instructions per second) computing power of Intel X86 processors and red diamonds show the cost of 1 MIPS for Texas Instruments processors. All these developments indicate an average reduction of cost of 35% per year. There is no other industry or technology known with such huge figures over a period of 50 years.

Let us give some insight into the real structure of a chip. Figure 4 shows a schematic of a chip and its different layers. This diagram refers to an older technology with only three layers of metal interconnect. Modern chips have up to 12 layers for wiring signal nets between the circuits. The cross-sectional view reveals the different layers generated on the chip by lithographic processes. By doping with foreign atoms so-called wells (N-wells and P-wells) are generated on the silicon substrate of the wafer. According to the doping the regions have either a surplus (emitter zones) or a demand (collector zones) of electrons. The space between these regions is controlled by a gate. The gates can be charged with different electrical potentials. This will effect that the space underneath the gate is blocked or that electrons can move, which means that the transistor as an electronic switch is either closed or open.

Figure 5 displays the structure of a real chip, visualized by a scanning tunneling microscope. We can identify emitter, collector and gates with their connector pins and some horizontal part of the connecting wires. This picture shows an older technology with aluminum as metal for the interconnect. Figure 6 shows the same technology. Each layer contains horizontal and/or vertical wires, and adjacent layers are separated by an insulation medium. One can also see vias, i.e., the connections between different metal layers. Vias can be considered as little holes in the insulating layer, filled with metal. Since approximately ten years ago aluminum has been replaced by copper for the interconnect wiring (Figure 7). This permits faster signal transmission.

1.2. The VLSI Design Problem: A High-Level View

Although all functions of a chip are composed of transistors and their interconnect, it is not useful to work on this level directly. Instead one uses a library for the design of a chip, where each element of the library represents a pre-designed configuration of sev-

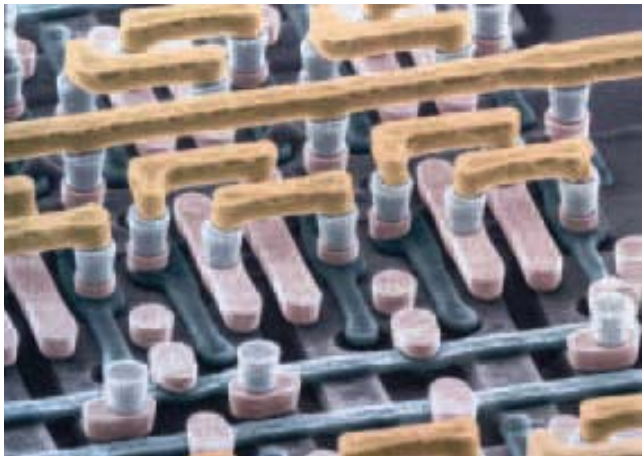

Figure 6. Layers of a chip visualized by electron microscopy

Figure 7. Microscopic view of a chip with copper-wiring (90 nm technology)

eral transistors, implementing a specific function. The elements of the library are called books.

Each book is a blueprint of a (smaller) integrated circuit itself: it contains a list of transistors and their layout, including internal interconnect. Most importantly, each book has at least one input and at least one output. Most books implement an elementary Boolean function, such as `and`, `or`, `invert`, `xor`, etc. For example, the output of an `and` is charged (logical 1) if both inputs are charged, and discharged (logical 0) otherwise. Other books implement registers.

Such simple cells are called *standard cells*. Their blueprints have unit height, because on the chip these cells have to be aligned vertically into so called *cell rows*. Thereby, the power supply pins in each cell row are also aligned and power supply wires can be arranged in straight horizontal lines across the chip.

Finally, there are complicated (and larger) books that represent complex structures like memory arrays, adders, or even complete microprocessors that have been designed earlier and are re-used several times. With each book there is also pre-computed information about its timing behavior. A simple view is that we know how long it takes that a change of an input bit is propagated to each output (if it has any effect).

A chip can contain many instances of the same book. These instances are called circuits or cells. For example, a logic chip can contain millions of inverters, but a typical library contains only a few dozen books that are different implementations of the `invert` function. These books have different layouts and different timing behavior, although they all implement the same function. Each circuit has a set of pins, and each of these pins corresponds to an input or an output of the corresponding book.

The most important part of an instance of the VLSI design problem is a *netlist*, which consists of a set of circuits, their pins, a set of additional pins that are inputs or outputs of the chip itself (I/O ports), and a set of nets, which are pairwise disjoint sets of pins. The layout problem consists of placing these circuits within the chip area, without any overlaps, and connecting the pins of each net by wires, such that wires of different nets are well separated from each other. Placement (Section 2) is a two-dimensional problem (it is currently not possible to put transistors on top of each other), but routing

(Section 5) is a three-dimensional problem as there are several (currently up to 12) layers that can be used for wiring. Of course there are additional rules for placement and routing that must be followed. Some of these are important for the nature of the problem and will be discussed later on, others are merely technical but cause no algorithmic problems; these will not be discussed in this paper.

Layout (placement and routing) is not the only interesting design task. Usually one of the main challenges is to meet timing constraints. In particular, all signals must be propagated in time not only through a single circuit, but also through long paths. In a simple setting, we have an arrival time at each input of the chip, and also a latest feasible arrival time at each output. Moreover, each register is controlled by a periodic clock signal, and the signal to be stored in the register must arrive in time, and can then be used for further computations. To make this possible, one can replace parts of the netlist equivalently (the new parts must compute the same Boolean function as the old ones). While the task to implement a given Boolean function optimally by a netlist (logic synthesis) is extremely hard and more or less completely unsolved, we concentrate on replacing smaller parts of the netlist or restrict to basic operations (Section 3). Another possibility to speed up timing is to schedule the clock signals for all registers (Section 4), thereby trading timing constraints of paths. This is one of the few tasks which we can solve optimally, even for the largest VLSI instances.

2. Placement

A chip is composed of basic elements, called cells, circuits, boxes, or modules. They usually have a rectangular shape, contain several transistors and internal connections, and have at least two pins (in addition to power supply). The pins have to be connected to certain pins of other cells by wires according to the netlist. A net is simply a set of pins that have to be connected, and the netlist is the set of all nets.

The basic placement task is to place the cells legally — without overlaps — in the chip area. A feasible placement determines an instance of the routing problem, which consists of implementing all nets by wires. The quality of a placement depends on the quality of a wiring that can be achieved for this instance.

For several reasons it is usually good if the wire length (the total length of the wires connecting the pins of a net) is as short as possible. The power consumption of a chip grows with the length of the interconnect wires, as higher electrical capacitances have to be charged and discharged. For the same reason signal delays increase with the wire length. Critical nets should be kept particularly short.

2.1. Estimating Net Length

An important question is how to measure (or estimate) wire length without actually routing the chip. First note that nets are wired in different layers with alternating orthogonal preference direction. Therefore the l_1-metric is the right metric for wire length. An exact wire length computation would require to find disjoint sets of wires for all nets (vertex-disjoint Steiner trees), which is an NP-hard problem. This even holds for the simplified problem of estimating the length of each net by a shortest two-dimensional rectilinear Steiner tree connecting the pins, ignoring disjointness and all routing constraints [7].

Table 1. Worst-case ratios of major net models. Entry (r, c) is $\sup c(N)/r(N)$ over all point sets N with $|N| = n$. Here $c(N)$ denotes a net length in the model of column c and r(N) in the model of row r.

	BB	STEINER	MST	CLIQUE	STAR
BB	1	1	1	1	1
STEINER	$\frac{n-1}{\lceil\sqrt{n}\rceil+\lceil n/\lceil\sqrt{n}\rceil\rceil-2}$ $\cdots$ $\frac{\lceil\sqrt{n-2}\rceil}{2}+\frac{3}{4}$	1	1	$\begin{cases}\frac{9}{8} & (n=4)\\ 1 & (n\neq 4)\end{cases}$	1
MST	$\left\lfloor\frac{\sqrt{2n-1}+1}{2}\right\rfloor$ $\cdots$ $\frac{\sqrt{n}}{\sqrt{2}}+\frac{3}{2}$	$\frac{3}{2}$	1	$1+\Theta\left(\frac{1}{n}\right)$ $\cdots$ $\frac{3}{2}$	$\begin{cases}\frac{4}{3} & (n=3)\\ \frac{3}{2} & (n=4)\\ \frac{6}{5} & (n=5)\\ 1 & (n>5)\end{cases}$
CLIQUE	$\frac{\lceil n/2\rceil\lfloor n/2\rfloor}{n-1}$	$\frac{\lceil n/2\rceil\lfloor n/2\rfloor}{n-1}$	$\frac{\lceil n/2\rceil\lfloor n/2\rfloor}{n-1}$	1	1
STAR	$\left\lfloor\frac{n}{2}\right\rfloor$	$\left\lfloor\frac{n}{2}\right\rfloor$	$\left\lfloor\frac{n}{2}\right\rfloor$	$\frac{n-1}{\lceil n/2\rceil}$	1

A simple and widely used estimate for the net length of a finite set $V \subset \mathbb{R}^2$ of pin coordinates (also called terminals) is the *bounding box* model BB, which is defined as half the perimeter of the bounding box:

$$\text{BB}(V) = \max_{(x,y)\in V} x - \min_{(x,y)\in V} x + \max_{(x,y)\in V} y - \min_{(x,y)\in V} y$$

The bounding box net length is a lower bound for the minimum Steiner tree length and computable in linear time. It is widely used for benchmarking and often also as an objective function in placement. Other useful measurements are the clique model CLIQUE which considers all pin to pin connections of a net

$$\text{CLIQUE}(V) = \frac{1}{|V|-1} \sum_{\{(x,y),(x',y')\}\in\binom{V}{2}} (|x-x'| + |y-y'|),$$

and the star model which is the minimum length of a star connecting all sinks to an optimally placed auxiliary point. It can be shown that the clique model is the best topology-independent approximation of the minimum Steiner length [8]. Therefore we use it in our optimization framework, which we will present in Section 2.3.

Table 1 gives an overview on major net models and their mutual worst-case ratios. They were proved in [8,9,10].

2.2. The Placement Problem

We now define the SIMPLIFIED PLACEMENT PROBLEM. It is called "simplified" as side constraints such as routability, timing constraints, decoupling capacitor densities, or nwell filling are neglected.[2] Moreover, wire length is estimated by the bounding box model. Nevertheless this formulation is very relevant in practice. Net weights are incorporated to

[2] Readers who are not acquainted with these terms might just think of additional constraints.

reflect timing criticalities and can be interpreted as Lagrange multipliers corresponding to delay constraints. Other constraints can be dealt with by adjusting densities.

Simplified Placement Problem

Instance:
- a rectangular chip area $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$
- a set of rectangular blockages
- a finite set C of (rectangular) cells
- a finite set P of pins, and a partition $\mathcal{N}$ of P into nets
- a weight $w(N) > 0$ for each net $N \in \mathcal{N}$
- an assignment $\gamma\colon P \to C \cup \{\square\}$ of the pins to cells [pins p with $\gamma(p) = \square$ are fixed; we set $x(\square) := y(\square) := 0$]
- offsets $x(p), y(p) \in \mathbb{R}$ of each pin $p \in P$

Task: Find a position $(x(c), y(c)) \in \mathbb{R}^2$ of each cell $c \in C$ such that
- each cell is contained in the chip area,
- no cell overlaps with another cell or a blockage,

and the weighted net length

$$\sum_{N \in \mathcal{N}} w(N) \operatorname{BB}(\{(x(\gamma(p)) + x(p), y(\gamma(p)) + y(p)) \mid p \in N\})$$

is minimum.

A special case of the Simplified Placement Problem is the Quadratic Assignment Problem (QAP), which is known to be one of the hardest combinatorial optimization problems in theory and practice (for example, it has no constant-factor approximation algorithm unless P = NP [11]).

Placement typically splits into global and detailed placement. Global placement ends with an infeasible placement, but with overlaps that can be removed by local moves: there is no large region that contains too many objects. The main objective of global placement is to minimize the weighted net length. Detailed placement, or legalization, takes the global placement as input and legalizes it by making only local changes. Here the objective is to ensure the previously neglected constraints while minimizing the perturbation of the global placement.

The global placement algorithm developed in [12,13,14,15] has two major components: quadratic placement and multisection.

At each stage the chip area $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ is partitioned by coordinates $x_{\min} = x_0 \le x_1 \le x_2 \le \cdots \le x_{n-1} \le x_n = x_{\max}$ and $y_{\min} = y_0 \le y_1 \le y_2 \le \cdots \le y_{m-1} \le y_m = y_{\max}$ into an array of regions $R_{ij} = [x_{i-1}, x_i] \times [y_{j-1}, y_j]$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Initially, $n = m = 1$. Each movable object is assigned to one region (cf. Figure 8).

In the course of global placement, columns and rows of this array, and thus the regions, are subdivided, and movable objects are assigned to subregions. After global placement, these rows correspond to cell rows with the height of standard cells, and the columns are small enough so that no region contains more than a few dozen movable objects. On a typical chip in 32 nm technology we have, depending on the library and die size, about 10,000 rows and 2,000 columns.

2.3. Quadratic Placement

Quadratic placement means solving

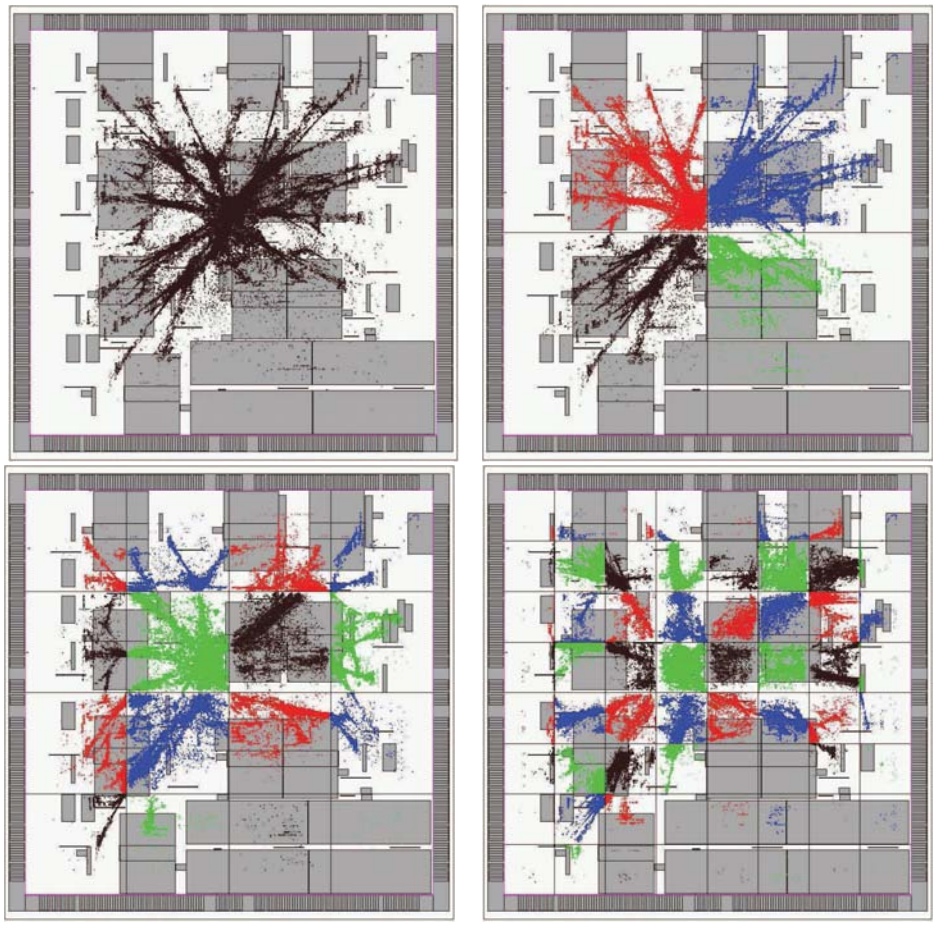

Figure 8. The initial four levels of the global placement with 1, 4, 16, and 64 regions. Colors indicate the assignment of the movable objects to the regions. The large grey objects are fixed and serve as blockages.

$$\min \sum_{N \in \mathcal{N}} \frac{w(N)}{|N| - 1} \sum_{p,q \in N} (X_{p,q} + Y_{p,q}),$$

where $\mathcal{N}$ is the set of nets, each net N is a set of pins, $|N|$ is its cardinality (which we assume to be at least two), and $w(N)$ is the weight of the net, which can be any positive number. For two pins p and q of the same net, $X_{p,q}$ is the function

1. $(x(c) + x(p) - x(d) - x(q))^2$ if p belongs to movable object c with offset $x(p)$, q belongs to movable object d with offset $x(q)$, and c and d are assigned to regions in the same column.
2. $(x(c) + x(p) - v)^2$ if p belongs to movable object c with offset $x(p)$, c is assigned to region $R_{i,j}$, q is fixed at a position with x-coordinate u, and $v = \max\{x_{i-1}, \min\{x_i, u\}\}$.
3. $(x(c) + x(p) - x_i)^2 + (x(d) + x(q) - x_{i'-1})^2$ if p belongs to movable object c with offset $x(p)$, q belongs to movable object d with offset $x(q)$, c is assigned to region $R_{i,j}$, d is assigned to region $R_{i',j'}$, and $i < i'$.

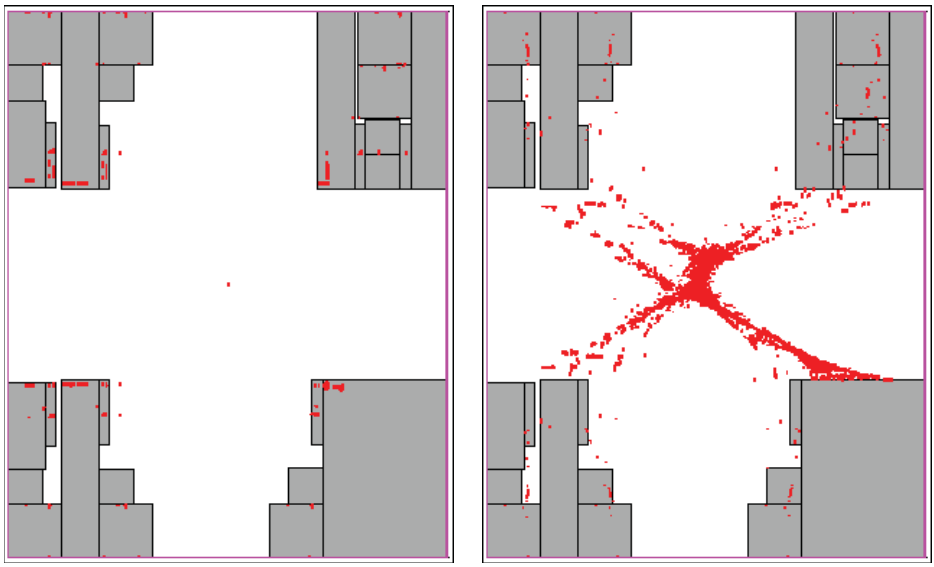

Figure 9. Minimizing the linear bounding box net length (left) gives hardly any information on relative positions compared to minimizing quadratic net length (right). As no disjointness constraints were considered yet, many cells share their position, especially on the left-hand side.

4. 0 if both p and q are fixed.

$Y_{p,q}$ is defined analogously, but with respect to y-coordinates, and with rows playing the role of columns.

In its simplest form, with $n = m = 1$, quadratic placement gives coordinates that optimize the weighted sum of squares of Euclidean distances of pin-to-pin connections (cf. the top left part of Figure 8). Replacing multi-terminal nets by cliques (i.e., considering a connection between p and q for all $p, q \in N$) is the best one can do as CLIQUE is the best topology-independent net model (see Section 2.1). Dividing the weight of a net by $|N|-1$ is necessary to prevent large nets from dominating the objective function. Splitting nets along cut coordinates as in (ii) and (iii), first proposed in [12], partially linearizes the objective function and reflects the fact that long nets will be buffered later.

There are several reasons for optimizing this quadratic objective function. Firstly, delay along unbuffered wires grows quadratically with the length. Secondly, quadratic placement yields unique positions for most movable objects, allowing one to deduce much more information than the solution to a linear objective function would yield (see Figure 9). Thirdly, as shown in [16], quadratic placement is stable, i.e., almost invariant to small netlist changes. Finally, quadratic placement can be solved extremely fast.

To compute a quadratic placement, first observe that the two independent quadratic forms, with respect to x- and y-coordinates, can be solved independently in parallel. Moreover, each row and column can be considered separately and in parallel. Each quadratic program is solved by the conjugate gradient method with incomplete Cholesky pre-conditioning. The running time depends on the number of variables, i.e., the number of movable objects, and the number of nonzero entries in the matrix, i.e., the number of pairs of movable objects that are connected. As large nets result in a quadratic number of connections, we replace large cliques, i.e., connections among large sets of pins in the

same net that belong to movable objects assigned to regions in the same column (or row when considering y-coordinates), equivalently by stars, introducing a new variable for the centre of a star. This was proposed in [12,14].

The running time to obtain sufficient accuracy grows slightly faster than linearly. There are linear-time multigrid solvers, but they do not seem to be faster in practice. We can compute a quadratic placement within at most a few minutes for 5 million movable objects. This is for the unpartitioned case $n = m = 1$; the problem becomes easier by partitioning, even when sequential running time is considered.

It is probably not possible to add linear inequality constraints to the quadratic program without a significant impact on the running time. However, linear equality constraints can be added easily, as was shown by [17]. Before partitioning, we analyze the quadratic program and add center-of-gravity constraints to those regions whose movable objects are not sufficiently spread. As the positions are the only information considered by partitioning, this is necessary to avoid random decisions. See also [18] for a survey on analytical placement.

2.4. Multisection

Quadratic placement usually has many overlaps which cannot be removed locally. Before legalization we have to ensure that no large region is overloaded. For this global placement has a second main ingredient, which we call multisection.

The basic idea is to partition a region and assign each movable object to a subregion. While capacity constraints have to be observed, the total movement should be minimized, i.e., the positions of the quadratic placement should be changed as little as possible. More precisely we have the following problem.

Multisection Problem

Instance:
- Finite sets C (cells) and R (regions),
- sizes $\text{size}\colon C \to \mathbb{R}_{\geq 0}$,
- capacities $\text{cap}\colon R \to \mathbb{R}_{\geq 0}$ and
- costs $d\colon C \times R \to \mathbb{R}$.

Task: Find an assignment $g\colon C \to R$ with
$\sum_{c \in C: g(c)=r} \text{size}(c) \leq \text{cap}(r)$ (for all $r \in R$)
minimizing the total cost $\sum_{c \in C} d(c, r)$.

This partitioning strategy has been proposed in [12] for $k = 4$ and l_1-distances as costs as the Quadrisection Problem, and was then generalized to arbitrary k and costs in [14]. It is a generalization of the Assignment Problem where the sizes and capacities are all 1. To decide whether a solution of the Multisection Problem exists is NP-complete (even for $|R| = 2$) since it contains the decision problem Partition. For our purpose it suffices to solve the fractional relaxation which is known as the Hitchcock Transportation Problem. Here each $c \in C$ can be assigned fractionally to several regions:

Hitchcock Transportation Problem

Instance:
- Finite sets C (cells) and R (regions),
- sizes size: $C \to \mathbb{R}_{\geq 0}$,
- capacities cap: $R \to \mathbb{R}_{\geq 0}$ and
- costs d: $C \times R \to \mathbb{R}$.

Task: Find a fractional assignment g: $C \times R \to \mathbb{R}_+$ with
$$\sum_{r \in R} g(c, r) = \text{size}(c) \text{ for all } c \in C$$
and
$$\sum_{c \in C} g(c, r) \leq \text{cap}(r) \text{ for all } r \in R$$
minimizing
$$\sum_{c \in C} \sum_{r \in R} g(c, r) d(c, r).$$

A nice characteristic of the fractional problem is, that one can easily find an optimum solution with only a few fractionally assigned cells. Most cells can be assigned to a unique region as shown by Vygen [13]:

Proposition 1. *From any optimum solution g to the* Hitchcock Problem *we can obtain another optimum solution $g^\star$ in $O(|C||R|^2)$ time that is integral up to $|R| - 1$ cells.*

Proof. W.l.o.g. let $R = \{1, \ldots, k\}$. Let g be an optimum solution. Define $\Phi(g) := |\{(c, r) \mid c \in C,\ r \in R,\ g(c, r) > 0\}|$.

Let G be the undirected graph with vertex set R. For every cell c that is not assigned integrally to one region, add an edge between the region i with least i and the region j with largest j that contain parts of c. (G may have parallel edges.) If $|E(G)| \leq k - 1$, we are done.

Otherwise G contains a circuit $(\{v_1, \ldots, v_j, v_{j+1} = v_1\}, \{\{v_i, v_{i+1}\} \mid i = 1, \ldots, j\})$. For each $i \in \{1, \ldots, j\}$ there is a $c_i \in C$ with $0 < g(c_i, v_i) < \text{size}(c_i)$ and $0 < g(c_i, v_{i+1}) < \textit{size}(c_i)$ (here $v_{j+1} := v_1$). $c_1, \ldots, c_j$ are pairwise distinct. Hence for a sufficiently small $\epsilon > 0$ we have that g' and g'' are feasible fractional partitions, where $g'(c_i, v_i) := g(c_i, v_i) - \epsilon$, $g'(c_i, v_{i+1}) := g(c_i, v_{i+1}) + \epsilon$, $g''(c_i, v_i) := g(c_i, v_i) + \epsilon$, $g''(c_i, v_{i+1}) := g(c_i, v_{i+1}) - \epsilon$ $(i = 1, \ldots, j)$ and $g'(c, r) := g''(c, r) := g(c, r)$ for $c \in C \setminus \{c_1, \ldots, c_j\}$ and $r \in R$.

The arithmetic mean of the objective function values of g' and g'' is precisely that of g, implying that g' and g'' are also optimum. If we choose ϵ as large as possible, $\Phi(g')$ or $\Phi(g'')$ is strictly smaller than $\Phi(g)$.

After $|C||R|$ iterations Φ must be zero. Note that each iterations can be performed in $O(|R|)$ time, including the update of G. □

The Hitchcock transportation problem can be modeled as a minimum cost flow problem as Figure 10 indicates. The fastest standard minimum cost flow algorithm runs in $O(n \log n(n \log n + kn))$ [19]. However, super-quadratic running times are too slow for VLSI instances. For the quadrisection case, where $k = 4$ and d is the l_1-distance, there is a linear-time algorithm by Vygen [13]. The algorithm is quite complicated but very efficient in practice. Recently, Brenner [20] proposed an $O(nk^2(\log n + k \log k))$-algorithm for the general case. This is extremely fast also in practice and has replaced the quadrisection algorithm of [13] in BonnPlace.

The idea is based on the well-known successive shortest paths algorithm (cf. [3]). Let the cells $C = \{c_1, c_2, \ldots, c_n\}$ be sorted by size $\text{size}(c_1) \geq \text{size}(c_2) \geq \cdots \geq \text{size}(c_n)$. We

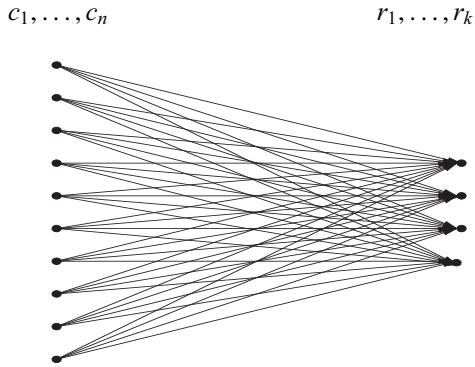

Figure 10. The Hitchcock transportation problem is a relaxation of the multisection problem. All arcs are oriented from left to right and are uncapacitated. The supply vertices on the left correspond to movable objects and have supply $\text{size}(c_1), \ldots, \text{size}(c_n)$. The demand vertices on the right correspond to subregions and have demand $\text{cap}(r_1), \ldots, \text{cap}(r_k)$. Arc costs are $d(c_i, r_j)$ for all $1 \le i \le n$, $1 \le j \le k$. Note that $k \ll n$ in this application.

assign the objects in this order. A key observation is that for doing this optimally we need to re-assign only $O(k^2)$ previously assigned objects and thus can apply a minimum cost flow algorithm in a digraph whose size depends on k only. Note that k is less than 10 in all our applications, while n can be in the millions. The relevant results for our purposes are summarized in following theorem.

Theorem 2 ([13,20]). *The Hitchcock transportation problem with $|R| = 4$ and l_1-distance can be solved in $O(n)$ time. The general case can be solved in $O(nk^2(\log n + k \log k))$ time, where $n = |C|$ and $k = |R|$.*

Figure 11 shows a multisection example where the movable objects are assigned optimally to nine regions.

2.5. Overall Global Placement and Macro Placement

With these two components, quadratic placement and multisection, the global placement can be described. Each level begins with a quadratic placement. Before subdividing the array of regions further, we fix macro cells that are too large to be assigned completely to a subregion. Macro placement uses minimum cost flow, branch-and-bound, and greedy techniques. We briefly describe its main component.

Assume that we want to place rectangular macros numbered $c_1, \ldots, c_n$, with widths $w_1, \ldots, w_n$ and heights $h_1, \ldots, h_n$ within an area $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$. The objective function is weighted bounding box netlength. If we fix a relation $r_{ij} \in \{W, S, E, N\}$ for each $1 \le i < j \le n$, then this can be written as linear program:

$$\min \sum_{N \in \mathcal{N}} w(N)(\overline{x}_N - \underline{x}_N + \overline{y}_N - \underline{y}_N) \tag{1}$$

subject to

$$x_i \ge x_{\min} \quad \text{for } i = 1, \ldots, n$$

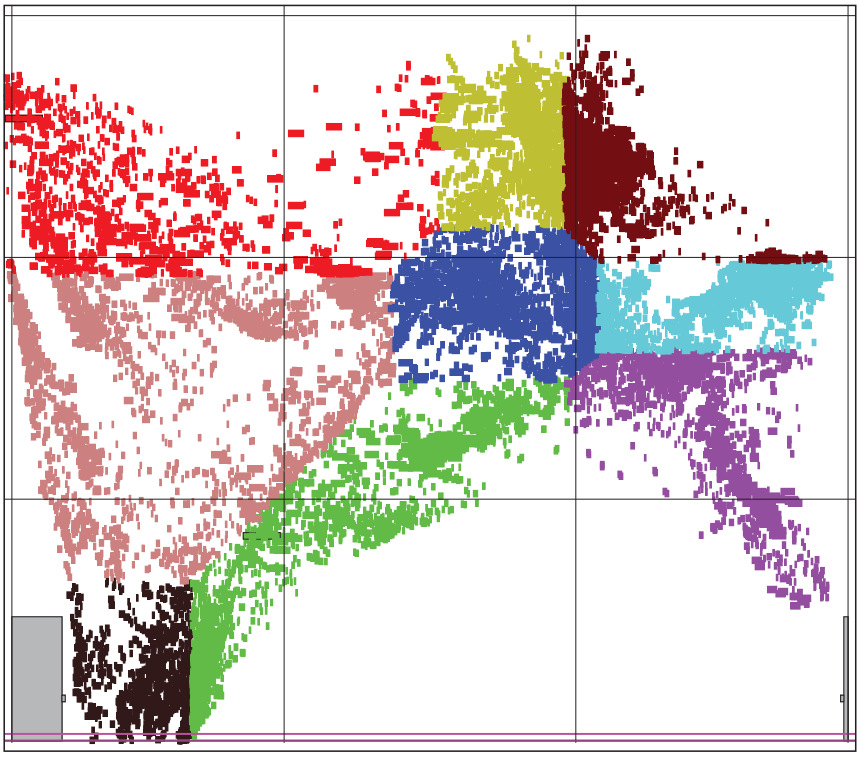

Figure 11. An example for multisection: objects are assigned to 3×3 subregions. The colors reflect the assignment: the red objects are assigned to the top left region, the yellow ones to the top middle region, and so on. This assignment is optimal with respect to total l_1-distance.

$$
\begin{aligned}
x_i + w_i &\le x_{\max} && \text{for } i = 1, \ldots, n \\
y_i &\ge y_{\min} && \text{for } i = 1, \ldots, n \\
y_i + h_i &\le y_{\max} && \text{for } i = 1, \ldots, n \\
x_i + w_i &\le x_j && \text{for } 1 \le i < j \le n \text{ with } r_{ij} = W \\
x_j + w_j &\le x_i && \text{for } 1 \le i < j \le n \text{ with } r_{ij} = E \\
y_i + h_i &\le y_j && \text{for } 1 \le i < j \le n \text{ with } r_{ij} = S \\
y_j + h_j &\le y_i && \text{for } 1 \le i < j \le n \text{ with } r_{ij} = N \\
x_i + x(p) &\ge \underline{x}_N && \text{for } i = 1, \ldots, n \text{ and } p \in P \text{ with } \gamma(p) = c_i \\
x(p) &\ge \underline{x}_N && \text{for } p \in P \text{ with } \gamma(p) = \square \\
x_i + x(p) &\le \overline{x}_N && \text{for } i = 1, \ldots, n \text{ and } p \in P \text{ with } \gamma(p) = c_i \\
x(p) &\le \overline{x}_N && \text{for } p \in P \text{ with } \gamma(p) = \square \\
y_i + y(p) &\ge \underline{y}_N && \text{for } i = 1, \ldots, n \text{ and } p \in P \text{ with } \gamma(p) = c_i \\
y(p) &\ge \underline{y}_N && \text{for } p \in P \text{ with } \gamma(p) = \square
\end{aligned}
$$

$$\begin{aligned} y_i + y(p) &\le \overline{y}_N \qquad \text{for } i = 1, \ldots, n \text{ and } p \in P \text{ with } \gamma(p) = c_i \\ y(p) &\le \overline{y}_N \qquad \text{for } p \in P \text{ with } \gamma(p) = \square \end{aligned} \tag{2}$$

This is the dual of an uncapacitated minimum cost flow problem. Hence we can find an optimum solution to Eq. (2) in $O((n + m)(p + n^2 + m \log m) \log(n + m))$ time, where $n = |C|$, $m = |\mathcal{N}|$, and $p = |P|$. Instead of enumerating all $2^{n(n-1)}$ possibilities for r, it suffices to enumerate all pairs of permutations π, ρ on $\{1, \ldots, n\}$. For $1 \le i < j \le n$ we then define $r_{ij} := W$ if i precedes j in π and ρ, $r_{ij} := E$ if j precedes i in π and ρ, $r_{ij} := S$ if i precedes j in π and j precedes i in ρ, and $r_{ij} := N$ if j precedes i in π and i precedes j in ρ. One of the $(n!)^2$ choices will lead to an optimum placement. This sequence-pair representation is due to [21] and [22]. In practice, however, a branch-and-bound approach is faster; Hougardy [23] solves instances up to approximately 20 circuits optimally. This is also used as part of a post-optimization heuristic.

However, interaction of small and large blocks in placement is still not fully understood [24,25], and placing large macros in practice often requires a significant amount of manual interaction.

After partitioning the array of regions, the movable objects are assigned to the resulting subregions. Several strategies are applied (see [15,26] for details), but the core subroutine in each case is the multisection described above. An important further step is repartitioning, where 2×2 or even 3×3 sub-arrays of regions are considered and all their movable objects are reassigned to these regions, essentially by computing a local quadratic placement followed by multisection.

There are further components which reduce routing congestion [27], deal with timing and resistance constraints, and handle other constraints like user-defined bounds on coordinates or distances of some objects. Global placement ends when the rows correspond to standard cell heights. Typically there are fewer columns than rows as most movable objects are wider than high. Therefore 2×3 partitioning is often used in the late stages of global placement.

2.6. Detailed Placement

Detailed placement, or legalization, considers standard cells only; all others are fixed beforehand. The task is to place the standard cells legally without changing the (illegal) input placement too much. Detailed placement does not only arise as a finishing step during the overall placement flow, but also in interaction with timing optimization and clock tree insertion. These steps add, remove or resize cells and thus require another legalization of the placement that has become illegal.

Due to technology constraints cells cannot be placed with arbitrary y-coordinates. Instead, they have to be arranged in cell rows. Cells that are higher than a cell row must be fixed before detailed placement. Thus we can assume unit height and have the following problem formulation:

Placement Legalization Problem

Instance:
- A rectangular chip area $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$,
- a set of rectangular blockages,
- a set C of rectangular cells with unit height,
- An equidistant subdivision $y_0^R = y_{\min} \le y_1^R \le \cdots \le y_{n_R}^R = y_{\max}$ of $[y_{\min}, y_{\max}]$ into *cell rows*
- a width $w(c)$ and a position $(x(c), y(c)) \in \mathbb{R}^2$ of each cell $c \in C$.

Task: Find new positions $(x'(c), y'(c)) \in \mathbb{Z}^2$ of the cells such that
- each cell is contained in the chip area,
- each cell snaps into a cell row (its y-coordinate is a cell row coordinate),
- no two cells overlap,
- no cell overlaps with any blockage,

and

$$\sum_{c \in C} \left((x(c) - x'(c))^2 + (y(c) - y'(c))^2 \right)$$

is minimum.

It is quite natural to model the legalization problem as a minimum cost flow problem, where flow goes from supply regions with too many objects to demand regions with extra space [28]. Brenner and Vygen [29] refined this approach. We describe this enhanced legalization algorithm in the following.

It consists of three phases. A *zone* is defined as a maximal part of a cell row that is not blocked by any fixed objects, i.e., can be used for legalization.

The first phase guarantees that no zone contains more cells than fit into it. The second phase places the cells legally within each zone in the given order. When minimizing quadratic movement, this can be done optimally in linear time by Algorithm 1, as shown in [29] (see also [30,31,32]). The algorithm gets as inputs a zone $[x_{\min}, x_{\max}]$, coordinates $x_1, \ldots, x_n \in \mathbb{R}$ and widths $w_1, \ldots, w_n > 0$ with $\sum_{i=1}^n w_i \le x_{\max} - x_{\min}$, and legalizes the circuits within $[x_{\min}, x_{\max}]$ in the given order. As all circuits stay within one zone, we do not consider y-coordinates. It places the circuits from left to right, each optimally and as far to the left as possible. If a circuit cannot be placed optimally, it is merged with its predecessor.

Theorem 3. *Algorithm 1 runs in linear time and computes coordinates $x'_1, \ldots, x'_n$ with $x_{\min} \le x'_1$, $x'_i + w_i \le x'_{i+1}$ $(i = 1, \ldots, n-1)$, and $x_n + w_n \le x_{\max}$, such that $\sum_{i=1}^n (x_i - x'_i)^2$ is minimum.*

Proof. Each iteration increases i by one or decreases $|\mathcal{L}|$ and i by one. As $1 \le i \le |\mathcal{L}| \le n + 1$, the total number of iterations is at most $2n$. Each takes constant time.

To prove correctness, the main observation is that we can merge circuits without losing optimality. So if we merge circuits h and i, we write $\mathcal{L}' := \{j \in \mathcal{L} \mid j < i\}$ and claim that there exists an optimum solution $(x_j^*)_{j \in \mathcal{L}' \cup \{i\}}$ of the subproblem defined by $(f_j, W_j)_{j \in \mathcal{L}' \cup \{i\}}$ where $x_h^* + W_h = x_i^*$.

Let $(x_j^*)_{j \in \mathcal{L}' \cup \{i\}}$ be an optimum solution of this subproblem. If $x_i^* - W_h \le \arg\min f_h$, then x_h^* can be set to $x_i^* - W_h$ without increasing $f_h(x_h^*)$.

Let $f_i\colon x \mapsto (x - x_i)^2$.
$x'_0 \leftarrow x_{\min}$, $W_0 \leftarrow 0$, $W_i \leftarrow w_i$ for $i = 1, \ldots, n$.
Let $\mathcal{L}$ be the list consisting of $0, 1, \ldots, n, n+1$.
$i \leftarrow 1$.
while $i < n + 1$ **do**
    Let h be the predecessor and j the successor of i in $\mathcal{L}$.
    if $h = 0$ or $x'_h + W_h \le \min\{x_{\max} - W_i, \arg\min f_i\}$ **then**
        $x'_i \leftarrow \max\{x_{\min}, \min\{x_{\max} - W_i, \arg\min f_i\}\}$.
        $i \leftarrow j$.
    else
        Redefine f_h by $f_h\colon x \mapsto f_h(x) + f_i(x + W_h)$.
        $W_h \leftarrow W_h + W_i$.
        Remove i from $\mathcal{L}$.
        $i \leftarrow h$.
    end if
end while
for $i \in \{1, \ldots, n\} \setminus \mathcal{L}$ **do**
    $x'_i \leftarrow x'_h + \sum_{j=h}^{i-1} w_j$, where h is the maximum index in $\mathcal{L}$ that is smaller than i.
end for

Algorithm 1. Single Row Placement Algorithm

So suppose that $x_i^* > x_h^* + W_h$ and $x_i^* > \arg\min f_h + W_h$. Then $x_i^* > \max\{x_{\min}, \arg\min f_h\} + W_h \ge x'_h + W_h > \min\{x_{\max} - W_i, \arg\min f_i\}$, a contradiction as decreasing x_i^* would reduce $f_i(x_i^*)$. □

Finally, some post-optimization heuristics (like exchanging two cells, but also much more complicated operations) are applied.

The most difficult and important phase is the first one, which we describe here in detail. If the global placement is very dense in some areas, a significant number of cells has to be moved. As phase two works in each zone separately, phase one has to guarantee that no zone contains more objects than fit into it.

In order to prevent long-distance movements within the zones later in phase two, wide zones are partitioned into regions. Each movable object is assigned to a region. This means that the centre of the movable object must be located in the assigned region. Parts of an object can overlap with neighboring regions.

Unless all movable objects that are assigned to a region R can be placed legally with their centre in R, some of them have to be moved out of R. But this is not sufficient: in addition, it may be necessary to move some objects out of certain sequences of consecutive regions. More precisely, for a sequence of consecutive regions $R_1, \ldots, R_k$ within a zone, we define its supply by

$$\begin{aligned}&\operatorname{supp}(R_1, \ldots, R_k)\\ &:= \max\left\{0, \sum_{i=1}^{k} (w(R_i) - a(R_i)) - \frac{1}{2}(w_l(R_1) + w_r(R_k)) - \sum_{\substack{1 \le i < j \le k\\ (i,j) \ne (1,k)}} \operatorname{supp}(R_i, \ldots, R_j)\right\},\end{aligned}$$

where $a(R_i)$ is the width of region R_i, $w(R_i)$ is the total width of cells that are currently assigned to region R_i, and $w_l(R_i)$ and $w_r(R_i)$ are the widths of the leftmost and rightmost cell in R_i, respectively, or zero if R_i is the leftmost (rightmost) region within the zone.

If $\text{supp}(R_1, \ldots, R_k)$ is positive, $(R_1, \ldots, R_k)$ is called a supply interval. Similarly, we define the demand of each sequence of consecutive regions, and the demand intervals. We now define a directed network $G = (V, E, c)$ on regions, supply intervals, and demand intervals, in which we compute a minimum cost flow that cancels demands and partly cancels supplies. Let vertices and edges be defined by

$$V(G) := \{\text{regions, supply intervals, demand intervals}\}$$

$$E(G) := \{(A, A') \mid A, A' \text{ adjacent regions}\}$$

$$\cup \{(A, A') \mid A \text{ supply interval}, A' \text{ maximal proper subset of } A\}$$

$$\cup \{(A, A') \mid A' \text{ demand interval}, A \text{ maximal proper subset of } A'\}.$$

Let the cost $c(A, A')$ between two adjacent regions A, A' be the expected cost of moving a cell of width 1 from A to A' and all other arcs costs be zero. The construction of this uncapacitated minimum cost flow instance is illustrated in Figure 12. We look for a minimum cost flow f which cancels all supplies:

$$f(\delta^+(v)) - f(\delta^-(v)) \geq \text{supp}(v) + \text{dem}(v) \quad \text{for all } v \in V(G).$$

This can be computed in $O(n^2 \log^2 n)$ time by Orlin's minimum cost flow algorithm [19]. Figure 13 shows part of a typical result on a real chip.

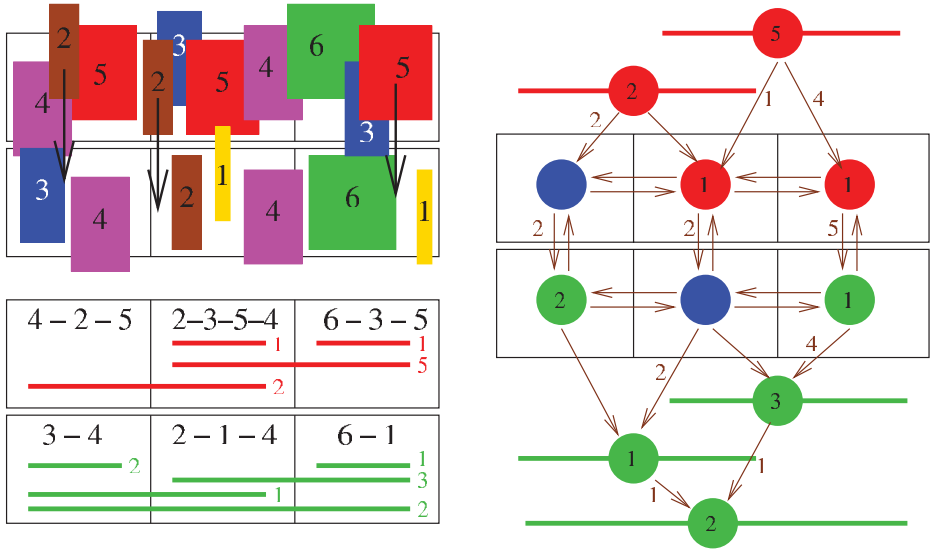

Figure 12. An example with two zones and six regions, each of width 10 (top left), the supply (red) and demand (green) regions and intervals with their supply and demand (bottom left), and the minimum cost flow instance (right) with a solution shown in brown numbers. To realize this flow, objects of size 2, 2, and 5, respectively, have to be moved from the top regions downwards.

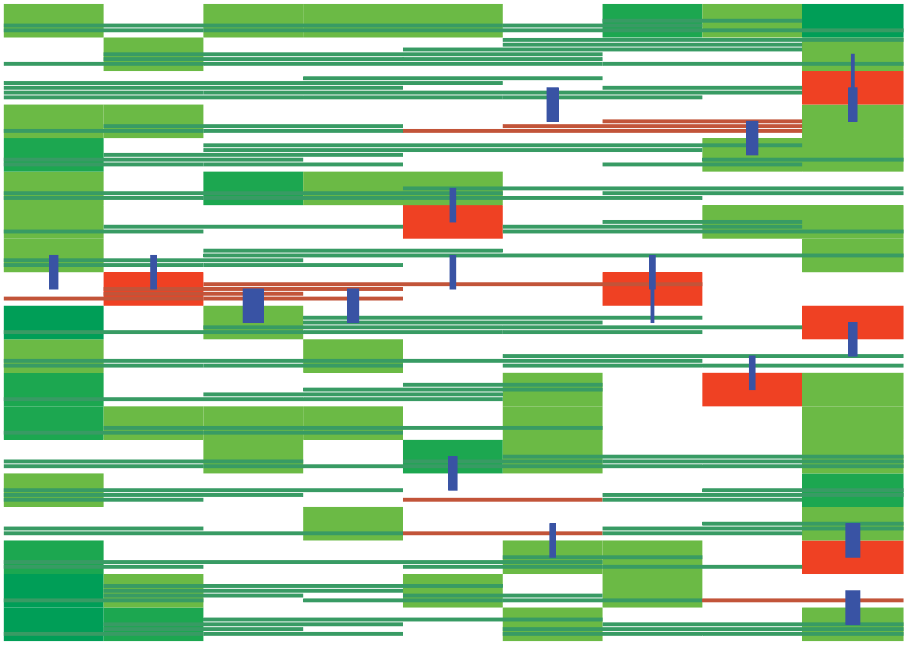

Figure 13. Small part of a real chip in legalization. Supply regions and intervals are shown in red, demand regions and intervals in green. The blue edges represent the minimum cost flow, and their width is proportional to the amount of flow.

Finally the flow is realized by moving objects along flow arcs. This means to move cells of total size $f(A, A')$ from region A to region A' for each pair of neighbors (A, A'). An exact realization may not exists as small amounts of flow may not be realizable by wide objects. Therefore the realization is approximated. We scan the arcs carrying flow in topological order and solve a multi-knapsack problem by dynamic programming for selecting the best set of cells to be moved for realizing the flow on each arc [28,29].

Of course zones can remain overloaded after realization. In this case phase one is repeated with increased region widths and decreased demand values. Typically, after a few iterations of phase 1 no overloaded zones remain.

The minimum cost flow formulation yields an optimum solution under some assumptions [29], and an excellent one in practice. Experimental results show that the gap between the computed solution and a theoretical lower bound is only approximately 10%, and neither routability nor timing is significantly affected [33].

3. Timing Optimization

In this section we describe the main ingredients of timing optimization. These include algorithms for the construction of timing- and routing-aware fan-out trees (repeater trees), for the timing-oriented logic restructuring and optimization, and for the timing- and power-aware choice of different physical realizations of individual gates. Each is based on new mathematical theory.

Altogether, these routines combined with appropriate net weight generation and iterative placement runs form the so-called timing-driven placement. Using the new algorithms introduced in this section the overall turn-around time for timing closure, including full placement and timing optimization, could be reduced from more than a week to 26 hours on the largest designs.

3.1. Timing Constraints

During optimization the signal propagation through a VLSI chip is estimated by a static timing analysis. We give a simplified description and refer the interested reader to [34] for further reading.

At every pin $v \in P$ the latest arrival time a_v of a possible signal occurrence is computed. Signals are propagated along the edges of the timing graph $G_{\mathcal{T}}$, which is a directed graph on the vertex set $V(G_{\mathcal{T}}) = P$ of pins in the design. $G_{\mathcal{T}}$ contains two type of edges. First "net" edges are inserted for each source-sink pin pair $(v, w) \in N$ of a net $N \in \mathcal{N}$, directed from the unique source of N to each sink. Second "circuit" edges $(v, w) \in E(G_{\mathcal{T}})$ are inserted for each input-output pin pair (v, w) of a cell, where a signal in v triggers a signal in w. For every edge $e \in E(G_{\mathcal{T}})$ a delay d_e is given. The delay depends on various parameters, e.g., circuit size, load capacitance (wire plus sink pin capacitances), net topologies, signal shapes.

The timing graph is acyclic as every cycle in the netlist is cut by a register, which does not have a direct input to output connection.[3] Arrival times are propagated in topological order. At each vertex $v \in V(G_{\mathcal{T}})$ with $\delta^-(v) = \varnothing$ a start time $\mathrm{AT}(v)$ is given as design constraint. It initializes the arrival time in v by $a_v = \mathrm{AT}(v)$. Then for each $v \in V(G_{\mathcal{T}})$ with $\delta^-(v) \neq \varnothing$, the arrival time a_v is the maximum over all incoming arrival times:

$$a_v := \max_{(u,v)\in\delta^-(v)} a_u + d_{(u,v)}. \tag{3}$$

At each endpoint pin $v \in V(G_{\mathcal{T}})$ with $\delta^+(v) = \varnothing$ of the combinational paths required arrival times $\mathrm{RAT}(v)$ are given as design constraints. The signals arrive in time at v if $a_v \leq \mathrm{RAT}(v)$. To measure timing feasibility on arbitrary pins, the maximum feasible required arrival time variable r_v is computed for each $v \in V(G_{\mathcal{T}})$. It is initialized by $r_v = \mathrm{RAT}(v)$ for all $v \in V(G_{\mathcal{T}}), \delta^+(v) = \varnothing$. For the remaining vertices $v \in V(G_{\mathcal{T}})$ with $\delta^+(v) \neq \varnothing$ they are computed in reverse topological order by

$$r_v := \min_{(v,w)\in\delta^+(v)} a_w - d_{(v,w)}. \tag{4}$$

The difference $\sigma_v := r_v - a_v$ is called *slack*. If it is nonnegative the timing constraints of all paths through v are satisfied. If $\sigma_v \geq 0$ for all endpoint pins $v \in V(G_{\mathcal{T}})$ with $\delta^+(v) = \varnothing$, all timing constraints are met, which implies $\sigma_v \geq 0$ for all nodes $v \in V(G_{\mathcal{T}})$. The slack can also be defined for an edge $(v, w) \in E(G_{\mathcal{T}})$ by $\sigma_{(v,w)} := r_w - d_{(v,w)} - a_v$ with following interpretation. When adding at most $\sigma_{(v,w)}$ to the delay $d_{(v,w)}$, all paths through (v, w) are fast enough. Note that $\sigma_{(v,w)}$ can also be negative; then delays must be reduced.

In some applications (e.g., Section 3.3) we are interested in a most critical path. Observe that there must be at least one path in $G_{\mathcal{T}}$ from a start pin $v \in V(G_{\mathcal{T}}), \delta^-(v) = \varnothing$

[3] We omit transparent latches here.

to an endpoint pin $v' \in V(G_{\mathcal{T}}), \delta^+(v') = \varnothing$, in which each vertex and edge has the overall worst slack $\min\{\sigma_v \mid v \in V(G_{\mathcal{T}})\}$. Such a path can be determined efficiently by backward search along a most critical incoming edge, starting from an endpoint pin $v' \in V(G_{\mathcal{T}}), \delta^+(v') = \varnothing$ with a smallest slack value $\sigma_{v'} = \min\{\sigma_v \mid v \in V(G_{\mathcal{T}})\}$.

Beside the introduced late mode constraints earliest arrival time or early mode constraints are given, too. Here signals must not arrive too early at the endpoints. Propagation is analog to Eqs. (3) and (4) with min and max being swapped. Together with the arrival times also signal shapes — slews[4] in VLSI terminology — are propagated. These are needed for proper delay calculation. When incorporating slews the above propagation rules become incorrect, as an early signal with a very large slew can result in later arrival times in subsequent stages than a late signal with a tight slew. In [35] we describe how to overcome these slew related problems by a slight modification of the propagation rules.

3.2. Fan-out Trees

On an abstract level the task of a fan-out tree is to carry a signal from one gate, the root r of the fan-out tree, to other gates, the sinks $s_1, \ldots, s_n$ of the fan-out tree, as specified by the netlist. If the involved gates are not too numerous and not too far apart, then this task can be fulfilled just by a metal connection of the involved pins, i.e., by a single net without any repeaters. But in general we need to insert repeaters (buffers or inverters). Inverters logically invert a signal while buffers implement the identity function. A *repeater tree* is a netlist in which all circuits are repeaters, r is the only input, and S is the set of outputs.

In fact, fan-out trees are a very good example for the observation mentioned in the introduction that the development of technology continually creates new complex design challenges that also require new mathematics for their solution. Whereas circuit delay traditionally dominated the interconnect delay and the construction of fan-out trees was of secondary importance for timing, the feature size shrinking is about to change this picture drastically.

Extending the current trends one can predict that in future technologies more than half of all circuits of a design will be needed just for bridging distances, i.e., in fan-out trees. The reason for this is that with decreasing feature sizes the wire resistances increase more than wire capacitances decrease. The delay over a pure metal wire is roughly proportional to the product of the total resistance and capacitance. It increases quadratically with its length, as both resistance and capacitance depend linearly on the length. But by inserting repeaters the growth rate can be kept linear. In current technologies buffers are realized by two subsequent inverters. Therefore, inverters are more flexible and typically faster than buffers.

Repeaters can be chosen from a finite library $\mathcal{L}$ of different sizes. Smaller sizes have a smaller drive strength. The smaller a repeater is, the higher is its delay sensitivity on the load capacitance $\partial\,\text{delay}\,/\partial\,\text{cap}$. On the other hand do smaller repeaters involve smaller input pin capacitances and therefore smaller load capacitances and delays for the predecessor. We now define the Repeater Tree Problem.

[4] The slew of a signal is an estimate for how fast the voltage changes

Repeater Tree Problem

Instance:
- A root r and a set S of sinks.
- a start time $\mathrm{AT}(r)$ at the root r and a required arrival time $\mathrm{RAT}(s)$ at each sink s,
- a parity in $\{+,-\}$ for each sink indicating whether it requires the signal or its inversion,
- placement information for the root and the sinks $\mathrm{Pl}(r), \mathrm{Pl}(s_1), \mathrm{Pl}(s_2), \ldots, \mathrm{Pl}(s_n) \in [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$,
- physical information about the driver strength of r and the input capacitances of the sinks $s \in S$, and
- physical information about the wiring and the library $\mathcal{L}$ of available repeaters (inverters and buffers).

Task: Find a repeater tree T that connects r with all sinks in S such that
- the desired parities are realized (i.e., for each sink s the number of inverters on the r-s-path in T is even iff s has parity +),
- the delay from r to s is at most $\mathrm{RAT}(s) - \mathrm{AT}(r)$, for each $s \in S$,
- and the power consumption is minimum.

In another formulation, $\mathrm{AT}(r)$ is not given but should be maximized. The procedure that we proposed for fan-out tree construction [36,37,38] works in two phases. The first phase generates a preliminary topology for the fan-out tree, which connects very critical sinks in such a way as to maximize the minimum slack, and which minimizes wiring for noncritical sinks. During the second phase the resulting topology is finalized and buffered in a bottom-up fashion using mainly inverters and respecting the parities of the sinks.

A *topology* for root r and set S of sinks is a pair (T, Pl) where T is an arborescence rooted at r in which the root has one child, the sinks have no children, and all other vertices have two children, and $\mathrm{Pl}\colon V(T) \setminus (\{r\} \cup S) \to [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ is an embedding of the internal vertices of T in the chip area. Let us denote by $T_{[r,s]}$ the unique path from r to s in T. A simplified delay model is used to model delay within (T, Pl). The delay on the path from the root r to a sink $s \in S$ is approximated by

$$c_{\mathrm{node}} \cdot (|E(T_{[r,s]})| - 1) + \sum_{(u,v) \in E(T_{[r,s]})} c_{\mathrm{wire}} \cdot \|\mathrm{Pl}(u) - \mathrm{Pl}(v)\|_1 \tag{5}$$

The second term in this formula accounts for the wire or distance delay of the r-s-path in T. This is a linear function in wire length as buffering from phase two is anticipated. The first term adds an additional delay of c_{node} for every bifurcation. This reflects the fact that a bifurcation adds additional capacitance. A constant adder is used as repeaters can be inserted later to shield large downstream capacitances in the branches.

The involved constants are derived in a pre-processing step. The accuracy of this very simple delay model is illustrated in Figure 14, which compares the estimated delay with the measured delay after buffering and sizing at the critical sinks.

Our topology generation algorithm inserts the sinks by a greedy strategy. First the sinks are sorted by their criticality. As criticality we use an upper bound for the slack at the sink, namely the slack that would result at $s \in S$ if we connected s to r by a shortest possible wire without any bifurcation:

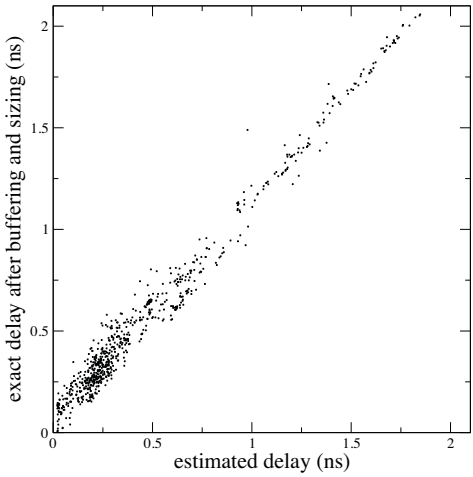

Figure 14. The simple timing model used for topology generation matches actual timing results after buffering well.

$$\sigma_s := \mathrm{RAT}(s) - \mathrm{AT}(r) - c_{\mathrm{wire}} \|\mathrm{Pl}(r) - \mathrm{Pl}(s)\|_1 . \tag{6}$$

The individual sinks are now inserted one by one into the preliminary topology in order of nonincreasing criticality, i.e., nondecreasing value of σ_s. When we insert a new sink s, we consider all arcs $e = (u, v) \in E(T)$ of the preliminary topology constructed so far and estimate the effect of subdividing e by a new internal node w and connecting s to w.

The sink s will be appended to a new vertex subdividing an arc e of T that maximizes $\xi\sigma_e - (100 - \xi)l_e$, where σ_e and l_e estimate the corresponding worst slack at r and the total length, respectively, when choosing e. The parameter $\xi \in [0, 100]$ allows us to favor slack maximization for timing critical instances or wiring minimization for noncritical instances. Figure 15 gives an example for a preliminary topology.

In most cases it is reasonable to choose values for ξ that are neither too small nor too large. Nevertheless, in order to mathematically validate our procedure we have proved optimality statements for the extreme values $\xi = 0$ and $\xi = 100$. If we ignore timing ($\xi = 0$) and choose an appropriate order of the sinks, the final length of the topology is at most $\frac{3}{2}$ times the minimum length of a rectilinear Steiner tree connecting the root and the sinks. If we ignore wiring ($\xi = 100$), the topology realizes the optimum slack with respect to our delay model (see below).

To measure the quality of the resulting topologies in practice, we can compute bounds for performance and power consumption to compare against. A minimum power topology obviously arises from a minimum Steiner tree on $S \cup \{r\}$. Some Steiner points may need to be replaced by two topology nodes with same coordinates. The following bound can be specified for the maximum achievable slack:

Theorem 4. *The maximum possible slack* $\sigma_{\max}$ *of a topology* (T, Pl) *with respect to our delay model is at most*

$$-c_{\mathrm{node}} \cdot \log_2\Bigg(\sum_{s \in S} 2^{-(\mathrm{RAT}(s) - \mathrm{AT}(r) - c_{\mathrm{wire}} \|\mathrm{Pl}(r) - \mathrm{Pl}(s)\|_1)/c_{\mathrm{node}}}\Bigg).$$

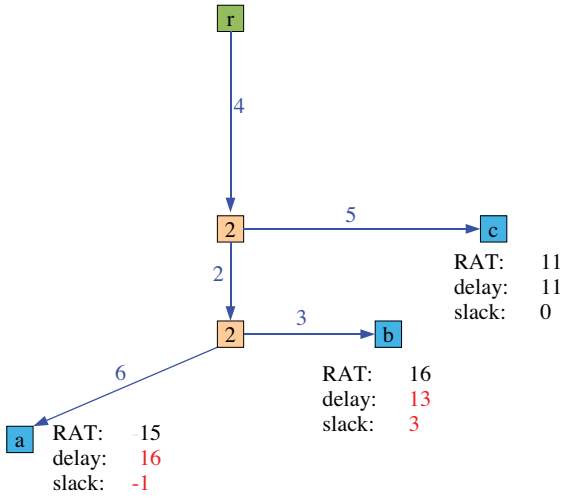

Figure 15. An example for topology generation with $\mathrm{AT}(r) = 0$, $c_{\mathrm{wire}} = 1$, $c_{\mathrm{node}} = 2$, and three sinks a, b and c with displayed required arrival times. The criticalities are $\sigma_a = 15-0-(4+2+6) = 3$, $\sigma_b = 16-0-(4+2+3) = 7$, and $\sigma_c = 11-0-(4+5) = 2$. Our algorithm first connects the most critical sink c to r. The next critical sink is a which is inserted into the only arc (r, c) creating an internal node w. For the insertion of the last sink b there are now three possible arcs (r, w), (w, a), and (w, c). Inserting b into (w, a) creates the displayed topology whose worst slack is -1, which is best possible here.

Proof. If $|S| = 1$, the statement is trivial. Let us assume $|S| > 1$. This means that we have at least one internal node in T. We can assume that all internal nodes are placed at $\mathrm{Pl}(r)$. The slack of a such a topology T is at least $\sigma_{\max}$ if and only if

$$\mathrm{RAT}(s) - \mathrm{AT}(r) - c_{\mathrm{wire}}\|\mathrm{Pl}(r) - \mathrm{Pl}(s)\|_1 - c_{\mathrm{node}} \cdot (|E(T_{[r,s]})| - 1) \geq \sigma_{\max},$$

for all sinks s. Equivalently,

$$|E(T_{[r,s]})| - 1 \leq \frac{\mathrm{RAT}(s) - \mathrm{AT}(r) - c_{\mathrm{wire}}\|\mathrm{Pl}(r) - \mathrm{Pl}(s)\|_1}{c_{\mathrm{node}}} - \frac{\sigma_{\max}}{c_{\mathrm{node}}}.$$

By Kraft's inequality [39] there exists a rooted binary tree with n leaves at depths $l_1, l_2, \ldots, l_n$ if and only if $\sum_{i=1}^{n} 2^{-l_i} \leq 1$. If we contract the arc incident to r in our topology we obtain a binary tree for which $(|E(T_{[r,s]})| - 1)$ is exactly the depth of sink s (remember $|S| > 1$). Now Kraft's inequality implies the theorem. □

It is possible to calculate a slightly better and numerically stable bound using Huffman coding [40]: if we set as in Eq. (6)

$$\sigma_s = \mathrm{RAT}(s) - \mathrm{AT}(r) - c_{\mathrm{wire}}\|\mathrm{Pl}(r) - \mathrm{Pl}(s)\|_1$$

for all $s \in S$, order these values $\sigma_{s_1} \leq \sigma_{s_2} \leq \cdots \leq \sigma_{s_n}$, and iteratively replace the largest two $\sigma_{s_{n-1}}$ and σ_{s_n} by $-c_{\mathrm{node}} + \min\{\sigma_{s_{n-1}}, \sigma_{s_n}\} = -c_{\mathrm{node}} + \sigma_{s_{n-1}}$ until only one value σ^* is left, then the maximum possible slack with respect to our delay model is at most this σ^*. This bound is never worse than the closed formula of Theorem 4. In fact, it corresponds to shortest wires from each sink to the source and an optimum topology

all internal nodes of which are at the position of the root. Such a topology would of cause waste too many wiring resources and lead to excessive power consumption. The topology generated by our algorithm is much better in these respects. Moreover we have the following guarantee.

Theorem 5. *For* $c_{\text{wire}} = 0$, *the topology constructed by the above procedure with* $\xi = 100$ *realizes the maximum possible slack with respect to our delay model.*

For $c_{\text{node}} = 1$ and integer values for AT(r) and RAT(s), $s \in S$, the theorem follows quite easily from Kraft's inequality, by induction on $|S|$ [36]. The general case is more complicated; see [38].

After inserting all sinks into the preliminary topology, the second phase begins, in which we insert the actual inverters [37]. For each sink s we create a cluster C containing only s. In general a cluster C is assigned a position Pl(C), a set of sinks $S(C)$ all of the same parity, and an estimate $W(C)$ for the wiring capacitance of a net connecting a circuit at position Pl(C) with the sinks in $S(C)$. The elements of $S(C)$ are either original sinks of the fan-out tree or inverters that have already been inserted.

There are three basic operations on clusters. Firstly, if $W(C)$ and the total input capacitance of the elements of $S(C)$ reach certain thresholds, we insert an inverter I at position Pl(C) and connect it by wire to all elements of $S(C)$. We create a new cluster C' at position Pl(C) with $S(C') = \{I\}$ and $W(C) = 0$. As long as the capacitance thresholds are not attained, we can move the cluster along arcs of the preliminary topology towards the root r. By this operation $W(C)$ increases while $S(C)$ remains unchanged. Finally, if two clusters happen to lie on a common position and their sinks are of the same parity, we can merge them, but we may also decide to add inverters for some of the involved sinks. This decision again depends on the capacitance thresholds and on the objectives timing and wire length.

During buffering, the root connects to the clusters via the preliminary topology and the clusters connect to the original sinks s_i via appropriately buffered nets. Once all clusters have been merged to one which arrives at the root r, the construction of the fan-out tree is completed.

The optimality statements which we proved within our delay model and the final experimental results show that the second phase nearly optimally buffers the desired connections. Our procedure is extremely fast. The topology generation solved 4.6 million instances with up to 10000 sinks from a current 90 nm design in less than 100 seconds on a 2.93 GHz Xeon machine [37], and the buffering is completed in less than 45 minutes. On average we deviated less than 1.5% from the minimum length of a rectilinear Steiner tree when minimizing wire length, and less than 3 ps from the theoretical upper slack bound when maximizing worst slack.

We are currently including enhanced buffering with respect to timing constraints, wire sizing, and plane assignment in our algorithm. We are also considering an improved topology generation, in particular when placement or routing resources are limited.

3.3. Fan-in trees

Whereas in the last section one signal had to be propagated to many destinations via a logically trivial structure, we now look at algorithmic tasks posed by the opposite situation in which several signals need to be combined to one signal as specified by some

Figure 16. A sequence of circuits with Boolean functions $g_1, g_2, \ldots, g_n$ on a critical path P.

Boolean expression. The netlist itself implicitly defines such a Boolean expression for all relevant signals on a design. The decisions about these representations were taken at a very early stage in the design process, i.e., in logic synthesis, in which physical effects could only be crudely estimated. At a relatively late stage of the physical layout process much more accurate estimates are available. If most aspects of the layout have already been optimized but we still see negative slack at some cells, changing the logic that feeds the cell producing the late signal is among the last possibilities for eliminating the timing problem. Traditionally, late changes in the logic are a delicate matter and only very local modifications replacing some few cells have been considered, also due to the lack of global algorithms.

To overcome the limitations of purely local and conservative changes, we have developed a totally novel approach that allows for the redesign of the logic on an entire critical path taking all timing and placement information into account [41]. Keep in mind that static timing analysis computes slack values for all pins of a design and that it reports timing problems for instance as lists of critical paths. Whereas most procedures for Boolean optimization of combinational logic are either purely heuristic or rely on exhaustive enumeration and are thus very time consuming, our approach is much more effective.

We consider a critical path P which combines a number of signals $x_1, x_2, \ldots, x_n$ arising at certain times $\mathrm{AT}(x_i)$ and locations $\mathrm{Pl}(x_i)$ by a sequence $g_1, g_2, \ldots, g_{n-1}$ of 2-input gates as in Figure 16. Then we try to re-synthesize P in a best possible way.

Critical Path Logic Resynthesis Problem

Instance:
- A set X of sources and a sink y,
- a start time $\mathrm{AT}(x)$ (and possibly a slew) at each source $x \in X$ and a required arrival time $\mathrm{RAT}(y)$ at the sink,
- placement information for the sources and the sink $\mathrm{Pl}(x_1)$, $\mathrm{Pl}(x_2)$, …, $\mathrm{Pl}(x_n)$, $\mathrm{Pl}(y) \in [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$,
- a Boolean expression of the form
$$y = f(x_1, x_2, \ldots, x_n) = g_{n-1}(\ldots g_3(g_2(g_1(x_1, x_2), x_3), x_4) \ldots, x_n)$$
where the g_i are elementary Boolean functions,
- physical information about the driver strength of the sources and the input capacitances of the sink, and
- physical information about the wiring and the library L of available elementary logical circuits (`and`, `or`, `nand`, `nor`, `invert`,…).

Task: Find a circuit representation of y as a function of the $x \in X$
- using elementary Boolean circuits,
- together with placement and sizing information for the circuits such that
- the computation of y completes before $\mathrm{RAT}(y)$, or as early as possible.

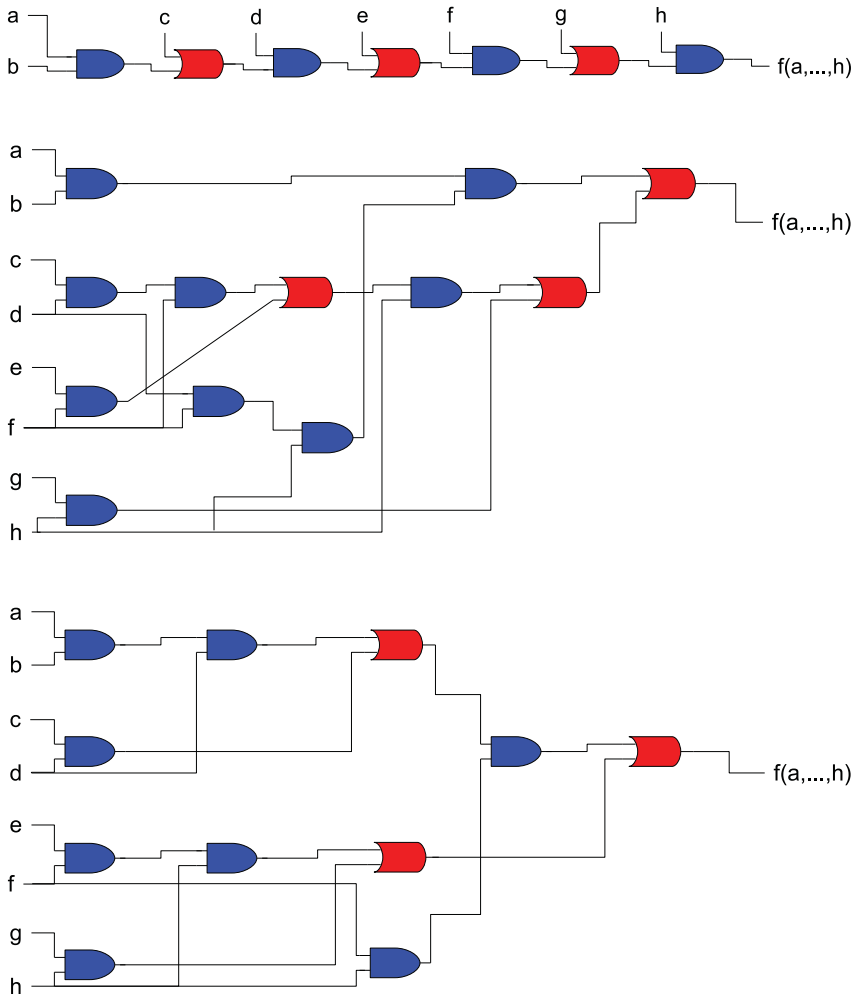

Figure 17. Three logically equivalent circuits for the function $f(a, b, \ldots, h)$ that correspond to the formulas $f(a, \ldots, h) = ((((((a \wedge b) \vee c) \wedge d) \vee e) \wedge f) \vee g) \wedge h$, $f(a, \ldots, h) = ((a \wedge b) \wedge ((d \wedge f) \wedge h)) \vee (((((c \wedge d) \wedge f) \vee (e \wedge f)) \wedge h) \vee (g \wedge h))$, and $f(a, \ldots, h) = ((((a \wedge b) \wedge d) \vee (c \wedge d)) \wedge (f \wedge h)) \vee (((e \wedge f) \wedge h) \vee (g \wedge h))$. The first path is a typical input of our procedure and the two alternative netlists have been obtained by the dynamic programming procedure based on the identity (7). Ignoring wiring and assuming unit delays for the circuits, the second netlist would for instance be optimal for $\mathrm{AT}(a) = \mathrm{AT}(b) = \mathrm{AT}(g) = \mathrm{AT}(h) = 3$, $\mathrm{AT}(e) = \mathrm{AT}(f) = 1$, and $\mathrm{AT}(c) = \mathrm{AT}(d) = 0$, leading to an arrival time of 6 for $f(a, \ldots, h)$ instead of 10 in the input path.

Our algorithm first generates a standard format. It decomposes complex circuits on P into elementary and- and or-circuits with fan-in two plus inversions. Applying the de Morgan rules we eliminate all inversions except for those on input signals of P. We arrive at a situation in which P is essentially represented by a sequence of and- and or-circuits. Equivalently, we could describe the procedure using nand-circuits only, and we will indeed use nands for the final realization. However, for the sake of a simpler description of our algorithm, and- and or-circuits are more suitable.

We now design an alternative, logically equivalent representation of the signal pro-

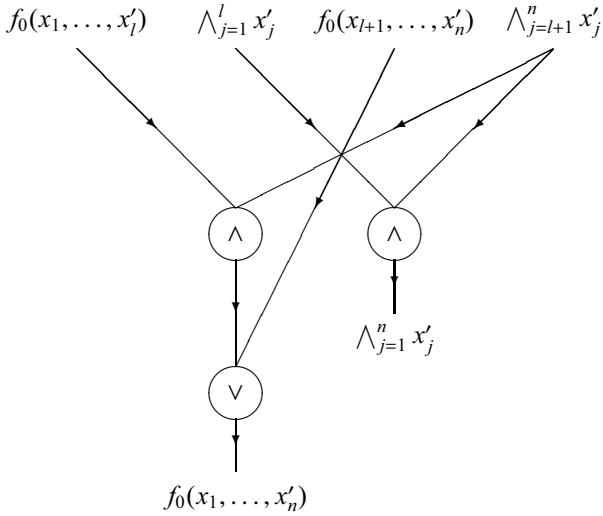

Figure 18. The logic circuit corresponding to Eq. (7).

duced by g_m as a function of the x_i in such a way that late input signals do not pass through too many logic stages of this alternative representation. This is easy if this sequence consists either just of `and`-circuits or just of `or`-circuits. In this situation every binary tree with n leaves leads to a representation of the Boolean function by identifying its leaves with the x_i and its internal nodes with `and`-circuits or `or`-circuits. If we consider only the arrival times $\mathrm{AT}(x_i)$ of the signals which might be justified because all locations are close together, we can easily construct and implement an optimal representation using Huffman coding. If we consider both $\mathrm{AT}(x_i)$ and $\mathrm{Pl}(x_i)$, then inverting time the problem is actually equivalent to the construction of a fan-out tree which we have described in Section 3.2.

The most difficult case occurs if the `and`- and `or`-circuits alternate, i.e., the function calculated by P is of the form

$$f(x_1, x'_1, x_2, x'_2, \ldots, x_n, x'_n) := ((\ldots(((x_1 \wedge x'_1) \vee x_2) \wedge x'_2)\ldots) \vee x_n) \wedge x'_n)$$
$$= \bigvee_{i=1}^{n} \left(x_i \wedge \left(\bigwedge_{j=i}^{n} x'_j \right) \right).$$

See Figure 17 for an example. In this case we apply dynamic programming based on identities like the following:

$$f(x_1, \ldots, x'_n) = \left(f(x_1, \ldots, x'_l) \wedge \left(\bigwedge_{j=l+1}^{n} x'_j \right) \right) \vee \; f(x_{l+1}, \ldots, x'_n). \tag{7}$$

Note that Eq. (7) corresponds to a circuit structure as shown in Figure 18.

Our dynamic programming procedure maintains sets of useful sub-functions such as $f(x_i, \ldots, x'_j)$ and $\bigwedge_{k=i}^{j} x'_k$ together with estimated timing and placement information. In order to produce the desired final signal, these sets of sub-functions are combined using small sets of circuits as shown for instance in Figure 18, and the timing and placement

information is updated. We maintain only those representations that are promising. The final result of our algorithm is found by backtracking through the data accumulated by the dynamic programming algorithm. After having produced a faster logical representation, we apply de Morgan rules once more and collapse several consecutive elementary circuits to more complex ones if this improves the timing behavior. In many cases this results in structures mainly consisting of `nand`-circuits and inverters. Certainly, the number of circuits used in the new representation is typically larger than in the old representation but the increase is at most linear.

Our procedure is very flexible and contains the purely local changes as a special case. Whereas the dynamic programming procedure is quite practical and easily allows us to incorporate physical insight as well as technical constraints, we can validate its quality theoretically by proving interesting optimality statements. In order to give an example for such a statement, let us neglect placement information, assume nonnegative integer arrival times and further assume a unit delay for `and`- and `or`-circuits. We proceed in two steps. First, we derive a lower bound on the arrival time of desired signal and then estimate the arrival time within our new logical representation.

Theorem 6. *If C is a circuit of fan-in 2 for some Boolean function f depending on the inputs $x_1, x_2, \ldots, x_n$ with arrival times $\mathrm{AT}_1, \mathrm{AT}_2, \ldots, \mathrm{AT}_n \in \mathbb{N}_0$, then*

$$\mathrm{AT}(f, C) \geq \left\lceil \log_2\left(\sum_{i=1}^{n} 2^{\mathrm{AT}_i}\right)\right\rceil,$$

where $\mathrm{AT}(f, C)$ denotes the arrival time of the value of f as computed by C assuming a unit delay for every circuit.

Proof. The existence of a circuit C of fan-in 2 that calculates the value of f by the time T implies the existence of a rooted binary tree with n leaves of depths $(T - \mathrm{AT}_1), (T - \mathrm{AT}_2), \ldots, (T - \mathrm{AT}_n) \in \mathbb{N}_0$. By Kraft's inequality, such a tree exists if and only if $\sum_{i=1}^{n} 2^{-(T-\mathrm{AT}_i)} \leq 1$ or, equivalently, $T \geq \log_2(\sum_{i=1}^{n} 2^{\mathrm{AT}_i})$, and the proof is complete. □

In order to estimate the arrival time within the new logical representation we have to analyze the growth behavior of recursions based on the decomposition identities used during the dynamic programming. If we just use Eq. (7) for instance, then we have to estimate the growth of the following recursion which reflects the additional delays incurred by the three circuits in Figure 18: For $n \geq 2$ and nonnegative integers $a, a_1, \ldots, a_n \in \mathbb{N}_0$ let

$$\mathrm{AT}(a) = a$$

$$\mathrm{AT}(a_1, \ldots, a_n) = \min_{1 \leq l \leq n-1} \max\{\mathrm{AT}(a_1, \ldots, a_l) + 2, \mathrm{AT}(a_{l+1}, \ldots, a_n) + 1\}.$$

We have demonstrated how to analyze such recursions in [41,42,43] and how to obtain results like the following.

Theorem 7. *If $a_1, a_2, \ldots, a_n \in \mathbb{N}_0$, then*

$$\mathrm{AT}(a_1, a_2, \ldots, a_n) \leq 1.44 \log_2\left(\sum_{i=1}^{n} 2^{a_i}\right) + 2.$$

Comparing the bounds in Theorems 6 and 7 implies that just using Eq. (7) during the dynamic programming would lead to an algorithm with asymptotic approximation ratio 1.44 [41]. Using further decomposition identities this can be reduced to $(1 + \epsilon)$ for arbitrary $\epsilon > 0$ [42]. Further details about practical application and computational results can be found in [44].

So far we have described an algorithm for the redesign of a critical path which is in fact a main issue during timing closure. This algorithm was "blind" for the actual function that was involved and hence applicable to almost every critical path. We have also devised procedures for the timing-aware design of more complex functions. As an example consider the following so-called prefix problem which is essential for the construction of fast binary adders.

Prefix Problem	
Instance:	An associative operation $\circ\colon D^2 \to D$ and inputs $x_1, x_2, \ldots, x_n \in D$.
Task:	Find a circuit computing $x_1 \circ x_2 \circ \cdots \circ x_i$ for all $1 \leq i \leq n$.

Applying the above algorithm to the n desired output functions would lead to a circuit with good delay properties but with a quadratic number of circuits. Similar constructions with close-to-optimal delay but quadratic size were described also by Liu et al. in [45]. In [46] we constructed circuits solving the prefix problem with close-to-optimal delay and much smaller sizes of $O(n \log(\log(n)))$.

For the addition of two n-bit binary numbers whose $2n$ bits arrive at times $t_1, t_2, \ldots, t_{2n} \in \mathbb{N}_0$ this leads to circuits over the basis $\{\vee, \wedge, \neg\}$ of fan-in 2 for $\vee$- or $\wedge$-circuits and fan-in 1 for $\neg$-circuits calculating the sum of the two numbers with size $O(n \log(\log(n)))$ and delay

$$2 \log_2\Big(\sum_{i=1}^{2n} (2^{t_i})\Big) + 6 \log_2(\log_2(n)) + O(1).$$

In view of Theorem 6, the delay bound is close-to-optimal and the bound on the size is optimal up to a factor of $O(\log(\log(n)))$. The best known adders are of depth $\log_2(n) + O(\sqrt{\log(n)})$ and size $O(n \log(n))$ [47] or size $O(n)$ [48], respectively. The adder developed in [49], which takes arrival times into account, has size $O(n \log(n))$, but no delay bound has been proved.

3.4. Gate Sizing and V_t-Assignment

The two problems considered in this section consist of making individual choices from some discrete sets of possible physical realizations for each circuit of the netlist such that some global objective function is optimized.

For gate sizing one has to determine the size of the individual circuits measured for instance by their area or power consumption. This size affects the input capacitance and driver strength of the circuit and therefore has an impact on timing. A larger circuit typically decreases downstream delay and increases upstream delay. Circuits with a single output pin are called (logic) gates. Assuming gates instead of general multi-output cir-

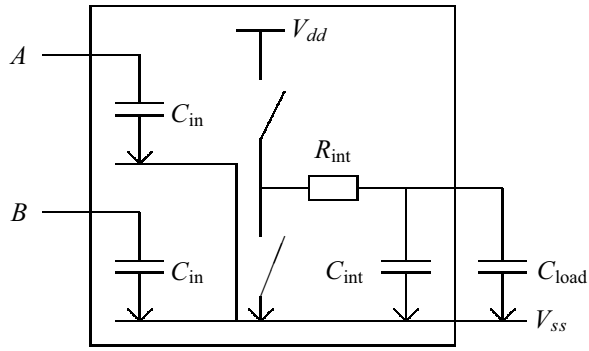

Figure 19. A simple electrical model of a circuit. The input capacitance C_{in} and the internal capacitance C_{int} are proportional to the scaling factor of the circuit while the internal resistance R_{int} is antiproportional.

cuits simplifies mathematical problem formulations. This is the reason why the problem is called rather gate sizing than circuit sizing.

Whereas the theoretically most well-founded approaches for the gate sizing problem rely on convex/geometric programming formulations [50,51,52], these approaches typically suffer from their algorithmic complexity and restricted timing models. In many situations, approaches that choose circuit sizes heuristically can produce competitive results because it is much easier to incorporate local physical insight into heuristic selection rules than into a sophisticated convex program. Furthermore, the convex programming formulations often assume continuous circuit size while standard cell libraries typically only offer a discrete set of different sizes. In BonnOpt we use both, a global formulation and convex programming for the general problem as well as heuristics for special purposes.

For the simplest form of the global formulation we consider a directed graph G which encodes the netlist of the design. G can be considered as the timing graph $G_{\mathcal{T}}$ from Section 3.1 after contracting all input pin vertices. For a set V_0 of nodes v — e.g., start and end nodes of maximal paths — we are given signal arrival times a_v and we must choose circuit sizes $x = (x_v)_{v \in V(G)} \in [l, u] \subseteq \mathbb{R}^{V(G)}$ and arrival times for nodes not in V_0 minimizing

$$\sum_{v \in V(G)} x_v$$

subject to the timing constraints

$$a_v + d_{(v,w)}(x) \leq a_w$$

for all arcs $(v, w) \in E(G)$. The circuit sizes x_v are scaling factors for the internal structures of the circuit v (see Figure 19).

For simplicity it is assumed that the input and internal capacitances of circuit v are proportional to x_v while the internal resistance is antiproportional to x_v. Using the Elmore delay model [53], the delay through circuit v is of the form $R_{\text{int}}(C_{\text{int}} + C_{\text{load}})$ where C_{load} is the sum of the wire capacitance and the input capacitances of the structures that are charged over the circuit v. Since C_{load} depends on the circuit sizes of the corresponding circuits in the same way, the delay $d_{(v,w)}(x)$ of some arc (v, w) of G is modeled by a linear

function with positive coefficients depending on terms of the form x_v, $1/x_v$ and x_w/x_v. Dualizing the timing constraints via Lagrange multipliers $\lambda_{(u,v)} \geq 0$ leads to the following Lagrange function.

$$\begin{aligned} L(x,a,\lambda) &= \sum_{u\in V(G)} x_u + \sum_{(u,v)\in E(G)} \lambda_{(u,v)}(a_u + d_{(u,v)}(x) - a_v) \\ &= \sum_{u\in V(G)} x_u + \sum_{(u,v)\in E(G)} \lambda_{(u,v)} d_{(u,v)}(x) + \sum_{(u,v)\in E(G)} \lambda_{(u,v)}(a_u - a_v) \\ &= \sum_{u\in V(G)} x_u + \sum_{(u,v)\in E(G)} \lambda_{(u,v)} d_{(u,v)}(x) \\ &\qquad + \sum_{u\in V(G)} a_u \Bigg(\sum_{v:(u,v)\in E(G)} \lambda_{(u,v)} - \sum_{v:(v,u)\in E(G)} \lambda_{(v,u)} \Bigg). \end{aligned}$$

Since after the dualization all arrival times $(a_v)_{v\in V(G)}$ (except those that are constant) are free variables, every optimal solution of the dual maximization problem has the property that

$$\sum_{u\in V(G)} a_u \Bigg(\sum_{v:(u,v)\in E(G)} \lambda_{(u,v)} - \sum_{v:(v,u)\in E(G)} \lambda_{(v,u)} \Bigg) = 0$$

for all choices of a_u, i.e., the Lagrange multipliers $\lambda_{(u,v)} \geq 0$ constitute a nonnegative flow on the timing graph [51].

Therefore, for given Lagrange multipliers the problem reduces to minimizing a weighted sum of the circuit sizes x and delays $d_{(u,v)}(x)$ subject to $x \in [l,u]$. This step is typically called local refinement. Generalizing results from [51,54,55,56] we proved that it can be solved by a very straightforward cyclic relaxation method with linear convergence rate in [57]. The overall algorithm is the classical constrained subgradient projection method (cf. [58]). The known convergence guarantees for this algorithm require an exact projection, which means that we have to determine the above-mentioned nonnegative flow on G that is closest to some given vector $(\lambda_e)_{e\in E}$.

Since this exact projection is actually the most time-consuming part, practical implementations use crude heuristics having unclear impact on convergence and quality. To overcome this limitation, we proved in [59] that the convergence of the algorithm is not affected by executing the projection in an approximate and much faster way. This is done by combining the subgradient projection method [60,61] in a careful way with the method of alternating projections [62] and results in a stable, fast, and theoretically well-founded implementation of the subgradient projection procedure for circuit sizing.

Nevertheless, in practice there exist advanced heuristics that are also good and much faster than the subgradient method. Such approaches improve all circuits iteratively based on the "dual" slack values. The improvement does not follow a strict mathematical formula but makes reasonable choices heuristically. Furthermore, time-consuming slack updates are not done after every single cell change but only once per iteration. Additional side constraints like load and slew limits, or placement density can be incorporated easily.

In [63], we developed a heuristic that yields competitive results compared to continuous mathematical optimization models, which lack accuracy due to simplified delay

models and rounding errors. The worst path delays are within 6% of a lower delay bound on average.

Global circuit sizing approaches are followed by local search on the critical paths. Here delay effects due to layout changes are computed accurately and slacks are updated after every circuit change. As accurate computations are extremely time consuming, only a few circuits (approximately 1%) are considered by this local search. Afterwards the worst path delays are within 2% of a lower delay bound on average.

The second optimization problem that we consider in this section is V_t-assignment. A physical consequence of feature size shrinking is that leakage power consumption represents a growing part of the overall power consumption of a chip. Increasing the threshold voltage of a circuit reduces its leakage but increases its delay. Modern libraries offer circuits with different threshold voltages. The optimization problem that we face is to choose the right threshold voltages for all circuits, which minimize the overall (leakage) power consumption while respecting timing restrictions.

As proposed in [64,65], we first consider a netlist in which every circuit is realized in its slowest and least-leaky version. We define an appropriate graph G whose arcs are assigned delays, and some of whose arcs correspond to circuits for which we could choose a faster yet more leaky realization. For each such arc e we can estimate the power cost c_e per unit delay reduction. We add a source node s joined to all primary inputs and to all output nodes of memory elements and a sink node t joined to all primary outputs and to all input nodes of memory elements. Then we perform a static timing analysis on this graph and determine the set of arcs E' that lie on critical paths.

The general step now consists in finding a cheapest s-t-cut $(S, \overline{S})$ in $G' = (V(G), E')$ by a max-flow calculation in an auxiliary network. Arcs leaving S that can be made faster contribute c_e to the cost of the cut, and arcs entering S that can be made slower contribute $-c_e$ to the cost of the cut. Furthermore, arcs leaving S that cannot be made faster contribute ∞ to the cost of the cut, and arcs entering S that cannot be made slower contribute 0 to the cost of the cut.

If we have found such a cut of finite cost, we can improve the timing at the lowest possible power cost per time unit by speeding up the arcs from S to $\overline{S}$ and slowing down (if possible) the arcs from $\overline{S}$ to S. The acceleration is performed until nonaccelerated paths become critical and the next iteration is performed on a growing auxiliary network. Figure 20 illustrates the algorithm. The optimality statement is proved in [66] subject to the simplifying assumptions that the delay/power dependence is linear and that we can realize arbitrary V_t-values within a given interval, which today's libraries typically do not allow. Nevertheless, the linearity of the delay/power dependence approximately holds locally and the discrete selectable values are close enough. There are strong connections to the so-called discrete time-cost tradeoff problem as studied for instance in [67].

We point out that the described approach is not limited to V_t-assignment. It can be applied whenever we consider roughly independent and local changes and want to find an optimal set of operations that corrects timing violations at minimum cost. This has been part of BonnTools for some time [68], but previously without using the possibility of slowing arcs from $\overline{S}$ to S, and thus without optimality properties.

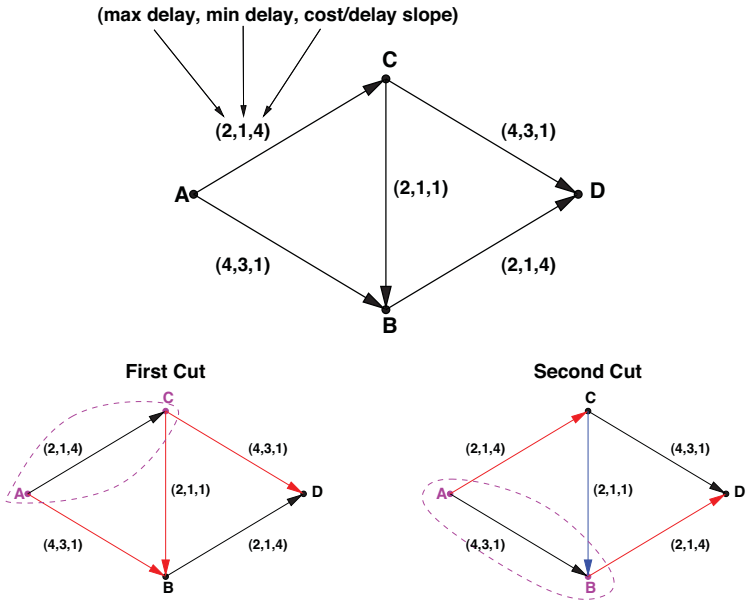

Figure 20. A time-cost-tradeoff instance (top). For each arc the maximum delay, the minimum delay, and the cost increase per unit delay decrease are specified. Initially every arc delay is chosen slowest possible. All three paths have the same delay of 6 time units. The minimum A-D-cut is $(\{A, C\}, \{B, D\})$ with value $3 = 1 + 1 + 1$ (bottom left). After accelerating the involved arcs $(A, B), (C, B)$ and (C, D) all paths have delay 5. Now the minimum cut is $(\{A, B\}, \{C, D\})$ with value $7 = 4 + 4 - 1$ (bottom right). Note that (A, C) and (B, D) are leaving the cut and therefore accelerated, but (C, B) is entering the cut and decelerated. All arcs except (C, B) reached their minimum delay therefore have infinity capacitance. Now the minimum cut has weight infinity and the algorithm stops. The critical paths $A \to B \to D$ and $A \to C \to D$ cannot be accelerated further.

4. Clock Scheduling and Clock Tree Construction

Most computations on chips are synchronized. They are performed in multiple cycles. The result of a cycle is stored in special memory elements (registers, flip-flops, latches) until it is used as input for the next cycle. Each memory element receives a periodic clock signal, controlling the times when the bit at the data input is to be stored and transferred to further computations in the next cycle. Today it is well-known that striving for simultaneous clock signals (zero skew), as most chip designers did for a long time, is not optimal. By clock skew scheduling, i.e., by choosing individual clock signal arrival times for the memory elements, one can improve the performance. However, this also makes clock tree synthesis more complicated. For nonzero skew designs it is very useful if clock tree synthesis does not have to meet specified points in time, but rather time intervals. We proposed this methodology together with new algorithms in [69,70,71]. Since then it has been successfully applied on many industrial high performance ASICs.

4.1. Clock Skew Scheduling

Let us define the latch graph as the digraph whose vertex set is the set of all memory elements and which contains an arc (v, w) if the netlist contains a path from the output of v to the input of w. Let $d_{(v,w)}$ denote the maximum delay of a path from v to w. If all

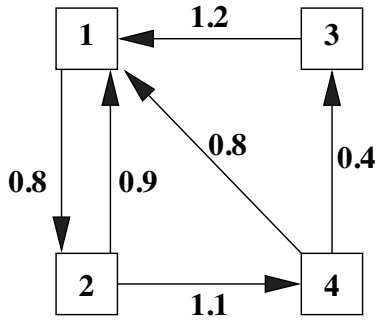

Figure 21. A latch graph with 4 latches (numbered boxes). The arc numbers specify the longest path delays. With a zero skew tree — when all latches switch simultaneously — the slowest path ($3 \to 1$) determines the cycle time $T_{zs} = 1.2$. With an optimum scheduled tree the slowest average delay cycle ($1 \to 2 \to 4 \to 1$) determines the cycle time $T_{opt} = 0.9$.

memory elements receive a periodic clock signal of the same frequency $1/T$ (i.e., their cycle time is T), then a zero skew solution is feasible only if all delays are at most T. Figure 21 shows a latch graph with four latches. In this example the minimum cycle time with a zero skew tree would be 1.2, bounded by the path $3 \to 1$.

With clock skew scheduling one can relax this condition. Let latch 3 in the above example switch earlier by 0.2 time units. Now signals on path $3 \to 1$ could spent 0.2 more time units per cycle. In turn the maximum allowed delay for signals on path $4 \to 3$ would decrease by 0.2 time units, which would not harm the overall cycle time as the path delay of 0.4 is very fast. Now path $2 \to 4$ determines a limit for the minimum cycle time of 1.1.

Motivated by this observation, the question arises how much can we improve the performance for given delays? We ask for arrival times a_v of clock signals at all memory elements x such that

$$a_v + d_{(v,w)} \leq a_w + T \tag{8}$$

holds for each arc (v, w) of the latch graph. We call such arrival times *feasible*. Now the best achievable cycle time T due to clock scheduling is given by following theorem.

Theorem 8. *Given a latch graph G with arcs delays d, the minimum cycle time T for which feasible arrival times $a\colon V(G) \to \mathbb{R}$ exist equals the maximum mean delay $d_{E(C)}/|E(C)|$ of a directed cycle in C in G.*

Proof. T is feasible if and only if there are arrival times a such that:

$$a_c + d_{(v,w)} \leq a_w + T \qquad \forall (v, w) \in E(G).$$

From shortest path theory it is known that such arrival times (node potentials) exist if and only if (G, c) does not contain any cycle with negative total cost, where $c(e) := T - d_e$. Equivalently,

$$T \geq \frac{d_{E(C)}}{|E(C)|} \qquad \text{for all cycles } C \text{ in } G.$$

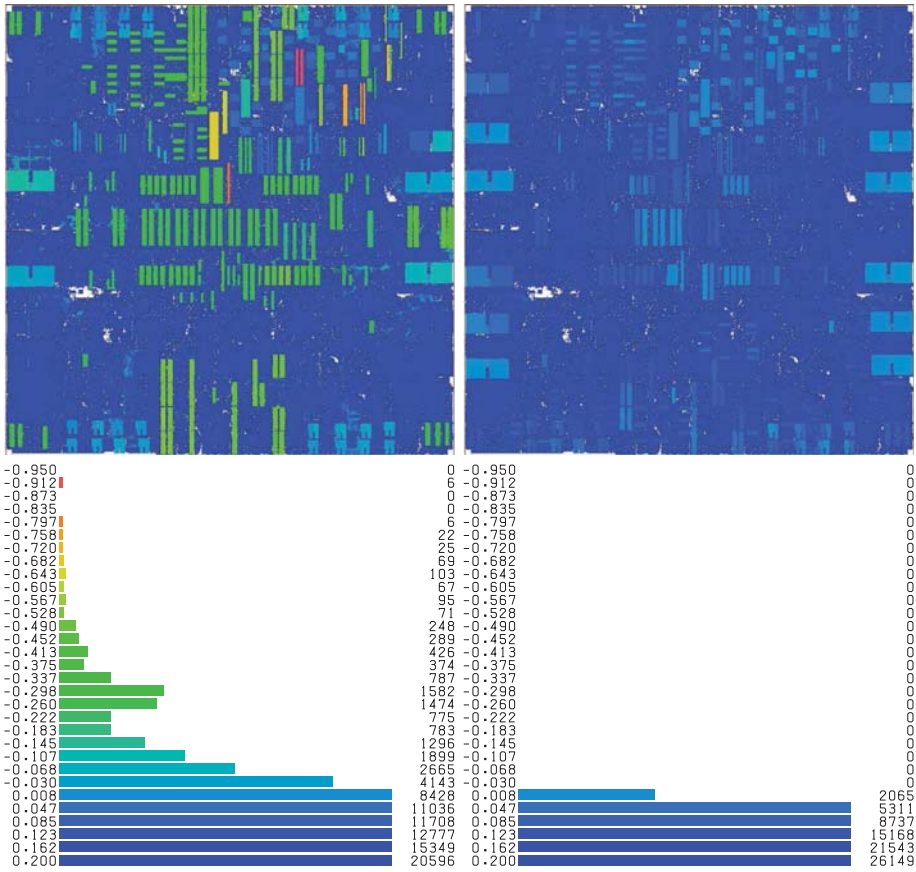

Figure 22. Slack histograms showing the improvement due to clock skew scheduling and appropriate clock tree synthesis; left: zero skew, right: with BonnClock trees. Each histogram row represents a slack interval (in ns) and shows the number of circuits with their worst slacks in this range. The placements on top are also colored according to the worst circuit slacks.

This shows that the minimum possible T equals the longest average delay of a cycle. □

In the example of Figure 21 this gives an optimum cycle time of 0.9, which can be computed easily by enumerating all three cycles. In general, the optimal feasible cycle time T and feasible clock signal arrival times $a(T)$ can be computed by minimum mean cycle algorithms, e.g., those of Karp [72] or Young, Tarjan, and Orlin [73].

However, this simple situation is unrealistic. Today systems on a chip have multiple frequencies and often several hundred different clock domains. The situation is further complicated by transparent latches, user-defined timing tests, and various advanced design methodologies.

Moreover, it is not sufficient to maximize the frequency only. The delays that are input to clock skew scheduling are necessarily estimates: detailed routing will be done later and will lead to different delays. Thus one would like to have as large a safety margin — positive slack — as possible. In analogy to the slack definition from Section 3.1 the arc slack $\sigma_{(v,w)}$ for given T and a is defined as $\sigma_{(v,w)} := a_w - a_v - d_{(v,w)} + T$ for every arc

$(v, w) \in E(G)$. Note that $a_w + T$ defines a required arrival time for signals entering latch w. In fact, maximizing the worst slack by clock scheduling is equivalent to minimizing the cycle time:

Proposition 9. *Let G be a latch graph with arc delays d. Let T' be the minimum possible cycle time for (G, d), $T > 0$, and*

$$\sigma'_T = \max_a \min_{(v,w)\in E(G)} (a_w - a_v - d_{(v,w)} + T)$$

be the maximum achievable worst slack for cycle time T. Then

$$T' = T - \sigma'_T.$$

Proof. This follows from the observation that $\sigma'_{T'} = 0$ and the difference $T - \sigma'_T$ is invariant in T. □

In practice the cycle time is a fixed input parameter and clock scheduling is used to achieve this cycle time and optimize the overall slack distribution.

Next, signals can also be too fast. This means that we are also given minimum delays $\delta_{(v,w)}$ of each path $(v, w) \in E(G)$, and a signal must not arrive in the previous cycle:

$$a_v + \delta_{(v,w)} \geq a_w, \qquad \forall (v, w) \in E(G).$$

Although such early-mode violations can be repaired by delay insertion via buffering, this can be very expensive in terms of placement and wiring resources as well as power consumption. Clock skew scheduling can remove most early-mode violations at almost no cost.

Finally, it is very hard to realize arbitrary individual arrival times exactly; moreover this would lead to high power consumption in clock trees. Computing time intervals rather than points in time is much better. Without making critical paths any worse, the power consumption (and use of space and wiring resources) by clock trees can be reduced drastically. Intervals for the clock arrival times can be introduced by splitting every $v \in V(G)$ into two nodes v_l, v_u and adding the constraints $a_{v_l} \leq a_{v_u}$. Delay constraints have to be reconnected to the corresponding split nodes, i.e. $a_v + d_{(v,w)} \leq a_w + T$ is replaced by $a_{v_u} + d_{(v,w)} \leq a_{w_l} + T$. Now $[a_{v_l}, a_{v_u}]$ defines an admissible arrival time interval for v.

A three-stage clock skew scheduling approach was proposed by Albrecht et al. [70]. Firstly, only late-mode slacks are considered. Then early-mode violations are reduced (more precisely, slacks that can be increased by inserting extra delays), without decreasing any negative or small positive late-mode slacks. Thirdly, a time interval for each memory element is computed such that whenever each clock signal arrives within the specified time interval, no negative or small positive slack will decrease. In the next section we discuss how to balance a certain set of slacks while not decreasing others.

4.2. Slack Balancing

In [70,74,65], generalizing the early work of Schneider and Schneider [75] and Young, Tarjan and Orlin [73], we have developed slack balancing algorithms for very general situations. The most general problem can be formulated as follows.

The Slack Balancing Problem

Instance: A directed graph G (the timing graph), $c : E(G) \to \mathbb{R}$ (delays), a set $F_0 \subseteq E(G)$ (arcs where we are not interested in positive slack) and a partition $\mathcal{F}$ of $E(G) \setminus F_0$ (groups of arcs in which we are interested in the worst slack only), and weights $w\colon E(G)\setminus F_0 \to \mathbb{R}_{>0}$ (sensitivity of slacks), such that there are no positive delay cycles in (V, F_0).

Task: Find arrival times $\pi\colon V(G) \to \mathbb{R}$ with

$$\pi(v) + c(e) \le \pi(w) \quad \text{for } e = (v, w) \in F_0 \tag{9}$$

such that the vector of relevant slacks

$$\left(\min\left\{\frac{\pi(w) - \pi(v) - c(e)}{w(e)} \;\middle|\; e = (v, w) \in F\right\}\right)_{F\in\mathcal{F}} \tag{10}$$

(after sorting entries in nondecreasing order) is lexicographically maximal.

Note that the delays c include cycle adjusts and thus can be negative (for the latch graph example above $c(e)$ is the propagation delay minus the cycle time T). The conditions for $e \in F_0$ correspond to edges on which slack must be nonnegative, but the actual amount is not of interest. In the following we denote $\sigma_\pi(e) := \pi(w) - \pi(v) - c(e)$ as the slack of $e \in E(G)$ with respect to the node potential π.

An alternative formulation of the Slack Balancing Problem is given by the following theorem.

Theorem 10. *Let $(G, c, w, \mathcal{F})$ be an instance of the* Slack Balancing Problem. *Let $\pi\colon V(G) \to \mathbb{R}$ with $\sigma_\pi(e) \ge 0$ for $e \in F_0$. For $F \in \mathcal{F}$ define*

$$F_\pi := \left\{e \in F \;\middle|\; \frac{\sigma_\pi(e)}{w(e)} \text{ minimal in } F\right\}.$$

and $E_\pi = \bigcup_{F\in\mathcal{F}} F_\pi$. Then π is an optimum solution if and only if there are no $F \in \mathcal{F}$ and $X_f \subset V(G)$ for $f \in F_\pi$ such that

$$\begin{aligned}
&f \in \delta^-(X_f) && \text{for } f \in F_\pi,\\
&\sigma_\pi(e) > 0 && \text{for } f \in F_\pi,\ e \in \delta^+(X_f) \cap F_0,\\
&\frac{\sigma_\pi(e)}{w(e)} > \frac{\sigma_\pi(f)}{w(f)} && \text{for } f \in F_\pi,\ e \in \delta^+(X_f) \cap E_\pi.
\end{aligned} \tag{11}$$

Proof. If there is an $F \in \mathcal{F}$ and $X_f \subset V(G)$ for $f \in F_\pi$ with Eq. (11), then π is not optimum, because setting $\pi'(v) := \pi(v) - \epsilon \cdot |\{f \in F_\pi : v \in X_f\}|$ for $v \in V(G)$ for a sufficiently small $\epsilon > 0$ increases the sorted vector (10) lexicographically (disproving optimality).

Let now $\pi, \pi'\colon V(G) \to \mathbb{R}$ be two vectors with Eqs. (9) and (11), and suppose there exists an $f = (p, q) \in E_\pi \cup E_{\pi'}$ with $\sigma_\pi(f) \ne \sigma_{\pi'}(f)$ and choose f such that $\min\{\sigma_\pi(f), \sigma_{\pi'}(f)\}/w(f)$ is minimum. Without loss of generality $\sigma_\pi(f) < \sigma_{\pi'}(f)$. Let $F \in \mathcal{F}$ be the set containing f. Then $f \in F_\pi$: The contrary assumption would imply that there exists $f' \in F_\pi$ with $\sigma_\pi(f')/w(f') < \sigma_\pi(f)/w(f)$. Then $\sigma_{\pi'}(f')/w(f') = \sigma_\pi(f')/w(f') < \sigma_\pi(f)/w(f) < \sigma_{\pi'}(f)/w(f)$ and thus $f \notin F_{\pi'}$, a contradiction.

For each $f' = (p, q) \in F_\pi$ let $X_{f'}$ be the set of all vertices reachable from q via arcs $e \in F_0$ with $\sigma_\pi(e) = 0$ or arcs $e \in E_\pi$ with $\sigma_\pi(e)/w(e) \leq \sigma_\pi(f)/w(f)$.

Then there exists an $f' = (p, q) \in F$ with $p \in X_{f'}$; for otherwise $(X_{f'})_{f' \in F}$ would satisfy Eq. (11). Hence there is a q-p-path P that consists only of arcs e with $\sigma_\pi(e) \leq \sigma_{\pi'}(e)$. Summation yields $\sigma_\pi(f) \geq \sigma_{\pi'}(f)$, a contradiction. □

This proof is essentially due to [35]. The special case $w \equiv 1$, $|F| = 1$ $\forall F \in \mathcal{F}$, was considered by Albrecht [76], and the *minimum balance problem* ($w \equiv 1$, $F_0 = \emptyset$, $\mathcal{F} = \{\{e\} \mid e \in E(G)\}$ by Schneider and Schneider [75].

Now we show how to solve the Slack Balancing Problem.

Theorem 11. *The* Slack Balancing Problem *can be solved in strongly polynomial time* $O(I \cdot (n^3 \log n + \min\{nm, n^3\} \cdot \log^2 n \log\log n + nm \log m))$, *where* $I := n + |\{F \in \mathcal{F} : |F| > 1\}|$, *or in pseudopolynomial time* $O(w_{\max}(nm + n^2 \log n) + I(n \log n + m))$ *for integral weights* $w\colon E(G) \to \mathbb{N}$ *with* $w_{\max} := \max\{w(e) \mid e \in E \setminus F_0\}$.

Sketch of proof. We may assume that $(V(G), F_0)$ contains no circuit of positive total weight. Set $w(e) := 0$ for $e \in F_0$ Analogously to the proof of Theorem 8, it is easy to see that the maximum worst weighted slack is given by the negative maximum weighted delay $-c(E(C))/w(E(C))$ of a cycle C in G with $w(E(C)) > 0$ and that arrival times exist that achieve this value.

Thus, an optimum solution for the Slack Balancing Problem can be obtained by iteratively identifying the maximum weighted delay cycle C, resolving all intersecting partitions, and and just preserving their respective minimum weighted slacks $-c(E(C))/w(E(C))$ in subsequent iterations. Algorithm 2 describes the overall procedure.

In each iteration, the edges $e \in F \cap E(C)$ determine the final worst weighted slack $\min\{\sigma_\pi(f)/w(f) : f \in F\}$ for their sets F. In lines 4 – 10 we fix the slack on all edges in sets $F \in \mathcal{F}$ that intersect C. Note that the delay and weighting modifications just preserve $\sigma_\pi(e)/w(e) \geq \lambda^\star$ in future iterations, but prevent the slacks of these edges from growing at the cost of less critical partitions.

If $|V(C)| > 1$, the critical cycle is contracted requiring adaption of incoming and outgoing edge costs in lines 12 – 17. Contraction may leave loops $e = (v, v)$ that can be removed unless $e \in F \in \mathcal{F}$ with $|F| > 1$. In this case it is not clear whether e or another edge from F will be the most critical edge in F. Thus, e must be kept, and may later lead to critical cycles C that are loops.

In each iteration either a cycle C with $|E(C)| > 1$ is contracted or, if C is a loop, a set F with $|F| > 1$ is removed from $\mathcal{F}$. Thus there are at most $I := n + |\{F \in \mathcal{F} : |F| > 1\}|$ iterations of the while loop.

The dominating factor in each iteration is the computation of the maximum weighted delay cycle. This can be done in strongly polynomial time $O(\min\{n^3 \log^2 n +$
$n^2 m \log m,$
$n^3 \log n + n^2 m \log^2 n \log\log n\})$ by an adaption [74] of Megiddo's [77] minimum ratio cycle algorithm for nonsimple graphs.

Alternatively, adopting the minimum balance algorithm of Young, Orlin and Tarjan [73] for our purpose, the cumulative running time for all cycle computations is bounded by the bound for a single maximum weighted delay cycle computation, which is $O(w_{\max}(mn + n^2 \log n))$. This is the fastest algorithm for small $w_{\max}$ (especially if $w\colon \mathbb{R} \to \{0, 1\}$). Detailed proofs can be found in [65], and in [70] for unit weights. □

```
1:  while (F ≠ ∅) do
2:      Compute the maximum weighted delay cycle C;
3:      λ* ← c(C)/w(C);
    /* Fixing (slack-preserving) delays in critical partitions */
4:      for F ∈ F with F ∩ E(C) ≠ ∅ do
5:          for f ∈ F do
6:              c(f) ← c(f) + λ*w(f);
7:              w(f) ← 0;
8:          end for
9:          F ← F \ {F};
10:     end for
    /* Critical cycle contraction*/
11:     if |V(C)| > 1 then
12:         for (x, y) ∈ δ⁻(V(C)) do
13:             c(x, y) ← c(x, y) − π(y);
14:         end for
15:         for (x, y) ∈ δ⁺(V(C)) do
16:             c(x, y) ← c(x, y) + π(x);
17:         end for
18:         Contract C;
19:         Remove irrelevant loops from G;
20:     end if
21: end while
```

Algorithm 2. Slack Balancing Algorithm

In practice, the Slack Balancing Problem can be solved much faster if we replace $(\pi(w) - \pi(v) - c(e))/w(e)$ by $\min\{\Theta, \pi(w) - \pi(v) - c(e)\}$ in (10), i.e., ignore slacks beyond a certain threshold Θ, which we typically set to small positive values. In our three stage clock scheduling approach optimizing only slacks below some threshold is necessary to enable optimization in the next stage. Otherwise all arrival time relations would be fixed already.

Positive slacks which have been obtained in a previous stage should not be decreased in later stages. This can be modeled by increasing the corresponding delays and setting their weights to 0, analogously to Algorithm 2. Time intervals for clock signal arrival times also correspond to positive slack on arcs described in the last section. Time intervals are maximized by the same algorithm.

By working on the timing graph — a direct representation of the timing analysis constraints — rather than on the latch graph, we can consider all complicated timing constraints, different frequencies, etc. directly. Furthermore its size is linear in the size of the netlist, while the latch graph can have a quadratic number of arcs and be much bigger.

On the other hand, in the timing graph model the vector of endpoints is optimized instead the vector of longest paths as in the latch graph model. In an optimum solution all paths entering the most critical cycle will obtain the most critical slack and give an overall worse timing result. In our experiments it turned out to be most efficient to use a combination of the latch graph — on most critical parts only — and the timing graph,

incorporating the advantages of both models.

Figure 22 shows a typical result on a leading-edge ASIC. The left-hand side shows the slacks after timing-driven placement, but without clock skew scheduling, assuming zero skew and estimating the on-chip variation on clock tree paths with 300 ps. The right-hand side shows exactly the same netlist after clock skew scheduling and clock tree synthesis. The slacks have been obtained with a full timing analysis as used for sign-off, also taking on-chip variation into account. All negative slacks have disappeared. In this case we improved the frequency of the most critical clock domain by 27%. The corresponding clock tree is shown in Figure 25. It runs at 1.033 GHz [71], which is a very high frequency for an ASIC design even today, several years later. Next we explain how to construct such a clock tree, using the input of clock skew scheduling.

4.3. Clock Tree Synthesis

The input to clock tree construction is a set of sinks, a time interval for each sink, a set of possible sources, a logically correct clock tree serving these sinks, a library of inverters and other books that can be used in the clock tree, and a few parameters, most importantly a slew target. The goal is to replace the initial tree by a logically equivalent tree which ensures that all clock signals arrive within the specified time intervals.

Current ASIC chips contain several hundred clock tree instances with up to a million sinks. For gigahertz frequencies manufacturing process variations already dissipate 20 – 30% of the cycle time. Therefore clock trees have to be constructed very carefully, especially when realizing the delay targets induced by the arrival time windows.

Traditionally the individual delay constraints were met by balancing wires (cf. Chao et al. [78]). Theoretically very exact delay targets can be met by tuning wire delay. The drawback of this approach is that it often requires a lot of wiring resources, the prescribed wiring layout is hard to achieve in detailed routing (see Section 5.4), and due to increasing wiring resistances in new technologies delays increase significantly.

We proposed a different approach [71], which assumes all inserted wires to be routed shortest possible. Delays are balanced by the constructed tree topology and accurate gate sizing.

First, the input tree is condensed to a minimal tree by identifying equivalent books and removing buffers and inverter pairs. For simplicity we will assume here that the tree contains no special logic and can be constructed with inverters only.

Next we do some preprocessing to determine the approximate distance to a source from every point on the chip, taking into account that some macros can prevent us from going straight towards a source.

The construction then proceeds in a bottom-up fashion (cf. Figure 23). Consider a sink s whose earliest feasible arrival time is latest, and consider all sinks whose arrival time intervals contain this point in time. Then we want to find a set of inverters that drives at least s but maybe also some of the other sinks. For each inverter we have a maximum capacitance which it can drive, and the goal is to minimize power consumption.

The input pins of the newly inserted inverters become new sinks, while the sinks driven by them are removed from the current set of sinks. When we insert an inverter, we fix neither its position nor its size. Rather we compute a set of octagons as feasible positions by taking all points with a certain maximal distance from the intersection of the sets of positions of its successors, and subtracting blocked areas and all points that are too far away from a source (cf. Figure 24). This can be computed efficiently [79].

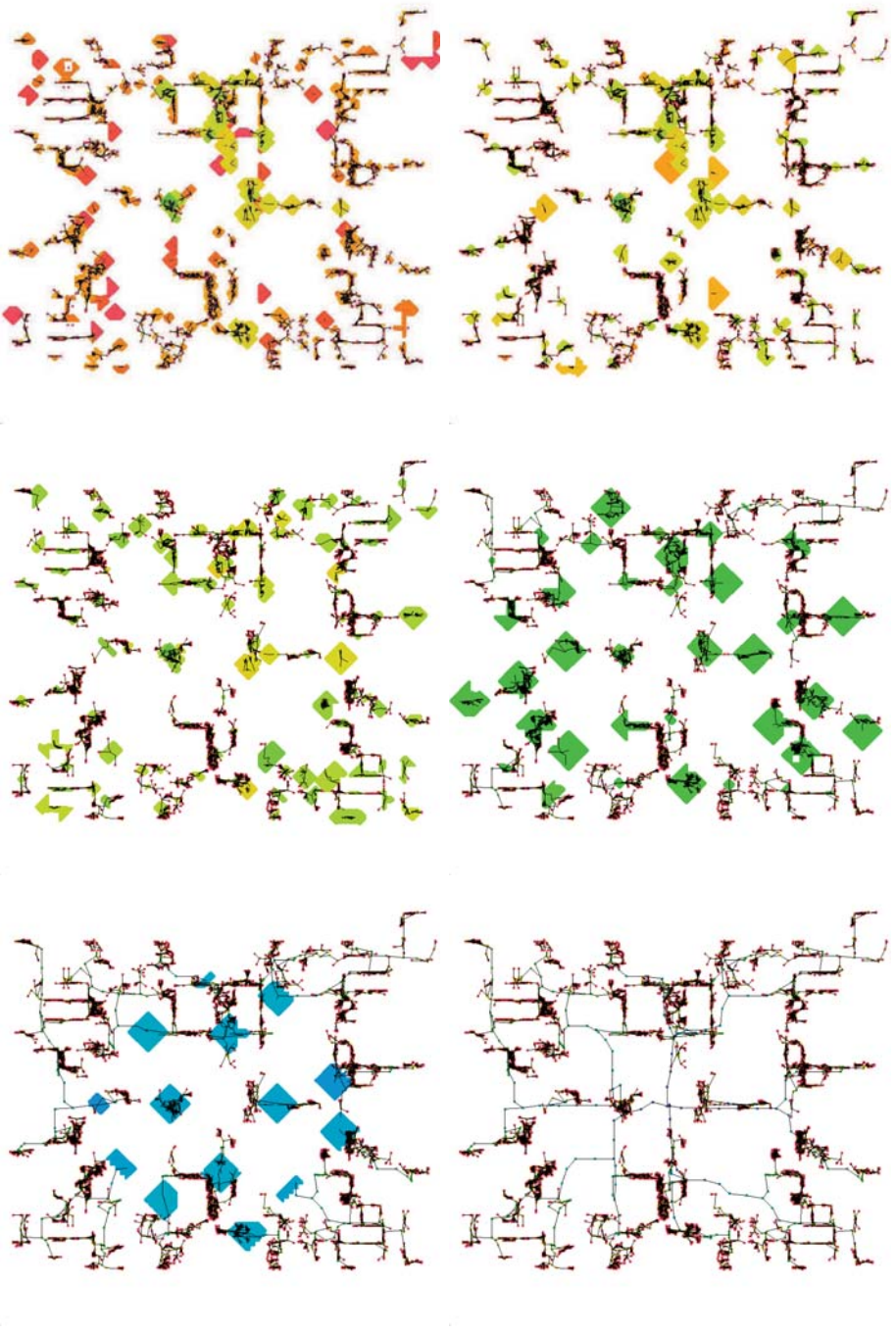

Figure 23. Different stages of a clock tree construction using BonnClock. The colored octagons indicate areas in which inverters (current sinks) can be placed. The colors correspond to arrival times within the clock tree: blue for signals close to the source, and green, yellow, and red for later arrival times. During the bottom-up construction the octagons slowly converge to the source, here located approximately at the centre of the chip.

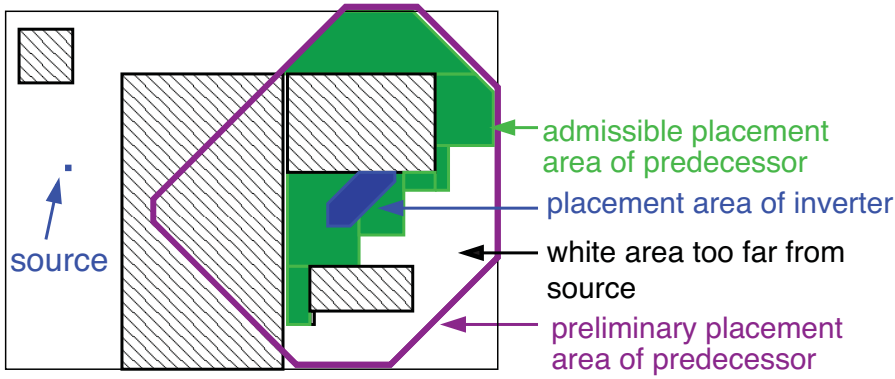

Figure 24. Computation of the feasible area for a predecessor of an inverter. From all points that are not too far away from the placement area of the inverter (blue) we subtract unusable areas (e.g., those blocked by macros) and points that are too far away from the source. The result (green) can again be represented as a union of octagons.

The inverter sizes are determined only at the very end after constructing the complete tree. During the construction we work with solution candidates. A solution candidate is associated with an inverter size, an input slew, a feasible arrival time interval for the input, and a solution candidate for each successor. We prune dominated candidates, i.e., those for which another candidate with the same input slew exists whose time interval contains the time interval of the former. Thus the time intervals imply a natural order of the solution candidates with a given input slew.

Given the set of solution candidates for each successor, we compute a set of solution candidates for a newly inserted inverter as follows. For each input slew at the successors we simultaneously scan the corresponding candidate lists in the natural order and choose maximal intersections of these time intervals. For such a nondominated candidate set we try all inverter sizes and a discrete set of input slews and check whether they generate the required input slews at the successors. If so, a new candidate is generated.

After an inverter is inserted but before its solution candidates are generated, the successors are placed at a final legal position. It may be necessary to move other objects, but with BonnPlace legalization (cf. Section 2.6) we can usually avoid moves with a large impact on timing. There are some other features which pull sinks towards sources, and which cause sinks that are ends of critical paths to be joined early in order to bound negative timing effects due to on-chip variation.

The inverter sizes are selected at the very end by choosing a solution candidate at the root. The best candidate (i.e., the best overall solution) with respect to timing (interval matching and tree latency) and power consumption is chosen. Due to discretizing slews, assuming bounded RC delays, and legalization, the timing targets may be missed by a small amount, in the order of 20 ps. But this impacts the overall timing result only if the deviation occurs in opposite directions at the ends of a critical path.

4.4. Sink Clustering

The overall power consumption of the clock trees is dominated by the bottom stage, where 80 – 90% of the power is consumed. Therefore this stage is very important.

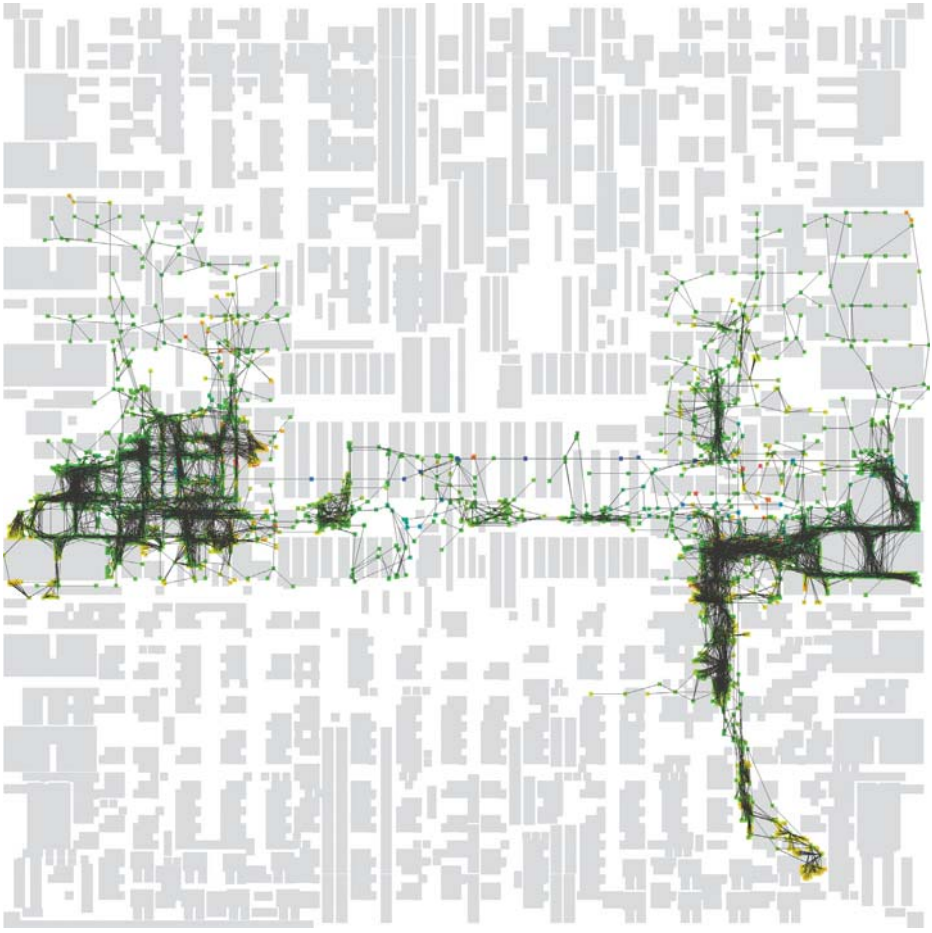

Figure 25. Gigahertz clock tree built by BonnClock based on the result of BonnCycleOpt shown in Figure 22. Colors indicate different arrival times as in Figure 23. Each net is represented by a star connecting the source to all sinks.

The basic mathematical problem that we face here can be formulated as following kind of *facility location problem*:

SINK CLUSTERING PROBLEM

Instance: A metric space (V, c),
a finite set $\mathcal{D} \subseteq V$ (terminals/clients),
demands $d\colon \mathcal{D} \to \mathbb{R}_+$ (input pin capacitances),
facility opening cost $f \in \mathbb{R}_+$ (cost for inserting a driver circuit),
capacity $u \in \mathbb{R}_+$ (capacity limit for a facility).

Task: Find a partition $\mathcal{D} = D_1 \dot\cup \ldots \dot\cup D_k$ and
Steiner trees T_i for D_i $(i = 1, \ldots, k)$ with

$$c(E(T_i)) + d(D_i) \le u \quad \text{for } i = 1, \ldots, k \tag{12}$$

such that $\sum_{i=1}^{k} c(E(T_i)) + kf$ is minimum.

The term $c(E(T_i)) + d(D_i)$ in Eq. (12) is the total *load* capacitance (wire plus input pin capacitances) that must be served/driven by a facility/cell. The objective function models power consumption. In our case, V is the plane and c is the l_1-metric.

The sink clustering problem is closely related to the soft-capacitated facility location problem. It contains the bin packing problem and the Steiner tree problem. The problem can therefore not be approximated arbitrary well [80]:

Theorem 12. *The* Sink Clustering Problem *has no* $(2-\epsilon)$*-approximation algorithm for any* $\epsilon > 0$ *for any class of metrics where the Steiner tree problem cannot be solved exactly in polynomial time.*

Proof. Assume that we have a $(2-\epsilon)$-approximation algorithm for some $\epsilon > 0$, and let $S = \{s_1, \ldots, s_n\}$, $k \in \mathbb{R}_+$ be an instance of the decision problem `Is there a Steiner tree for` S `with length` $\leq k$`?`. We construct an instance of the Sink Clustering Problem by taking S as the set of terminals, setting $d(s) = 0\ \forall s \in S$, $u = k$ and $f = 2k/\epsilon$. Then the $(2-\epsilon)$-approximation algorithm computes a solution consisting of one facility if and only if there is a Steiner tree of length $\leq k$. This implies that the above decision problem, and hence the Steiner tree problem, can be solved in polynomial time. □

The first constant-factor approximation algorithms for this problem were given by Maßberg and Vygen [80]. One of them has a very fast running time of $O(n \log n)$ and is described now.

Let F_1 be a minimum spanning tree for $(\mathcal{D}, c)$ and $e_1, \ldots, e_{n-1}$ be the edges of F_1 in sorted order such that $c(e_1) \geq \cdots \geq c(e_{n-1})$. Let us further define a sequence of forests by $F_k := F_{k-1} \setminus \{e_{k-1}\}$ for $k = 2, \ldots, n$. Exploiting the matroid property of forests it is easy to see that each F_k, $k = 1, \ldots, n$ is a minimum weight spanning forest with exactly k connected components. By a k-Steiner forest we mean a forest F with exactly k connected components and $\mathcal{D} \subseteq V(F)$. By extending the Steiner ratio[5] from minimum spanning trees to minimum spanning forests we get:

Lemma 13. $(1/\alpha)c(F_k)$ *is a lower bound for the cost of a minimum weight* k*-Steiner forest, where* α *is the Steiner ratio.*

We now compute a lower bound on the cost of an optimum solution. A *feasible* k-Steiner forest is a k-Steiner forest where inequality (12) holds for each of the connected components $T_1, \ldots, T_k$.

Let t' be the smallest integer such that $(1/\alpha)c(F_{t'}) + d(\mathcal{D}) \leq t' \cdot u$. By inequality (12) and Lemma 13 this is a lower bound for the number of facilities:

Lemma 14. t' *is a lower bound for the number of facilities in any feasible solution.*

Let further t'' be an integer in $\{t', \ldots, n\}$ minimizing $\frac{1}{\alpha}c(F_{t''}) + t'' \cdot f$.

Theorem 15. $(1/\alpha)c(F_{t''}) + t'' \cdot f$ *is a lower bound for the cost of an optimal solution.*

Denote $L_r := (1/\alpha)c(F_{t''})$, and $L_f := t'' \cdot f$. Then $L_r + L_f$ is a lower bound on the cost of an optimum solution, and

[5] The Steiner ratio of a metric space (V, c) is the worst-case ratio, over all terminal sets T, of the lengths of a minimum spanning tree for (T, c) and a shortest Steiner tree for T in (V, c).

$$L_r + d(D) \leq L_f \frac{u}{f}. \tag{13}$$

Based on these lower bound considerations the algorithm proceeds as follows. First it computes a minimum spanning tree on (D, c). Second t'' and $F_{t''}$ are computed according to Theorem 15. If a component T of $F_{t''}$ violates Eq. (12) it must be decomposed into smaller components.

Thus overloaded components (with $c(E(T)) + d(D_i) > u$) are split. We do this in such a way that at least $u/2$ of the load will be removed whenever we introduce a new component. This can be done by considering a minimal overloaded subtree and applying the next-fit algorithm for bin packing. Splitting continues until no overloaded component exists. The number of new components is at most $\frac{2}{u}$ times the load of T.

Thus the total cost of the solution that we obtain is at most $c(F_{t''}) + t''f + (2/u)(c(F_{t''}) + d(D))f = \alpha L_r + L_f + (2f/u)(\alpha L_r + d(D))$. As $(f/u)L_r \leq L_f$ by Eq. (13), we get [80]:

Corollary 16. *The above algorithm computes a solution of cost at most* $(2\alpha + 1)$ *times the optimum in* $O(n \log n)$ *time, where* α *is the Steiner ratio.*

[80] also proved better approximation guarantees for other metric spaces, but for the rectilinear plane the above performance ratio of 4 is still the best known. However, Maßberg [81] proved stronger lower bounds. In practice, the ratio of the cost of the computed solution over a tight lower bound is typically less than 1.1. Furthermore, an exchange and merge heuristic is used to improve the clustering further as a post-optimization step. The above approximation algorithm also proves extremely fast in practice; we used it on instances with up to one million sinks.

In general, nonoverlapping time windows might restrict the clustering. In addition to being computed by BonnCycleOpt, time windows occur naturally in upper levels of the clock tree even with uniform windows at the leaves. An extension of the approximation algorithm for the problem with time windows is given in [81]. By exploiting the time intervals, which are single points only for the few most critical memory elements, and by using an algorithm with provable performance guarantee the clock tree power consumption could be reduced substantially.

5. Routing

Due to the enormous instance sizes, most routers comprise at least two major parts, global and detailed routing. Global routing defines an area for each net to which the search for actual wires in detailed routing is restricted. As global routing works on a much smaller graph, we can globally optimize the most important design objectives. Moreover, global routing has another important function: decide for each placement whether a feasible routing exists and if not, give a certificate of unfeasibility.

5.1. The global routing graph

The global router works on a three-dimensional grid graph which is obtained by partitioning the chip area into regions. For classical Manhattan routing this can be done by an axis-parallel grid. In any case, these regions are the vertices of the global routing graph.

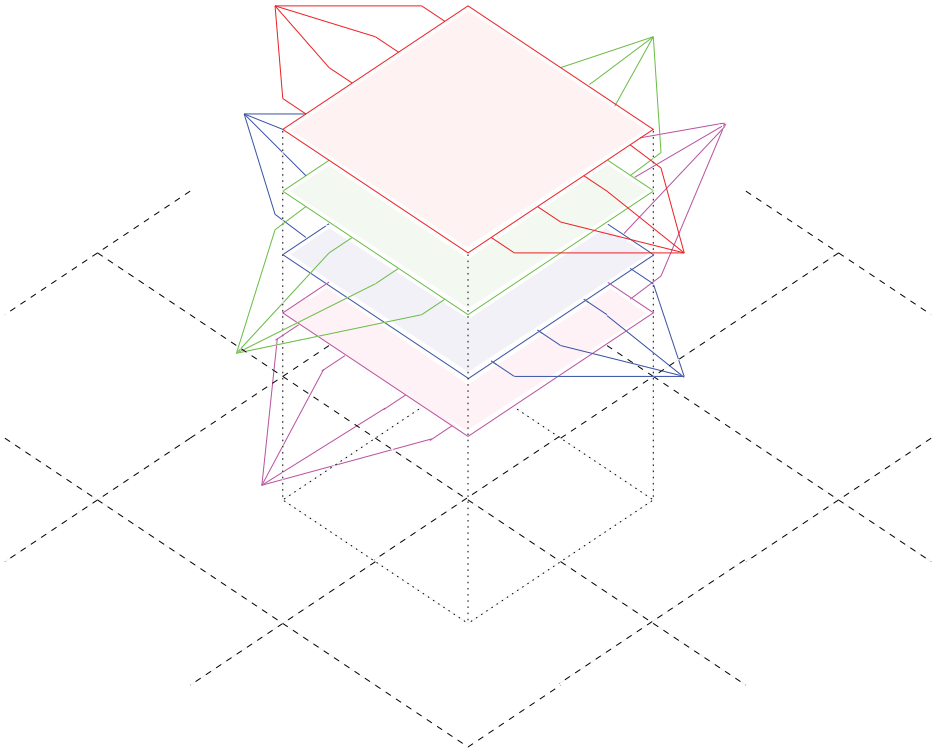

Figure 26. An instance of the vertex-disjoint paths problem for estimating global routing capacities. Dashed lines bound global routing regions. Here we show four wiring planes, each with a commodity (shown in different colors), in alternating preference directions.

Adjacent regions are joined by an edge, with a capacity value indicating how many wires of unit width can join the two regions. Each routing plane has a preference direction (horizontal or vertical), and we remove edges orthogonal to this direction in the global routing graph.

For each net we consider the regions that contain at least one of its pins. These vertices of the global routing graph have to be connected by a Steiner tree. If a pin consists of shapes in more than one region, we may assign it to one of them, say the one which is closest to the center of gravity of the whole net, or by solving a group Steiner tree problem.

The quality of the global routing depends heavily on the capacities of the global routing edges. A rough estimate has to consider blockages and certain resources for nets whose pins lie in one region only. These nets are not considered in global routing. However, they may use global routing capacity. Therefore we route very short nets, which lie in one region or in two adjacent regions, first in the routing flow, i.e., before global routing. They are then viewed as blockages in global routing. Yet these nets may be rerouted later in local routing if necessary.

Routing short nets before global routing makes better capacity estimates possible, but this also requires more sophisticated algorithms than are usually used for this task. We consider a vertex-disjoint paths problem for every set of four adjacent global routing

regions, illustrated in Figure 26. There is a commodity for each wiring plane, and we try to find as many paths for each commodity as possible. Each path may use the plane of its commodity in preference direction and adjacent planes in the orthogonal direction.

An upper bound on the total number of such paths can be obtained by considering each commodity independently and solving a maximum flow problem. However, this is too optimistic and too slow. Instead we compute a set of vertex-disjoint paths (i.e., a lower bound) by a very fast multicommodity flow heuristic [82]. It is essentially an augmenting path algorithm but exploits the special structure of a grid graph. For each augmenting path it requires only $O(k)$ constant-time bit pattern operations, where k is the number of edges orthogonal to the preferred wiring direction in the respective layer. In practice, k is less than three for most paths.

This very fast heuristic finds a number of vertex-disjoint paths in the region of 90% of the (weak) max-flow upper bound. For a complete chip with about one billion paths it needs 5 minutes of computing time whereas a complete max-flow computation with our implementation of the Goldberg-Tarjan algorithm would need more than a week.

Please note that this algorithm is used only for a better capacity estimation, i.e., for generating accurate input to the main global routing algorithm. However, this better capacity estimate yields much better global routing solutions and allows the detailed router to realize these solutions.

5.2. Classical Global Routing

In its simplest version, the global routing problem amounts to packing Steiner trees in a graph with edge capacities. A fractional relaxation of this problem can be efficiently solved by an extension of methods for the multicommodity flow problem. However, the approach does not consider today's main design objectives which are timing, signal integrity, power consumption, and manufacturing yield. Minimizing the total length of all Steiner trees is no longer important. Instead, minimizing a weighted sum of the capacitances of all Steiner trees, which is equivalent to minimizing power consumption, is an important objective. Delays on critical paths also depend on the capacitances of their nets. Wire capacitances can no longer be assumed to be proportional to the length, since coupling between neighboring wires plays an increasingly important role. Small detours of nets are often better than the densest possible packing. Spreading wires can also improve the yield.

Our global router is the first algorithm with a provable performance guarantee which takes timing, coupling, yield, and power consumption into account directly. Our algorithm extends earlier work on multicommodity flows, fractional global routing, the min-max resource sharing problem, and randomized rounding.

Let G be the global routing graph, with edge capacities $u\colon E(G) \to \mathbb{R}_+$ and lengths $l\colon E(G) \to \mathbb{R}_+$. Let $\mathcal{N}$ be the set of nets. For each $N \in \mathcal{N}$ we have a set $\mathcal{Y}_N$ of feasible Steiner trees. The set $\mathcal{Y}_N$ may contain all delay-optimal Steiner trees of N or, in many cases, it may simply contain all possible Steiner trees for N in G. Actually, we do not need to know the set $\mathcal{Y}_N$ explicitly. The only assumption which we make is that for each $N \in \mathcal{N}$ and any $\psi\colon E(G) \to \mathbb{R}_+$ we can find a Steiner tree $Y \in \mathcal{Y}_N$ with $\sum_{e\in E(Y)} \psi(e)$ (almost) minimum sufficiently fast. This assumption is justified since in practical instances almost all nets have less than, say, 10 pins. We can use a dynamic programming algorithm for

finding an optimum Steiner tree for small nets and a fast approximation algorithm for others. With $w(N, e) \in \mathbb{R}_+$ we denote the width of net N at edge e. A straightforward integer programming formulation of the classical global routing problem is:

$$\begin{aligned}
\min \sum_{N \in \mathcal{N}} \sum_{e \in E(G)} l(e) \sum_{Y \in \mathcal{Y}_N : e \in E(Y)} x_{N,Y} & \\
\text{s.t.} \sum_{N \in \mathcal{N}} \sum_{Y \in \mathcal{Y}_N : e \in E(Y)} w(N, e) x_{N,Y} \leq u(e) & \qquad (e \in E(G)) \\
\sum_{Y \in \mathcal{Y}_N} x_{N,Y} = 1 & \qquad (N \in \mathcal{N}) \\
x_{N,Y} \in \{0, 1\} & \qquad (N \in \mathcal{N}, Y \in \mathcal{Y}_N)
\end{aligned}$$

Here the decision variable $x_{N,Y}$ is 1 iff the Steiner tree Y is chosen for net N. The decision whether this integer programming problem has a feasible solution is already NP-complete. Thus, we relax the problem by allowing $x_{N,Y} \in [0, 1]$. Raghavan and Thompson [83,84] proposed solving the LP relaxation first, and then using randomized rounding to obtain an integral solution whose maximum capacity violation can be bounded. Although the LP relaxation has exponentially many variables, it can be solved in practice for moderate instance sizes since it has only $|E(G)| + |\mathcal{N}|$ many constraints. Therefore all but $|E(G)| + |\mathcal{N}|$ variables are zero in an optimum basic solution. However, for current complex chips with millions of nets and edges, all exact algorithms for solving the LP relaxation are far too slow.

Fortunately, there exist combinatorial fully polynomial approximation schemes, i.e., algorithms that compute a feasible solution of the LP relaxation which is within a factor of $1 + \epsilon$ of the optimum, and whose running time is bounded by a polynomial in $|V(G)|$ and $1/\epsilon$, for any accuracy $\epsilon > 0$. If each net has exactly two pins, $\mathcal{Y}_N$ contains all possible paths connecting N, and $w \equiv 1$, the global routing problem reduces to the edge-disjoint paths problem whose fractional relaxation is the multicommodity flow problem. Shahrokhi and Matula [85] developed the first fully polynomial approximation scheme for multicommodity flows. Carden, Li and Cheng [86] first applied this approach to global routing, while Albrecht [87] applied a modification of the approximation algorithm by Garg and Könemann [88]. However, these approaches did not consider the above-mentioned design objectives, like timing, power, and yield.

5.3. Advanced Global Routing

The power consumption of a chip induced by its wires is proportional to the weighted sum of all capacitances (see Figure 27), weighted by switching activities. The coupling capacitance depends on the distance between adjacent wires. In older technologies coupling capacitances were quite small and therefore could be ignored. In deep submicron technologies coupling matters a lot.

To account for this in global routing, we assign a certain space to each edge e and each net N using this edge. We write $s(e, N) \geq 0$ for the extra space that we assign in addition to the width $w(e, N)$. The contribution of edge e to the total capacitance of N is then a convex function of $s(e, N)$.

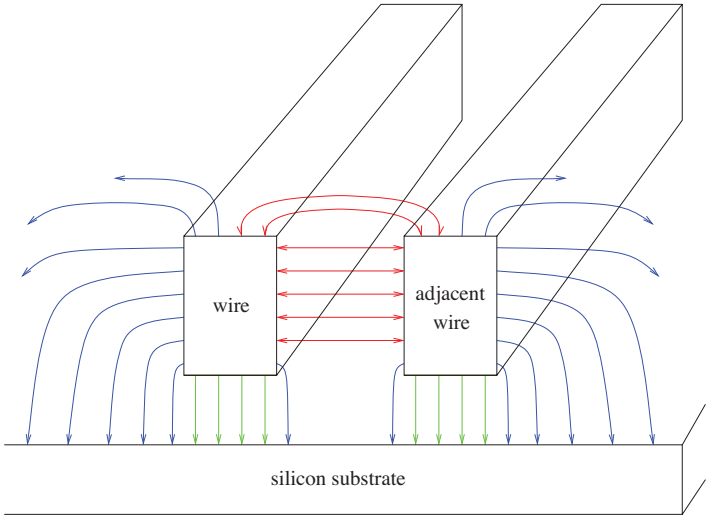

Figure 27. The capacitance of a net consists of area capacitance (green) between the undersurface of the wire and the substrate, proportional to length times width, fringing capacitance (blue) between side face of the wire and substrate, proportional to length, and coupling capacitance (red), proportional to length if adjacent wires exist. Note that the wire height is fixed within each plane, while the width varies.

Similarly to minimizing power consumption based on the above capacitance model, we can optimize yield by replacing capacitance by "critical area", i.e., the sensitivity of a layout to random defects [89]. Such random defects are caused by small particles that contaminate the chip during lithography. They can either disconnect a wire or connect two wires to a short.

Moreover, we can also consider timing restrictions. This can be done by excluding from the set $\mathcal{Y}_N$ all Steiner trees with large detours, or by imposing upper bounds on the weighted sums of capacitances of nets that belong to critical paths. For this purpose, we first do a static timing analysis under the assumption that every net has some expected capacitance. The set $\mathcal{Y}_N$ will contain only Steiner trees with capacitance below this expected value. We enumerate all paths which have negative slacks under this assumption. We compute the sensitivity of the nets of negative slack paths to capacitance changes, and use these values to translate the delay bound to appropriate bounds on the weighted sum of capacitances for each path. To compute reasonable expected capacitances we can apply weighted slack balancing (cf. Section 4.2) using delay sensitivity and congestion information.

Altogether we get a family $\mathcal{M}$ of subsets of $\mathcal{N}$ with $\mathcal{N} \in \mathcal{M}$, bounds $U\colon \mathcal{M} \to \mathbb{R}_+$ and convex functions $g(e, N, M)\colon \mathbb{R}_+ \to \mathbb{R}_+$ for $N \in M \in \mathcal{M}$ and $e \in E(G)$. We also treat the objective function as a constraint (we can apply binary search to compute the optimum value approximately, but in practice we can guess an excellent bound).

With these additional assumptions and this notation we can generalize the original integer programming formulation of the global routing problem to:

General Global Routing Problem

Instance:
- An undirected graph G with edge capacities $u\colon E(G) \to \mathbb{R}_+$,
- a set $\mathcal{N}$ of nets and a set $\mathcal{Y}_N$ of feasible Steiner trees for each net N,
- wire widths $w\colon E(G) \times \mathcal{N} \to \mathbb{R}_+$,
- A family $\mathcal{M}$ of subsets of $\mathcal{N}$ with bounds $U\colon \mathcal{M} \to \mathbb{R}_+$ and convex functions $g(e, N, M)\colon \mathbb{R}_+ \to \mathbb{R}_+$ for $N \in M \in \mathcal{M}$ and $e \in E(G)$.

Task: Find a Steiner tree $Y_N \in \mathcal{Y}_N$ and numbers $s(e, N) \geq 0$ for each $N \in \mathcal{N}$ and $e \in E(Y_N)$, such that
- $\sum_{N\in\mathcal{N}:e\in E(Y_N)}(w(e, N) + s(e, N)) \leq u(e)$ for each edge $e \in E(G)$,
- $\sum_{N\in M}\sum_{e\in E(Y_N)} g(e, N, M)(s(e, N)) \leq U(M)$ for each $M \in \mathcal{M}$.

This general formulation was proposed by [90]. Reformulating it, we look for a feasible solution (λ, x, s) to the following nonlinear optimization problem, where x is integral and $\lambda = 1$. As this is hard, we first solve the following fractional relaxation approximately and then apply randomized rounding to obtain an integral solution.

$$\min \lambda \quad \text{s.t.}$$

$$\sum_{Y\in\mathcal{Y}_N} x_{N,Y} = 1 \qquad (N \in \mathcal{N})$$

$$\sum_{N\in M}\Big(\sum_{Y\in\mathcal{Y}_N} x_{N,Y} \sum_{e\in E(Y)} g(e, N, M)(s(e, N))\Big) \leq \lambda\, U(M) \qquad (M \in \mathcal{M})$$

$$\sum_{N\in\mathcal{N}}\Big(\sum_{Y\in\mathcal{Y}_N:e\in E(Y)} x_{N,Y}(w(e, N) + s(e, N))\Big) \leq \lambda\, u(e) \qquad (e \in E(G))$$

$$s(e, N) \geq 0 \qquad (e \in E(G), N \in \mathcal{N})$$

$$x_{N,Y} \geq 0 \qquad (N \in \mathcal{N}, Y \in \mathcal{Y}_N) \qquad (14)$$

This can be transformed to an instance of the Min-Max Resource Sharing Problem, defined as follows. Given finite sets $\mathcal{R}$ of resources and $\mathcal{N}$ of customers, an implicitly given convex set $\mathcal{B}_N$, called block, and a convex resource consumption function $g_N\colon \mathcal{B}_N \to \mathbb{R}_+^{\mathcal{R}}$ for every $N \in \mathcal{N}$, the task is to find $b_N \in \mathcal{B}_N$ $(N \in \mathcal{N})$ approximately attaining $\lambda^* := \inf\{\max_{r\in\mathcal{R}} \sum_{N\in\mathcal{N}}(g_N(b_N))_r \mid b_N \in \mathcal{B}_N(N \in \mathcal{N})\}$. In the general problem formulation we have access to the sets $\mathcal{B}_N$ only via oracle functions $f_N\colon \mathbb{R}_+^{\mathcal{R}} \to \mathcal{B}_N$, called block solvers, which for $N \in \mathcal{N}$ and $y \in \mathbb{R}_+^{\mathcal{R}}$ return an element $b_N \in \mathcal{B}_N$ with $y^\top g_N(b_N) \leq \sigma \inf_{b\in\mathcal{B}_N} y^\top g_N(b)$. Here $\sigma \geq 1$ is a given constant.

In our application the customers are the nets, and the resources are the elements of $E(G) \cup \mathcal{M}$. We can define

$$\mathcal{B}_N := \operatorname{conv}(\{(\chi(Y), s) \mid Y \in \mathcal{Y}_N,\ s \in \mathbb{R}_+^{E(G)},\ s_e = 0 \text{ for } e \notin E(Y)\}),$$

where $\chi(Y) \in \{0, 1\}^{E(G)}$ denote the edge-incidence vector of a Steiner tree Y. The functions g_N are then given by

/* Initialization */
$y_r \leftarrow 1$ for $r \in \mathcal{R}$;
$x_{N,b} \leftarrow 0$ for $N \in \mathcal{N}$, $b \in \mathcal{B}_N$;
$X_N \leftarrow 0$ for $N \in \mathcal{N}$;

/* Main Loop */
for $p := 1$ **to** t **do**
  for $N \in \mathcal{N}$ **do**
    while $X_N < p$ **do**

/* Call block solver */
      $b \leftarrow f_N(y)$;
      $a \leftarrow g_N(b)$;

/* Update variables */
      $\xi \leftarrow \min\{p - X_N, 1/\max\{a_r \mid r \in \mathcal{R}\}\}$;
      $x_{N,b} \leftarrow x_{N,b} + \xi$ and $X_N \leftarrow X_N + \xi$;

/* Update prices */
      for $r \in \mathcal{R}$ **do**
        $y_r \leftarrow y_r \mathrm{e}^{\epsilon\xi a_r}$;
      end for
    end while
  end for
end for
/* Take Average */
$x_{N,b} \leftarrow (1/t)x_{N,b}$ for $N \in \mathcal{N}$ and $b \in \mathcal{B}_N$.

Algorithm 3. Resource Sharing Algorithm

$$(g_N(x,s))_e := (x_e w(e,N) + s_e)/u(e) \qquad (e \in E(G))$$
$$(g_N(x,s))_M := \Big(\sum_{e \in E(G): x_e > 0} x_e g(e,N,M)(s_e/x_e)\Big)\Big/U(M) \qquad (M \in \mathcal{M}) \tag{15}$$

for each $N \in \mathcal{N}$ and $(x,s) \in \mathcal{B}_N$.

We showed in [91] that the block solvers can be implemented by an approximation algorithm for the Steiner tree problem in weighted graphs. Then they always return an extreme point $b \in \mathcal{B}_N$, corresponding to a single Steiner tree, which we denote by Y_b.

Algorithm 3 solves the Min-Max Resource Sharing Problem, and hence Eq. (14), approximately. It is a primal-dual algorithm which takes two parameters $0 < \epsilon < 1$ and $t \in \mathbb{N}$. They control the approximation guarantee and running time.

The algorithm proceeds in t iterations where it calls the block solver for every net based on current resource prices. After each individual choice the prices are updated.

Let $\mathrm{opt}_N(y) := \inf_{b \in \mathcal{B}_N} y^\top g_N(b)$. The analysis of the algorithm relies on weak duality: any set of prices yields a lower bound on the optimum:

Lemma 17. *Let $y \in \mathbb{R}_+^{\mathcal{R}}$ be some cost vector with $\mathbb{1}^\top y \neq 0$. Then*

$$\frac{\sum_{N\in\mathcal{N}} opt_N(y)}{\mathbb{1}^\top y} \le \lambda^*.$$

Proof. Let $\delta > 0$ and $(b_N \in \mathcal{B}_N)_{N\in\mathcal{N}}$ a solution with $\max_{r\in\mathcal{R}} \sum_{N\in\mathcal{N}} (g_N(b_N))_r < (1+\delta)\lambda^*$. Then

$$\frac{\sum_{N\in\mathcal{N}} \operatorname{opt}_N(y)}{\mathbb{1}^\top y} \le \frac{\sum_{N\in\mathcal{N}} y^\top g_N(b_N)}{\mathbb{1}^\top y} < \frac{(1+\delta)\lambda^* \mathbb{1}^\top y}{\mathbb{1}^\top y} = (1+\delta)\lambda^*. \qquad \square$$

The Resource Sharing Algorithm yields $x_{N,b} \ge 0$ for all $b \in \mathcal{B}_N$ with $\sum_{b\in\mathcal{B}_N} x_{N,b} = 1$. Hence we have a convex combination of vectors in $\mathcal{B}_N$ for each $N \in \mathcal{N}$. To estimate the quality of the solution we prove two lemmas. Let $y^{(p,i)}$ denote y at the end of the ith innermost iteration and k_p the total number of innermost iterations within the pth outer iteration. We call the outer iterations phases. Let $y^{(p)}$ denote y at the end of phase p. Similar for the other variables in the algorithm.

Lemma 18. *Let (x, y) be the output of the* Resource Sharing Algorithm. *Then*

$$\max_{r\in\mathcal{R}} \sum_{N\in\mathcal{N}} \left(g_N\Big(\sum_{b\in\mathcal{B}_N} x_{N,b} b \Big) \right)_r \le \max_{r\in\mathcal{R}} \sum_{N\in\mathcal{N}} \sum_{b\in\mathcal{B}_N} x_{N,b} (g_N(b))_r \le \frac{1}{\epsilon t} \ln(\mathbb{1}^\top y).$$

Proof. The first inequality follows from the convexity of the functions g_N. For the second inequality, note that for $r \in \mathcal{R}$:

$$\sum_{N\in\mathcal{N}} \sum_{b\in\mathcal{B}_N} x_{N,b} (g_N(b))_r = \frac{1}{t} \sum_{p=1}^{t} \sum_{i=1}^{k_p} \xi^{(p,i)} (a^{(p,i)})_r = \frac{1}{\epsilon t} \ln y_r^{(t)} \le \frac{1}{\epsilon t} \ln\left(\mathbb{1}^\top y^{(t)}\right). \qquad \square$$

Lemma 19. *Let $\sigma \ge 1$ such that $y^\top g_N(f_N(y)) \le \sigma \operatorname{opt}_N(y)$ for all y. Let $\epsilon > 0$ and $\epsilon' := (e^\epsilon - 1)\sigma$. If $\epsilon' \lambda^* < 1$, then*

$$\mathbb{1}^\top y^{(t)} \le |\mathcal{R}| e^{t\epsilon'\lambda^*/(1-\epsilon'\lambda^*)}.$$

Proof. We will consider the term $\mathbb{1}^\top y^{(p)}$ for all phases p. Initially we have $\mathbb{1}^\top y^{(0)} = |\mathcal{R}|$. We can estimate the increase of the resource prices as follows:

$$\begin{aligned} \sum_{r\in\mathcal{R}} y_r^{(p,i)} &= \sum_{r\in\mathcal{R}} y_r^{(p,i-1)} e^{\epsilon \xi^{(p,i)} (a^{(p,i)})_r} \\ &\le \sum_{r\in\mathcal{R}} y_r^{(p,i-1)} + (e^\epsilon - 1) \sum_{r\in\mathcal{R}} y_r^{(p,i-1)} \xi^{(p,i)} (a^{(p,i)})_r, \end{aligned} \tag{16}$$

because $\xi^{(p,i)} (a^{(p,i)})_r \le 1$ for $r \in \mathcal{R}$, and $e^x \le 1 + \frac{e^\epsilon - 1}{\epsilon} x$ for $0 \le x \le \epsilon$.

Moreover,

$$\sum_{r\in\mathcal{R}} y_r^{(p,i-1)} (a^{(p,i)})_r \le \sigma \operatorname{opt}_{N^{(p,i)}}(y^{(p,i-1)}). \tag{17}$$

Using Eqs. (16), (17), the monotonicity of y, the fact $\sum_{i:N^{(p,i)}=N} \xi^{(p,i)} = 1$ for all N, and Lemma 17 we get

$$\begin{aligned}\mathbb{1}^\top y^{(p)} &\le \mathbb{1}^\top y^{(p-1)} + (\mathrm{e}^\epsilon - 1)\sigma \sum_{i=1}^{k_p} \xi^{(p,i)} \operatorname{opt}_{N^{(p,i)}}(y^{(p,i-1)}) \\ &\le \mathbb{1}^\top y^{(p-1)} + \epsilon' \sum_{N\in\mathcal{N}} \operatorname{opt}_N(y^{(p)}) \\ &\le \mathbb{1}^\top y^{(p-1)} + \epsilon'\lambda^* \mathbb{1}^\top y^{(p)}\end{aligned}$$

and hence

$$\mathbb{1}^\top y^{(p)} \le \frac{\mathbb{1}^\top y^{(p-1)}}{1-\epsilon'\lambda^*}.$$

Combining this with $\mathbb{1}^\top y^{(0)} = |\mathcal{R}|$ and $1 + x \le \mathrm{e}^x$ for $x \ge 0$ we get, if $\epsilon'\lambda^* < 1$:

$$\mathbb{1}^\top y^{(t)} \le \frac{|\mathcal{R}|}{(1-\epsilon'\lambda^*)^t} = |\mathcal{R}|\left(1 + \frac{\epsilon'\lambda^*}{1-\epsilon'\lambda^*}\right)^t \le |\mathcal{R}|\mathrm{e}^{t\epsilon'\lambda^*/(1-\epsilon'\lambda^*)}. \qquad \square$$

Combining Lemmas 18 and 19 we get:

Theorem 20. *Let λ^* be the optimum LP value, $\lambda^* \ge \frac{1}{2}$, $\sigma\lambda^* \le \frac{5}{2}$, and $0 < \epsilon \le \frac{1}{3}$, and $t\lambda^* > \log|\mathcal{R}|$. Then the algorithm computes a feasible solution whose value differs from the optimum by at most a factor*

$$\frac{2\ln|\mathcal{R}|}{\epsilon t} + \sigma\frac{(\mathrm{e}^\epsilon - 1)}{\epsilon(1 - \frac{5}{2}(\mathrm{e}^\epsilon - 1))}.$$

By choosing ϵ and t appropriately, we get a $(\sigma+\omega)$-optimal solution in $O(\omega^{-2}\ln|\mathcal{R}|)$ iterations, for any $\omega > 0$.

Although these assumptions on λ^* and σ are realistic in practice, one can also get rid of them and obtain a $(\sigma + \omega)$-optimal solution with $O(\log|\mathcal{R}|((|\mathcal{N}| + |\mathcal{R}|)\log\log|\mathcal{R}| + (|\mathcal{N}| + |\mathcal{R}|)\omega^{-2}))$ oracle calls in general [91].

Moreover, we proposed several speedup techniques and an extremely efficient parallel implementation [92,91]. This makes the approach applicable even on the largest VLSI instances. One can obtain a solution which is provably within a few percent of the optimum for an instance with millions of nets and constraints in a few hours of computing time.

The algorithm always gives a dual solution and can therefore, by Lemma 17, give a certificate of unfeasibility if a given placement is not routable. We also showed how to make randomized rounding work [90,91].

This approach is quite general. It allows us to add further constraints. Here we have modeled timing, yield, and power consumption, but we may think of other constraints if further technological or design restrictions come up.

Figure 28 shows a typical result of global routing. In the dense (red and orange) areas the main challenge is to find a feasible solution, while in other areas there is room for optimizing objectives like power or yield. Experimental results show a significant improvement over previous approaches which optimized net length and number of vias, both in terms of power consumption and expected manufacturing yield [89,92].

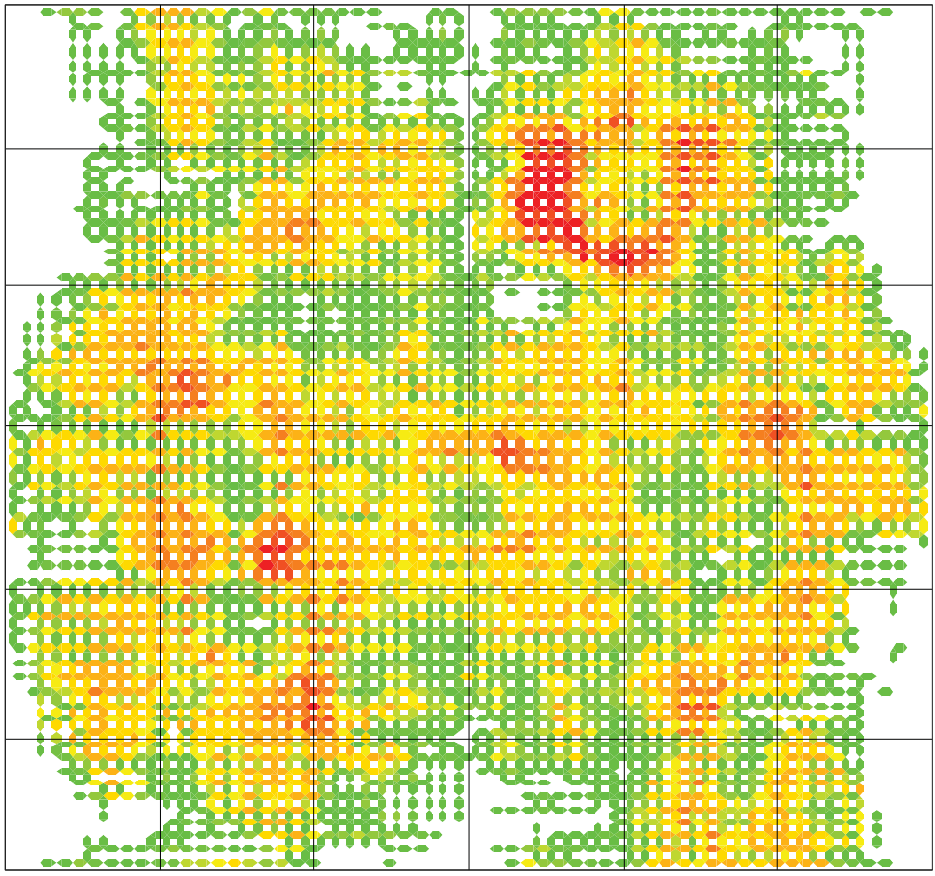

Figure 28. A typical global routing congestion map. Each edge corresponds to approximately 10x10 global routing edges (and to approximately 1 000 detailed routing channels). Red, orange, yellow, green, and white edges correspond to an average load of approximately 90 – 100%, 70 – 90%, 60 – 70%, 40 – 60%, and less than 40%.

We conclude this section by pointing out that this problem is not restricted to VLSI design. It is in fact equivalent to routing traffic flow, with hard capacity bounds on edges (streets), without capacity bounds on vertices, with flows statically repeated over time, with bounds on weighted sums of travel times. Algorithm 3 can then be interpreted as selfish routing with taxes that depend exponentially on congestion.

5.4. Detailed Routing

The task of detailed routing is to determine the exact layout of the metal realizations of the nets. Efficient data structures are used to store all metal shapes and allow fast queries. Grid-based routers define routing tracks (and minimum distances) and work with a detailed routing graph G which is an incomplete three-dimensional grid graph, i.e., $V(G) \subseteq \{x_{\min}, \ldots, x_{\max}\} \times \{y_{\min}, \ldots, y_{\max}\} \times \{1, \ldots, z_{\max}\}$ and $((x, y, z), (x', y', z')) \in E(G)$ only if $|x - x'| + |y - y'| + |z - z'| = 1$.

The z-coordinate models the different routing layers of the chip and $z_{\max}$ is typically around $10-12$. We can assume without loss of generality that the x- and y-coordinates correspond to the routing tracks; typically the number of routing tracks in each plane, and hence $x_{\max} - x_{\min}$ and $y_{\max} - y_{\min}$, is in the order of magnitude of 10^5, resulting in a graph with more than 10^{11} vertices. The graph is incomplete because some parts are reserved for internal circuit structures or power supply, and some nets may have been routed earlier.

To find millions of vertex-disjoint Steiner trees in such a huge graph is very challenging. Thus we decompose this task, route the nets and even the two-point connections making up the Steiner tree for each net individually. Then the elementary algorithmic task is to determine shortest paths within the detailed routing graph (or within a part of it, as specified by global routing).

Whereas the computation of shortest paths is probably the most basic and well-studied algorithmic problem of discrete mathematics [3], the size of G and the number of shortest paths that have to be found concurrently makes the use of textbook versions of shortest path algorithms impossible. The basic algorithm for finding a shortest path connecting two given vertices in a digraph with nonnegative arc weights is Dijkstra's algorithm. Its theoretically fastest implementation, with Fibonacci heaps, runs in $O(m + n \log n)$ time, where n and m denote the number of vertices and edges, respectively [93]. For our purposes this is much too slow. Various strategies are applied to speed up Dijkstra's algorithm.

Since we are not just looking for one path but have to embed millions of disjoint trees, the information provided by global routing is most important. For each two-point connection global routing determines a corridor essentially consisting of the global routing tiles to which this net was assigned in global routing. If we find a shortest path for the two-point connection within this corridor, the capacity estimates used during global routing approximately guarantee that all desired paths can be realized disjointly. Furthermore, we get a dramatic speedup by restricting the path search to this corridor, which usually represents a very small fraction of the entire routing graph.

The second important factor speeding up our shortest path algorithm is the way in which distance information is stored. Whereas Dijkstra's algorithm labels individual vertices, we consider intervals of consecutive vertices that are similar with respect to their usability and their distance properties. Since the layers are assigned preferred routing directions, the intervals are chosen parallel to these. By the similarity of the vertices in one interval we mean that their distance properties can be encoded more efficiently than by storing numbers for each individual vertex. If e.g. the distance increases by one unit from vertex to vertex we just need to store the distance information for one vertex and the increment direction. Hetzel's version of Dijkstra's algorithm [94], generalized by [95] and [96], labels intervals instead of vertices, and its time complexity therefore depends on the number of intervals, which is typically about 50 times smaller than the number of vertices. A sophisticated data structure for storing the intervals and answering queries very fast is the basis of this algorithm and also of its efficient shared-memory parallelization.

The last factor speeding up the path search is the use of a future cost estimate, which is a lower bound on the distance of vertices to a given target set of vertices. This is a well-known technique. Suppose we are looking for a path from s to t in G with respect to edge weights $c\colon E(G) \to \mathbb{R}_+$, which reflect higher costs for vias and wires orthogonal

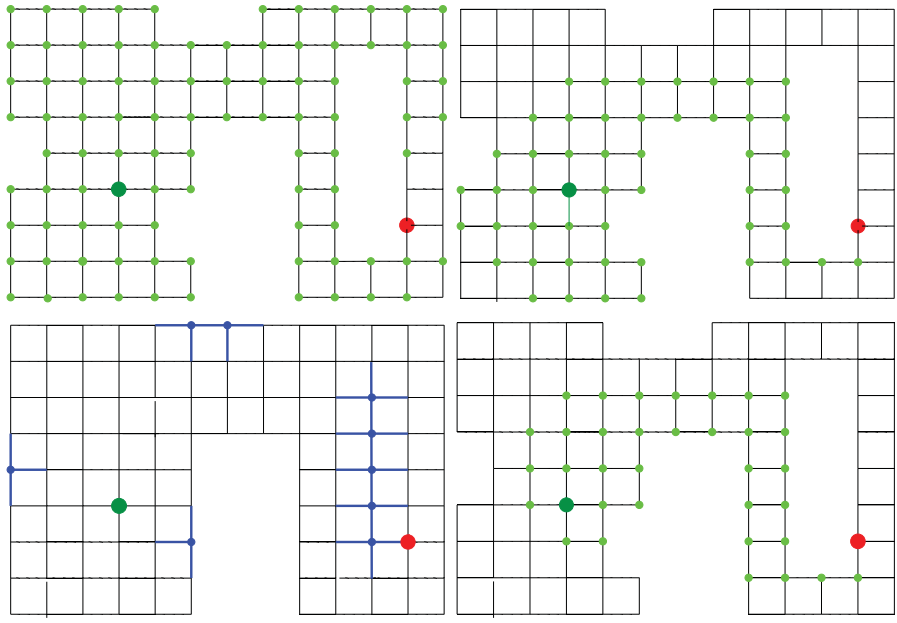

Figure 29. An instance of the shortest paths problem with unit edge weights. We look for a shortest path from the big dark green vertex in the left to the big red vertex in the right part. The figures show Dijkstra's algorithm without future costs (top left), with l_1-distances as future costs (top right), and with improved future costs (bottom right). The improved future costs are based on distances in a grid graph arising by filling small holes (blue vertices and edges in the bottom left). Points labeled by Dijkstra's algorithm are marked light green. The running time is roughly proportional to the number of labeled points (93 versus 51 versus 36).

to the preferred direction and can also be used to find optimal rip-up sets. Let $l(x)$ be a lower bound on the distance from x to t (the future cost) for any vertex $x \in V$. Then we may apply Dijkstra's algorithm to the costs $c'(x,y) := c(\{x,y\}) - l(x) + l(y)$. For any s-t-path P we have $c'(P) = c(P) - l(s) + l(t)$, and hence shortest paths with respect to c' are also shortest paths with respect to c. If l is a good lower bound, i.e., close to the exact distance, and satisfies the natural condition $l(x) \leq c(\{x,y\}) + l(y)$ for all $\{x,y\} \in E(G)$, then this results in a significant speedup.

If the future cost estimate is exact, our procedure will only label intervals that contain vertices lying on shortest paths.

Clearly, improving the accuracy of the future cost estimate improves the running time of the path search and there is a tradeoff between the time needed to improve the future cost and the time saved during path search. Hetzel [94] used l_1-distances as future cost estimates. In [95] we showed how to obtain and use much better estimates efficiently by computing distances in a condensed graph whose vertices correspond to rectangles. This leads to significant reductions of the running time as illustrated by Figure 29.

6. Conclusion

We have demonstrated that mathematics can yield better solutions for leading-edge chips. Several complete microprocessor series (cf., e.g., [68,97]) and many leading-edge ASICs

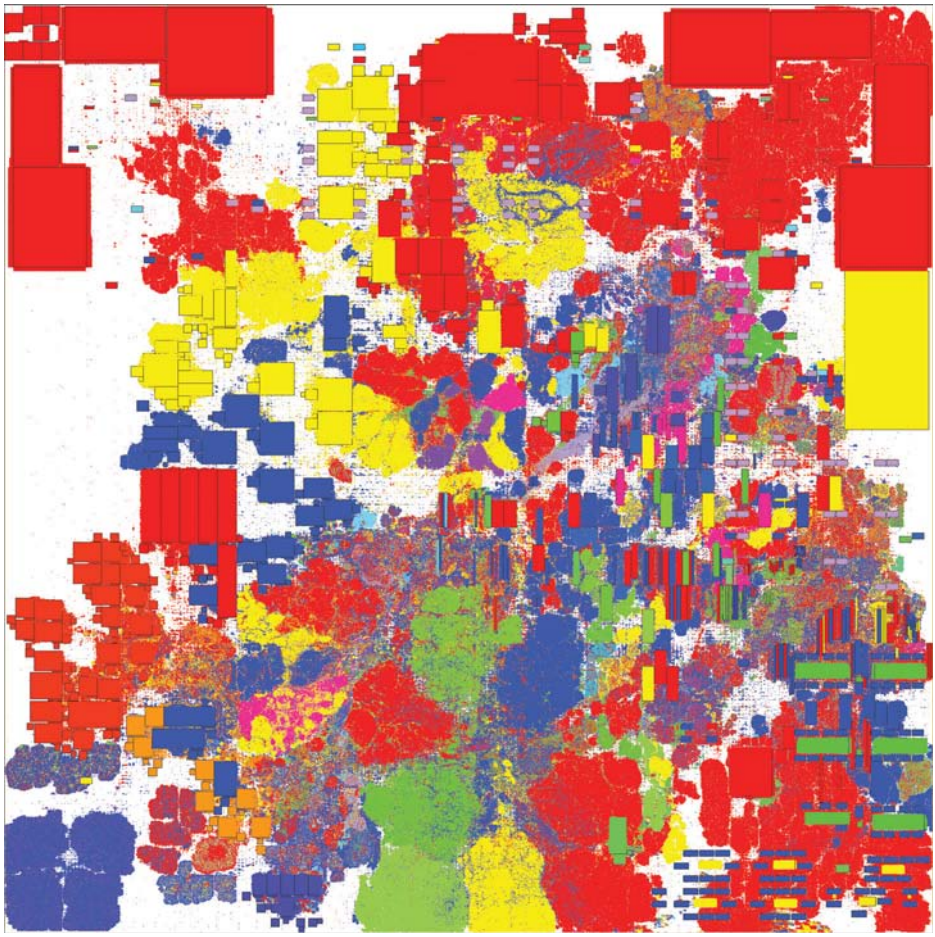

Figure 30. A system on a chip designed in 2006 with BonnTools. This 90nm design for a forthcoming IBM server has more than 5 million circuits and runs with frequencies up to 1.5 GHz. Colors reflect the structure of the underlying logic blocks.

(cf., e.g., [98,71]) have been designed with BonnTools. Many additional ones are in the design centers at the time of writing. Figures 30 and 31 show examples of chips that have been and are currently designed by IBM with BonnTools.

Chip design is inspiring a great deal of interesting work in mathematics. Indeed, most classical problems in combinatorial optimization, and many new ones, have been applied to chip design. Some algorithms originally developed for VLSI design automation are applied also in other contexts.

However, there remains a lot of work to do. Exponentially increasing instance sizes continue to pose challenges. Even some classical problems (e.g., logic synthesis) have no satisfactory solution yet, and future technologies continuously bring new problems. Yet we strongly believe that mathematics will continue to play a vital role in facing these challenges.

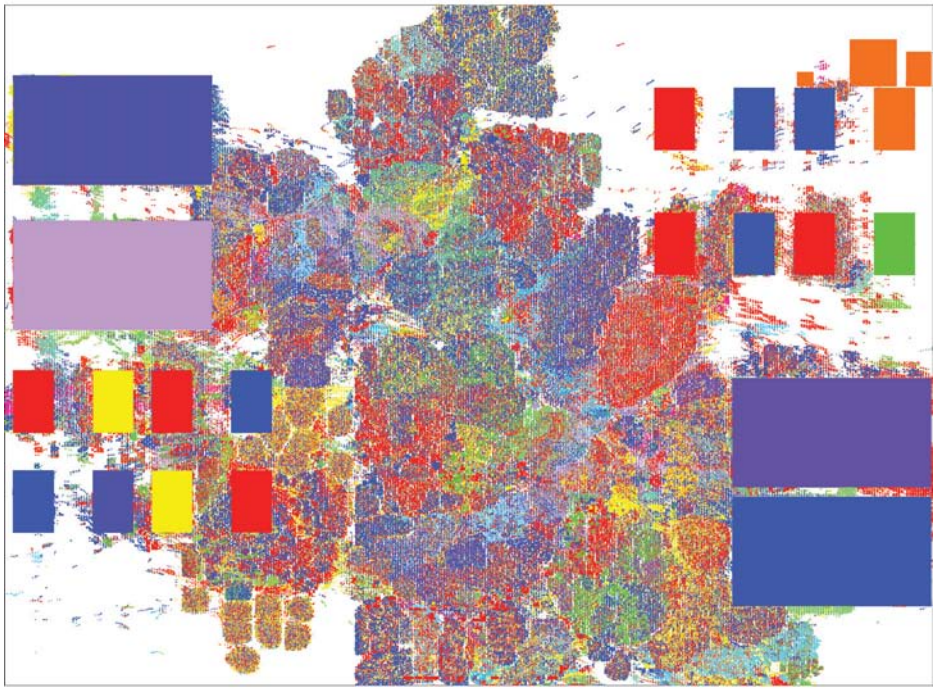

Figure 31. A vector processing unit currently designed with BonnTools. This 22 nm prototype runs with a frequency of 4.7 GHz.

Acknowledgments

We thank all current and former members of our team in Bonn. Moreover, we thank our cooperation partners, in particular at IBM. Last but not least we thank Vašek Chvátal for inviting us to give lectures on the topics discussed in this paper at the 45th Session of the Séminaire de mathématiques supérieures at Montréal.

References

[1] Korte, B., Rautenbach, D., and Vygen, J. (2007) BonnTools: Mathematical innovation for layout and timing closure of systems on a chip. *Proceedings of the IEEE*, **95**, 555 –572.

[2] Korte, B., Lovász, L., Prömel, H. J., and Schrijver, A. (eds.) (1990) *Paths, Flows, and VLSI-Layout*, vol. 9 of *Algorithms and Combinatorics*. Springer, Berlin.

[3] Korte, B. and Vygen, J. (2008) *Combinatorial Optimization*, vol. 21 of *Algorithms and Combinatorics*. Springer, Berlin, 4th edn.

[4] Cook, W. J., Lovász, L., and Vygen, J. (eds.) (2009) *Research Trends in Combinatorial Optimization*. Springer, Berlin.

[5] Korte, B. and Vygen, J. (2008) Combinatorial problems in chip design. Grötschel, M. and Katona, G. O. H. (eds.), *Building Bridges*, vol. 19 of *Bolyai Society Mathematical Studies*, pp. 333–368, Springer, Berlin.

[6] Moore, G. E. (1965) Cramming more components onto integrated circuits. *Electronics*, **38**, 114–117.

[7] Garey, M. R. and Johnson, D. S. (1977) The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, **32**, 826–834.

[8] Brenner, U. and Vygen, J. (2001) Worst-case ratios of networks in the rectilinear plane. *Networks*, **38**, 126–139.
[9] Hwang, F. K. (1976) On Steiner minimal trees with rectilinear distance. *SIAM Journal on Applied Mathematics*, **30**, 104–114.
[10] Rautenbach, D. (2004) Rectilinear spanning trees versus bounding boxes. *Electronic Journal of Combinatorics*, **11**, Note 12.
[11] Queyranne, M. (1986) Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems. *Operations Research Letters*, **4**, 231–234.
[12] Vygen, J. (1997) Algorithm for large-scale flat placement. *Proceedings of the 34th Design Automation Conference*, (Anaheim, CA, 1997), pp. 746–751, ACM, New York, NY.
[13] Vygen, J. (2005) Geometric quadrisection in linear time, with application to VLSI placement. *Discrete Optimization*, **2**, 362–390.
[14] Brenner, U. and Struzyna, M. (2005) Faster and better global placement by a new transportation algorithm. Joyner, J., William H., Martin, G., and Kahng, A. B. (eds.), *Proceedings of the 42nd Design Automation Conference*, (San Diego, CA, 2005), pp. 591–596, ACM, New York, NY.
[15] Brenner, U., Struzyna, M., and Vygen, J. (2008) BonnPlace: Placement of leading-edge chips by advanced combinatorial algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **27**, 1607–1620.
[16] Vygen, J. (2007) New theoretical results on quadratic placement. *Integration, the VLSI Journal*, **40**, 305–314.
[17] Kleinhans, J. M., Sigl, G., Johannes, F. M., and Antreich, K. J. (1991) GORDIAN: VLSI placement by quadratic programming and slicing optimization placement by quadratic programming and slicing optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **10**, 356–365.
[18] Brenner, U. and Vygen, J. (2008) Analytical methods in VLSI placement. Alpert, C. J., Mehta, D. P., and Sapatnekar, S. S. (eds.), *Handbook of Algorithms for Physical Design Automation*, chap. 17, pp. 327–346, CRC Press, Boca Raton, FL.
[19] Orlin, J. B. (1993) A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, **41**, 338–350.
[20] Brenner, U. (2008) A faster polynomial algorithm for the unbalanced Hitchcock transportation problem. *Operations Research Letters*, **36**, 408–413.
[21] Jerrum, M. (1985) Complementary partial orders and rectangle packing. Tech. rep., Department of Computer Science, University of Edinburgh.
[22] Murata, H., Fujiyoshi, K., Nakatake, S., and Kajitani, Y. (1996) VLSI module placement based on rectangle-packing by the sequence-pair. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **15**, 1518–1524.
[23] Hougardy, S. (2010). personal communication.
[24] Brenner, U. (2005) *Theory and Practice of VLSI Placement*. Dissertation, Universität Bonn.
[25] Schneider, J. (2009) *Macro-Platzierung im VLSI-Design*. Diploma thesis, Universität Bonn.
[26] Struzyna, M. (2010) *Flow-Based Partitioning and Fast Global Placement in Chip Design*. Dissertation, Universität Bonn.
[27] Brenner, U. and Rohe, A. (2003) An effective congestion-driven placement framework. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **22**, 387–394.
[28] Vygen, J. (1998) Algorithms for detailed placement of standard cells. *1998 Design, Automation and Test in Europe*, (Paris, 1998), pp. 321–324, IEEE Computer Society, Los Alamitos, CA.
[29] Brenner, U. and Vygen, J. (2004) Legalizing a placement with minimum total movement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **23**, 1597–1613.
[30] Kahng, A. B., Tucker, P., and Zelikovsky, A. (1999) Optimization of linear placements for wirelength minimization with free sites. *Proceedings of the 1999 Asia and South Pacific Design Automation Conference*, (Wanchai, 1999), pp. 241–244, IEEE Press, Piscataway, NJ.
[31] Brenner, U. and Vygen, J. (2000) Faster optimal single-row placement with fixed ordering. *2000 Design, Automation and Test in Europe*, (Paris, 2000), pp. 117–121, IEEE Computer Society, Los Alamitos, CA.
[32] Suhl, U. (2010) *Row-Placement in VLSI Design: The Clumping Algorithm and a Generalization*. Diploma thesis, Universität Bonn.
[33] Brenner, U., Pauli, A., and Vygen, J. (2004) Almost optimum placement legalization by minimum cost flow and dynamic programming. *Proceedings of the 2004 International Symposium on Physical Design*,

(Phoenix, AZ, 2004), pp. 2–9, ACM, New York, NY.
[34] Sapatnekar, S. S. (2004) *Timing*. Kluwer, Boston, MA.
[35] Vygen, J. (2006) Slack in static timing analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **25**, 1876–1885.
[36] Bartoschek, C., Held, S., Rautenbach, D., and Vygen, J. (2006) Efficient generation of short and fast repeater tree topologies. *Proceedings of the 2006 International Symposium on Physical Design*, (San Jose, CA, 2006), pp. 120–127, ACM, New York, NY.
[37] Bartoschek, C., Held, S., Rautenbach, D., and Vygen, J. (2009) Fast buffering for optimizing worst slack and resource consumption in repeater trees. *Proceedings of the 2009 International Symposium on Physical Design*, (San Diego, CA, 2009), pp. 43–50, ACM, New York, NY.
[38] Bartoschek, C., Held, S., Maßberg, J., Rautenbach, D., and Vygen, J. (2010) The repeater tree construction problem. *Information Processing Letters*, **110**, 1079–1083.
[39] Kraft, L. G. (1949) *A device for quantizing grouping and coding amplitude modulated pulses*. Master's thesis, MIT.
[40] Huffman, D. A. (1952) A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, **40**, 1098–1101.
[41] Rautenbach, D., Szegedy, C., and Werber, J. (2006) Delay optimization of linear depth Boolean circuits with prescribed input arrival times. *Journal of Discrete Algorithms*, **4**, 526–537.
[42] Rautenbach, D., Szegedy, C., and Werber, J. (2003) Asymptotically optimal Boolean circuits for functions of the form $g_{n-1}(g_{n-2}(\cdots g_3(g_2(g_1(x_1, x_2), x_3), x_4)\ldots, x_{n-1}), x_n)$. Tech. Rep. 03931, Forschungsinstitut für Diskrete Mathematik, Universität Bonn.
[43] Rautenbach, D., Szegedy, C., and Werber, J. (2008) On the cost of optimal alphabetic code trees with unequal letter costs. *European Journal of Combinatorics*, **29**, 386–394.
[44] Werber, J., Rautenbach, D., and Szegedy, C. (2007) Timing optimization by restructuring long combinatorial paths. *Proceedings of the 2007 IEEE/ACM International Conference on Computer-Aided Design*, (San Jose, CA, 2007), pp. 536–543, IEEE Press, Piscataway, NJ.
[45] Liu, J., Zhou, S., Zhu, H., and Cheng, C.-K. (2003) An algorithmic approach for generic parallel adders. *2003 International Conference on Computer Aided Design*, (San Jose, CA, 2003), pp. 734–740, IEEE Computer Society, Los Alamitos, CA.
[46] Rautenbach, D., Szegedy, C., and Werber, J. (2007) The delay of circuits whose inputs have specified arrival times. *Discrete Applied Mathematics*, **155**, 1233–1243.
[47] Brent, R. (1970) On the addition of binary numbers. *IEEE Transactions on Computers*, **19**, 758–759.
[48] Khrapchenko, V. M. (1970) Asymptotic estimation of addition time of a parallel adder. *Systems Theory Research*, **19**, 105–122.
[49] Yeh, W.-C. and Jen, C.-W. (2003) Generalized earliest-first fast addition algorithm. *IEEE Transactions on Computers*, **52**, 1233–1242.
[50] Boyd, S. P., Kim, S.-J., Patil, D. D., and Horowitz, M. A. (2005) Digital circuit optimization via geometric programming. *Operations Research*, **53**, 899–932.
[51] Chen, C.-P., Chu, C. C. N., and Wong, D. F. (1998) Fast and exact simultaneous gate and wire sizing by Lagrangian relaxation. *Proceedings of the 1998 IEEE/ACM International Conference on Computer-Aided Design*, (San Jose, CA, 1998), pp. 617–624, ACM, New York, NY.
[52] Fishburn, J. P. and Dunlop, A. E. (1985) TILOS: A posynomial programming approach to transistor sizing. *Proceedings of the 1985 International Conference on Computer-Aided Design*, (Santa Clara, CA, 1985), pp. 326–328.
[53] Elmore, W. C. (1948) The transient response of damped linear networks with particular regard to wideband amplifiers. *Journal of Applied Physics*, **19**, 55–63.
[54] Chu, C. C. N. and Wong, D. F. (1998) Greedy wire-sizing is linear time. *Proceedings of the 1998 International Symposium on Physical Design*, (Monterey, CA, 1998), pp. 39–44, ACM, New York, NY.
[55] Cong, J. and He, L. (1999) Theory and algorithm of local-refinement-based optimization with application to device and interconnect sizing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **18**, 406–420.
[56] Langkau, K. (2000) *Gate-Sizing im VLSI-Design*. Diploma thesis, Universität Bonn.
[57] Rautenbach, D. and Szegedy, C. (2008) A class of problems for which cyclic relaxation converges linearly. *Computational Optimization and Applications*, **41**, 53–60.
[58] Minoux, M. (1986) *Mathematical Programming: Theory and Algorithms*. Wiley, Chichester.
[59] Rautenbach, D. and Szegedy, C. (2004) A subgradient method using alternating projections. Tech. Rep.

04940, Forschungsinstitut für Diskrete Mathematik, Universität Bonn.
[60] Polyak, B. T. (1967) A general method for solving extremal problems. *Doklady Akademii Nauk SSSR*, **174**, 33–36 (Russian), english translation, Soviet Mathematics Doklady 8 (1967), 593–597].
[61] Polyak, B. T. (1969) Minimization of unsmooth functionals. *USSR Computational Mathematics and Mathematical Physics*, **9**, 14–29.
[62] Cheney, W. and Goldstein, A. A. (1959) Proximity maps for convex sets. *Proceedings of the American Mathematical Society*, **10**, 448–450.
[63] Held, S. (2009) Gate sizing for large cell-based designs. *2009 Design, Automation and Test in Europe*, (Nice, 2009), pp. 827–832, IEEE Press, Piscataway, NJ.
[64] Schmedding, R. (2007) *Time-Cost-Tradeoff-Probleme und eine Anwendung in der Timing-Optimierung im VLSI-Design*. Diploma thesis, Universität Bonn.
[65] Held, S. (2008) *Timing Closure in Chip Design*. Dissertation, Universität Bonn.
[66] Phillips, J., Steve and Dessouky, M. I. (1977) Solving the project time/cost tradeoff problem using the minimal cut concept. *Management Science*, **24**, 393–400.
[67] Skutella, M. (1998) Approximation algorithms for the discrete time-cost tradeoff problem. *Mathematics of Operations Research*, **23**, 909–929.
[68] Fassnacht, U. and Schietke, J. (1998) Timing analysis and optimization of a high-performance CMOS processor chipset. *1998 Design, Automation and Test in Europe*, (Paris, 1998), pp. 325–331, IEEE Computer Society, Los Alamitos, CA.
[69] Albrecht, C., Korte, B., Schietke, J., and Vygen, J. (1999) Cycle time and slack optimization for VLSI-chips. *1999 IEEE/ACM International Conference on Computer-Aided Design*, (San Jose, CA, 1999), pp. 232–238, IEEE Press, Piscataway, NJ.
[70] Albrecht, C., Korte, B., Schietke, J., and Vygen, J. (2002) Maximum mean weight cycle in a digraph and minimizing cycle time of a logic chip. *Discrete Applied Mathematics*, **123**, 103–127.
[71] Held, S., Korte, B., Maßberg, J., Ringe, M., and Vygen, J. (2003) Clock scheduling and clocktree construction for high performance asics. *2003 International Conference on Computer Aided Design*, (San Jose, CA, 2003), pp. 232–239, IEEE Computer Society, Los Alamitos, CA.
[72] Karp, R. M. (1978) A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, **23**, 309–311.
[73] Young, N. E., Tarjan, R. E., and Orlin, J. B. (1991) Faster parametric shortest path and minimum-balance algorithms. *Networks*, **21**, 205–221.
[74] Held, S. (2001) *Algorithmen für Potential-Balancierungs-Probleme und Anwendungen im VLSI-Design*. Diploma thesis, Universität Bonn.
[75] Schneider, H. and Schneider, M. H. (1991) Max-balancing weighted directed graphs and matrix scaling. *Mathematics of Operations Research*, **16**, 208–222.
[76] Albrecht, C. (2001) *Zwei kombinatorische Optimierungsprobleme im VLSI-Design: Optimierung der Zykluszeit und der Slackverteilung und globale Verdrahtung*. Ph.D. thesis, Universität Bonn.
[77] Megiddo, N. (1983) Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM*, **30**, 852–865.
[78] Chao, T.-H., Hsu, Y.-C., and Ho, J.-M. (1992) Zero skew clock net routing. *Proceedings of the 29th ACM/IEEE Design Automation Conference*, (Anaheim, CA, 1992), pp. 518–523, IEEE Computer Society, Los Alamitos, CA.
[79] Gester, M. (2009) *Voronoi-Diagramme von Achtecken in der Maximum-Metrik*. Diploma thesis, Universität Bonn.
[80] Maßberg, J. and Vygen, J. (2008) Approximation algorithms for a facility location problem with service capacities. *ACM Transactions on Algorithms*, **4**, Art. 50.
[81] Maßberg, J. (2009) *Facility Location and Clock Tree Synthesis*. Dissertation, Universität Bonn.
[82] Müller, D. (2002) *Bestimmung der Verdrahtungskapazitäten im Global Routing*. Diploma thesis, Universität Bonn.
[83] Raghavan, P. and Thompson, C. D. (1987) Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, **7**, 365–374.
[84] Raghavan, P. and Thompson, C. D. (1991) Multiterminal global routing: A deterministic approximation scheme. *Algorithmica*, **6**, 73–82.
[85] Shahrokhi, F. and Matula, D. W. (1990) The maximum concurrent flow problem. *Journal of the ACM*, **37**, 318–334.
[86] Carden, I., R. C., Li, J., and Cheng, C.-K. (1996) A global router with a theoretical bound on the optimal

solution. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **15**, 208–216.

[87] Albrecht, C. (2001) Global routing by new approximation algorithms for multicommodity flow. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **20**, 622–632.

[88] Garg, N. and Könemann, J. (1998) Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, (Palo Alto, CA, 1998), pp. 300–309, IEEE Computer Society, Los Alamitos, CA.

[89] Müller, D. (2006) Optimizing yield in global routing. *Proceedings of the 2006 IIEEE/ACM International Conference on Computer-Aided Design*, (San Jose, CA, 2006), pp. 480–486, ACM, New York, NY.

[90] Vygen, J. (2004) Near-optimum global routing with coupling, delay bounds, and power consumption. Nemhauser, G. L. and Bienstock, D. (eds.), *Integer Programming and Combinatorial Optimization*, (New York, NY, 2004), vol. 3064 of *Lecture Notes in Computer Science*, pp. 308–324, Springer, Berlin.

[91] Müller, D., Radke, K., and Vygen, J. (2010) Faster min-max resource sharing in theory and practice. Tech. Rep. 101013, Forschungsinstitut für Diskrete Mathematik, Universität Bonn.

[92] Müller, D. (2009) *Fast Resource Sharing in VLSI Routing*. Dissertation, Universität Bonn.

[93] Fredman, M. L. and Tarjan, R. E. (1987) Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, **34**, 596–615.

[94] Hetzel, A. (1998) A sequential detailed router for huge grid graphs. *1998 Design, Automation and Test in Europe*, (Paris, 1998), pp. 332–338, IEEE Computer Society, Los Alamitos, CA.

[95] Peyer, S., Rautenbach, D., and Vygen, J. (2009) A generalization of Dijkstra's shortest path algorithm with applications to VLSI routing. *Journal of Discrete Algorithms*, **7**, 377–390.

[96] Humpola, J. (2009) *Schneller Algorithmus für kürzeste Wege in irregulären Gittergraphen*. Diploma thesis, Universität Bonn.

[97] Koehl, J., Baur, U., Ludwig, T., Kick, B., and Pflueger, T. (1998) A flat, timing-driven design system for a high-performance cmos processor chipset. *1998 Design, Automation and Test in Europe*, (Paris, 1998), pp. 312–320, IEEE Computer Society, Los Alamitos, CA.

[98] Koehl, J., Lackey, D. E., and Doerre, G. (2003) IBM's 50 million gate ASICs. *Proceedings of the 2003 Asia and South Pacific Design Automation Conference*, (Kitakyushu, 2003), pp. 628–634, ACM, New York, NY.

Combinatorial Optimization: Methods and Applications
V. Chvátal (Ed.)
IOS Press, 2011

doi:10.3233/978-1-60750-718-5-97

Facility Location: Discrete Models and Local Search Methods

Yury KOCHETOV [1]
Sobolev Institute of Mathematics, Russia

Abstract. Discrete location theory is one of the most dynamic areas of operations research. We present the basic mathematical models used in this field, as well as their properties and relationship with pseudo-Boolean functions. We also investigate the theory of PLS-complete problems, average and worst case computational complexity of the local search algorithms, and approximate local search. Finally, we discuss computationally difficult test instances and promising directions for further research.

Keywords. Local search, PLS-complete problems, pseudo-Boolean function, Karush – Kuhn – Tucker conditions, metaheuristics

Introduction

Facility location constitutes a broad spectrum of mathematical models, methods, and applications in operations research. It is an interesting topic for theoretical studies, experimental research, and real-world applications. Examples include storage facilities, warehouses, police and fire stations, base stations for wireless services, and others [1]. Who actually proposed the first mathematical model in this field will probably never be known. It is most common to credit Pierre de Fermat (1601 – 1665) and Evangelista Torricelli (1608 – 1647), who studied a basic form of the spacial median problem (see [2] for a historical review of the literature).

Surely no paper can cover all aspects of facility location. In these lecture notes, we discuss only the basic discrete models and present theoretical results for local search methods. In Section 1 we consider the well-known uncapacitated facility location problem and its generalizations. The main idea of this section is to show the useful relationship between facility location models and pseudo-Boolean functions. In Section 2 we introduce some simple neighborhoods and discuss the relations between local optima and classical Karush – Kuhn – Tucker conditions. In Section 3 we define the class PLS (polynomial time local search problems) and show that some local search problems in facility location are the most difficult in this class. In Section 4 we investigate the quality of local optima. More precisely, we introduce the class GLO (Guaranteed Local Optima) and show that the closure of GLO under PTAS reductions coincides with the class APX, which is the set of optimization problems with underlying decision problem in NP that allow polynomial-time constant-factor approximation algorithms. Finally, in Section 5 we discuss difficult test instances for the local search methods.

[1] Sobolev Institute of Mathematics, Akademika Koptyuga pr. 4, Novosibirsk, 630090, Russia; E-mail: jkochet@math.nsc.ru

1. Discrete Facility Location Models

1.1. The Uncapacitated Facility Location Problem

This problem (UFLP) has been also called *the simple plant location problem* in the early literature [3, 4]. Its input consists of a finite set I of *sites* with nonnegative *fixed costs* f_i of opening a facility in site i and a finite set J of *users* with a nonnegative *production-transportation costs* c_{ij} of servicing user j from a facility opened in site i. The goal is to find a set of sites such that opening facilities there minimizes the total cost of servicing all users. This means finding a nonempty subset S of I, which minimizes the objective function F defined by

$$F(S) = \sum_{i \in S} f_i + \sum_{j \in J} \min_{i \in S} c_{ij} :$$

the first term is the fixed cost for opening facilities in all sites in S and the second term is the production-transportation cost for servicing all users from these sites.

This problem is NP-hard in the strong sense and it is difficult to approximate: unless P = NP, it admits no constant-factor polynomial-time approximation algorithm, and so it does not belong to the class APX. Polynomially solvable cases and approximation algorithms are described in [5, 6].

For the metric case, when the matrix (c_{ij}) satisfies the triangle inequality, the problem is strongly NP-hard again and Max SNP-hard. The best known approximation algorithm has a guaranteed performance ratio of 1.52 and is suggested in [7]. A 1.463 factor approximation algorithm would imply P = NP [8]. For the special case of the metric UFLP when facilities and users are points in d-dimensional Euclidean space and the production-transportation costs are geometrical distances between the points, an approximation scheme is suggested meaning an $(1+\varepsilon)$-factor approximation algorithm for each positive ε with running time polynomial in $|I|$ and $|J|$ and exponential in d and $1/\varepsilon$. An excellent review of different techniques for approximations in the metric case can be found in [9].

Exact branch and bound methods with lower bounds based on linear programming relaxations have been developed by research teams in several countries. In describing these bounds, we shall find it convenient to assume that $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, n\}$. The first approaches used a *weak* linear 0-1 programming formulation [10]. Let us introduce the decision variables:

$$x_i = \begin{cases} 1 & \text{if facility } i \text{ is opened,} \\ 0 & \text{otherwise,} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if user } j \text{ is serviced from facility } i, \\ 0 & \text{otherwise.} \end{cases}$$

Now the UFLP can be written as a 0-1 program:

$$\min\Big\{\sum_{i\in I} f_i x_i + \sum_{j\in J}\sum_{i\in I} c_{ij}x_{ij}\Big\}$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{i\in I} x_{ij} = 1, && j \in J,\\
& nx_i \geq \sum_{j\in J} x_{ij}, && i \in I,\\
& x_i, x_{ij} \in \{0,1\}, && i \in I, j \in J.
\end{aligned}$$

If we replace the last restriction by $x_i, x_{ij} \in [0,1]$ for all $i \in I$, $j \in J$ we see that every optimal solution satisfies

$$x_i = \frac{1}{n}\sum_{j\in J} x_{ij}, \quad i \in I$$

and a lower bound can be computed as the optimal value of the following trivial linear programming problem:

$$\begin{aligned}
\min \quad & \sum_{j\in J}\sum_{i\in I}(f_i/n + c_{ij})x_{ij}\\
\text{subject to} \quad & \sum_{i\in I} x_{ij} = 1, && j \in J,\\
& 0 \leq x_{ij} \leq 1, && i \in I, j \in J.
\end{aligned}$$

We may compute the optimal solution for the problem easily, but this lower bound is not sharp. If we replace m restrictions $nx_i \geq \sum_{j\in J} x_{ij}$ by $n \times m$ restrictions

$$x_i \geq x_{ij}, \quad i \in I, j \in J,$$

we will get an equivalent reformulation of the UFLP with a better lower bound. Note that we do not have an analytical solution yet. This *strong* reformulation has been used in the exact methods by researchers of Russia [11, 12], Ukraine [13], Scandinavia [14], and the USA [15] independently. Recently, the strong formulation was used for solving large scale instances as well [16, 17]. Therefore, for combinatorial optimization problems we can find different equivalent reformulations and the choice of formulation is important.

There is a useful relationship between the UFLP and minimization problem for pseudo-Boolean functions. It was first noted by P. Hammer and S. Rudeanu [18, 19]. Later, V. Beresnev [20] suggested another reduction of the UFLP to a minimization problem for pseudo-Boolean polynomial with positive coefficients for nonlinear terms. Moreover, it has been shown that these combinatorial optimization problems are equivalent. Below we discuss the reduction in details.

For a vector $g = (g_1, \ldots, g_m)$ with ranking

$$g_{i_1} \leq g_{i_2} \leq \cdots \leq g_{i_m},$$

we introduce a vector $\Delta g = (\Delta g_0, \ldots, \Delta g_m)$ in the following way:

$$\Delta g_0 = g_{i_1};$$
$$\Delta g_l = g_{i_{l+1}} - g_{i_l}, \quad 1 \le l < m;$$
$$\Delta g_m = g_{i_m}.$$

Lemma 1 ([11, 20]). *For each* 0-1 *vector* $z = (z_1, \dots, z_m)$ *distinct from* $(1, \dots, 1)$ *the following equations hold*:

$$\min_{i|z_i=0} g_i = \Delta g_0 + \sum_{l=1}^{m-1} \Delta g_l z_{i_1} \dots z_{i_l};$$
$$\max_{i|z_i=0} g_i = \Delta g_m - \sum_{l=1}^{m-1} \Delta g_{m-l} z_{i_{m-l+1}} \cdots z_{i_m}.$$

Let the ranking for column j of the matrix (c_{ij}) be

$$c_{i_1^j j} \le c_{i_2^j j} \le \cdots \le c_{i_m^j j}$$

Using Lemma 1, we can get a pseudo-Boolean function for the UFLP:

$$b(z) = \sum_{i \in I} f_i(1 - z_i) + \sum_{j \in J} \sum_{l=0}^{m-1} \Delta c_{lj} z_{i_1^j} \dots z_{i_l^j}.$$

Below we will see the relationship between the UFLP and the minimization problem for this real-valued function defined on the $2^n - 1$ points of the hypercube distinct from $(1, \dots, 1)$.

Theorem 1 ([11, 20]). *The minimization problem for the pseudo-Boolean function* $b(z)$ *for* $z \ne (1, \dots, 1)$ *and the* UFLP *are equivalent. For optimal solutions* z^*, S^* *of these problems we have* $F(S^*) = b(z^*)$ *and* $z_i^* = 0 \Leftrightarrow i \in S^*$ *for all* $i \in I$.

Let us consider an illustrative example. Put $I = J = \{1, 2, 3\}$,

$$f_i = \begin{pmatrix} 10 \\ 10 \\ 10 \end{pmatrix} \quad c_{ij} = \begin{pmatrix} 0 & 3 & 10 \\ 5 & 0 & 0 \\ 10 & 20 & 7 \end{pmatrix}.$$

According to Lemma 1, the corresponding function $b(z)$ is the following:

$$b(z) = 10(1 - z_1) + 10(1 - z_2) + 10(1 - z_3) + (5z_1 + 5z_1z_2) + (3z_2 + 17z_1z_2) + (7z_2 + 3z_2z_3)$$
$$= 15 + 5(1 - z_1) + 0(1 - z_2) + 10(1 - z_3) + 22z_1z_2 + 3z_2z_3.$$

Let us try to reconstruct an instance of the UFLP: we obtain $I' = I$, $J' = \{1, 2\}$,

$$f_i' = \begin{pmatrix} 5 \\ 0 \\ 10 \end{pmatrix} \quad c_{ij}' = \begin{pmatrix} 0 & 3 \\ 0 & 0 \\ 22 & 0 \end{pmatrix}.$$

The dimension of the new instance is less than the dimension of the original one, $|J'| < |J|$. Moreover, $f'_2 = 0$, hence, we may open the second facility without loss of optimality. In other words, we get a new equivalent instance of the UFLP with smaller dimension. Different instances of the UFLP can lead to the same function $b(z)$. Thus, we may try to reduce the dimension before solving the problem.

Theorem 2 ([21]). *For the minimization problem of the pseudo-Boolean function $b(z)$ with positive coefficients in the nonlinear terms, the equivalent instance of the* UFLP *with a minimal number of users can be found in polynomial time from n and m.*

Idea of proof. Consider an arbitrary pseudo-Boolean function $b(z)$ with positive coefficients in the nonlinear terms. Let L be the set of nonlinear terms and the function $b(z)$ defined by

$$b(z) = \sum_{i \in I} \alpha_i (1 - z_i) + \sum_{l \in L} \beta_l \prod_{i \in I_l} z_i, \quad \text{where } \beta_l > 0 \text{ and } I_l \subset I \text{ for all } l \in L.$$

The family of subsets $\{I_l\}_{l \in L}$ of the set I with order relation $I_{l'} < I_{l''} \Leftrightarrow I_{l'} \subset I_{l''}$ forms a partially ordered set (poset). An arbitrary sequence of subsets $I_{l_1} < \cdots < I_{l_k}$ is called a *chain*. An arbitrary partition of the family $\{I_l\}_{l \in L}$ into nonoverlapping chains induces a matrix (c_{ij}) for the UFLP. Each element of the partition corresponds to a user. The requirement to find an instance of the UFLP with a minimal number of users is equivalent to finding a partition of the poset into the minimal number of nonoverlapping chains. This is a well-known problem, which can be solved in polynomial time (see Dilworth's Theorem [22]). □

The minimization problem for $b(z)$ is equivalent to the UFLP but it has some new properties. Let us consider this problem for continuous variables z_i from the interval $[0, 1]$. In the UFLP this replacement leads to an integrality gap:

$$\text{gap} = (F(S^*) - F_{\text{LP}})/F(S^*),$$

where F_{LP} is the optimal value for the linear programming relaxation. It can be arbitrary close to 1 [23]. For the minimization problem of $b(z)$ the gap equals 0.

Theorem 3 ([24]). *The set of optimal solutions of the minimization problem for arbitrary pseudo-Boolean function with continuous variables contains a pure integer solution.*

Suppose now that the fixed costs are the same for all facilities and that we open exactly p facilities. The first item in the objective function of the UFLP is a constant and we wish to minimize the objective function F defined by

$$F(S) = \sum_{j \in J} \min_{i \in S} c_{ij}$$

for every subset S from I with cardinality p. This problem is known as the discrete p-median problem. It is NP-hard in the strong sense and existence of a $2^{q(n,m)}$-factor approximation algorithm for a polynomial q would imply P = NP [25]. In other words, this problem does not belong to the class APX and a good approximate solution is hard to find, as is the optimal one. The p-median problem can be reduced to the minimization

problem for a pseudo-Boolean function as well. However, now one additional restriction $\sum_{i\in I} z_i = m - p$ is added. Reformulations of Theorems 2 and 3 for the problem are valid as well.

1.2. The Multi-Stage Facility Location Problem

Let us consider a more complicated situation, where we need several facilities to produce the goods for users. We assume that there are k types of facilities (plants, warehouses, distribution centers and others) and we need facilities of all types to produce the goods. As in the previous model, the set of facilities I and the set of users J are finite. We assume that $I = I_1 \cup \cdots \cup I_k$, where I_l is the set of facilities of type l and $I_{l_1} \cap I_{l_2} = \varnothing$ whenever $l_1 \neq l_2$. Let P denote the set of all admissible facility paths. For each path p from P we know a sequence of facilities $p = \{i_1, \ldots, i_k\}$, where $i_l \in I_l$ for each l from 1 to k. For each facility i we have the following nonnegative parameters: c_i is the production cost, $d_{ii'}$ is the transportation cost between facilities i and i', and the set P_i of facility paths, which contain this facility. Denote by D_{pj} the total transportation cost for the facility path p and user j:

$$D_{pj} = d_{i_1 i_2} + \cdots + d_{i_{k-1} i_k} + d_{i_k j}.$$

Similarly, the total production cost for the facility path p is the following:

$$C_p = c_{i_1} + \cdots + c_{i_k}.$$

The amount of goods for user j we denote by φ_j. In the Multi-Stage Uncapacitated Facility Location Problem (MSUFLP) we need to find a subset of facilities and a subset of facility paths in such a way to service all users with minimal total cost for opening facilities, producing the goods and transporting them to users. Using similar variables as for the UFLP, we can write the problem in the following way:

$$\min\left\{\sum_{i\in I} f_i x_i + \sum_{j\in J} \varphi_j \sum_{p\in P} (C_p + D_{pj}) x_{pj}\right\}$$

$$\text{subject to} \quad \sum_{p\in P} x_{pj} = 1, \qquad j \in J,$$

$$\sum_{p\in P_i} x_{pj} \leq x_i, \qquad i \in I, j \in J,$$

$$x_{pj}, x_i \in \{0, 1\}, \qquad i \in I, p \in P, j \in J.$$

The objective function is the total cost. The first restriction ensures that all users are satisfied. The second restriction allows us to use opening facilities only for servicing the users.

The problem is strongly NP-hard and closely related to standardization and unification problems [11, 14], pseudo-Boolean functions, and as we will see below, the bilevel facility location problems. If each facility path contains exactly one facility then we get the UFLP. Therefore, it is a difficult problem for approximation. For the metric case some approximation algorithms are developed in [26, 27]. The branch and bound meth-

ods based on heuristics for the dual linear programming problems are studied in [28,29]. Polynomially solvable cases are described in [30,31].

The restriction $x_{pj} \leq x_i$ for all $p \in P_i$, $i \in I$, $j \in J$, can be used instead of

$$\sum_{p \in P_i} x_{pj} \leq x_i, \quad i \in I, j \in J.$$

It is easy to check that this replacement gives the same integer optimal solution but a weak linear programming relaxation. The same effect will occur (see [32]) if we introduce new variables $y_p = 1$ if the facility path p is used and $y_p = 0$ otherwise, and replace the restriction $\sum_{p \in P_i} x_{pj} \leq x_i$ for all $i \in I$, $j \in J$, by the following:

$$y_p \geq x_{pj}, \quad j \in J, p \in P,$$
$$x_i \geq y_p, \quad p \in P_i, i \in I.$$

Again, we have the same 0-1 optimal solution and a weak linear programming relaxation. Moreover, there is a family of instances for the MSUFLP where the ratio of optimal value of the original and new linear programming relaxations may be arbitrarily large. Let us rewrite the MSUFLP as an unconstrained problem. Define $C_{pj} = \varphi_j(C_p + D_{pj})$ for all $p \in P$, $j \in J$. Now the MSUFLP can be written as the minimization problem for the function F defined by

$$F(y) = \sum_{i \in I} f_i \max_{p \in P_i}\{y_p\} + \sum_{j \in J} \min_{p \in P}\{C_{pj} \mid y_p = 1\}.$$

Using Lemma 1 we can obtain the following pseudo-Boolean function:

$$B(z) = \sum_{i \in I} f_i (1 - \prod_{p \in P_i} z_p) + \sum_{j \in J} \sum_{l=0}^{|P|-1} \Delta C_{lj} z_{i_1^j} \cdots z_{i_l^j}.$$

Note that this function has positive and negative nonlinear terms.

Theorem 4 ([11, 20])**.** *The* MSUFLP *is equivalent to the minimization problem for pseudo-Boolean function $B(z)$ for $z \neq (1, \ldots, 1)$. For optimal solutions z^*, y^* of these problems we have $B(z^*) = F(y^*)$ and $z_p^* = 1 - y_p^*$ for all $p \in P$.*

Therefore, the MSUFLP can be reduced to a minimization problem for a pseudo-Boolean function. For an arbitrary pseudo-Boolean function we can reconstruct an equivalent instance of the MSUFLP with the minimal number of users.

1.3. Facility Location with User Preferences

Thus far we have assumed that there was only one decision maker who tried to minimize the total cost of opening facilities and servicing users. However, users may be free to choose the facility. They may have their own preferences, for example, the travel time to a facility. They do not have to minimize the production and transportation costs of the firm. Hence, we should include user preferences in the mathematical model [33].

Let the matrix (g_{ij}) define the user preferences on the set I. If $g_{i_1 j} < g_{i_2 j}$, then user j prefers facility i_1. We assume for simplicity that all elements are different in each column of the matrix. Otherwise, we must consider cooperative and noncooperative strategies for the decision maker and users [34]. Therefore, the decision maker wishes to find a subset S of opening facilities in such a way that all users will be serviced with minimal total cost, taking into account user preferences. For this case, the mathematical model can be presented as the 0-1 bilevel linear programming problem [35,36]: minimize the objective function F defined by

$$F(x_i) = \sum_{i \in I} f_i x_i + \sum_{j \in J} \sum_{i \in I} c_{ij} x^*_{ij}(x_i)$$

where $x^*_{ij}(x_i)$ is an optimal solution for the user problem:

$$\begin{aligned} &\min_{x_{ij}} \sum_{j \in J} \sum_{i \in I} g_{ij} x_{ij} \\ &\text{subject to} \quad \sum_{i \in I} x_{ij} = 1, \qquad j \in J, \\ &\qquad\qquad x_{ij} \le x_i, \qquad i \in I, j \in J, \\ &\qquad\qquad x_{ij} \in \{0, 1\}, \qquad i \in I, j \in J. \end{aligned}$$

The objective function of the decision maker is, as before, the total cost of opening facilities and servicing all users. However, now the feasible domain is described by constraints $x_i \in \{0, 1\}$ for all $i \in I$ and the auxiliary optimization problem (the user problem). The values of variables x_i are known for the auxiliary problem. The bilevel problem is a new type of optimization problem. Such problems can be NP-hard even with continuous variables, linear constraints and linear objective functions [34].

The uncapacitated facility location problem with user preferences (UFLPUP) can be reduced to a single level problem [35, 36]. Observe that only the ranking of the g_{ij}'s for each j is of importance and not their numerical values. Let the ranking for user j be

$$g_{i_1 j} < g_{i_2 j} < \cdots < g_{i_m j}.$$

Put $S_{ij} = \{l \in I | \, g_{lj} < g_{ij}\}$ for all $i \in I$. For an optimal solution $x^*_{ij}(x_i)$ of the user problem we have $x^*_{ij} = 1 \Rightarrow x_l = 0$ for all $l \in S_{ij}$. We may therefore rewrite the UFLPUP as follows:

$$\begin{aligned} &\min \sum_{i \in I} f_i x_i + \sum_{j \in J} \sum_{i \in I} c_{ij} x_{ij} \\ &\text{subject to} \quad x_{ij} + x_l \le 1, \qquad l \in S_{ij}, i \in I, j \in J, \\ &\qquad\qquad \sum_{i \in I} x_{ij} = 1, \qquad j \in J, \\ &\qquad\qquad x_{ij} \le x_i, \qquad i \in I, j \in J, \\ &\qquad\qquad x_i, x_{ij} \in \{0, 1\}, \qquad i \in I, j \in J. \end{aligned}$$

Indeed, in every optimal solution of the problem all constraints of UFLP will be satisfied and the first constraint will ensure that x_{ij} is an optimal solution for the user problem. The number of variables in the problem is the same as in the UFLP. However, while the UFLP already has the large number of constraints $n + nm$, the UFLPUP has $O(m^2 n)$ additional ones. This prohibits a direct resolution except in small instances. In order to avoid additional constraints from becoming too numerous we can rewrite them in the equivalent form:

$$\sum_{l \in S_{ij}} x_l \le |S_{ij}|(1 - x_{ij}), \quad i \in I, j \in J,$$

or

$$x_i \le x_{ij} + \sum_{l \in S_{ij}} x_l, \quad i \in I, j \in J,$$

or

$$x_i \le x_{ij} + \sum_{l \in S_{ij}} x_{lj}, \quad i \in I, j \in J.$$

It is not difficult to show that the last inequality produces a better linear programming relaxation than the three previous ones [36].

The special case of the UFLPUP when $f_i = 0$ for all $i \in I$ is also interesting. For the UFLP this case is trivial; the optimal solution can be computed in linear time. However, for the UFLPUP this case is NP-hard and the integrality gap can be arbitrarily close to 1. If $c_{ij} = g_{ij}$ then we get the UFLP. If $c_{ij} = -g_{ij}$ then we can solve the problem in polynomial time [37]. Other reformulations, valid inequalities, branch and cut methods and computational results for local search methods can be found in [38–40].

As with the previous location problems, the UFLPUP can be reduced to the minimization problem for the pseudo-Boolean functions. For each $j \in J$ we put

$$\nabla c_{i_1 j} = c_{i_1 j}$$
$$\nabla c_{i_l j} = c_{i_l j} - c_{i_{l-1} j}, \quad 1 < l \le m,$$

and define the pseudo-Boolean function $B(z)$ in the following way:

$$B(z) = \sum_{i \in I} f_i(1 - z_i) + \sum_{j \in J} \sum_{i \in I} \nabla c_{ij} \prod_{l \in S_{ij}} z_l.$$

Theorem 5 ([35])**.** *The* UFLPUP *is equivalent to the minimization problem for the pseudo-Boolean function $B(z)$ for $z \ne (1, \ldots, 1)$. For the optimal solutions z^*, x^* of these problems we have $B(z^*) = F(x^*)$ and $z_i^* = 1 - x_i^*$ for all $i \in I$.*

Note that the coefficients ∇c_{ij} can be positive or negative. In other words, for an arbitrary pseudo-Boolean function we can reconstruct an equivalent instance of the UFLPUP and vice versa. Moreover, we can reconstruct an instance of the UFLPUP with a minimal number of users in polynomial time by the same method as in the proof of Theorem 2.

1.4. Competitive Location with Foresight

Let us assume that two firms want to open facilities. The first firm, which we will refer to as the leader, opens its own set of facilities X from the set I. We assume that $|X| = p$. Later, the second firm, which we will refer to as the follower, opens its own set of facilities Y from the set $I \setminus X$. We assume that $|Y| = r$. Each user selects one facility from the union $X \cup Y$ according to its own preferences, for example, according to distances to the facilities. Each firm will get a positive profit w_j if it services the user j. The firms try to maximize own profits. They do not have the same rights. The leader makes a decision first. The follower makes a decision by analyzing the set X. It is a Stakelberg game for two players, where we need to maximize the total profit of the leader [41–43].

Let us introduce the decision variables:

$$x_i = \begin{cases} 1 & \text{if the leader opens facility } i, \\ 0 & \text{otherwise,} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{if the follower opens facility } i, \\ 0 & \text{otherwise,} \end{cases}$$

$$z_j = \begin{cases} 1 & \text{if user } j \text{ is serviced by a leader facility,} \\ 0 & \text{if user } j \text{ is serviced by a follower facility.} \end{cases}$$

For each vector x and each user j we can define the set of facilities

$$I_j(x) = \{i \in I \mid g_{ij} < \min_{l \in I}(g_{lj} \mid x_l = 1)\},$$

which allow "capturing" user j by the follower. Note that we consider conservative users [43]. If a user has the same distances to the closest leader and the closest follower facilities, he prefers the leader facility. Now the model can be written as a linear 0-1 bilevel programming problem [44]:

$$\max_x \sum_{j \in J} w_j z_j^*(x)$$

$$\text{subject to} \quad \sum_{i \in I} x_i = p,$$

$$x_i \in \{0, 1\}, \qquad i \in I,$$

where $z_j^*(x), y_i^*(x)$ is the optimal solution of the follower problem:

$$\max_{z,y} \sum_{j \in J} w_j (1 - z_j)$$

$$\text{subject to} \quad 1 - z_j \le \sum_{i \in I_j(x)} y_i, \qquad j \in J,$$

$$\sum_{i \in I} y_i = r,$$

$$y_i, z_j \in \{0, 1\}, \qquad i \in I, j \in J.$$

The objective function of the upper level defines the total profit of the leader. The feasible domain is described by two constraints and an auxiliary optimization problem of the follower. The vector x and the sets $I_j(x)$ for all $j \in J$ are known in the follower problem. The objective function of the lower level defines the total profit of the follower. The first constraint guarantees that user j is serviced by the leader if the follower has no facilities in the set $I_j(x)$. The second constraint requires opening exactly r facilities for the follower.

In [43] the follower problem is called the *medianoid* problem. It is shown there that the medianoid problem is NP-hard and that the original bilevel problem is NP-hard even for $r = 1$. Note that $z_j^*(x) = \prod_{i \in I_j(x)}(1 - y_i^*(x))$ for all $j \in J$. Hence, the follower problem can be rewritten for the polynomial $P(y, x) = \sum_{j \in J} w_j(1 - \prod_{i \in I_j(x)}(1 - y_i))$ as follows:

$$\max_y \left\{ P(y, x) \;\middle|\; \sum_{i \in I} y_i = r, y_i \in \{0, 1\} \text{ for all } i \in I \right\}.$$

Recall that each user is serviced by the leader or by the follower. Thus, the sum of the objective functions for the upper and lower levels is a constant. Hence, we can present the bilevel problem as a min-max problem as follows:

$$\min_x \max_y \left\{ P(y, x) \;\middle|\; \sum_{i \in I} y_i = r, \sum_{i \in I} x_i = p, x_i, y_i \in \{0, 1\} \text{ for all } i \in I \right\}.$$

In [45] it is shown that the problem is Σ_2^{P}-hard. Therefore, we are dealing with a more difficult problem than the NP-complete problems. Polynomially solvable cases and complexity results can be found in [46]. In order to get an upper bound for the total profit of the leader we can rewrite the bilevel problem as a single level mixed integer linear program with an exponential number of constraints and variables. A similar approach is suggested in [42] for a partial enumeration algorithm. If we extract a subfamily of constraints and variables, then we may get an upper bound. In [44] a nonclassical column generation method is applied to find an optimal solution for the bilevel problem. Computational experiments for the test instances from the benchmark library *Discrete Location Problems* (http://math.nsc.ru/AP/benchmarks/english.html) indicate that the exact method allows us to find the global optimum for $p = r = 5$, $n = m = 100$.

For higher dimensions we may apply heuristics or metaheuristics. The simplest heuristic for the leader is to ignore the follower. The leader opens its own facilities to minimize the total distance between users and his facilities. He wishes to service all users and solves the classical p-median problem. This strategy is not so bad despite ignoring the follower. Computational experiments show that this lower bound can be improved by a few percent only.

The second strategy is more sophisticated. The leader anticipates that the follower will react to his decision. Therefore, $(p + r)$ facilities will be opened. According to the second heuristic, the leader solves the $(p + r)$-median problem and opens the p most profitable facilities. Unfortunately, this strategy is weak.

There is a third strategy suggested for continuous locations [47]. This heuristic is iterative. For a solution of one decision maker, we find the optimal solution for the other one. In discrete case this strategy produces a cycle. The best solution in the cycle is the result of the approach. If we use the previous strategies to create a starting solution, we can improve the profit of the leader. Surely, this is a more time consuming procedure.

One of the most powerful approaches is a hybrid memetic algorithm, where a tabu search is used to improve the elements of the population [48]. To evaluate neighboring solutions for the leader, the linear programming relaxation of the follower problem is solved by CPLEX software. To reduce the running time at each step of the tabu search, the idea of randomized neighborhoods is used. Other heuristics can be found in [49, 50].

2. Local Optima and Karush – Kuhn – Tucker Conditions

Let us consider the minimization problem for the pseudo-Boolean function $B(z) = \sum_{l\in L} \gamma_l \prod_{i\in I_l} z_i$ with arbitrary coefficients γ_l. The UFLP, MSUFLP, and UFLPUP can be reduced to this problem. We wish to show the relationship between Flip-minimal solutions for $B(z)$ and solutions that satisfy the classical Karush – Kuhn – Tucker conditions for the Lagrange function L defined by

$$L(z, \sigma, \mu) = B(z) + \sum_{i\in I} \sigma_i(z_i - 1) - \sum_{i\in I} \mu_i z_i$$

for continuous variables $z_i \in [0, 1]$ and nonnegative multipliers σ_i, μ_i corresponding to constraints $z_i - 1 \le 0$ and $z_i \ge 0$ for each $i \in I$. Recall that the Flip neighborhood for solution z, or Flip(z) for short, is the set of 0-1 solutions that can be obtained from z by flipping exactly one variable. A solution is called local minimal, for example Flip-minimal, if it does not have neighboring solution with smaller value of the objective function.

Theorem 6 ([51])**.** *A* 0-1 *vector* z^* *is* Flip*-minimal if and only if there exist nonnegative multipliers* σ_i^*, μ_i^**, for all* $i \in I$ *such that the vector* (z^*, σ^*, μ^*) *satisfies the Karush – Kuhn – Tucker conditions*:

(i) $(\partial L/\partial z_i)(z^*, \sigma^*, \mu^*) = \sum_{l\in L|i\in I_l} \gamma_l \prod_{j\in I_l\setminus\{i\}} z_j^* - \mu_i^* + \sigma_i^* = 0$, $i \in I$;
(ii) $z_i^*\mu_i^* = 0$, $i \in I$;
(iii) $\sigma_i^*(z_i^* - 1) = 0$, $i \in I$.

Proof. Suppose that the vector (z^*, σ^*, μ^*) satisfies the conditions (i) – (iii). Let $z' \in$ Flip(z^*) and $z_i^* = 0, z_i' = 1$ for an index $i \in I$. We then have

$$B(z^*) - B(z') = -\sum_{l\in L|i\in I_l} \gamma_l \prod_{j\in I_l\setminus\{i\}} z_j^* = \sigma_i^* - \mu_i^* = -\mu_i^* \le 0.$$

Assume that $z_i^* = 1, z_i' = 0$. We have

$$B(z^*) - B(z') = \sum_{l\in L|i\in I_l} \gamma_l \prod_{j\in I_l\setminus\{i\}} z_j^* = \mu_i^* - \sigma_i^* = -\sigma_i^* \le 0.$$

For both cases $B(z^*) \le B(z')$, and z^* is Flip-minimal.

Consider a Flip-minimal vector z^*. Denote by $B_i'(z)$ the first derivative of the function $B(z)$ by the variable z_i. Put

$$\mu_i^* = \begin{cases} 0 & \text{if } z_i^* = 1 \\ B_i'(z^*) & \text{if } z_i^* = 0 \end{cases}, \quad \sigma_i^* = \begin{cases} 0 & \text{if } z_i^* = 0 \\ -B_i'(z^*) & \text{if } z_i^* = 1 \end{cases}, \qquad i \in I.$$

If $z_i^* = 0$, $z'_i = 1$, then $B(z^*) - B(z') = -B'_i(z^*) = -\mu_i^*$. Since $B(z^*) \le B(z')$, we have $\mu_i^* \ge 0$. If $z_i^* = 1$, $z'_i = 0$, then $B(z^*) - B(z') = B'_i(z^*) = -\sigma_i^*$, and we have $\sigma_i^* \ge 0$. For both cases the conditions (i) – (iii) hold, which completes the proof. □

Let us introduce an additional constraint $\sum_{i\in I} z_i = m - p$ into the minimization problem for $B(z)$. This new problem corresponds to the p-median problem and its generalizations. The Swap-neighborhood for z, or Swap(z) for short, is the set of 0-1 solutions, which can be obtained from z by flipping exactly two variables with different values.

The Lagrange function L with multiplier λ and nonnegative multipliers μ_i, σ_i for each $i \in I$ is defined as follows:

$$L(z, \lambda, \mu, \sigma) = B(z) + \lambda(m - p - \sum_{i\in I} z_i) + \sum_{i\in I} \sigma_i(z_i - 1) - \sum_{i\in I} \mu_i z_i.$$

The corresponding Karush – Kuhn – Tucker conditions are presented as:

$$\begin{aligned}
&\frac{\partial L}{\partial z_i}(z, \lambda, \mu, \sigma) = B'_i(z) - \lambda + \sigma_i - \mu_i = 0, \quad && i \in I,\\
&\sum_{i\in I} z_i = m - p,\\
&\sigma_i(z_i - 1) = 0, && i \in I,\\
&\mu_i z_i = 0, && i \in I.
\end{aligned}$$

The vector $(z^*, \lambda^*, \mu^*, \sigma^*)$ is called *a saddle point* with respect to Swap-neighborhood or Swap-*saddle point* if

$$L(z^*, \lambda, \mu, \sigma) \le L(z^*, \lambda^*, \mu^*, \sigma^*) \le L(z, \lambda^*, \mu^*, \sigma^*)$$

for all 0-1 vectors $z \in$ Swap(z^*), for all λ, and all nonnegative multipliers μ, σ.

Theorem 7 ([52]). *For each* 0-1 *vector* z^* *the following properties are equivalent*:

(i) z^* *is* Swap-*minimal.*
(ii) z^* *satisfies the KKT conditions.*
(iii) *There are the multiplier* λ^* *and nonnegative multipliers* μ_i^*, σ_i^* *for each* $i \in I$ *such that the vector* $(z^*, \lambda^*, \mu^*, \sigma^*)$ *is the* Swap-*saddle point of the Lagrange function* $L(z, \lambda, \mu, \sigma)$.

The reductions of the facility location problems to the minimization problem for the function $B(z)$ save the objective function value. Hence, the vector z is Flip-minimal for $B(z)$ (Swap-minimal) if and only if the corresponding solution S defined by $S(z) = \{i \in I \mid z_i = 0\}$ is Flip-minimal (Swap-minimal) for the location problem. When we use wider neighborhoods, for example, k-Flip, Lin – Kernighan neighborhoods, Fiduccia – Mattheyses neighborhoods and others [52], the set of local optima is decreased. However, all local optima satisfy the KKT conditions. Hence, the large neighborhoods extract the highest quality KKT points and we should use them in local search methods. Other theoretical properties of polynomially searchable neighborhoods can be found in [53–56].

3. Complexity of Local Search

3.1. The Class PLS *and* PLS-*Complete Problems*

In order to introduce the concept of local search problems, let us recall a formal definition of an optimization problem. *An optimization problem* OP is defined by the quadruple $\langle \mathcal{I}, \text{Sol}, F, \text{goal} \rangle$, where

1. $\mathcal{I}$ is the set of instances of OP;
2. Sol is a function that associates to every input instance x of OP the set of its feasible solutions;
3. F is the cost function that assigns an integer $F(s, x)$ for every feasible solution s of x;
4. goal $\in \{\min, \max\}$ specifies whether OP is a maximization or a minimization problem.

In the problem OP we need to find an optimal solution for a given instance.

Definition 1. A *local search problem* Π is a pair (OP, N), where OP is an optimization problem and N is a function that, for every pair (x, s), assigns a set $N(s, x)$ of neighboring feasible solutions. In the local search problem we need to compute a solution that does not have a better neighboring solution.

We will assume that for each instance x its feasible solutions have length bounded by a polynomial in the length of x.

Definition 2. A local search problem Π is in the class PLS if there are three polynomial-time algorithms A, B, C with the following properties:

1. For each string x, algorithm A determines whether x is an instance ($x \in \mathcal{I}$), and in this case it produces a feasible solution.
2. For each instance x and each string s, algorithm B determines whether s is a feasible solution for x and if so, B computes the cost $F(s, x)$.
3. For each instance x and each feasible solution s, algorithm C determines whether s is a local optimum, and if it is not, C outputs a neighboring solution for s with better cost.

This definition gives rise directly to the standard local search algorithm, which starts from the initial solution generated by the algorithm A, and then applies repeatedly algorithm C until it reaches a local optimum. The precise algorithm is determined by the chosen pivoting rule. For a current solution that is not a local optimum, the pivoting rule selects a neighboring solution with strictly better cost.

The class PLS is not empty. A lot of well-known combinatorial optimization problems with natural polynomial neighborhoods belong to it, for example, the traveling salesman problem with a polynomially searchable neighborhood, or the uncapacitated facility location problem with the Flip or Swap neighborhoods.

Definition 3. A local search problem Π from the class PLS belongs to the class P_{PLS} if there exists a polynomial time algorithm that returns a local optimum for every instance of the problem.

The class P_{PLS} is the polynomially solvable part of the class PLS. The relationship between the classes PLS and P_{PLS} is fundamental to complexity theory. If $P_{PLS} \neq PLS$ then $P \neq NP$.

The class P_{PLS} contains many "unweighted" problems. For example, the maximal clique problem with the Flip-neighborhood is in it. The number of steps of the standard local search algorithm is bounded by the number of vertices of the graph. The unweighted set covering problem with each polynomial neighborhood belongs to the class P_{PLS} as well. A nontrivial example of the local search problem from P_{PLS} is the linear programming problem with an arbitrary polynomially searchable neighborhood. It is known that the optimal solution for this problem can be found in polynomial time by the ellipsoid method. Hence, this local search problem belongs to the class P_{PLS} in spite of the fact that the simplex method is not polynomial in the worst case for many well-known pivoting rules. Note that the simplex method is in fact a local search. It moves from one basic feasible solution to another one by exchanging a variable of the basis for another variable outside the basis.

Theorem 8 ([57])**.** *If a* PLS *problem* Π *is* NP*-hard then* NP = co-NP.

This statement shows that it is very unlikely that the class PLS contains an NP-hard problem. Therefore, the local search problems may not be so difficult. In other words, there are no NP-complete problems that can be reduced to a local search problem from the class PLS in polynomial time. Therefore, the complexity of problems in this class is lower than that of NP-complete problems. Note that the conjecture $NP \neq$ co-NP is stronger than the conjecture $P \neq NP$, since the coincidence of the latter two classes implies the coincidence of the former ones.

Definition 4. Let Π_1 and Π_2 be two local search problems. A *PLS-reduction* from Π_1 to Π_2 consists of two polynomial time computable functions h and g such that:

1. h maps instances x of Π_1 to instances $h(x)$ of Π_2.
2. g maps (solution of $h(x)$, x) pairs to solutions of x.
3. For all instances x of Π_1, if s is a local optimum for instance $h(x)$ of Π_2, then $g(s, x)$ is a local optimum for x.

PLS-reductions have the following standard properties.

Propopsition 1. *If* Π_1 PLS*-reduces to* Π_2 *and* Π_2 PLS*-reduces to* Π_3 *then* Π_1 PLS*-reduces to* Π_3*. Moreover*, $\Pi_1 \in P_{PLS}$ *if* $\Pi_2 \in P_{PLS}$.

Propopsition 2. *Let* $\Pi_1 = (OP, N_1)$*,* $\Pi_2 = (OP, N_2)$ *be two local search problems in* PLS *and each local optimum under an* N_2 *neighborhood is a local optimum under an* N_1 *neighborhood. Then* Π_1 PLS*-reduces to* Π_2.

We say that a problem Π in PLS is PLS-*complete* if every problem in PLS can be PLS-reduced to it. We describe below the first PLS-complete problem, which is the basis of further reductions. This problem is called (Circuit, Flip). An instance of this problem is a Boolean circuit x, which consists of AND, OR, and NOT gates. The circuit x has m inputs and n outputs. The set of feasible solutions consists of all the binary strings of length m. The neighborhood Flip(s) of a solution s consists of all the binary strings

of length m whose Hamming distance equals one from s. The objective function F is defined as

$$F(s) = \sum_{j=1}^{n} 2^{j-1} y_j,$$

where y_j is the jth output of the circuit with input s.

Theorem 9 ([58]). *Both the maximization version and the minimization versions of* (Circuit, Flip) *are* PLS-*complete.*

The following local search problems are PLS-complete:

1. Graph partitioning under the following neighborhoods: KL [58], Swap, FM, FM_1 [57]. Given an undirected graph $G = (V, E)$ with $|V| = 2n$ and positive integer weights on its edges, we wish to find a partition of V into two subsets V_1 and V_2 with $|V_1| = |V_2| = n$, such that the sum of the weights of the edges that have one endpoint in V_1 and one endpoint in V_2 is minimal. Swap neighborhood is defined as follows: a partition (V_1, V_2) has as neighbors all the partitions that can be produced by swapping a node in V_1 with a node in V_2.

In the Kernighan – Lin neighborhood we replace the single swap by a well-chosen sequence of n swaps. At each step of the sequence we choose to swap the best pair of nodes among those that have not been used in previous steps of the sequence. By the term *best*, as above, we mean that the swap produces the best improvement of the objective function. The FM neighborhood is defined in a similar way but now each step consists of the two substeps. In the first substep, we examine all the nodes that have not moved since the beginning of the sequence and choose to move the best such node from one side to the other. In the second substep, we move the best node that has not yet been moved from the opposite side. The neighborhood FM_1 contains one neighboring solution only. This solution is obtained after the first step of the FM procedure.

2. Traveling salesman problem under the k-Opt and LK′ neighborhoods. Given a complete undirected graph of n nodes with positive integer weights on its edges, we wish to find the least-weight tour that passes exactly once through each node. Neighboring tours for the k-Opt neighborhood are defined as follows. We delete k edges from the tour in order to obtain k nonconnected paths. Then we reconnect these k paths so that a new tour is produced. The TSP under the k-Opt neighborhood is PLS-complete for large k [59].

The main idea of the Lin – Kernighan neighborhood is as follows. Given a tour, we delete an edge (a, b) and obtain a Hamiltonian path with end nodes a and b. Let a be stable and b variable. If we add an edge (b, c) then a circle is created. There is a unique edge (c, d) that is incident on node c, whose deletion breaks the circle, producing a new Hamiltonian path with a new variable end node d. This procedure is called rotation. We can close a tour by adding an edge between the stable end a and the variable end d. A move from the current tour to a neighboring tour consists of removing an edge, then performing a sequence of "greedy" rotations, and finally the reconnecting of the two ends to form a tour. There are many variations of this main framework depending on how exactly the rotation is chosen in each step, and on the restrictions on edges to enter and leave the tour. A variant is denoted by LK′. The PLS-completeness is shown in [60].

3. Max-Cut problem under the Flip neighborhood [57]. Given an undirected graph $G = (V, E)$ with positive integer weights on its edges, we wish to find a partition of the set V into two not necessarily equal sets V_1, V_2 such that the sum of the weights of the edges that have one end point in V_1 and one end point in V_2 is maximal. The maximization and minimization versions are not equivalent. The minimization version can be solved in polynomial time, whereas the maximization version is NP-hard. In the Flip neighborhood two partitions are neighbors if one can be obtained from other by moving a node to another subset.

4. Max-Sat problem under the Flip neighborhood [57, 61]. The input is a Boolean formula in a conjunctive normal form with a positive integer weight for each clause. A solution is an assignment of 0 or 1 to all variables. We wish to find an assignment such that the sum of the weights of the satisfied clauses is maximized. The restriction of Max-Sat to instances with at most k literals in each clause is called Max-kSat. In the Flip neighborhood two assignments are neighbors if one can be obtained from the other by flipping the value of one variable. This local search problem is PLS-complete even for $k = 2$.

5. Not-all-equal Max-Sat problem under the Flip neighborhood. The input is a set of clauses of the form $(\alpha_1, \ldots, \alpha_K)$, where α_i is either a literal or a constant 0 or 1. Such a clause is satisfied if its elements do not all have the same value. Each clause is assigned a positive integer weight. A set of feasible solutions of the problem consists of all assignments of 0 or 1 to the variables. We wish to find an assignment maximizing the sum of the weights of the satisfied clauses. If we restrict the clauses to have at most k literals then we get the NAE Max-kSat problem. The restriction to instances with no negative literals in their clauses is called Pos NAE Max-Sat. The Flip neighborhood is defined as for Max-Sat. The local search problem Pos NAE Max-3Sat under the Flip neighborhood is PLS-complete [57, 61].

6. Stable configurations neural networks in the Hopfield model. The input is an undirected graph $G = (V, E)$ with a weight on each edge and a threshold t_v for each node v. A configuration assigns to each node v a state $s_v \in \{-1, 1\}$. A node is "happy" if $s_v = 1$ and $\sum_u w_{(uv)} s_u s_v + t_v \geq 0$ or $s_v = -1$ and $\sum_u w_{(uv)} s_u s_v + t_v \leq 0$. A configuration is stable if all the nodes are happy. The problem is to find a stable configuration. We get an optimization problem if we introduce the cost function $\sum_{(uv)\in E} w_{(uv)} s_u s_v + \sum_{v\in V} t_v s_v$. It is known [62] that if a node is unhappy then changing its state will increase the cost. In fact, the stable configurations coincide with the locally optimal solutions with respect to the Flip neighborhood. This local search problem is PLS-complete [57].

Figure 1 shows the sequence of PLS-reductions for the local search problems described above and for two local search problems in facility location. The neighborhoods FM_1, FM, and KL for the p-median problem are defined in a way similar to the corresponding neighborhoods for the graph partitioning problem.

3.2. PLS-*Complete Facility Location Problems*

Theorem 10 ([63])**.** *The local search problem* (UFLP, Flip) *is* PLS-*complete.*

Proof. Let us consider the PLS-complete problem (Max-Cut, Flip). Given a graph $G = (V, E)$ with positive weight w_e on each edge e, find a partition of the set V into two subsets V_1 and V_2 with maximal weight of the cut $W(V_1, V_2)$. We will reduce (Max-Cut, Flip)

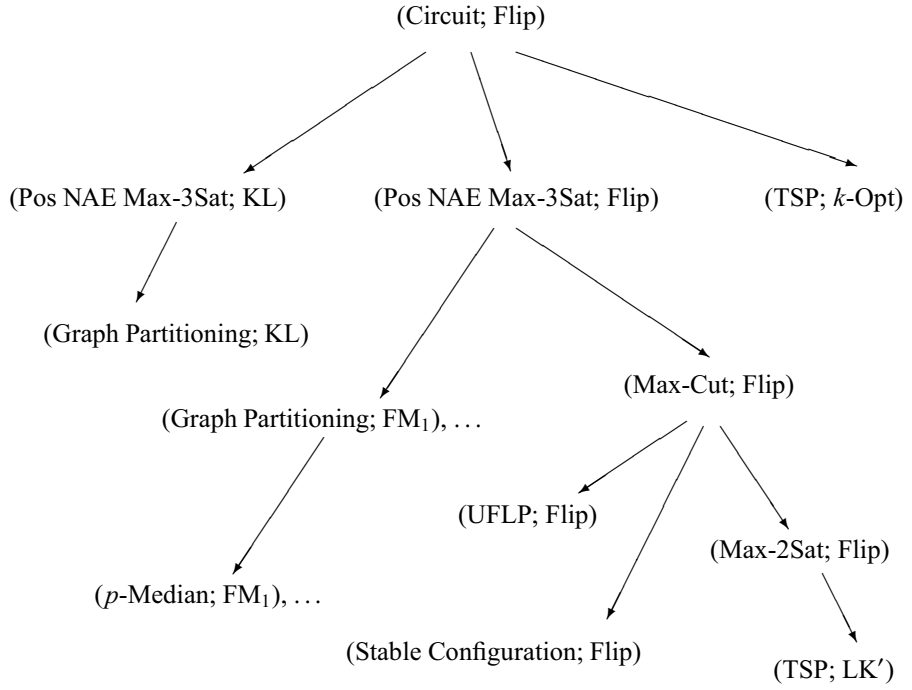

Figure 1. Reductions of the PLS-complete problems

to (UFLP, Flip). To this end we present the functions h, g from Definition 4 with the required properties.

Denote by $E(i)$ the set of edges in G that are incident to the vertex $i \in V$. Put $I = V$, $J = E$ and

$$f_i = \sum_{e \in E(i)} w_e, \quad c_{ie} = \begin{cases} 0 & \text{if } (i = i_1) \text{ or } (i = i_2), \\ 2w_e & \text{otherwise,} \end{cases} \qquad e = (i_1, i_2).$$

For each solution S of the UFLP we define a partition (V_1, V_2) in the following way: $V_1 = S$, $V_2 = V \setminus V_1$. We claim that

$$\sum_{i \in S} f_i + \sum_{j \in J} \min_{i \in S} c_{ij} + W(V_1, V_2) = 2 \sum_{e \in E} w_e.$$

This guarantees the desired properties of the reduction. Let us consider three cases for an edge $e = (i_1, i_2)$.

Case 1: $i_1, i_2 \in V_1$. The weight w_e is included in f_{i_1} and f_{i_2}, $\min_{i \in S} c_{ie} = 0$, and w_e is not included in $W(V_1, V_2)$. Hence, the value w_e is presented twice in both parts of the equation.

Case 2: $i_1, i_2 \in V_2$. The values f_{i_1} and f_{i_2} are not included into the first term, $\min_{i \in S} c_{ie} = 2w_e$, and w_e is not included in $W(V_1, V_2)$.

Case 3: $i_1 \in V_1$, $i_2 \in V_2$. The weight w_e is included in f_{i_1} and $W(V_1, V_2)$ but $\min_{i \in S} c_{ie} = 0$.

Thus, we get the desired result. □

Theorem 11 ([52]). *The local search problem* (p-*median*, FM_1) *is* PLS-*complete.*

Proof. Let us consider the PLS-complete problem (Graph Partitioning, FM_1). Given an undirected graph $G = (V, E)$ with even number of vertices and positive weight w_e on each edge e, find a partition of the set V into two subsets V_1, V_2 with the same cardinalities and a maximal weight of the cut $W(V_1, V_2)$. We reduce this problem under the FM_1 neighborhood to (p-median, FM_1). Put

$$W_i = \sum_{e \in E(i)} w_e, \quad W = \sum_{e \in E} w_e,$$

$$I = \{1, \ldots, |V|\}, \quad J = \{1, \ldots, |E| + |V|\}, \quad p = |V|/2.$$

For each index $j = 1, \ldots, |E|$ we assign the edge $e \in E$ and put

$$c_{ij} = \begin{cases} 0 & \text{if } (i = i_1) \text{ or } (i = i_2), \\ 2w_e & \text{otherwise,} \end{cases} \qquad e = (i_1, i_2).$$

For each $j = |E| + 1, \ldots, |E| + |V|$ we define

$$c_{ij} = \begin{cases} 0 & \text{if } i = j - |E|, \\ W - W_i & \text{otherwise.} \end{cases}$$

For the partition (V_1, V_2) we put $S = V_1$. The proof of the theorem is based on the following equality:

$$\sum_{j \in J} \min_{i \in S} c_{ij} + W(V_1, V_2) = pW.$$

By definition we have

$$\sum_{j=1}^{|E|} \min_{i \in S} c_{ij} = 2 \sum (w_e \mid e = (i_1, i_2), i_1, i_2 \notin S)$$

and

$$\sum_{j=1+|E|}^{|J|} \min_{i \in S} c_{ij} = \sum_{i \notin S} (W - W_i) = pW - \sum_{i \notin S} W_i.$$

Note that

$$\sum_{i \notin S} W_i = W(V_1, V_2) + \sum_{j=1}^{|E|} \min_{i \in S} c_{ij},$$

which completes the proof. □

3.3. Complexity of the Standard Local Search Algorithm

As we have mentioned above, there are polynomially solvable local search problems in the class PLS. However, we still do not know of polynomial time algorithms for PLS-complete problems. Whether such algorithms exist or not is still an open question for further research. Below we will observe that the standard local search algorithm is not appropriate since it takes, in the worst case, an exponential number of iterations to reach a local optimum for every pivoting rule. In other words, we need a fresh idea (new framework, new "ellipsoid" method) to show that PLS = P_{PLS}, if it is true.

Definition 5. Let Π be a local search problem and x be an instance of Π. *The transition graph* $TG_\Pi(x)$ of the instance x is a directed graph with one node for each feasible solution of x and with an arc $(s \to t)$ whenever $t \in N(s, x)$ and $F(t, x) < F(s, x)$. The *height of a node* v is the length of the shortest path in $TG_\Pi(x)$ from v to a sink (a vertex with no outgoing arcs). The *height of* $TG_\Pi(x)$ is the largest height of a node.

The height of a node v is a lower bound on the number of iterations needed by the standard local search algorithm even if it uses the best possible pivoting rule.

Definition 6. Suppose Π_1 and Π_2 are problems in PLS and let (h, g) be a PLS-reduction from Π_1 to Π_2. This reduction is *tight* if for every instance x of Π_1 we can choose a subset R of feasible solutions for the image instance $h(x)$ so that the following properties are satisfied:

1. R contains all local optima of $h(x)$.
2. For every solution t of x we can construct in polynomial time a solution $q \in R$ of $h(x)$ such that $g(q, x) = t$.
3. Suppose that the transition graph of $h(x)$, $TG_{\Pi_2}(h(x))$, contains an arc from $q \in R$ to $q' \in R$ or a directed path from q to q' such that all internal path nodes are outside R, and let $t = g(q, x)$ and $t' = g(q', x)$ be the corresponding solutions of x. Then either $t = t'$ or $TG_{\Pi_1}(x)$ contains an arc from t to t'.

Tight reductions allow us to transfer lower bounds on the running time of the standard local search algorithm from one problem to another. Thus, if the standard local search algorithm of Π_1 takes exponential time in the worst case, then so does the standard algorithm for Π_2.

All PLS-complete problems that we have referred to are complete under tight PLS reductions. We now want to show that in the worst case the running time of the standard local search algorithm is exponential for the tightly PLS-complete problems. To prove this it suffices to find a local search problem in the class PLS which has this property.

Lemma 2 ([64])**.** *There is a local search problem in* PLS *whose standard local search algorithm takes exponential time.*

Proof. Consider the following artificial minimization problem. For every instance x of size n, the solution set consists of all n-bit integers $0, \ldots, 2^n - 1$. For each solution i, its cost is i, and if $i > 0$ it has one neighbor, $i - 1$. Thus, there is a unique local and global minimum, namely 0, and the transition graph is a path from $2^n - 1$ down to 0. The local search algorithm starting at $2^n - 1$ will follow this path. □

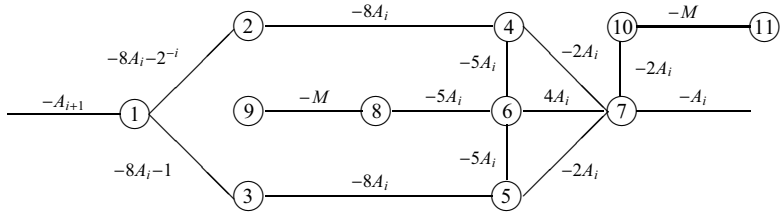

Figure 2. Module i: $A_i = 20^{i-1}$

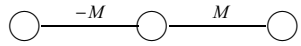

Figure 3. Chain

Theorem 12. *The standard local search algorithm takes exponential time in the worst case for the following problems, regardless of the tie-breaking and pivoting rules used*:

- *Graph partitioning under the neighborhoods* KL [58], Swap, FM, FM_1 [57].
- *Traveling salesman problem under the k*-Opt *neighborhood for some constant k* [59], *and the* LK′ *neighborhood* [60].
- Max-Cut, Max-2Sat *and* Pos NAE Max-3Sat *under the* Flip *neighborhood* [57, 61].
- *Stable configuration for neural networks* [57].
- *Uncapacitated facility location problem under the* Flip *neighborhood* [63].
- *p-median problem under the* FM_1, Swap, KL, FM *neighborhoods* [65].

It is interesting to see a family of nonartificial instances and initial solutions for which the local search algorithm takes an exponential number of iterations to find a local optimum. In order to do so we consider the Generalized Graph 2-Coloring problem (2-GGCP) with the Flip-neighborhood: given an undirected graph $G = (V, E)$ with integer weight w_e on each edge e, find a color assignment $c\colon V \to \{1, 2\}$ of the vertices that minimizes the total weight of the monochromatic edges. For each solution $c(V)$, a Flip neighbor is obtained by choosing a vertex and assigning a new color. A solution is Flip-optimal if the flipping of every single vertex does not decrease the total weight of monochromatic edges. The local search problem (2-GGCP, Flip) is tightly PLS-complete as a reformulation of the Max-Cut problem under the Flip neighborhood.

To illustrate the exponential number of iterations needed for finding a Flip-optimal solution, we present an example of a graph and an initial solution for 2-GGCP for which the *best improvement*, meaning always flipping the best vertex, takes an exponential number of iterations. This graph consists of K modules with weights on the edges as shown in Figure 2 for $i = 1, \ldots, K$ and a chain of three additional vertices as shown in Figure 3. Vertex 1 is called the input node and vertex 7 is called the output node of a module. The input node of module i is adjacent to the output node of module $i+1$, for $i = K-1, \ldots, 1$, and the input node of module K is adjacent to the rightmost vertex of the chain of Figure 3. The output node of module 1 is only adjacent to vertices 4, 5, 6, and 10 of this module. An edge of weight $-M$, where M is some large positive value, makes sure that the two vertices incident to this edge have the same color.

Let us consider a starting solution with vertices of the same color. In this case, only flipping the rightmost vertex of the chain yields an improvement. This flip results in a solution in which the input node of module K is "unhappy."

Theorem 13 ([66]). *If the input node of module K is the only unhappy node, then the output node of module* 1 *flips* 2^K *times.*

3.4. Average Case Behavior

The worst-case analysis yields bad predictions about local search and metaheuristics such as Genetic algorithms, Variable Neighborhood Search, Ant Colony algorithms, and others [67–69]. However, computational experiments show a lot of positive results for local search. In practice, it finds a local optimum quickly, with a small number of iterations as compared to the size of the search space. It is interesting to understand why the local search methods are so successfully used in applications. Is the standard local search algorithm polynomial on average? The first results in this direction have been obtained by C. Tovey [70, 71].

Consider the problem of maximizing a real valued function F defined on the vertices of the n-dimensional 0-1 hypercube. We assume that the values of F are distinct. In the Flip neighborhood for the hypercube, two vertices are neighbors if they differ in exactly one component. For each F we can construct an *ordering*: a list of the vertices from best to worst function value. The random distribution we consider is such that all orderings are equally likely to occur.

Theorem 14 ([72]). *Under the assumption that all orderings are equally likely, the expected number of iterations of the standard local search algorithm with the* Flip *neighborhood and every pivoting rule is less than* $\frac{3}{2}en$*, where e is the logarithmic constant.*

Note, that the statement holds for every kind of pivoting rule. Even a careful rule such as picking the worst better neighbor has an expected performance of less than $\frac{3}{2}en$.

Theorem 15 ([72]). *Suppose the ratio of probabilities of occurrence satisfies*

$$\frac{\text{Prob}[v]}{\text{Prob}[v']} \leq 2^{\alpha n}$$

for all orderings v, v' *and a positive constant* α*. Then the expected number of iterations of every local search algorithm with the* Flip *neighborhood is less than* $(\alpha + 2)en$.

This statement can be extended for more powerful neighborhoods.

Theorem 16 ([72]). *Suppose the vertices of the hypercube are assigned neighbors in such a way that every vertex has at most* $q(n)$ *neighbors, where* $q(n) \geq n$ *is a polynomial. Then for every probability distribution satisfying*

$$\frac{\text{Prob}[v]}{\text{Prob}[v']} \leq 2^{\alpha n} \quad \textit{for all orderings } v, v',$$

the expected number of iterations of every local search algorithm is less than $e(\alpha+2)q(n)$.

Other results for more powerful neighborhoods and other random distributions can be found in [72]. It is not clear how to use these results for the facility location models. Can we get a polynomial upper bound for the running time of the standard local search algorithm on average for (UFLP, Flip), (p-median, Swap), and others? This is an open question for further research.

3.5. Approximate Local Search

Computational studies of local search algorithms and metaheuristics have been extensively reported in the literature for various NP-hard optimization problems. Empirically, local search heuristics appear to converge very quickly, in low-order polynomial time. The standard local search algorithm terminates in a pseudo polynomial number of iterations, but polynomial-time algorithms for finding a local optimum are not known in general. It is interesting to explore the possibility of identifying *approximately locally optimal* solutions in polynomial time. We say that a feasible solution s^ε to an instance of a combinatorial optimization problem OP with neighborhood function N is an *ε-local minimum* if

$$F(s^\varepsilon) \le (1+\varepsilon)F(s) \quad \text{for all } s \in N(s^\varepsilon) \text{ and some positive } \varepsilon.$$

Hence, while s^ε is not necessarily a local minimum, it *almost* is. A family of algorithms $(A_\varepsilon)_{\varepsilon>0}$ for the local search problem is an *ε-local optimization scheme* if A_ε produces an ε-local optimum. If the running time of algorithm A_ε is polynomial in the input size and $1/\varepsilon$, it is called a *fully polynomial-time ε-local optimization scheme*. In [73] it is shown that every local search problem in the class PLS with a linear objective function has a fully polynomial-time ε-local optimization scheme. In particular, an ε-locally optimal solution can be computed in polynomial time for the PLS-complete problems mentioned above.

Let us consider a linear combinatorial optimization problem OP, where the set Sol of feasible solutions is a family of subsets of a finite ground set $E = \{1, \dots, n\}$. The objective function $F\colon 2^E \to Q_+$ assigns a nonnegative cost to every feasible solution $s \in$ Sol through $F(s) = \sum_{e\in s} f_e$. Note that we focus on a linear combinatorial optimization problem as opposed to a *general* combinatorial optimization problem. The class of problems we are looking at is equivalent to that of 0-1 integer linear programming problems. The UFLP, MSUFLP and UFLPUP belong to this class.

The algorithm starts with a feasible solution s^0. We then alter the element costs f_e for $e \in E$ according to a prescribed scaling rule to generate a modified instance. Using local search on this modified problem, we look for a solution with an objective function value (with respect to the original cost) that is half that of $F(s^0)$. If no solution is found then we are at a local optimum for the modified problem and output this solution. Otherwise, we replace s^0 by the solution, which has a cost of at most $0.5\,F(s^0)$, and the algorithm is repeated. All details of the algorithm are described above. Note that the modification of the cost coefficients in Step 2 merely amounts to rounding them up to the closest integer multiple of q.

Theorem 17 ([73])**.** *Algorithm ε-Local Search produces an ε-local minimum and its running time is polynomial in the input size and $1/\varepsilon$.*

Proof. Let s^ε be the solution produced by the algorithm and $s \in N(s^\varepsilon)$. Note that

$$F(s^\varepsilon) = \sum_{e\in s^\varepsilon} f_e \le \sum_{e\in s^\varepsilon} \left\lceil \frac{f_e}{q} \right\rceil q \le \sum_{e\in s} \left\lceil \frac{f_e}{q} \right\rceil q \le \sum_{e\in s} q\left(\frac{f_e}{q} + 1\right) \le \sum_{e\in s} f_e + nq = F(s) + nq.$$

If $F(s) \ge K/2$ then

1: Find $s^0 \in \mathrm{Sol}(x)$ and put $i := 0$
2: Put $K := F(s^i)$, $q := \varepsilon K/(2n(1+\varepsilon))$, $f'_e := \lceil f_e/q \rceil q$, $e \in E$.
3: Put $j := 0$ and $s^{ij} := s^i$.
4: **repeat**
5: **if** s^{ij} is a local minimum **then**
6: $s^\varepsilon := s^{ij}$, STOP
7: **else**
8: select better solution $s^{ij+1} \in N(s^{ij})$, $F(s^{ij+1}) < F(s^{ij})$ and put $j := j + 1$
9: **until** $F(s^{ij}) \le K/2$
10: Put $s^{i+1} := s^{ij}$, $i := i + 1$ and goto 2.

Algorithm 1. Algorithm ε-Local Search

$$\frac{F(s^\varepsilon) - F(s)}{F(s)} \le \frac{nq}{F(s)} \le \frac{nq}{F(s^\varepsilon) - nq} \le \frac{2nq}{K - 2nq} = \varepsilon.$$

Let us analyze the running time. Step 1 is polynomial because the local search problem is in the class PLS. In each improvement move in Step 4 we get an improvement of at least of q units. Thus the number of local search iterations in Step 4 is $O(n(1+\varepsilon)/\varepsilon) = O(n/\varepsilon)$. Step 2 is executed at most $\log F(s^0)$ times. Thus, the total number of local search iterations is $O(n \log F(s^0)/\varepsilon)$. More accurate calculations give $O(n^2\varepsilon^{-1} \log n)$ iterations. □

If we replace the relative error in the definition of an ε-local optimum with the absolute error, then the existence of a polynomial ε-Local Search algorithm will imply $\mathrm{PLS} = \mathrm{P_{PLS}}$.

Theorem 18 ([73]). *If there is an algorithm that for every instance x of a* PLS*-complete local search problem* (OP, N) *finds a feasible solution s^ε in polynomial time, such that*

$$F(s^\varepsilon) \le F(s) + \varepsilon \quad \textit{for all } s \in N(s^\varepsilon)$$

for some fixed positive ε, then $\mathrm{PLS} = \mathrm{P_{PLS}}$.

Proof. Recall that the objective function is an integer-value. For each instance x we create a new instance x' with the same set of feasible solutions, $\mathrm{Sol}(x') = \mathrm{Sol}(x)$, and a new objective function defined by

$$F'(s) = \sum_{e \in s} f'_e, \quad \text{for all } s \in \mathrm{Sol}(x'), \text{ where } f'_e = f_e(1+\varepsilon) \text{ for every } e \in E.$$

We apply the algorithm to the new instance x' and let s' be the resulting solution. Then, $F'(s') - F'(s) \le \varepsilon$ for all $s \in N(s')$. Thus, $F(s') - F(s) \le \varepsilon/(\varepsilon+1) < 1$ for all $s \in N(s')$ and s' is a local optimum for x. □

Theorem 19 ([73]). *If a* PLS*-complete local search problem* (OP, N) *has a fully polynomial time ε-local optimization scheme $(A_\varepsilon)_{\varepsilon>0}$ such that the actual running time of A_ε is polynomial in the input size and* $\log 1/\varepsilon$, *then* $\mathrm{PLS} = \mathrm{P_{PLS}}$.

Proof. Choose $\varepsilon = 1/(nf_{\max} + 1)$, $f_{\max} = \max_{e \in E} f_e$ and apply A_ε. Note that its running time is polynomial in the input size. If s^ε is the solution returned by the algorithm, then $F(s^\varepsilon) \le (1 + \varepsilon)F(s) < F(s) + 1$ for all $s \in N(s^\varepsilon)$. Hence, s^ε is a local optimum. □

Observe that the results hold for the facility location problems as well.

4. The Quality of Local Optima

We say that a neighborhood is exact if each local optimum is a global one. The standard local search algorithm with an exact neighborhood produces the optimal solution of the problem. However, for the p-median problem (the traveling salesman problem [74] and some others) the existence of the polynomially searchable exact neighborhoods implies P = NP [65]. In general, this property can be presented in the following way.

An optimization problem OP with optimal value $F^*(x)$ is called *pseudo polynomially bounded* if there is a polynomial q in the length of $x \in \mathcal{I}$ and $\mathrm{Max}(x)$, which is the maximal absolute value of the components of x, such that

$$|F(s) - F^*(x)| \le q(|x|, \mathrm{Max}(x))$$

for all $x \in \mathcal{I}$ and for all $s \in Sol(x)$. The set of the pseudo polynomially bounded problems is denoted by NPO_B.

Theorem 20 ([64, 65])**.** *Let* OP $\in \mathrm{NPO}_B$ *and* $(\mathrm{OP}, N) \in$ PLS. *If the approximation of* OP *within a factor* ε *is strongly* NP*-hard then* N *cannot guarantee a ratio of* ε *unless* P = NP.

For $\varepsilon = 1$ we have the following.

Corollary 1. *If* OP $\in \mathrm{NPO}_B$, $(\mathrm{OP}, N) \in$ PLS *and* OP *is strongly* NP*-hard then* N *cannot be exact unless* P = NP.

Theorem 21 ([64])**.** *If* $(\mathrm{OP}, N) \in$ PLS *and the approximation of* OP *within a factor* ε *is* NP*-hard then* N *cannot guarantee a ratio of* ε *unless* NP = co-NP.

Corollary 2. *If* $(\mathrm{OP}, N) \in$ PLS *and* OP *is* NP*-hard, then* N *cannot be exact unless* NP = co-NP.

As we have mentioned in Section 1, the uncapacitated facility location problem is NP-hard in the strong sense even for the metric case. Moreover, the existence of a polynomial time 1.463-factor approximation algorithm for this problem implies P = NP. Therefore, it is difficult to find an exact polynomially searchable neighborhood for this problem or a neighborhood that guarantees the ratio $\varepsilon \le 1.463$. This is bad news. However, below we will present some good news.

We say that an optimization problem OP is *polynomially bounded* if there exists a polynomial r such that $F(s) \le r(|x|)$ for every instance x of OP and every feasible solution s of x.

Definition 7. An optimization problem OP has *guaranteed local optima* if there exists a polynomial time searchable neighborhood N and a constant k such that $F(s) \le kF^*(x)$ for every instance x of OP and every local minimum s of x with respect to N.

Definition 8. For an instance x and feasible solutions s, s' of x we say that s' is an *h-bounded neighbor of s* if the Hamming distance between s and s' is at most h. A neighborhood N is said to be *h-bounded* if there exists a constant h such that every neighbor of s is an h-bounded for every feasible solution s and every instance x.

Definition 9. Let OP be a polynomially bounded optimization problem. We say that OP *belongs to the class Guaranteed Local Optima* (GLO) if the following two conditions hold:

- at least one feasible solution s of x can be computed in polynomial time for every instance x of OP,
- there exists a constant h such that OP has a guaranteed local optima with respect to a suitable h-bounded neighborhood.

Definition 10. Let A and B be two optimization problems. A is said to be PTAS-*reducible* to B (in symbol $A \leq_{\text{PTAS}} B$) if three functions f, g, c exist such that:

- for every $x \in \mathcal{I}_A$ and for every $\varepsilon \in (0,1)_Q$, (Q is the set of rational numbers) $f(x,\varepsilon) \in \mathcal{I}_B$ is computable in polynomial time with respect to $|x|$;
- for every $x \in \mathcal{I}_A$, for every $s \in \text{Sol}_B(f(x,\varepsilon))$, and for every $\varepsilon \in (0,1)_Q$, $g(x,s,\varepsilon) \in \text{Sol}_A(x)$ is computable in time polynomial with respect to both $|x|$ and $|s|$;
- $c\colon (0,1)_Q \to (0,1)_Q$ is computable and surjective;
- for every $x \in \mathcal{I}_A$, for every $s \in \text{Sol}_B(f(x,\varepsilon))$, and for every $\varepsilon \in (0,1)_Q$ $E_B(f(x,\varepsilon),s) \leq c(\varepsilon)$ implies $E_A(x, g(x,s,\varepsilon)) \leq \varepsilon$, where $E(x,s)$ is the relative error of s for x,

$$E(x,s) = \frac{|F(s) - F^*(x)|}{\max\{F(s), F^*(x)\}}.$$

Suppose that $A \leq_{PTAS} B$ and $B \in \text{APX}$, then $A \in \text{APX}$. If C is a class of optimization problems, then by $\overline{C}$ we denote the *closure* of C under PTAS reductions, that is, the set of problems defined by

$$\overline{C} = \{A \mid \exists B \in C \text{ such that } A \leq_{\text{PTAS}} B\}.$$

Theorem 22 ([75]). $\overline{\text{GLO}} = \text{APX}$.

In other words, "*... the basis of approximability of a large class problems stands an important combinatorial property, namely, the fact that all local optima have guaranteed quality with respect to global optima*" G. Ausiello, M. Protasi [75].

5. Computationally Difficult Instances

The iterative local search methods show high performance for many combinatorial optimization problems in business, engineering, and science. These metaheuristics provide fast and robust tools, producing high quality solutions for location problems as well [76–79]. As a rule, they deal with the set of local optima under polynomially searchable neighborhoods. If the local optima cluster is in a small part of the feasible domain, as for the metric TSP [80], we understand why these heuristics are so effective. Conversely,

if we wish to generate computationally difficult instances for the local search methods then we may try to create instances, where local optima are scattered all around the feasible domain. Such computationally difficult instances for the UFLP based on binary perfect codes, finite projective planes, and others [63] are described below.

5.1. Polynomially Solvable Instances

Let us consider a finite projective plane of order k [81], which is a collection of $n = k^2 + k + 1$ points $x_1, \dots x_n$ and lines $L_1, \dots, L_n$. An incidence matrix A is an $n \times n$ matrix defining the following: $a_{ij} = 1$ if $x_j \in L_i$ and $a_{ij} = 0$ otherwise. The incidence matrix A satisfying the following properties:

1. A has constant row sum $k + 1$;
2. A has constant column sum $k + 1$;
3. the inner product of every pair of distinct rows of A is 1;
4. the inner product of every pair of distinct columns of A is 1.

These matrices exist if k is a power of a prime. A set of lines $B_j = \{L_i \mid x_j \in L_i\}$ is called a bundle for the point x_j. The cardinality of each bundle is $k+1$ and $|B_{j1} \cap B_{j2}| = 1$ for every pair of different points x_{j1} and x_{j2}. Let us define a class of instances for the UFLP. Put $I = J = \{1, \dots, n\}$ and

$$c_{ij} = \begin{cases} \xi_{ij} & \text{if } a_{ij} = 1, \\ +\infty & \text{otherwise,} \end{cases} \qquad f_i = f \text{ for all } i \in I, \text{ where } f > \sum_{i \in I} \sum_{j \in J} \xi_{ij}.$$

We denote this class by FPP_k. It is easy to see that the optimal solution for FPP_k corresponds to a bundle. Hence, the problem can be solved in polynomial time.

Every bundle corresponds to a strong local optimum of the UFLP under the neighborhood Flip ∪ Swap. Global optimum is one of them. The Hamming distance for an arbitrary pair of the strong local optima equals $2k$. Hence, the diameter of the area, where local optima are located, is quite large. Moreover, there are no other local optima with distance to the bundle less than or equal to k. As we will see in some computational results, the local optima have large basins of attraction. For metaheuristics it is an additional obstacle for moving from one local optimum to another. In Tabu Search we have to use a large tabu list. For Simulated Annealing we need high temperatures. If the population in Genetic Algorithm is a collection of the bundles then the crossover operators produce "bad" local optima or the same bundles. For the GRASP heuristic this class is difficult too [79].

5.2. Instances with Exponential Number of Strong Local Optima

Let us consider two classes of instances, where the number of strong local optima grows exponentially as dimension increases. The first class uses binary perfect codes with code distance 3. The second class involves a chess board.

Let B_k be a set of words (or vectors) of length k over an alphabet $\{0,1\}$. A binary code of length k is an arbitrary nonempty subset of B_k. A binary perfect code C with distance 3 is a subset of B_k with property $|C| = 2^k/(k+1)$ such that the Hamming distance

$d(c_1, c_2)$ for all $c_1, c_2 \in C$ is at least 3 whenever $c_1 \neq c_2$. These codes exist for $k = 2^r - 1$ and $r > 1$, integer. Put $n = 2^k, I = J = \{1, \ldots, n\}$. Every element $i \in I$ corresponds to a vertex $x(i)$ of the binary hypercube $\mathbb{Z}_2^k$. Therefore, we may use a distance $d_{ij} = d(x(i), x(j))$ for every pair of elements $i, j \in I$. Now we define

$$c_{ij} = \begin{cases} \xi_{ij} & \text{if } d(x(i), x(j)) \leq 1, \\ +\infty & \text{otherwise,} \end{cases} \qquad f_i = f \text{ for all } i \in I.$$

An arbitrary perfect code C produces a partition of $\mathbb{Z}_2^k$ into $2^k/(k+1)$ disjointed spheres of radius 1 and corresponds to a strong local optimum for the UFLP. The number of perfect codes $\aleph(k)$ grows exponentially as k increases. The best known lower bound [82] is

$$\aleph(k) \geq 2^{2^{((k+1)/2)\log_2(k+1)}} \times 3^{2^{(k-3)/4}} \times 2^{2^{((k+5)/4)\log_2(k+1)}}.$$

The minimal distance between two perfect codes or strong local minima is at least $2^{(k+1)/2}$. We denote the class of benchmarks by BPC_k.

Let us glue boundaries of the $3k \times 3k$ chess board so that we get a torus. Put $r = 3k$. Each cell of the torus has 8 neighboring cells. For example, the cell $(1, 1)$ has the following neighbors: $(1, 2), (1, r), (2, 1), (2, 2), (2, r), (r, 1), (r, 2), (r, r)$. Define $n = 9k^2$, $I = J = \{1, \ldots, n\}$ and

$$c_{ij} = \begin{cases} \xi_{ij} & \text{if the cells } i, j \text{ are neighbors,} \\ +\infty & \text{otherwise,} \end{cases} \qquad f_i = f \text{ for all } i \in I.$$

The torus is divided into k^2 squares with 9 cells in each of them. Every cover of the torus by k^2 squares corresponds to a strong local optimum for the UFLP. The total number of these strong local optima is $2 \times 3^{k+1} - 9$. The minimal distance between them is $2k$. We denote this class of benchmarks by CB_k.

5.3. Instances with Large Integrability Gap

As we will see later, the integrality gap for the described classes is quite small. Therefore, the branch and bound algorithm finds an optimal solution and proves the optimality quickly. It is interesting to design benchmarks, which are computationally difficult for both metaheuristics and branch and bound methods.

As in previous cases, let the $n \times n$ matrix (c_{ij}) have the following property: each row and column have the same number of finite elements. We denote this number by l. The value l/n is called the density of the matrix. Now we present an algorithm to generate random matrices (c_{ij}) with the fixed density.

The array Column[j] keeps the number of small elements in the jth column of the generating matrix. Variable l_0 is used to count the columns, where small elements must be located in the ith row. These columns are detected in advance (line 7) and removed from the set J (line 11). Note that we may get random matrices with exactly l small elements for each row only if we remove lines 6 – 11 from the algorithm. By transposing we get random matrices with this property for columns only. Now we introduce three classes of benchmarks:

Gap-A: each column of c_{ij} has exactly l small elements;

```
1: J ← {1, ... n}
2: Column [j] ← 0 for all j ∈ J
3: c[i, j] ← +∞ for all i, j ∈ J
4: for i ← 1 to n do
5:     l_0 ← 0
6:     for j ← 1 to n do
7:         if n − i + 1 = l − Column[j] then
8:             c[i, j] ← ξ[i, j]
9:             l_0 ← l_0 + 1
10:            Column[j] ← Column[j] + 1
11:            J ← J \ j
12:    select a subset J' ⊂ J, |J'| = l − l_0 at random and put c[i, j] ← ξ[i, j], for j ∈ J'.
```

Algorithm 2. Random matrix generator (l, n)

Gap-B: each row of c_{ij} has exactly l small elements;

Gap-C: each column and row of c_{ij} have exactly l small elements.

For these classes we save $I = J = \{1, \ldots, n\}$ and $f_i = f$ for all $i \in I$. The instances have a significant integrality gap $\delta = 100\%(F^* - F_{\mathrm{LP}})/F^*$, where F_{LP} is an optimal value for the linear programming relaxation. For $l = 10$, $n = 100$ we observe that $\delta \in [21\%, 29\%]$. As a consequence, the branch and bound algorithm evaluates about 0.5×10^9 nodes in the branching tree for most of the instances from the class Gap-C.

5.4. Computational Experiments

To study the behavior of metaheuristics, we generate 30 random test instances for each class. The values of ξ_{ij} are taken from the set $\{0, 1, 2, 3, 4\}$ at random and $f = 3000$. All instances are available at http://www.math.nsc.ru/AP/benchmarks/english.html. Optimal solutions are found by the branch and bound algorithm. Table 1 shows the performance of the algorithm on average. Column *Running time* presents the execution times on a PC Pentium 1200 MHz, RAM 128 Mb. Column *Iterations B&B* shows the total number of iterations or number of evaluated nodes in the branching tree. Column *The best iteration* shows iterations for which optimal solutions were discovered. For comparison we include two well known classes:

Uniform: values c_{ij} are selected in the interval $[0, 10^4]$ at random with uniform distribution and independently from each other.

Euclidean: values c_{ij} are Euclidean distances between points i and j in two-dimensional space. The points are selected in a square of size 7000×7000 at random with uniform distribution and independently from each other.

For these classes $f = 3000$. The interval and size of the square are taken in such a way that optimal solutions have the same cardinality as in the previous classes. Table 1 confirms that classes Gap-A, Gap-B, and Gap-C have a large integrality gap and they are the most difficult for the branch and bound algorithm. The classes BPC_7, CB_4, and Euclidean have a small integrality gap. Nevertheless, the classes BPC_7 and CB_4 are more

Table 1. Performance of the branch and bound algorithm in average

Benchmarks classes	n	Gap δ	Iterations B&B	The best iteration	Running time
BPC_7	128	0.1	374 264	371 646	00:00:15
CB_4	144	0.1	138 674	136 236	00:00:06
FPP_{11}	133	7.5	6 656 713	6 635 295	00:05:20
Gap-A	100	25.6	10 105 775	3 280 342	00:04:52
Gap-B	100	21.1	30 202 621	14 656 960	00:12:24
Gap-C	100	28.4	541 320 830	323 594 521	01:42:51
Uniform	100	4.7	9 834	2 748	<00:00:01
Euclidean	100	0.1	1 084	552	<00:00:01

Table 2. Attributes of the local optima allocation

Benchmarks classes	N	Diameter	Radius			R_{100}	R^*
			min	ave	max		
BPC_7	8868	55	1	3	357	24	52
CB_4	8009	50	1	13	178	78	53
FPP_{11}	8987	51	1	2	8	3	1
Gap-A	6022	36	1	53	291	199	7
Gap-B	8131	42	1	18	164	98	16
Gap-C	8465	41	1	14	229	134	21
Uniform	1018	33	1	31	101	61	1
Euclidean	40	21	11	13	18	—	10

difficult than the Euclidean class. This has a simple explanation: classes BPC_7 and CB_4 have many strong local optima with small waste over the global optimum.

In order to understand the difference between classes from the point of view of local optima allocation, we produce the following computational experiment. For 9000 random starting points we apply the standard local improvement algorithm with the Flip ∪ Swap neighborhood and get a set of local optima, some of which are identical. Impressive differences between benchmark classes become clear when the cardinality of the local optima sets are compared. Classes Uniform and Euclidean have small pathological cardinalities of local optima sets and, as we will see below, these classes are very easy for metaheuristics. In Table 2 the column N shows the cardinalities for typical instances in each class. The column *Diameter* yields a lower bound for the diameter of area, where local optima are located. This value equals the maximal mutual Hamming distance over all pairs of local optima obtained.

Figures 4 – 11 plot the costs of local optima against their distances from the global optimum. For every local optimum we draw a sphere. The center of the sphere has coordinates (x, y), where x is the distance, and y is the value of the objective function. The radius of the sphere is the number of local optima, which are located near this local optimum. More precisely, the Hamming distance d is less than or equal to 10. In Table 2 columns *min*, *ave*, and *max* show the minimal, average, and maximal radiuses for the corresponding sets of local optima. The class FPP_{11} has an extremely small maximal and average value of the radiuses. Hence, the basins of attraction for the local optima are quite large. In Figure 6 the local optima, which correspond to the bundles, are shown by two lower

spheres. The distance between them is 22. One of them is the global optimum. All other local optima are located quite far from global optimum and have higher values of the objective function. The distance from global optimum to the nearest local optimum is 12. Column R^* in Table 2 gives the radius of the sphere for the global optimum. It equals 1 for the classes FPP_{11} and Uniform. The maximal values 53 and 52 belong to classes CB_4 and BPC_7. Note that for all classes the sphere of global optima is not the maximal or minimal one. It seems that there is no correlation between the density of local optima in the feasible domain and the objective function. Column R_{100} shows the minimal radius for the 100 biggest spheres. This column indicates that the classes Gap-A, Gap-B, Gap-C have many large spheres. Class FPP_{11} has spheres that are not so large. In Figures 4, 5, 9, and 10 there are many small spheres quite far from the global optimum. If we imagine the set of local optima as a galaxy then we observe many small spheres at the border of the galaxy and a lot of large spheres at the center. The region of high concentration of local optima consists of the spheres with high and low values of the objective function. The global optimum can be located at the center part of the galaxy and has a large sphere (see Figures 4, 5) or very far from the center and has a small sphere (Figures 6, 10). We cannot predict its radius or place in the galaxy. We can only observe that easy classes have a

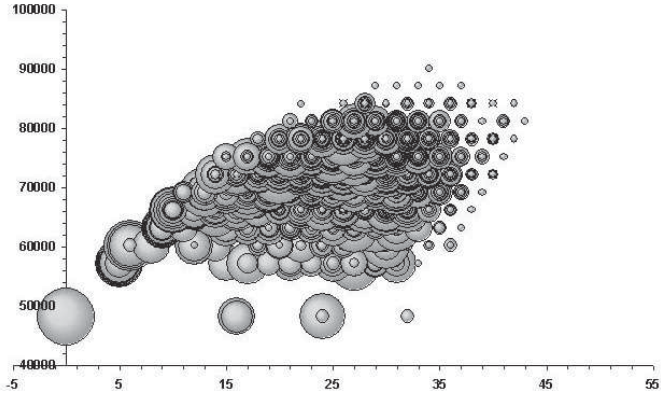

Figure 4. Analysis of local optima for the class BPC_7

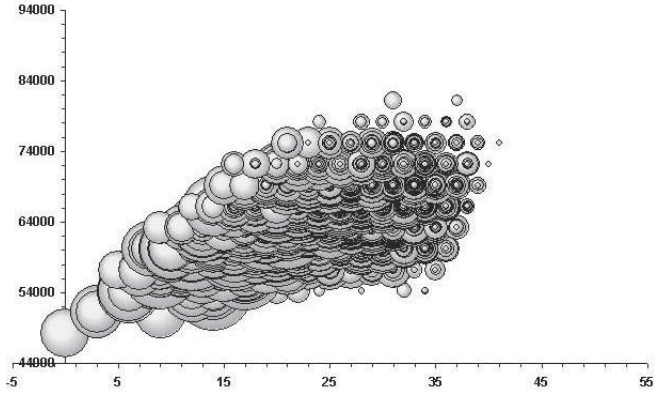

Figure 5. Analysis of local optima for the class CB_4

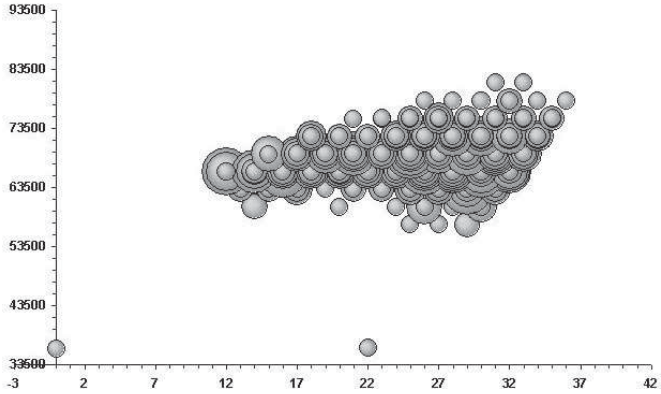

Figure 6. Analysis of local optima for the class FPP_{11}

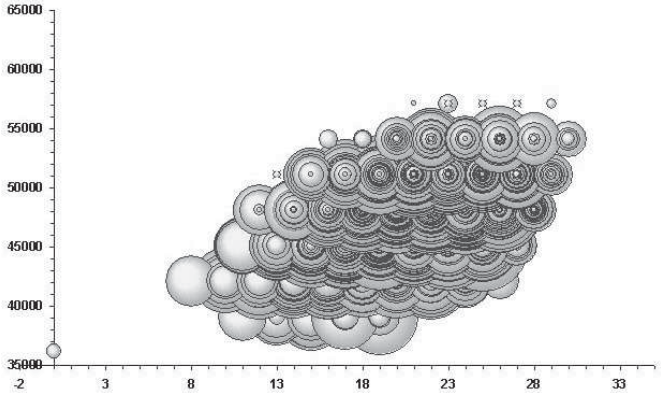

Figure 7. Analysis of local optima for the class GAP-A

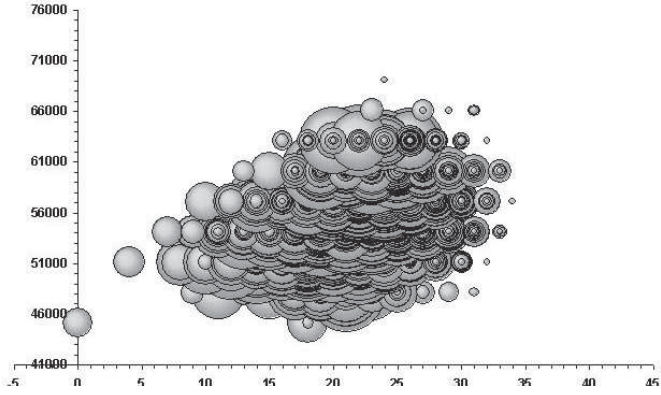

Figure 8. Analysis of local optima for the class GAP-B

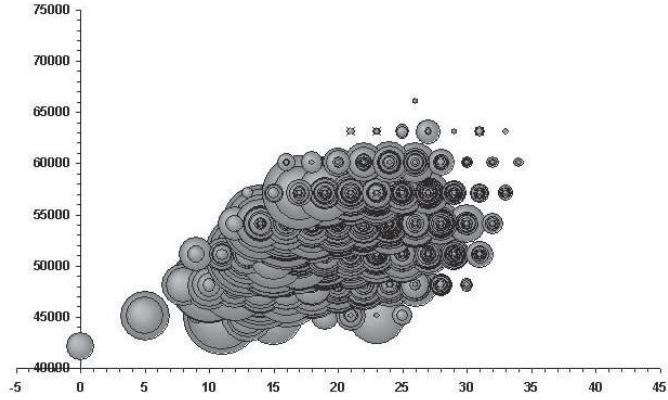

Figure 9. Analysis of local optima for the class GAP-C

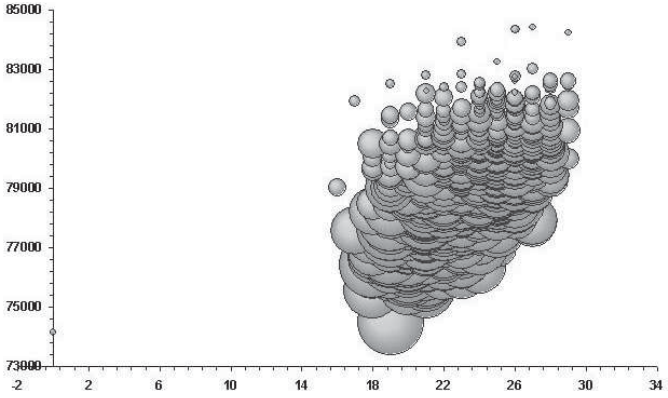

Figure 10. Analysis of local optima for the class Uniform

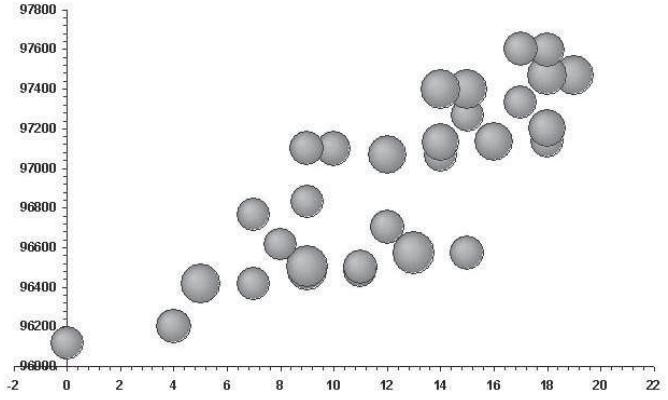

Figure 11. Analysis of local optima for the class Euclidean

small galaxy, $N = 1018$ for class Uniform and $N = 40$ for class Euclidean. The difficult classes have large galaxies, $N \approx 9000$ for classes BPC_7, FPP_{11}, Gap-C. It is possible that the most difficult classes correspond to the cases, where several galaxies are separated from each other by regions with high values of the objective function. Test instances with this property are very interesting for future research [83]. Theoretical properties of different landscapes for NP-hard combinatorial problems can be found in [84].

Table 3 shows the frequency of finding an optimal solution using the following metaheuristics: Probabilistic Tabu Search (PTS), Genetic Algorithm (GA) and Greedy Randomizes Adaptive Search Procedure with Local Improvement (GRASP+LI). The stopping criterion for the algorithms is the maximal number of steps by the neighborhood Flip $\cup$ Swap. We use number 10^4 as the criteria. The genetic algorithm uses local improvements for each offspring during the evolution. Table 3 indicates that classes Euclidean and Uniform are easy. The average number of steps before an optimal solution is reached is less than 10^3 for all algorithms.

6. Conclusions

We have presented some discrete facility location models, as well as theoretical and experimental results for local search methods. We have explained why the concept of local optimality is important for the theory of computational complexity and numerical methods. There are many open questions in this area. For example, the well-known set covering problem can be reformulated as a facility location problem. We still know very little about the complexity of the corresponding local search problems. In [85] a one-to-one correspondence between Nash equilibria in a facility location game and local optima in a special facility location model is presented. This property helps to determine the computational complexity of finding Nash equilibrium [21]. For the theory of approximation algorithms the concept of local optimality plays an important role as well. In regards to combinatorial optimization problems, the property that all local optima have guaranteed quality with respect to global optima is the basis of the approximability of a large class of problems [75].

Table 3. Frequency of obtaining optimal solutions by metaheuristics

Benchmarks classes	Dimension	PTS	GA	GRASP+LI
BPC_7	128	0.93	0.90	0.99
CB_4	144	0.99	0.88	0.68
FPP_{11}	133	0.67	0.46	0.99
Gap-A	100	0.85	0.76	0.87
Gap-B	100	0.59	0.44	0.49
Gap-C	100	0.53	0.32	0.42
Uniform	100	1.0	1.0	1.0
Euclidean	100	1.0	1.0	1.0

Acknowledgments

Many thanks to Vašek Chvătal and Mark Goldsmith for their efforts for improving the presentation of the paper. This work was supported by RFBR grants 09-01-00059, 09-06-00032.

References

[1] Mirchandani, B. P. and Francis, R. L. (eds.) (1990) *Discrete Location Theory*. Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, New York.

[2] Drezner, Z., Klamroth, K., Schobel, A., and Wesolowsky, G. (2004) The Weber problem. Drezner, Z. and Hamacher, H. W. (eds.), *Facility Location*, pp. 1–36, Springer, Berlin, 2nd edn.

[3] Balinski, M. L. and Wolfe, P. (1963) On Benders decomposition and a plant location problem. Working paper ARO-27, Mathematica, Princeton, NJ.

[4] Efroymson, M. A. and Ray, T. L. (1966) A branch and bound algorithm for plant location. *Operations Research*, **14**, 361–368.

[5] Ageev, A. A. and Beresnev, V. L. (1990) Polynomially solvable cases of simple plant location problem. *Proceedings of the 1st Integer Programming and Combinatorial Optimization Conference IPCO 1990*, pp. 1–6, University of Waterloo Press, Waterloo, ON.

[6] Hochbaum, D. (1982) Heuristics for the fixed cost median problem. *Mathematical Programming*, **22**, 148–162.

[7] Mahdian, Y., Ye, M., and Zhang, J. (2002) Improved approximation algorithms for metric facility location problems. *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, vol. 2462 of *Lecture Notes in Computer Science*, pp. 229–242, Springer, London.

[8] Guha, S. and Khuller, S. (1999) Greedy strikes back: improved facility location algorithms. *Journal of Algorithms*, **31**, 228–248.

[9] Korte, B. and Vygen, J. (2006) *Combinatorial Optimization*, vol. 21 of *Algorithms and Combinatorics*. Springer, Berlin, 2nd edn.

[10] Khumawala, B. (1972) An efficient branch and bound algorithm for the warehouse location problem. *Management Science*, **18**, 718–731.

[11] Beresnev, V. L., Gimadi, E. K., and Dement′ev, V. T. (1978) *Extremal Standardization Problems*. Nauka, Novosibirsk, in Russian.

[12] Lebedev, S. S. and Kovalevskaya, M. I. (1976) The lagrange multipliers in the simple plant location problem. Fridman, A. A. (ed.), *Studies in Discrete Optimization*, pp. 170–180, Nauka, Moscow, in Russian.

[13] Mikhalevich, V. S., Trubin, V. A., and Shor, N. Z. (1986) *Optimization Problems of Production-Transportation Planning*. Nauka, Moscow, in Russian.

[14] Bilde, O. and Krarup, J. (1977) Sharp lower bounds and efficient algorithms for the simple plant location problem. Hammer, P. L., Johnson, E. L., Korte, B. H., and Nemhauser, G. L. (eds.), *Studies in Integer Programming*, vol. 1 of *Annals of Discrete Mathematics*, pp. 79–97, North-Holland, Amsterdam.

[15] Erlenkotter, D. (1978) A dual-based procedure for uncapacitated facility location. *Operations Research*, **26**, 992–1009.

[16] Avella, P., Sassano, A., and Vasil′ev, I. (2006) Computational study of large-scale *p*-median problems. *Mathematical Programming*, **109(1)**, 89–114.

[17] Barahona, F. and Chudak, F. (2000) Solving large scale uncapacitated facility location problems. Pardalos, P. M. (ed.), *Approximation and Complexity in Numerical Optimization*, vol. 42 of *Nonconvex Optimization and Its Applications*, Kluwer.

[18] Hammer, P. L. and Rudeanu, S. (1968) *Boolean Methods in Operations Research and Related Areas*, vol. 7 of *Econometrics and Operations Research*. Springer, New York.

[19] Hammer, P. (1968) Plant location – a pseudo-Boolean approach. *Israel Journal of Technology*, **6**, 330–332.

[20] Beresnev, V. L. (1979) Algorithms for the minimization of polynomials from Boolean variables. *Problemy Kibernetiki*, **36**, 225–246, in Russian.

[21] Kochetov, Y., Kononov, A., and Plyasunov, A. (2009) Competitive facility location models. *Computational Mathematics and Mathematical Physics*, **49**, 994–1009.

[22] Schrijver, A. (2003) *Combinatorial Optimization. Polyhedra and Efficiency*, vol. 24 of *Algorithms and Combinatorics*. Springer, Berlin.
[23] Krarup, J. and Pruzan, P. M. (1983) The simple plant location problem: survey and synthesis. *European Journal of Operational Research*, **12**, 36–81.
[24] Boros, E. and Hammer, P. (2002) Pseudo-Boolean optimization. *Discrete Applied Mathematics*, **123**, 155–225.
[25] Nemhauser, G. L. and Wolsey, L. A. (1988) *Integer and Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, New York.
[26] Aardal, K., Chudak, F. A., and Shmoys, D. B. (1999) A 3-approximation algorithm for the k-level uncapacitated facility location problem. *Information Processing Letters*, **72**, 161–167.
[27] Ageev, A., Ye, Y., and Zhang, J. (2003) Improved combinatorial approximation algorithms for the k-level facility location problem. Baeten, J. C. M., Lenstra, J. K., Parrow, J., and Woeginger, G. J. (eds.), *Automata, Languages and Programming*, Eindhoven, 2003, vol. 2719 of *Lecture Notes in Computer Science*, pp. 145–156, Springer, Berlin.
[28] Goncharov, E. (1998) Branch and bound algorithm for the two-level uncapacitated facility location problem. *Discrete Analysis and Operations Research, Series 2*, **5**, 19–39, in Russian.
[29] Tcha, D. W. and Lee, B. I. (1984) A branch-and-bound algorithm for the multi-level uncapacitated facility location problem. *European Journal of Operational Research*, **18**, 35–43.
[30] Gimadi, E. K. (1997) Effective algorithms for solving multi-level plant location problems. Korshunov, A. D. (ed.), *Operations Research and Discrete Analysis*, vol. 391 of *Mathematics and Its Applications*, pp. 51–69, Kluwer, Dordrecht.
[31] Gimadi, E. K. (1997) Exact algorithm for some multi-level location problems on a chain and a tree. Zimmermann, U., Derigs, U., Gaul, W., Möhring, R. H., and Schuster, K.-P. (eds.), *Operations Research Proceedings 1996*, Braunschweig, 1996, pp. 72–77, Springer, Berlin.
[32] Barros, A. I. and Labbé, M. (1994) A general model for the uncapacitated facility and depot location problem. *Location Science*, **2**, 173–191.
[33] Hanjoul, P. and Peeters, D. (1987) A facility location problem with clients' preference orderings. *Regional Science and Urban Economics*, **17**, 451–473.
[34] Dempe, S. (2002) *Foundations of Bilevel Programming*. Kluwer, Dordrecht.
[35] Gorbachevskaya, L. E. (1998) *Polynomially solvable and* NP*-hard bilevel standardization problems*. Ph.D. thesis, Sobolev Institute of Mathematics, in Russian.
[36] Hansen, P., Kochetov, Y., and Mladenović, N. (2004) Lower bounds for the uncapacitated facility location problem with user preferences. Tech. Rep. G-2004-24, GERAD, Montréal.
[37] Rhys, J. M. W. (1970) A selection problem of shared fixed costs and network flows. *Management Science*, **17**, 200–207.
[38] Alekseeva, E. and Kochetov, Y. (2007) Genetic local search for the p-median problem with user preferences. *Discrete Analysis and Operations Research. Series 2*, **14**, 3–31, in Russian.
[39] Canovas, L., Garcia, S., Labbe, M., and Marin, A. (2007) A strengthened formulation for the simple plant location problem with order. *Operations Research Letters*, **35**, 141–150.
[40] Vasil′ev, I., Klimentova, K., and Kochetov, Y. (2009) New lower bounds for the facility location problem with user preferences. *Computational Mathematics and Mathematical Physics*, **49**, 1055–1066.
[41] Plastria, F. and Vanhaverbeke, L. (2008) Discrete models for competitive location with foresight. *Computers and Operations Research*, **35**, 683–700.
[42] Rodriguez, C. M. C. and Perez, J. A. M. (2008) Multiple voting location problems. *European Journal of Operational Research*, **191**, 437–453.
[43] Hakimi, S. L. (1990) Locations with spatial interactions: competitive locations and games. Mirchandani, P. B. and Francis, R. L. (eds.), *Discrete Location Theory*, pp. 439–478, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, New York.
[44] Alekseeva, E., Kochetova, N., Kochetov, Y., and Plyasunov, A. (2010) A heuristic and exact methods for the discrete $(r \mid p)$-centroid problem. Cowling, P. I. and Merz, P. (eds.), *Evolutionary Computation in Combinatorial Optimization*, vol. 6022 of *Lecture Notes in Computer Science*, pp. 11–22, Springer, Berlin.
[45] Noltemeier, H., Spoerhase, J., and Wirth, H. C. (2007) Multiple voting location and single voting location on trees. *European Journal Operational Research*, **181**, 654–667.
[46] Spoerhase, J. and Wirth, H. C. (2008) (r, p)-centroid problems on paths and trees. Tech. Rep. 441, Inst. Comp. Science, University of Wurzburg.

[47] Bhadury, J., Eiselt, Y., and Jaramillo, J. (2003) An alternating heuristic for medianoid and centroid problems in the plane. *Computers and Operations Research*, **30**, 553–565.
[48] Alekseeva, E., Kochetova, N., Kochetov, Y., and Plyasunov, A. (2009) A hybrid memetic algorithm for the competitive p-median problem. *Proceedings of INCOM 2009*, pp. 1516–1520.
[49] Alekseeva, E. and Kochetova, N. (2008) Upper and lower bounds for the competitive p-median problem. *Proceedings of XIV Baikal International School-Seminar*, pp. 563–569, in Russian.
[50] Benati, S. and Laporte, G. (1994) Tabu search algorithms for the $(r|X_p)$-medianoid and $(r|p)$-centroid problems. *Location Science*, **2**, 193–204.
[51] Plyasunov, A. Personal communication.
[52] Alekseeva, E., Kochetov, Y., and Plyasunov, A. (2008) Complexity of local search for the p-median problem. *European Journal of Operational Research*, **191**, 736–752.
[53] Grover, L. K. (1992) Local search and the local structure of NP-complete problems. *Operations Research Letters*, **12**, 235–243.
[54] Gutin, G. (1999) Exponential neighbourhood local search for the traveling salesman problem. *Computers & Operations Research*, **26**, 313–320.
[55] Gutin, G. and Yeo, A. (1999) Small diameter neighbourhood graphs for the traveling salesman problem: at most four moves from tour to tour. *Computers & Operations Research*, **26**, 321–327.
[56] Ahuja, R. K., Ergun, Ö., Orlin, J. B., and Punnen, A. P. (2002) A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, **123**, 75–102.
[57] Schäffer, A. A. and Yannakakis, M. (1991) Simple local search problems that are hard to solve. *SIAM Journal on Computing*, **20**, 56–87.
[58] Johnson, D. C., Papadimitriou, C. H., and Yannakakis, M. (1988) How easy is local search? *Journal of Computer and System Sciences*, **37**, 56–87.
[59] Krentel, M. W. (1989) Structure in locally optimal solutions. *30th Annual Symposium on Foundations of Computer Science*, pp. 216–222, IEEE Comput. Soc. Press, Los Alamitos, CA.
[60] Papadimitriou, C. H. (1992) The complexity of the Lin – Kegnighan heuristic for the traveling salesman problem. *SIAM Journal on Computing*, **21**, 450–465.
[61] Krentel, M. W. (1990) On finding and verifying locally optimal solutions. *SIAM Journal on Computing*, **19**, 742–751.
[62] Hopfield, J. J. (1982) Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA*, **79**, 2554–2558.
[63] Kochetov, Y. and Ivanenko, D. (2005) Computationally difficult instances for the uncapacitated facility location problem. Ibaraki, T., Nonobe, K., and Yagiura, M. (eds.), *Metaheuristics: Progress as Real Solvers*, pp. 351–367, Springer.
[64] Yannakakis, M. (1997) Computational complexity. Aarts, E. and Lenstra, J. K. (eds.), *Local Search in Combinatorial Optimization*, pp. 19–55, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, Chichester.
[65] Kochetov, Y., Paschenko, M., and Plyasunov, A. (2005) On complexity of local search for the p-median problem. *Discrete Analysis and Operations Research, Series 2*, **12**, 44–71, in Russian.
[66] Vredeveld, T. and Lenstra, J. K. (2003) On local search for the generalized graph coloring problem. *Operations Research Letters*, **31**, 28–34.
[67] Dréo, J., Pétrowski, A., Siarry, P., and Taillard, E. (2006) *Metaheuristics for Hard Optimization*. Springer, Berlin.
[68] Glover, F. and Laguna, M. (1997) *Tabu Search*. Kluwer, Dordrecht.
[69] Hansen, P. and Mladenović, N. (2001) Variable neighborhood search: principles and applications. *European Journal of Operational Research*, **130**, 449–467.
[70] Tovey, C. A. (1983) On the number of iterations of local improvement algorithms. *Operations Research Letters*, **2**, 231–238.
[71] Tovey, C. A. (1985) Hill climbing with multiple local optima. *SIAM Journal on Algebraic and Discrete Methods*, **6**, 384–393.
[72] Tovey, C. A. (1997) Local improvement on discrete structures. Aarts, E. and Lenstra, J. K. (eds.), *Local Search in Combinatorial Optimization*, pp. 57–89, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, Chichester.
[73] Orlin, J., Punnen, A., and Schulz, A. S. (2004) Approximate local search in combinatorial optimization. *SIAM Journal of Computing*, **33**, 1201–1214.
[74] Papadimitriou, C. H. and Steiglitz, K. (1977) On the complexity of local search for the traveling salesman

problem. *SIAM Journal of Computing*, **6**, 76–83.
[75] Ausiello, G. and Protasi, M. (1995) Local search, reducibility and approximability of NP optimization problems. *Information Processing Letters*, **54**, 73–79.
[76] Kochetov, Y., Alekseeva, E., Levanova, T., and Loresh, M. (2005) Large neighborhood local search for the *p*-median problem. *Yugoslav Journal of Operations Research*, **15**, 53–63.
[77] Levanova, T. V. and Loresh, M. A. (2004) Algorithms of ant system and simulated annealing for the *p*-median problem. *Automation and Remote Control*, **65**, 431–438.
[78] Mladenović, N., Brimberg, J., Hansen, P., and Moreno-Perez, J. (2007) The *p*-median problem: A survey of metaheuristic approaches. *European Journal Operational Research*, **179**, 927–939.
[79] Resende, M. G. C. and Werneck, R. F. (2006) A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operational Research*, **174**, 54–68.
[80] Boese, K. D., Kahng, A. B., and Muddu, S. (1994) A new adaptive multi-start technique for combinatorial global optimization. *Operations Research Letters*, **16**, 101–113.
[81] Hall, M., Jr. (1967) *Combinatorial Theory*. Blaisdell, Waltham, MA.
[82] Krotov, D. (2000) Lower bounds for number of *m*-quasi groups of order 4 and number of perfect binary codes. *Diskretnyĭ Analiz i Issledovanie Operatsiĭ. Seriya 1*, **7**, 47–53, in Russian.
[83] Qasem, M. and Prügel-Bennett, A. (2010) Learning the large-scale structure of the MAX-SAT landscape using populations. *IEEE Transactions on Evolutionary Computation*, **14**, 518–529.
[84] Angel, E. and Zissimopoulos, V. (2000) On the classification of NP-complete problems in terms of their correlation coefficient. *Discrete Applied Mathematics*, **99**, 261–277.
[85] Tardos, É. and Wexler, T. (2007) Network formation games and the potential function method. Nisan, N., Roughgarden, T., Tardos, É. T., and Vazirani, V. V. (eds.), *Algorithmic Game Theory*, pp. 487–516, Cambridge Univ. Press, Cambridge.

Combinatorial Optimization: Methods and Applications
V. Chvátal (Ed.)
IOS Press, 2011

doi:10.3233/978-1-60750-718-5-135

Discrete Convexity and Its Applications

G.A. KOSHEVOY [1]
CEMI, Russian Academy of Sciences, Russia

Abstract. We explain what subsets of the lattice $\mathbb{Z}^n$ and what functions on the lattice $\mathbb{Z}^n$ could be called convex. The basis of the theory is the following three main postulates of classical convex analysis: concave functions are closed under sums; they are also closed under convolutions; and the superdifferential of a concave function is nonempty at each point of the domain. Interesting (and even dual) classes of discrete concave functions arise if we require either the existence of superdifferentials and closeness under convolutions or the existence of superdifferentials and closeness under sums. The corresponding classes of convex sets are obtained as the affinity domains of such discretely concave functions. The classes of the first type are closed under (Minkowski) sums, and the classes of the second type are closed under intersections. In both classes, the separation theorem holds true. Unimodular sets play an important role in the classification of such classes. The so-called polymatroidal discretely concave functions, most interesting for applications, are related to the unimodular system $A_n := \{\pm e_i, e_i - e_j\}$. We demonstrate that such functions naturally appear in mathematical economics, in combinatorics, play an important role for solution of the Horn problem, for describing submodule invariants over discrete valuation rings, etc.

Keywords. Integer polyhedron, base polyhedron, pure subgroup, pure system, unimodular system, laminarization, pseudoconvexity, convolution, discrete valuation ring, Horn problem, octohedron recurrence

Introduction

My lectures are devoted to the following questions: what subsets of the lattice $\mathbb{Z}^n$ should be regarded as "convex," and what functions on $\mathbb{Z}^n$ should be regarded as "convex" and "concave"?

Lecture 1

There are several equivalent definition of convexity in the usual Euclidean spaces and of convex/concave functions on these spaces. For instance, a subset of $\mathbb{R}^n$ is called convex if, with every pair of its points x and y, it contains the whole line segment $[x, y]$, defined as $\{\alpha x + (1 - \alpha)y : \alpha \in [0, 1]\}$. A cornerstone of classical convexity theory asserts that a subset X of $\mathbb{R}^n$ is convex in this sense if and only if for every point z of $\mathbb{R}^n$ located outside X, there exists a hyperplane which separates X and z; this property further implies that

[1] Laboratory of Mathematical Economics, CEMI, Nakhimosky Prospekt, 47, 117418, Moscow, Russia; E-mail: koshevoy@cemi.rssi.ru.

every pair of nonintersecting convex sets can be separated by a hyperplane. Our aim is to construct a theory of convexity preserving the latter strong separation property in the discrete setup, for subsets of integer points $\mathbb{Z}^n$ in $\mathbb{R}^n$.

In the discrete setup there are several instances where separation takes place. We are going to list a few without defining the terms that we use: Frank (1982) [1] proved the separation theorem for submodular and supermodular functions. Later on, Lovász (1984) [2] proposed a convex continuation of a submodular function to the unit cube, using a natural triangulation of the cube (this is a particular form of Choquet integration (1953) [3]). In 1991, Dress and Wentzel [4] introduced valuated matroids and later Murota (1996) [5] considered their ultradiscretization, $M^\natural$-convex/concave functions. These functions form a class of functions for which the separation holds true (integer valued $M^\natural$-functions form class which is Fenchel dual to the class of supermodular functions on the cube). In 1998, Danilov, Murota and I [6] found that discrete convexity plays the key role in the existence of equilibria in economies with indivisibles. Let us point out that in the economic framework, a class of functions which is equivalent to the ultradiscretization of valuated matroids was implicitly introduced in 1982 by Kelso and Crawford [7].

General theory was elaborated in the paper [8] by Danilov and myself. In the book [9], a particularly important variant of general theory corresponding to a class of discrete convexity related to generalized polymatroids was studied in depth. One can speculate that in the discrete case there might be a variant of discrete convexity. The point is that in the discrete case there is no unique theory of convexity, contrary to the usual Euclidean case, and we will explain reasons for that.

Convex Sets in $\mathbb{Z}^n$ and Separation

Let us define "convex" sets in $\mathbb{Z}^n$ using an exterior approach via separation: let us call a subset X of $\mathbb{Z}^n$ "convex" if it can be separated by a hyperplane (in $\mathbb{R}^n$) from any point in $\mathbb{Z}^n \setminus X$.

It is easy to check that such a "convex" set coincides with the set of all integer points of its convex hull. To state this observation in symbols, we let $\mathrm{co}(X)$ denote the convex hull of X and for every subset S of $\mathbb{R}^n$, we let $S(\mathbb{Z})$ denote $S \cap \mathbb{Z}^n$. In this notation, every "convex" set coincides with $\mathrm{co}(X)(\mathbb{Z})$. Since we want to deal with "closed convex sets," we add the requirement that $\mathrm{co}(X)$ be a polyhedron:

Definition. A subset X of $\mathbb{Z}^n$ is said to be *pseudoconvex* if $X = \mathrm{co}(X)(\mathbb{Z})$ and $\mathrm{co}(X)$ is a polyhedron (the intersection of finitely many half-spaces).

We let $\mathcal{PC}$ denote the set of pseudoconvex sets.

We call these sets pseudoconvex and not convex, because they do not possess the strong separation property. To see this, consider the following simple example in $\mathbb{Z}^2$: If $X = \{(0,0),(1,1)\}$ and $Y = \{(0,1),(1,0)\}$, then both X and Y are pseudoconvex and they do not intersect, but they cannot be separated by a hyperplane. This example shows that separation of non-intersecting pseudoconvex sets does not follow from separation of pseudoconvex sets and points outside them.

To construct an interesting theory of discrete convexity, we need the separation of non-intersecting "convex" sets, and so we have to consider narrower classes of subsets of $\mathbb{Z}^n$ than the class $\mathcal{PC}$.

Recall that a polyhedron P in a finite-dimensional Euclidean space V (V is isomorphic to $\mathbb{R}^n$ with some n, integer points of V we denote by M, $M \cong \mathbb{Z}^n$) is said to be *rational* if it is the set of solutions to a finite system of integer linear inequalities. A polyhedron P is *integer* if it is rational and if every (nonempty) face of P contains an integer point.

For example, a polytope (a bounded polyhedron) is integer if and only if all its vertices are integer points.

Proposition 1. *Suppose $X \subset M$. The following assertions are equivalent*:

(a) *X is pseudoconvex*;

(b) *$X = P(\mathbb{Z})$ for some integer polyhedron $P \subset V$*;

(c) *X is the set of integer solutions of a finite system of linear inequalities with integer coefficients.*

Proof. The implication (a) $\Rightarrow$ (b) is almost obvious; it suffices to take P to be $\mathrm{co}(X)$. The implication (b) $\Rightarrow$ (c) is obvious. Finally, implication (c) $\Rightarrow$ (a) is precisely Meyer's theorem (see [10, p. 430]). □

Denote by $\mathcal{IPh}$ the class of all integer polyhedra in V. By Proposition 1, we have a natural bijection between the classes $\mathcal{IPh}$ and $\mathcal{PC}$, which is given by the mappings $\phi\colon \mathcal{IPh} \to \mathcal{PC}$, $P \mapsto P(\mathbb{Z})$ and $\phi^{-1}\colon X \mapsto \mathrm{co}(X)$. Both of these classes are stable under integer translations ($X \mapsto X + m$, $m \in \mathbb{M}$), under reflection ($X \mapsto -X$), and under taking faces ($X \mapsto X \cap F$, where F is a face of the polyhedron $\mathrm{co}(X)$). Furthermore, the class of pseudoconvex sets $\mathcal{PC}$ is stable under intersection and is not stable under summation, whereas the class of integer polyhedra $\mathcal{IPh}$ is stable under summation and is not stable under intersection (the sum of two pseudoconvex sets need not be pseudoconvex, while the intersection of integer polyhedra need not be an integer polyhedron).

In the following commutative diagrams the bottom mappings do not take the form of intersection and summation, respectively.

$$\begin{array}{ccc} \mathcal{PC}\times\mathcal{PC} & \xrightarrow{\cap} & \mathcal{PC} \\ \downarrow{\scriptstyle \phi^{-1}\times\phi^{-1}} & & \downarrow{\scriptstyle \phi^{-1}} \\ \mathcal{IPh}\times\mathcal{IPh} & \longrightarrow & \mathcal{IPh} \end{array} \qquad \begin{array}{ccc} \mathcal{IPh}\times\mathcal{IPh} & \xrightarrow{+} & \mathcal{IPh} \\ \downarrow{\scriptstyle \phi\times\phi} & & \downarrow{\scriptstyle \phi} \\ \mathcal{PC}\times\mathcal{PC} & \longrightarrow & \mathcal{PC} \end{array}$$

From the next proposition it follows that in a narrowed class $\mathcal{K} \subset \mathcal{PC}$, which possesses separation of pseudoconvex sets, and in $\mathcal{P} = \phi(\mathcal{K})$, we have the following commuting diagrams

$$\begin{array}{ccc} \mathcal{K}\times\mathcal{K} & \xrightarrow{\cap} & \mathcal{PC} \\ \downarrow{\scriptstyle \phi^{-1}\times\phi^{-1}} & & \downarrow{\scriptstyle \phi^{-1}} \\ \mathcal{IPh}\times\mathcal{IPh} & \longrightarrow & \mathcal{IPh} \end{array} \qquad \begin{array}{ccc} \mathcal{P}\times\mathcal{P} & \xrightarrow{+} & \mathcal{IPh} \\ \downarrow{\scriptstyle \phi\times\phi} & & \downarrow{\scriptstyle \phi} \\ \mathcal{PC}\times\mathcal{PC} & \longrightarrow & \mathcal{PC} \end{array}$$

We say that a class $\mathcal{K} \subset \mathcal{PC}$ is *ample* if $\mathcal{K}$ is stable under (a) integer translations, (b) reflection, and (c) faces. In the same way we understand ampleness of a polyhedral class $\mathcal{P} \subset \mathcal{IP}h$.

Proposition 2. *Let $\mathcal{K} \subset \mathcal{PC}$ be an ample class. The following four properties of $\mathcal{K}$ are equivalent*:

(Sep) *if sets X and Y of $\mathcal{K}$ do not intersect, then there exists a (integer) linear functional $p\colon V \to \mathbb{R}$ such that $p(x) > p(y)$ for any $x \in X$, $y \in Y$;*

(Add) *for every $X, Y \in \mathcal{K}$ the sets $X \pm Y$ are pseudoconvex;*

(Int) *if sets X and Y of $\mathcal{K}$ do not intersect, then the polyhedra $\mathrm{co}(X)$ and $\mathrm{co}(Y)$ do not intersect as well*;

(Edm) *for every $X, Y \in \mathcal{K}$ the polyhedron $\mathrm{co}(X) \cap \mathrm{co}(Y)$ is integer.*

Proof. (Add) $\Rightarrow$ (Sep). If X and Y have an empty intersection, then $0 \notin X - Y$. Since the set $X - Y$ is pseudoconvex, 0 does not belong to the polyhedron $\mathrm{co}(X-Y) = \mathrm{co}(X) - \mathrm{co}(Y)$. Hence there exists a linear (integer) functional $p\colon V \to \mathbb{R}$ which is strictly positive on $\mathrm{co}(X - Y)$. Therefore $p(x) > p(y)$ for $x \in X$ and $y \in Y$.

(Sep) $\Rightarrow$ (Int). This one is obvious.

(Int) $\Rightarrow$ (Add). Let us show that $X - Y$ is pseudoconvex. Since $\mathrm{co}(X - Y) = \mathrm{co}(X) - \mathrm{co}(Y)$ is a polyhedron, we need to prove that $X - Y = \mathrm{co}(X - Y) \cap M$. Suppose the integer point m lies in $\mathrm{co}(X - Y) = \mathrm{co}(X) - \mathrm{co}(Y)$. Then the polyhedra $\mathrm{co}(X)$ and $m + \mathrm{co}(Y) = \mathrm{co}(m + Y)$ intersect. Applying (*Int*) to the sets X and $m + Y$, we see that these sets also intersect, that is $m \in X - Y$.

(Edm) $\Rightarrow$ (Int). This implication is obvious.

(Int) $\Rightarrow$ (Edm). Suppose $X, Y \in \mathcal{K}$, $P = \mathrm{co}(X)$, and $Q = \mathrm{co}(Y)$. We need to show that $P \cap Q$ is an integer polyhedron. Obviously $P \cap Q$ is rational. Therefore we need to establish that every (non-empty) face of $P \cap Q$ contains an integer point. We assume here, without loss of generality, that the face is minimal.

Suppose F is a minimal (nonempty) face of the polyhedron $P \cap Q$. Let P' (resp. Q') be a minimal face of P (resp. Q) which contains F. We claim that $F = P' \cap Q'$.

Projecting V along F, we may suppose additionally that F is of dimension zero. That is, F consists of a single point, which is a vertex of $P \cap Q$. Suppose, on the contrary, that $P' \cap Q'$ contains some other point a. Since the point F is relatively interior both in P' and in Q', then F is an interior point of some segment $[a, b]$, lying in both P' and Q'. But in such a case the segment $[a, b] \subset P' \cap Q' \subset P \cap Q$, and F can not be a vertex of $P \cap Q$. Contradiction.

Thus, $F = P' \cap Q'$. Since our class $\mathcal{K}$ is stable under faces, the sets $P'(\mathbb{Z})$ and $Q'(\mathbb{Z})$ belong to $\mathcal{K}$. The property *(Int)* implies that the sets $P'(\mathbb{Z})$ and $Q'(\mathbb{Z})$ intersect. Because of this, F is an integer singleton. □

Definition. An ample class $\mathcal{K} \subset \mathcal{PC}$ is a *class of discrete convexity* (or a *DC-class*) if it possesses any of the properties from Proposition 2.

In the language of integer polyhedra, the definition of discrete convexity is formulated as follows. A class $\mathcal{P}$ of integer polyhedra is a *polyhedral class of discrete convexity* if it is ample and the following variant of the Edmonds' condition holds:

(Edm′) The intersection of any two polyhedra from $\mathcal{P}$ is an integer polyhedron (not necessarily in $\mathcal{P}$).

Jack Edmonds did not invent this property. We attributed it to him because of his famous polymatroids intersection theorem.

According to Proposition 2, an equivalent condition is:

(Add′) $(P + Q)(\mathbb{Z}) = P(\mathbb{Z}) + Q(\mathbb{Z})$ for every $P, Q \in \mathcal{P}$.

Let us give a few examples of DC-classes.

Example 1 (One-dimensional case)**.** Let $V \cong \mathbb{R}$ and $M \cong \mathbb{Z}$. Then the class $\mathcal{PC}$ of all pseudoconvex sets in M is a *DC*-class. This is not the case in higher dimensions of course.

The class of integer rectangles in the plane $\mathbb{R}^2$ is a *DC*-class. More generally, if $\mathcal{K}_1$ and $\mathcal{K}_2$ are *DC*-classes in the free Abelian groups M_1 and M_2, respectively, then the class of sets of the form $X_1 \times X_2$ with $X_i \in \mathcal{K}_i$, $i = 1, 2$, is a *DC*-class in $M_1 \times M_2$ as well.

Example 2 (Hexagons)**.** Let us consider a more interesting than rectangles class $\mathcal{H}$ of polyhedra in $\mathbb{R}^2$ defined by the inequalities $a_1 \le x_1 \le b_1$, $a_2 \le x_2 \le b_2$, $c \le x_1 + x_2 \le d$, where a_1, a_2, b_1, b_2, c and d are integers. It is easy to check that such a hexagon has integer vertices (generally speaking, this hexagon can be degenerated to a polyhedron with a smaller number of edges). Obviously, $\mathcal{H}$ is stable under integer translations, reflection and faces. Since the intersection of hexagons yields a hexagon, we conclude that $\mathcal{H}$ is a polyhedral *DC*-class.

Next two examples present two possible higher dimensional generalizations of Example 2.

Example 3 (Base polyhedra)**.** Let N be a finite set, and let V be the set of measures on N, that is the dual space to $\mathbb{R}^N$.

A function $b\colon 2^N \to \mathbb{R} \cup \{+\infty\}$ is called *submodular* if for any $S, T \subset N$, the following inequality holds

$$b(S) + b(T) \ge b(S \cup T) + b(S \cap T).$$

Given a submodular function b, a *base polyhedron* is a polyhedron of the following form

$$B(b) = \{x \in V \mid x(S) \le b(S), S \subset N, \text{ and } x(N) = b(N)\}.$$

Denote by $\mathcal{B}$ the class if base-polyhedra in V defined by integer-valued submodular functions. Obviously, the class $\mathcal{B}$ is stable under integer translations and under reflection. Less trivial is that $\mathcal{B}$ is stable under faces (see, e.g., [11]), and hence, each base polyhedron has integer vertices. The well-known theorem by Edmonds [12] ensures that the property (Edm) holds, and thus $\mathcal{B}$ is a polyhedral *DC*-class.

Example 4. Let N be a finite set, and let $V = \mathbb{R}^N$ be the space of real-valued functions on N. Consider the class $\mathcal{L}$ of polyhedra in V, given by inequalities of the form $a_i \le x(i) \le b_i$ and $a_{ij} \le x(i) - x(j) \le b_{ij}$, where $i, j \in N$, and all a's and b's are integers. We claim

that these polyhedra are integer. Indeed, their vertices are given by equalities of the form $x(i) = c_i$ and $x(i) - x(j) = c_{ij}$ where the c's are integers. It is clear that such a system has an integer solution.

Thus, the class $\mathcal{L}$ consists of integer polytopes. Since it is stable under intersection, the axiom (Edm′) is satisfied automatically, and $\mathcal{L}$ is a polyhedral *DC*-class.

Examples 3 and 4 have to convince a reader that classes of discrete convexity are non trivial and interesting objects. However, one can ask: do other classes exist and how to construct such classes.

General Construction of Classes of Discrete Convexity

It is well known that any polyhedron can be decomposed into Minkowski sum of a linear subspace (a lineal), a cone, and a polytope. There are two view points on polytopes, either as the convex hull of a collection of points or as the set of solutions to a finite system of linear inequalities.

For aims of construction of DC-classes we exploit another view points. Namely, we are looking on polyhedra from the perspective of "directions" of its faces.

Homogenization

Let $P \subset V$ be a polyhedron and let F be a face of P. The "direction" of F is the linear space $\mathrm{Tan}(F) := \mathbb{R}(F - F) = \{\alpha(f - f'), \alpha \in \mathbb{R}, f, f' \in F\}$, the *tangent space* to F. For example, for $F = P$, the space $\mathrm{Tan}(P)$ is the minimal linear subspace of V which contains P; for a vertex $x \in P$, the tangent space $\mathrm{Tan}(x)$ is the null-dimensional subspace of V.

Linear subspaces are the simplest polyhedral, and to a polyhedron P we associate the collection of directions of its faces

$$\mathcal{U}(P) := \{\mathrm{Tan}(F), F \text{ is a face of } P\}.$$

We call the collection $\mathcal{U}(P)$ of linear subspaces of V the *homogenization* of P.

For a class of polyhedra $\mathcal{P}$ we associate the collection of directions of all polyhedra in the class, a *homogenization of the class*,

$$\mathcal{U}(\mathcal{P}) := \{\mathcal{U}(P), P \in \mathcal{P}\}.$$

For an ample class $\mathcal{P}$, there holds $\mathcal{U}(\mathcal{P}) := \{\mathrm{Tan}(P), P \in \mathcal{P}\}$.

We are going to show that a class $\mathcal{P}$ forms a class of discrete convexity if and only if the homogenization $\mathcal{U}(\mathcal{P})$ forms a pure system of linear subspaces in V. To formulate such a purity property we use the language of pseudoconvex sets.

For a (rational) vector subspace $F \subset V \cong \mathbb{R}^n$ the set $S = F(\mathbb{Z})$ of all integer points of F is an Abelian subgroup of $M \cong \mathbb{Z}^n$. Such subgroups of M are called *pure*. Equivalently, a subgroup $S \subset M$ is pure if and only if S is a pseudoconvex subset of M.

In general, the sum of pure subgroups of M need not be a pure subgroup of M. For example, if $M = \mathbb{Z}^2$, $S = \mathbb{Z}(1, 1)$, and $S' = \mathbb{Z}(1, -1)$ then the group $S + S'$ has index 2 in M.

Definition. Pure subgroups S and S' of M are called *mutually pure* if the sum $S + S'$ is a pure subgroup of M. Two (rational) linear subspaces L and L' of V are *mutually pure* if the subgroups $L(\mathbb{Z})$ and $L'(\mathbb{Z})$ are mutually pure.

For a pseudoconvex subset X in M, consider the tangent space $\mathrm{Tan}(X) := \mathbb{R}(X - X)$ in V, and the subgroup $S = \mathbb{Z}(X - X)$ in M. Of course, $S \subseteq \mathrm{Tan}(X)(\mathbb{Z})$, and in the general case this inclusion is proper. Hence, in the general case, S needs not be a pure subgroup of M. Nevertheless, there is an instance when we can guarantee the purity of S.

For a natural number n and $X \subset M$, we denote by $[n]X$ the sum of n copies of X; for example, $[2]X = X + X$.

Proposition 3. *Let $X \subset M$. Suppose that $[n]X$ is a pseudoconvex set for every $n = 1, \ldots$. Then the subgroup $\mathbb{Z}(X - X)$ is pure.*

Given an ample class $\mathcal{K}$ of pseudoconvex sets, we associate to it the following system of linear subspaces in V, the *homogenization* of $\mathcal{K}$,

$$\mathcal{U}(\mathcal{K}) = \{\mathrm{Tan}(X), X \in \mathcal{K}\}.$$

Definition. A collection $\mathcal{U}$ of linear subspaces in V is called a *pure system* if every pair $F, G \in \mathcal{U}$ forms a pair of mutually pure subspaces. Elements of a pure system are called *flats*.

Say that an ample class $\mathcal{P}$ of integer polyhedra is *very ample* if it contains the polyhedron nP with any integer n and any polyhedron $P \in \mathcal{P}$.

The following proposition states that the homogenization of a DC-classes leads to a pure system.

Proposition 4. *Let $\mathcal{P}$ be a very ample DC-class $\mathcal{P}$ of integer polyhedra. Then $\mathcal{U}(\mathcal{P})$ is a pure system.*

Definition. A pure system $\mathcal{U}$ is said to be a *pure S-system* (correspondingly, a *pure I-system*) if $F + G$ (correspondingly, $F \cap G$) belongs to $\mathcal{U}$ for any $F, G \in \mathcal{U}$.

It is clear that the homogenization of an S-class is a pure S-system, and the homogenization of an I-class is a pure I-system.

Main properties of pure systems are collected in the following.

Theorem 1. (a) *Any pure system contains finitely many subspaces.*

(b) *Let $\mathcal{U}$ be a pure system* (*in M*), *then the orthogonal system $\mathcal{U}^\perp$ is pure* (*in the Abelian group M^* of the dual space V^**), *where $\mathcal{U}^\perp := \{L^\perp, L \in \mathcal{U}\}$ and $L^\perp = \{p \in V^*, p(v) = 0 \; \forall \, v \in L\}$.*

(c) *$\mathcal{U}$ is an S-system if and only if $\mathcal{U}^\perp$ is an I-system.*

Proofs can be found in [8].

Remark. Pure systems form a category. The objects of the category are pairs $(M, \mathcal{U})$, a free Abelian group and a pure system in it, and a morphism from $(M, \mathcal{U})$ to $(M', \mathcal{U}')$ is a homomorphism of groups $f\colon M \to M'$ such that $f(\mathcal{U}) \subset \mathcal{U}'$.

The class of integer base-polyhedra $\mathcal{B}$ is a CD-class. Therefore its homogenization $\mathcal{U}(\mathcal{B})$ is a pure system. We denote this pure system by $\mathcal{U}(\mathbb{A}(N))$.

Example 5 (The homogenization of base polyhedra)**.** Let $B(b)$ be the base-polyhedron defined by a submodular function $b\colon 2^N \to \mathbb{R} \cup \{+\infty\}$. Here we can assume that $B(b)$ is a symmetric base polyhedron (with respect to the origin 0). This means that $b(S) = b(N \setminus S)$; in particular, $b(N) = 0$. It is clear, that $nB(b) = B(nb)$. Therefore, the tangent space $\mathrm{Tan}(B(b))$ is the base polyhedron $B(\infty b)$ that is a linear subspace given by the following equations:

$$x(S) = 0, \quad S \in \mathcal{F}(b),$$

where $\mathcal{F}(b) = \{S \subset N, b(S) = 0\}$. Obviously, $\emptyset, N \in \mathcal{F}(b)$. The symmetry of $B(b)$ implies that $N \setminus S \in \mathcal{F}(b)$ with any $S \in \mathcal{F}(b)$. Submodularity of b implies that $S \cup T$ and $S \cap T$ belong to $\mathcal{F}(b)$ with any $S, T \in \mathcal{F}(b)$. Thus, $\mathcal{F}(b)$ is a Boolean subalgebra of 2^N.

Thus, the set of flats in $\mathcal{U}(\mathbb{A}(N))$ is isomorphic to the set of Boolean subalgebras in 2^N. The codimension of a flat, corresponding to a Boolean subalgebra, is equal to the number of union-irreducible elements (atoms) in the subalgebra.

Let us consider, for instance, one-dimensional flats (directions of edges of base-polyhedra). These flats correspond to subalgebras with $N - 1$ atoms. Such a subalgebra has one two-element atom and other atoms are singletons. Thus one-dimensional flats have the form $\mathbb{R}(e_i - e_j)$, $i \neq j$, and (e_i), $i \in N$, denote the Dirac measure at the point $i \in N$.

Now we explain a construction of DC-classes from pure systems.

Dehomogenization

Let $\mathcal{U}$ be a pure system in V. Then the collection of all integer translations of flats of $\mathcal{U}$ is a polyhedral DC-class. However, this class is of little interest. For instance, it contains no polytopes (except possibly zero-dimensional ones). Below we define a more interesting (maximal) DC-class $\mathcal{P}h(\mathcal{U}, \mathbb{Z})$ of integer polyhedra associated to a given pure system $\mathcal{U}$. In words, polyhedra of such a class have directions of faces from $\mathcal{U}$.

Definition. Let $\mathcal{U}$ be a collection of (rational) vector subspaces in V. A polyhedron P is said to be $\mathcal{U}$-*convex* (or $\mathcal{U}$-*polyhedron*) if, for any face F of P, the tangent space $\mathrm{Tan}(F) = \mathbb{R}(F - F)$ belongs to $\mathcal{U}$.

Let $\mathcal{P}h(\mathcal{U})$ be the set of $\mathcal{U}$-polyhedra, and let $\mathcal{P}h(\mathcal{U}, \mathbb{Z})$ be the set of integer $\mathcal{U}$-polyhedra. The homogenization of $\mathcal{P}h(\mathcal{U}, \mathbb{Z})$ brings us back to $\mathcal{U}$.

Theorem 2 ([8])**.** *A class $\mathcal{P}h(\mathcal{U}, \mathbb{Z})$ is a DC-class of integer polyhedra if and only if $\mathcal{U}$ is a pure system.*

Because of this theorem, we get a bijection between (maximal) classes of discrete convexity and (maximal) pure systems. We will show that there exists non-isomorphic maximal pure systems. This is a reason for variety of theories of discrete convexity.

Open problem. To characterize maximal pure systems.

Lecture 2

We established a construction of a bijection between maximal DC-classes and maximal pure systems.

The implication: {DC-sets} $\Rightarrow$ {pure systems} is given by the rule

$$\mathcal{K} \to \operatorname{Tan}\mathcal{K} := \{\mathbb{Z}(X - X), X \in \mathcal{K}\}.$$

The reverse implication is given by

$$\mathcal{U} \to \mathcal{P}h(\mathcal{U}, \mathbb{Z}),$$

where we let $\mathcal{P}h(\mathcal{U}, \mathbb{Z})$ denote the collection of integer polyhedra with direction of faces being flats of $\mathcal{U}$, that is $P \in \mathcal{P}h(\mathcal{U}, \mathbb{Z})$ if and only if, for any face $F \subset P$, the pure subgroup $\mathbb{R}(F - F) \cap \mathbb{Z}^n$ belongs to the pure system $\mathcal{U}$.

Note, that $\mathcal{P}h(\mathcal{U}, \mathbb{Z})$ is a polyhedral DC-class, that is, for any $P, Q \in \mathcal{P}h(\mathcal{U}, \mathbb{Z})$, $P \cap Q$ is an integer polyhedron.

Thus, in order to classify DC-classes we have to classify pure systems. This is an open problem in general, but we can characterize classes which are generated by one-dimensional flats.

Unimodular Systems

Pure S-systems generated by one-dimensional flats are classified by unimodular sets.

Definition. A subset $\mathcal{R} \subset M = \mathbb{Z}^n$ is called *unimodular* if, for any subset $B \subset \mathcal{R}$ the subgroup $\mathbb{Z}B \subset M$ is pure. A *unimodular system* is a pair $(M, \mathcal{R})$ where $\mathcal{R}$ is a unimodular set in M. Non-zero elements of $\mathcal{R}$ are called *roots*.

We call a *flat* (or $\mathcal{R}$-*flat*) a subspace of the form $\mathbb{R}B$ with some $B \subset \mathcal{R}$.

The set collection of all $\mathcal{R}$-flats, $\mathcal{U}(\mathcal{R})$, forms a pure S-system. Thus, the DC-class $\mathcal{P}h(\mathcal{U}(\mathcal{R}), \mathbb{Z})$ is an S-class of discrete convexity. We call elements of this class $\mathcal{R}$-polyhedra (directions of the edges of any $\mathcal{R}$-polytope is a subset of $\mathcal{R}$).

The DC-class $\mathcal{P}h(\mathcal{U}^{\perp}(\mathcal{R}), \mathbb{Z})$ is an I-class of discrete convexity, we call elements of this class $*\mathcal{R}$-polyhedra (directions of the normal vectors to facets of any $*\mathcal{R}$-polytope is a subset of $\mathcal{R}$).

Note, that results of McMullen [13] on zonotope-type tiling of $\mathbb{R}^n$ and of Erdal and Ryzhkov [14] on lattice structure of dicings can be deduced from that, for a unimodular $\mathcal{R}$, the classes of $\mathcal{R}$-polyhedra and $*\mathcal{R}$-polyhedra form S-class and I-class, respectively.

Examples of $\mathcal{R}$-polyhedra are zonotopes $\sum_{r \in R}[0, a_r]r$, $a_r \in \mathbb{Z}_+$; $*\mathcal{R}$-polyhedra are Hoffman polyhedra of the form $x(r) \le b_r$, $b_r \in \mathbb{Z}$, $r \in \mathcal{R}$, $(x \in (\mathbb{R}^n)^*)$.

Let us remind some useful facts on unimodular systems: unimodular systems are closely related to totally unimodular matrices. A matrix is totally unimodular if all of its minors are equal to 0 or ± 1. Suppose that a unimodular set $\mathcal{R}$ is of full dimension, or, equivalently, spans V. If we pick a basis $B \subset \mathcal{R}$ and represent vectors of $\mathcal{R}$ as linear combinations of the basis vectors, then the matrix of coefficients is totally unimodular. In particular, the coefficients of this matrix are either 0 or ± 1, which proves finiteness of any unimodular set. Conversely, columns of a totally unimodular $n \times m$ matrix yield a unimod-

ular set in $\mathbb{Z}^n$. Thus, unimodular systems are nothing but coordinate-free representations of totally unimodular matrices.

By the homogenization of base polyhedra we get the pure system $\mathbb{A}(N)$, which is spanned by one-dimensional flats $\mathbb{Z}(e_i - e_j)$, $i, j \in N$. Thus, the set of vectors $e_i - e_j$, $i, j \in N$, is a unimodular set (in $M^* \cong (\mathbb{Z}^N)^*$). This system is not of full dimension, since it belongs to the hyperplane $x(N) = 0$.

If we project the set $\mathbb{A}(N \cup \{0\})$ along the axis $\mathbb{R}e_0$ onto the space $(\mathbb{R}^N)^*$, we obtain the full-dimensional unimodular system consisting of the vectors $\pm e_i$ and $e_i - e_j$, $i, j \in N$, in $(\mathbb{Z}^N)^*$. We denote this system by $\mathbb{A}_N$. Of course, we could construct $\mathbb{A}_N$ simply by adding the basis $(\pm e_i, i \in N)$ to the system $\mathbb{A}(N)$. We shall show that $\mathbb{A}_N$-polyhedra are nothing but generalized polymatroids.

Subsystems $\mathcal{R} \subset \mathbb{A}_N$ (more precisely, symmetrical subsystems, which contain 0 and $-r$ for any $r \in \mathcal{R}$) are identical to so-called *graphic* unimodular systems.

To any graph G one can associate the so called *cographic* unimodular system $\mathbb{D}(G)$. In a non-invariant way, it might be obtained as a system corresponding to the unimodular matrix (E_m, A^t), where (E_n, A) is the unimodular matrix for the graphical system with the graph G.

Cubic (or three-valent) graphs give the most interesting examples of cographic systems. The simplest example of such a graph is the complete graph K_4 with 4 vertices. The corresponding system $\mathbb{D}(K_4)$ is isomorphic to $\mathbb{A}_3$. The bipartite graph $K_{3,3}$ yields a more interesting example.

Example 6. The system $\mathbb{D}(K_{3,3})$ consists of the following 19 vectors in $\mathbb{R}^4$: $\{0, \pm e_i, i = 1, \ldots, 4, \pm(e_1 + e_2), \pm(e_2 + e_3), \pm(e_3 + e_4), \pm(e_4 + e_1), \pm(e_1 + e_2 + e_3 + e_4)\}$.

One can check that $\mathbb{D}(K_{3,3})$ is not a graphic system.

Example 7. There is an exceptional unimodular system $\mathbb{E}_5$ in dimension five which is neither graphic no cographic. It consists of the following 21 vectors: $\{0, \pm e_i, i = 1, \ldots, 5, \pm(e_1 - e_2 + e_3), \pm(e_2 - e_3 + e_4), \pm(e_3 - e_4 + e_5), \pm(e_4 - e_5 + e_1), \pm(e_5 - e_1 + e_2)\}$.

According to the Seymour theorem [15], every unimodular system can be constructed by combining graphic systems, cographic systems, and the system $\mathbb{E}_5$.

In dimension N, the maximal number of vectors in a unimodular system is less than or equal to $N(N + 1)$, and the system $\mathbb{A}_N$ contains exactly $N(N + 1)$ vectors.

Given a unimodular system $\mathcal{R}$, define the set of all its subsystems which are isomorphic to $\mathbb{A}_k$ with some k, and such k is maximal,

$$\mathrm{Fram}(\mathcal{R}) = \{\mathcal{H} \subset \mathcal{R} \mid \mathcal{H} \text{ is } \mathbb{A}_k\text{-type flat}\}.$$

For example $\mathrm{Fram}(\mathbb{A}_n) = \mathbb{A}_n$, and $\mathrm{Fram}(E_5)$ is constituted from one-dimensional flats of E_5.

Conjecture. Let $\mathcal{R}$ be a maximal unimodular system and let P be an $\mathcal{R}$-polyhedron. Then P can be represented in the form of a sum of $\mathbb{A}_k$-type polyhedra, namely P is equal to a sum of $\mathcal{H}$-polyhedra, $\mathcal{H} \in \mathrm{Fram}(\mathcal{R})$.

This conjecture was proposed by V. Danilov and holds true for some cases, for example for the exceptional system $\mathbb{E}_5$, but still open for a general case.

In order to better understand the structure of $\mathcal{R}$-polyhedra ($*\mathcal{R}$-polyhedra) we apply two view-points: interior and exterior.

Exterior Description

A fan Σ in V is a collection of cones, such that the intersection of any two cones of Σ is a cone of Σ and it is a common face of both, and the union of all cones in Σ is equal to V.

A convex function f on V^* is *compatible* with a fan Σ if f is linear on every cone σ from Σ. In this case, it is easy to show that the subdifferential $\partial(f)$ is a polytope.

$$\partial(f) := \{x \in V \mid x(p) \leq f(p) \; \forall \, p \in V^*\}. \tag{1}$$

More precisely, let σ be a full-dimensional cone of the fan Σ; denote by v_σ a (unique) linear function on the space V^*, which coincides with f on the cone σ. Then v_σ (being considered as an element of V) is a vertex of the polytope $\partial(f)$ and all vertices of the polytope are of that form. In particular, a polytope P is integer if and only if its support function $\phi(P, \cdot)$ has integer values in integer points.

The support function of any polytope P is compatible with the normal fan $\mathcal{N}(P)$. Given a point $x \in P$, the following cone in the dual space V^*

$$\mathrm{Con}^*(P, x) = \{p \in V^*, p(x) \geq p(y) \; \forall y \in P\}$$

is said to be the *cotangent* cone to P at x. The collection of all cotangent cones $\mathrm{Con}^*(P, x)$, $x \in P$, forms the *cotangent fan* (or the *normal fan*) $\mathcal{N}(P)$ of the polytope P. Cones of the normal fan $\mathcal{N}(P)$ one-to-one correspond to faces of P.

Let $\mathcal{R}$ be a unimodular system. Element r of $\mathcal{R}$ can be identified with morphisms of $\mathbb{A}_1$ to $\mathcal{R}$. Conversely, morphisms of $\mathcal{R}$ to $\mathbb{A}_1$ are called *coroots*. In other words, a co-root is a homomorphism of groups $\phi\colon M \to \mathbb{Z}$ such that $|\phi(r)| \leq 1$ for any root $r \in \mathcal{R}$. The set of co-roots is denoted by $\mathcal{R}^*$.

Consider the pure system $\mathcal{U}^\perp = \mathcal{U}(\mathcal{R})^\perp$, it consists of hyperplanes $H_r(0) = (\mathbb{R}r)^\perp$, $r \in \mathcal{R}$, and all possible intersections of these hyperplanes. The hyperplanes cut the space V^* onto a finite number of cones (*cameras*) which constitute the fan $\Sigma(\mathcal{R})$. One-dimensional flats of $\mathcal{U}^\perp$ are called *crossings* as well as their primitive generators. (Of course, the crossings exist only if the unimodular system $\mathcal{R}$ is of full dimension.) A crossing is a surjective homomorphism of Abelian groups $\xi\colon M \to \mathbb{Z}$ such that the kernel of ξ is a flat of $\mathcal{R}$. Let us denote by $\mathcal{R}^\vee$ the set of crossings.

Lemma 1 ([8]). $\mathcal{R}^\vee \subset \mathcal{R}^*$.

The support function to an $\mathcal{R}$-polytope is compatible with the fan $\Sigma(\mathcal{R})$ and is uniquely determined by its restriction on $\mathcal{R}^\vee$. This follows from the following

Proposition 5. *Let $\mathcal{U}$ be a pure system in V, and let $P \subset V$ be a convex polytope. The following assertions are equivalent*:

(a) *P is a $\mathcal{U}$-convex polytope*;
(b) *the normal fan $\mathcal{N}(P)$ consists of $\mathcal{U}^\perp$-cones.*

Therefore, the support function f to an $\mathcal{R}$-polytope P is characterized by the family of real numbers $(f(\xi), \xi \in \mathcal{R}^\vee)$. However, the values $f(\xi)$, $\xi \in \mathcal{R}^\vee$ are not arbitrary. Being values of a convex function (the support function to a convex set is a convex homogeneous function on V^*), they have to satisfy some kind of "submodularity" relations.

These relations may be divided into two groups. The first group of relations addresses the functions' linearity on each cone of the fan. The second group of the relations yields convexity. Let us formulate these relations more explicitly:

I. Suppose that crossings $\xi_1, \dots, \xi_m \in \mathcal{R}^\vee$ belong to a cone $\sigma \in \Sigma(\mathcal{R})$. Then any linear relation $\sum_i \alpha_i \xi_i = 0$ should imply the similar relation $\sum_i \alpha_i f(\xi_i) = 0$.
Of course, if the cone σ is simplicial (as in the case of $\mathbb{A}_n$), these relations disappear.

II. Suppose that we have two adjacent (full-dimensional) cones σ and σ' of the fan, separated by a wall τ. Let τ be spanned by the crossings $\xi_1, \dots, \xi_m$, and let ξ, ξ' be crossings from σ, σ' respectively, which do not belong to the wall τ. Then any relation $\alpha\xi + \alpha'\xi' = \sum_i \alpha_i \xi_i$, where $\alpha, \alpha' > 0$, implies the relation $\alpha f(\xi) + \alpha' f(\xi') \geq \sum_i \alpha_i f(\xi_i)$. (According to Lemma 1, we can assume that $\alpha = \alpha' = 1$.)

Example 8 (Generalized polymatroids)**.** Let us check that the class of generalized polymatroids in $(\mathbb{R}^N)^*$ coincides with the class of $\mathbb{A}_N$-polyhedra. The arrangement $\mathcal{A}(\mathbb{A}_N)$ consists of hyperplanes $p(i) = 0$, $i \in N$, and $p(i) = p(j)$, $i, j \in N$. The collection of vectors $\{\pm \mathbf{1}_S, S \subset N\}$ is the set of crossings. Cones of $\Sigma(\mathbb{A}_n)$ are in a one-to-one correspondence with pairs of orders $(\preceq_W, \preceq_{W'})$ on partitions (W, W') of N. These partitions derive from the partitions of coordinates in non-negative and negative parts; W denotes the nonnegative coordinates of vectors of a cone, whereas W' denotes the negative ones.

Now let f be a convex function on $\mathbb{A}_N$, which is compatible with the fan $(\Sigma(\mathbb{A}_N))$. Consider the following two functions a and b on 2^N: $a(S) := -f(-\mathbf{1}_S)$ and $b(S) := f(\mathbf{1}_S)$ for $S \subset N$. There are three kinds of relations between crossings: $\mathbf{1}_S + \mathbf{1}_T = \mathbf{1}_{S\cup T} + \mathbf{1}_{S\cap T}$, $-\mathbf{1}_S - \mathbf{1}_T = -\mathbf{1}_{S\cup T} - \mathbf{1}_{S\cap T}$, and

$$\mathbf{1}_S + (-\mathbf{1}_T) = \mathbf{1}_{S-T} + (-\mathbf{1}_{T-S}). \tag{2}$$

The first two yield submodularity of b and supermodularity of a, respectively, while the third yields the following inequalities

$$b(S) - a(T) = f(\mathbf{1}_S) + f(-\mathbf{1}_T) \geq f(\mathbf{1}_{S-T}) + f(-\mathbf{1}_{T-S}) = b(S - T) - a(T - S). \tag{3}$$

Thus, the pair (b, a) is a strong pair in the sense of [11]. The corresponding polyhedron ∂f is given by the inequalities

$$a(S) \leq x(S) \leq b(S),$$

where $S \subset N$ and, by definition, ∂f is a generalized polymatroid.

Conversely, we can extend any strong pair (b, a) to a convex function on $\mathbb{R}^N$ compatible with the fan $\Sigma(\mathbb{A}_N)$. Thus, the class of (bounded) generalized polymatroids coincides with the class of $\mathbb{A}_N$-polytopes. Similarly, the class of all generalized polymatroids coincides with the class of $\mathbb{A}_N$-polyhedra, and the class of integer generalized polymatroids coincides with the class of integer $\mathbb{A}_N$-polytopes.

The normal fan of an $*\mathcal{R}$-polyhedron is spanned by $\mathcal{R}$-cones. Therefore a polyhedron $x(r) \leq b_r$, $r \in \mathcal{R}$, $b_r \in \mathbb{Z}$ is an $*\mathcal{R}$-polyhedron (no submodularity condition!). However, a $\mathcal{R}^\vee$-polytope might not be an $*\mathcal{R}$-polytope.

Laminarization

Recall, that, for a unimodular set $\mathcal{R} \subset \mathbb{Z}^n$, we are interested in the polyhedral DC-classes $\mathcal{R}$-polyhedra (S-class) and $*\mathcal{R}$-polyhedra (I-class).

How do we construct such polyhedra? The class of $\mathcal{R}$-polyhedra is stable under summation and, for example, contains the $\mathcal{R}$-zonotopes. An $\mathcal{R}$-zonotope is a sum of line segments directed parallel to roots, $Z = \sum_{r\in\mathcal{R}}[0, a_r r]$, $a_r \in \mathbb{Z}_+$, $r \in \mathcal{R}$.

We have

Proposition 6. *A polytope P is $\mathcal{R}$-convex if and only if there exists a polytope P' such that $P + P'$ is an $\mathcal{R}$-zonotope.*

Let us note that despite the fact that the definition of the zonotope is transparent, and so is its structure (a projection of a unit cube), but the number of its vertices might be huge, for example, an $\mathbb{A}_n$-zonotope has $n!$ vertices.

In a general case, the set $\mathcal{R}^\vee$ is not a unimodular system. However, if we can find a unimodular system $\mathcal{Q}$ in $\mathcal{R}^\vee$ such that its crossings form a subset in $\mathcal{R}$ (we call this a laminarization of $\mathcal{R}$), this will bring us an advantage. Indeed, in such a case any $*\mathcal{Q}$-polyhedron is an $\mathcal{R}$-polyhedron, and therefore a polyhedron defined by systems of linear inequalities

$$\{v \in V, \xi(v) \le a(\xi), \xi \in \mathcal{Q}\}$$

is an $\mathcal{R}$-polyhedron for arbitrary "right parts" $a(\xi)$, $\xi \in \mathcal{Q}$. Of course, when the $a(\xi)$ are integer, the corresponding polyhedron is integer too. Let us give a more particular realization of this idea.

Example 9. A family $\mathcal{T}$ of subsets of a finite set N is called *laminar* if for any $A, B \in \mathcal{T}$, either $A \subset B$, or $B \subset A$, or $A \cap B = \varnothing$. Without loss of generality we can assume that any singleton belongs to $\mathcal{T}$.

Let $\mathcal{T}$ be a laminar family. We assert that the set $\mathcal{Q} = \{\pm\mathbf{1}_T, T \in \mathcal{T}\}$ is a unimodular set in the space $\mathbb{R}^N$. That is, $\mathcal{Q}$ is a laminarization of the system $\mathbb{A}_N$. Since the orthogonal hyperplanes $(\mathbf{1}_T)^\perp$ are $\mathbb{A}_N$-flats, we have to check that any intersections of such hyperplanes are also $\mathbb{A}_N$-flats.

Let us recall (Example 5 from Lecture 1) that an $\mathbb{A}_N$-flat has the form

$$F(A_1, \dots, A_k) := \{x \in (\mathbb{R}^N)^*, x(A_j) = 0 \text{ for } j = 1, \dots, k\},$$

where $A_1, \dots, A_k$ are disjoint subsets of N. (The codimension of $F(A_1, \dots, A_k)$ is equal to a number of non-empty A_j-s.) In particular, the hyperplane $(\mathbf{1}_T)^\perp$ is $F(T)$. Let us show that the intersection of hyperplanes $F(T_1), \dots, F(T_k)$, where $T_J \in \mathcal{T}$, has a form $F(A_1, \dots, A_k)$. For this we write A_j explicitly. Namely, A_j is equal to T_j minus the union of those of the T_i which are contained in T_j. Indeed, using the laminarity of $\mathcal{T}$, we can assume that the T_i-s do not intersect. Therefore, the vanishing $x(T_j)$-s are equivalent to the vanishing $x(A_j)$-s.

In particular, for a laminar family $\mathcal{T}$ in N, the polyhedron defined by the inequalities

$$a(S) \le x(S) \le b(S), \quad S \in \mathcal{T},$$

is an $\mathbb{A}_N$-polyhedron for any functions $a, b\colon \mathcal{T} \to \mathbb{R} \cup \{\infty\}$, and is an integer $\mathbb{A}_N$-polyhedron for integer-valued a and b.

Dicing

In order to represent integer $*\mathcal{R}$-polyhedra visually, it is convenient to use the notion of a dicing. A dicing is the following regular polyhedral decomposition of V^*. Let us consider the following counting (but locally finite) collection of hyperplanes $H_r(a) = \{p \in V^*,\ p(r) = a\}$, where $r \in \mathcal{R}$ and $a \in \mathbb{Z}$. These hyperplanes dissect the space V^* on connected parts, the *regions* of the dicing. Regions are bounded sets if $\mathcal{R}$ is of full dimension. Closure of regions, as well as their faces, are called *chambers* of the dicing. The set $\mathcal{D}(\mathcal{R})$ of the chambers form a polyhedral decomposition of V^*, that is the chambers meet only their faces and cover the whole space V^*. If $\mathcal{R}$ is of full dimension then nodes of the dicing (that is zero-dimensional chambers) are integer points of V^*.

Each chamber of the dicing $\mathcal{D}(\mathcal{R})$ is an integer $*\mathcal{R}$-polyhedron. Conversely, any integer $*\mathcal{R}$-polyhedron is a union of chambers of $\mathcal{D}(\mathcal{R})$. Thus, an integer $*\mathcal{R}$-polyhedron is none other than a convex set composed from some chambers. Note that for the unimodular system $\mathbb{A}_n$, the number of different types of full-dimensional chambers is equal to $n!$.

Example 10. Let us consider the *dicing star* $\mathbf{St}(\mathcal{R})$. It is composed from those chambers of the dicing $\mathcal{D}(\mathcal{R})$, which contain the origin 0. In order to establish the convexity of $\mathbf{St}(\mathcal{R})$, we show that:

$$\mathbf{St}(\mathcal{R}) = \{p \in V^*, r(p) \le 1, \text{ where } r \in \mathcal{R}\}.$$

For the time being, we call $\mathbf{St}'$ the polyhedron appearing on the right hand of the formula. Obviously, any chamber which contains 0 belongs to $\mathbf{St}'$. Hence $\mathbf{St}(\mathcal{R}) \subset \mathbf{St}'$.

Conversely, let $p \in \mathbf{St}' \setminus \mathbf{St}(\mathcal{R})$. Assume we move from p to 0 along the segment $[0, p]$. At some time t, $0 < t < 1$, the point tp will be on the boundary of $\mathbf{St}(\mathcal{R})$. Hence, there exists $r \in \mathcal{R}$ with $r(tp) = 1$. This implies that $r(p) = 1/t > 1$, a contradiction.

From this description of $\mathbf{St}(\mathcal{R})$ we see that integer points of $\mathbf{St}(\mathcal{R})$ are the coroots of $\mathcal{R}$,

$$\mathbf{St}(\mathcal{R})(\mathbb{Z}) = \mathcal{R}^*.$$

Conversely, $\mathbf{St}(\mathcal{R}) = \mathrm{co}(\mathcal{R}^*)$.

It is not difficult to check that the convex hull $\mathrm{co}\{r, r \in \mathcal{R}\}$ is a $\mathcal{R}$-polytope iff

$$\mathcal{R}^\vee = \mathcal{R}^*.$$

That is the case for $\mathcal{R} = \mathbb{A}_n$, but not the case, for example, for systems $\mathbb{D}_4$ and $\mathbb{E}_5$. For each of these systems the crossings constitute a proper subset of coroots.

Problem. To describe unimodular sets with the property $\mathcal{R}^\vee = \mathcal{R}^*$.

Interior Description

For a pseudoconvex set $X \subset M$, we define the tangent cone $\mathrm{Con}(X, x)$ as the set of vectors v such that $x + tv \in \mathrm{co}(X)$ for some $t > 0$.

Theorem 3. *A subset $X \in \mathbb{Z}^n$ is $\mathcal{R}$-convex if and only if, for any $x \in X$, the following two conditions are fulfilled*

(A) *the tangent cone* $\mathrm{Con}(X, x)$ *is $\mathcal{R}$-convex, i.e.,* $\mathbb{R}_+(\mathrm{Con}(x, X) \cap \mathcal{R}) = \mathrm{Con}(x, X)$;
(B) $x + (\mathrm{Con}(X, x) \cap \mathcal{R}) \subset X$, *that is for any $x \in X$ and every root from the tangent cone $r \in \mathrm{Con}(X, x) \cap \mathcal{R}$, the point $x + r$ belongs to X.*

For a set of points $X \subset M$, we can define an $\mathcal{R}$-convex hull of X, $\mathcal{R}$-$co(X)$, as an $\mathcal{R}$-convex set Y, such that $X \subset Y$, and there is not an $\mathcal{R}$-convex set $Z \neq Y$ with $X \subset Z \subset Y$. One can check that, for any set there exists an $\mathcal{R}$-convex hull, but there exists sets with non-unique convex hull. Denote by $\mathrm{CH}_{\mathcal{R}}(X)$ the set of all convex hulls of X.

The following criterion of $\mathcal{R}$-convexity holds true.

Proposition 7. *A set $X \subset M \cong \mathbb{Z}^n$ is $\mathcal{R}$-convex if and only if together with every pair of points $x, y \in X$ there exists $\mathcal{R}$-segment $Z \in \mathrm{CH}_{\mathcal{R}}(\{x, y\})$ such that $Z \subset X$.*

An interior description of $*\mathcal{R}$-convex sets is dual to Theorem 3.

Theorem 4. *A subset $Z \subset M^*$ is $*\mathcal{R}$-convex if and only if for any $p \in Z$, the following two conditions are fulfilled.*

(A) *The cone* $\mathrm{Con}(Z, p)$ *is $*\mathcal{R}$-convex*;
(B) $p + \mathrm{Con}(Z, p) \cap \mathcal{R}^\vee \subset Z$.

Recall that the fan $\Sigma(\mathcal{R})$ is the fan of an arrangement of hyperplanes

$$r^\perp = \{\xi \in V^* \mid r(\xi) = 0\}, r \in \mathcal{R}.$$

Note that $\Sigma(\mathcal{R})$ decomposes V^* in cones of the form $\sigma(p)$, $p \in V^*$. The cone $\sigma(p)$ consists of vectors $q \in V^*$ which are *comonotone* with p, i.e., we have that if $p(r) \geq 0$ for $r \in \mathcal{R}$, then $q(r) \geq 0$. Thus, we have

$$\sigma(p) = \{q \in V^* \mid p(r) \geq 0,\ r \in \mathcal{R} \Rightarrow q(r) \geq 0\}.$$

So $\sigma(p)$ is $*\mathcal{R}$-polyhedron, and moreover, this is the minimal $*\mathcal{R}$-convex cone which contains p.

The class of $*\mathcal{R}$-convex sets is stable under intersection. For a set $Z \subset M^*$, the intersection of all $*\mathcal{R}$-convex sets which contains Z is said to be the $*\mathcal{R}$-*convex hull* of Z, $*\mathcal{R}$-co(Z). Of course, such a convex hull $*\mathcal{R}$-co(Z) is the set of integer solutions to the following inequalities

$$r(\cdot) \leq \max_{p \in Z} r(p), \quad r \in \mathcal{R}. \tag{4}$$

In particular, the convex hull of a pair of points x and $y \in M^*$, $*\mathcal{R}$-co($\{x, y\}$), is the $*\mathcal{R}$-segment joining x and y. Contrary to the case of $\mathcal{R}$-segments, the $*\mathcal{R}$-segment is uniquely defined for any pair of points, and the following holds

$$*\mathcal{R}\text{- co}(\{x, y\}) = (\{x + \sigma(y - x)\} \cap \{y + \sigma(x - y)\}) \cap M^*, \tag{5}$$

where $\sigma(p)$ is the minimal cone of the fan $\Sigma(\mathcal{R})$ that contains $p \in V^*$.

Now, suppose a set $Z \subset M^*$, with any pair of points $p, q \in Z$, contains their $*\mathcal{R}$-segment $*\mathcal{R}$-co($\{p, q\}$). Does this imply that Z is $*\mathcal{R}$-convex? This is true for unimodular systems of $\mathbb{A}_n$-type, but not true in general. Here is an example.

Example 11. Consider the unimodular set $\mathbb{D}_4$. The fan $\Sigma(\mathbb{D}_4)$ is not simplicial. Specifically, a cone constituted from the set of linear functionals being positive on the roots $\{e_1, e_2, e_4, -e_1 - e_3, -e_3 - e_4, e_1 + e_2 + e_3 + e_4\}$ is not simplicial. This cone is generated by the following crossings (functionals) $\xi_1 = x_2 - x_3$, $\xi_2 = x_1 - x_3 + x_4$, $\xi_3 = x_2$, $\xi_4 = x_1 - x_3$ and $\xi_5 = -x_3 + x_4$. The plane spanned by ξ_1 and ξ_2 is not $*\mathcal{R}$-convex, because ξ_1 is annihilated by e_1, e_4 and $e_1 + e_2 + e_3 + e_4$, ξ_2 is annihilated by $e_2, -e_1 - e_3$ and $-e_3 - e_4$. The difference $\xi_2 - \xi_1 = x_1 - x_2 + x_4$ is also a crossing, and it is annihilated by e_3, $e_1 + e_2$ and $e_2 + e_4$. We exhausted all roots and there are no roots left to annihilate the plane. The triangle co$(0, \xi_1, \xi_2)$ has edges in $*\mathbb{D}_4$, but it is not $*\mathbb{D}_4$-convex.

Remark. The pure system $\mathcal{U}(\mathbb{D}_4)$ is maximal unimodular system, but it is not a maximal pure system. Namely we can add the orthogonal plane to the plane spanned ξ_1 and ξ_2 to the set of generators of $\mathcal{U}(\mathbb{D}_4)$ and get a bigger pure system $\widehat{\mathbb{D}}_4$.

However, if we will require that a set $Z \subset M^*$ and all its multipliers kZ, $k = 1, 2, \ldots$, possess such a property, then Z (and of course all its multipliers) will be $*\mathcal{R}$-convex.

Specifically, we have

Theorem 5. *A set $Z \subset M^*$ is $*\mathcal{R}$-convex iff for every $k = 1, \ldots$, any $p, q \in kZ$ and every $\xi \in \sigma(q - p) \cap \mathcal{R}^\vee$, the following condition holds*

$$p + \xi \text{ and } q - \xi \text{ belong to } kZ.$$

Lecture 3

Discrete Convex Functions

Pseudo-convexity

Let f be a function on the lattice $\mathbb{Z}^n$ (or on some subset $D \subset \mathbb{Z}^n$). Similarly to the case with sets in $\mathbb{Z}^n$, the pseudoconcavity is the first approximation to discrete concave functions. Specifically, an affine function l is called a *superdifferential* of f at a point x if $l \geq f$ and $l(x) = f(x)$.

Proposition 8. *For a function $f\colon \mathbb{Z}^n \to \mathbb{R} \cup \{-\infty\}$, the following two properties are equivalent*:

1. *f has a superdifferential at each point of its domain*;

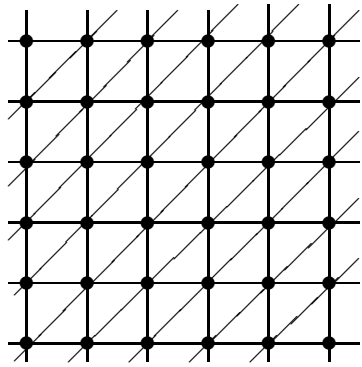

Figure 1. Standard triangulation of $\mathbb{R}^2$ with vertices in $\mathbb{Z}^2$

2. *f is the restriction to $\mathbb{Z}^n$ of a concave function defined on $\mathbb{R}^n$.*

For pseudoconvex function the separation theorem does hold in general. Here is an example: the function $f\colon \{0,1\}^2 \to \mathbb{Z}$, defined by $f(0,0) = f(0,1) = f(1,0) = 0$, $f(1,1) = 1$ is pseudoconcave. Let $g = f$. Then f is pseudoconvex, g is pseudoconcave, and $g \geq f$, but we can not separate these functions by an affine function.

Before going on to explain the general theory, we begin with examples of discrete convex functions in dimensions one, two and three.

Dimension One

The class of pseudoconcave functions constitutes the class of discrete concave functions. This class is stable under summation and the convolution.

Recall, that the *convolution* $f \star g$ of functions f and g is defined by

$$(f \star g)(z) = \inf_{z=x+y} (f(x) + g(y)),$$

where the infimum is taken over all decompositions of $z = x + y$ with $x \in \operatorname{dom} f$ and $y \in \operatorname{dom} g$.

Dimension Two

In order to define such functions, let us cut the plane $\mathbb{R}^2$ into parts by lines of three sorts:

$$x = a,\ a \in \mathbb{Z}; \quad y = b,\ b \in \mathbb{Z}; \quad x - y = c,\ c \in \mathbb{Z}.$$

We obtain a standard triangulation of the plane (see Figure 1).

Let us extend a function $f\colon \mathbb{Z}^2 \to \mathbb{R}$ over each triangle by linearity; as a result of such a linear interpolation, we obtain a function $\tilde{f}\colon \mathbb{R}^2 \to \mathbb{R}$. We say that the function f is discretely concave if $\tilde{f}$ is a concave function on $\mathbb{R}^2$.

The concavity is equivalent to the rhombus conditions: there are three types of rhombuses which might occur.

This leads to

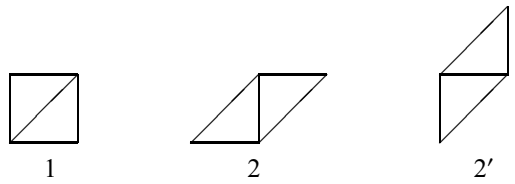

Figure 2. Three types of rhombuses of the form of unions of two adjoin triangles of the standard triangulation

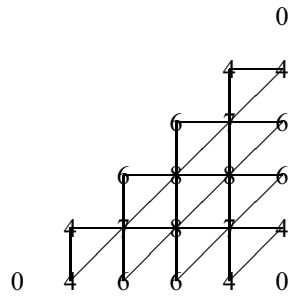

Figure 3. Affinity areas of the function $nx - (x^3 + y^3)/(x + y)$ on the grid $\Delta(n)$ for $n = 5$

Definition. A function f defined on the set H of integer points of a convex set constituted from triangles of the above type is *discretely concave* (*or a DC-function*) if, for any integers i and $j \in H$, the following inequalities hold:

(i) $f(i, j) + f(i + 1, j + 1) \geq f(i + 1, j) + f(i, j + 1)$;
(ii) $f(i + 1, j) + f(i + 1, j + 1) \geq f(i, j) + f(i + 2, j + 1)$;
(iii) $f(i, j + 1) + f(i + 1, j + 1) \geq f(i, j) + f(i + 1, j + 2)$.

For example, Any affine function is discretely concave. For such a function, all inequalities (i)–(iii) are equalities indeed.

There exist more interesting quasi-separable functions. Suppose we are given three discretely concave functions f, g, and h on one (integer) variable. Then, the function

$$F(x, y) = f(x) + g(y) + h(x - y),$$

on two (integer) variables is, obviously, a DC function.

If we let f, g, and h be equal to the same function $y(k) = k(n - k)/2$, then

$$F_0(x, y) = [x(n - x) + y(n - y) + (x - y)(n - x + y)]/2 = nx - (x^3 + y^3)/(x + y)$$

is a DC function on the triangle $\Delta(n) = \{(i, j) \in \mathbb{Z}^2, 1 \leq y \leq x \leq n\}$. Figure 3 shows this function and its affinity areas for $n = 5$.

Though, there are many nonseparable DC functions.

We can complete characterize discrete concave function defined on integer points of the positive octant $\mathbb{Z}^2_+$. Namely, let M be a module of finite length over the discrete valuation ring $K[\![T]\!]$, that is, in fact, a finite-dimensional K-space with nilpotent action of an operator T. Assume that we are given submodules N_1 and N_2. Define a function $c = c(M; N_1, N_2) : \mathbb{Z}^2_+ \to \mathbb{Z}$ by the rule

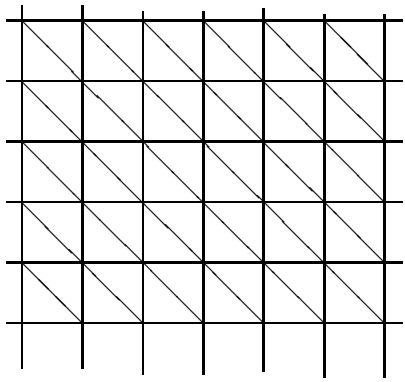

Figure 4. Dual triangulation of the standard one

$$c(k_1, k_2) = \dim(M/T^{k_1}N_1 + T^{K_2}N_2).$$

We have the following characterization.

Theorem 6 ([16])**.** 1. *The function c is discrete concave.*
2. *Let* $c\colon \mathbb{Z}^2_+ \to \mathbb{Z}$ *be a monotone discrete concave functions with* $c(0) \geq 0$*. Then there exist a module M and submodules* N_1 *and* N_2 *such that* $c = c(M; N_1, N_2)$*.*

An old question that goes back to 19th century: assume we know spectra of Hermitian (symmetric) matrices A and B (of size n); what are the possible values of the spectrum of their sum $C = A + B$? For a history on solution of this problem, called the Horn problem, see a nice paper by Fulton [17]. The crucial step was made by Klyachko [18] and by Knutson and Tao [19] in proving the Horn conjecture. Klyachko's proof is based on difficult theorems from algebraic geometry. Danilov and I ([20]) gave almost combinatorial solution to this problem. Namely, we formulate the answer in the form: $\mathrm{Sp}\, A$, $\mathrm{Sp}\, B$ and $\mathrm{Sp}\, C$ can be spectra of matrices A, B and $C = A + B$ if and only if there exists a discrete concave function on the triangle grid with vertices $(0, 0)$, $(n, 0)$, and (n, n), such that the restriction of this function to the sides of the triangle have increments $\mathrm{Sp}\, A$, $\mathrm{Sp}\, B$ and $\mathrm{Sp}\, C$, respectively. We formulated a conjecture is that there should be a canonical bijection between the pairs of Hermitian matrices A, B and DC-functions on the triangle grids having increments $\mathrm{Sp}\, A$, $\mathrm{Sp}\, B$ and $\mathrm{Sp}(A + B)$ (this conjecture is proven in several cases, for details see [20]).

In dimension two, there is another "conjugate" class of functions which have the affinity areas of the form of 'rotated' hexagons. This class is isomorphic to the class of DC-functions with respect to the unimodular transformation $\left(\begin{smallmatrix} 1 & -1 \\ 0 & 1 \end{smallmatrix}\right)$. A function of the conjugate class is coherent with the dicing depicted in Figure 3, that is a function $f\colon \mathbb{Z}^2 \to \mathbb{R}$ of this class being interpolated by affinity on each triangle of the dicing $\mathbb{R}^2$ with the lines $x = a$, $y = b$, $x + y = c$, $a, b, c \in \mathbb{Z}$ (see Figure 4) yields a concave function $\tilde{f}\colon \mathbb{R}^2 \to \mathbb{R}$.

In other words, such a function satisfies the following three types of inequalities (for each type of rhombus composed from the triangles of the dicing):

(i)′ $f(i, j) + f(i + 1, j + 1) \leq f(i + 1, j) + f(i, j + 1)$;
(ii)′ $f(i + 1, j) + f(i + 1, j + 1) \geq f(i, j + 1) + f(i + 2, j)$;

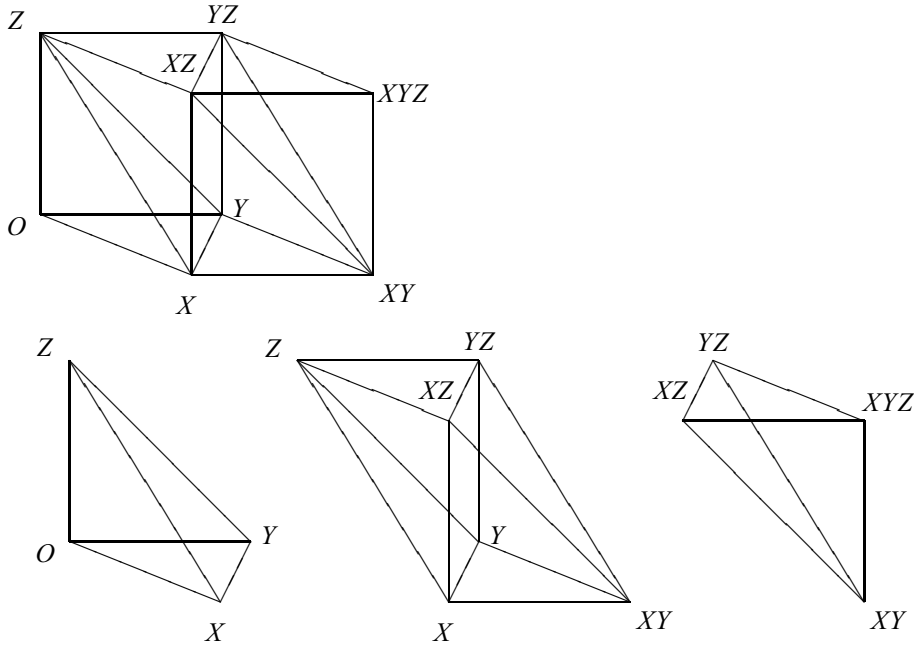

Figure 5. Decomposition of a cube into union of three g-polymatroids, two simplices and one octahedron

(iii)′ $f(i, j + 1) + f(i + 1, j + 1) \geq f(i + 1, j) + f(i, j + 2)$.

The functions of this class have affinity areas from the class of 2-dimensional generalized polymatroids (hexagons).

The class of integer-valued DC-functions and this class of conjugate functions are Fenchel conjugate to each other. Recall, that a function

$$f^*(p) = \max_{x \in \mathbb{Z}^n} f(x) - p(x), \quad p \in (\mathbb{Z}^n)^*,$$

is Fenchel dual to a function $f : \mathbb{Z}^n \to \mathbb{Z}$.

For 2-dimensional DC-functions the separation theorem holds true. The reason is: the intersection of any two triangles of the dicing is an integer polytope.

Lecture 4

Dimension Three

In this case we also propose a direct construction of functions with affinity areas being generalized polymatroids (in $\mathbb{Z}^3$). Let us cut the space $\mathbb{R}^3$ into parts by planes of four sorts: $x = a$, where a are integers; $y = b$, where b are integers, $z = c$, where c are integers; and $x + y + z = d$, where d are integers. The tiles of this dicing are of three types: two types of simplexes and the octahedron (see Figure 5).

These two simplexes and the octahedron (as well as their integer translations) are g-polymatroids. Halves of the octahedron also g-polymatroids, that is if we cut the oc-

tahedron by the wall (X, Y, XZ, YZ) we will obtain two halves $co(X, Y, XZ, YZ, Z)$ and $\mathrm{co}(X, Y, XZ, YZ, XY)$, which are g-polymatroids. The same holds for cutting either by the wall (Z, XZ, YZ, XY), or by the wall (Z, ZY, X, XY).

Now we proceed similarly to the case of dimension two. Namely, we extend a function $f\colon \mathbb{Z}^3 \to \mathbb{R}$ to a function $\tilde{f}\colon \mathbb{R}^3 \to \mathbb{R}$ by interpolating it on each simplex and octahedron. In order to have a correct interpolation to the octahedron the following condition must hold:

At least two of the sums of the values of a function f at the endpoints of the diagonals of the octahedron, $A = f(X) + f(YZ)$, $B = f(XZ) + f(Y)$, $C = f(Z) + f(XY)$, have to be equal, that is either $A = B$, or $B = C$, or $A = C$.

If this condition does not hold in some octahedron, we can not interpolate the function. We suppose validity of this condition.

In order to get a concave interpolation $\tilde{f}$, the *octahedron condition* has to hold:

(OCT) *at least two of values $A = f(X) + f(YZ)$, $B = f(XZ) + f(Y)$, $C = f(Z) + f(XY)$ are equal and they are not less than the third one.*

For example, the case $A = C \geq B$ corresponds to the case of linearity of $\tilde{f}$ on halves $\mathrm{co}(Z, ZY, X, XY, Y)$ and $\mathrm{co}(Z, ZY, X, XY, XZ)$ of the octahedron. However, two-dimensional conditions have to be added. Namely, simplices from adjoint cubes which share a common edge and simplices adjoining to an octahedron give rhombuses; for concavity of $\tilde{f}$ the corresponding rhombus-type inequality must hold. It is easy to see that all these rhombuses are located on planes $x = a$, $y = b$, $z = c$, $x + y + z = d$, $a, b, c, d \in \mathbb{Z}$. Thus, we come to

Definition. A function $f\colon \mathbb{Z}^3 \to \mathbb{R}$ is a *polymatroidal concave* function if its restrictions to any plane of the form $x = a$, or $y = b$, or $z = c$, or $x + y + z = d$ for integers a, b, c, d, satisfy the rhombus inequalities (i)′„ (ii)′„ (iii)′, and in each octahedron the octahedron condition (OCT) holds true.

Examples of such functions:

$$f(x) + g(y) + h(z) + f'(x + y) + k'(x + y + z),$$

with concave one dimensional functions $f, g, \ldots$;

$$f(x) + g(y) + h(z) + g'(y + z) + k'(x + y + z);$$
$$f(x) + g(y) + h(z) + h'(z + x) + k'(x + y + z);$$

convolutions of these functions.

In the class of the polymatroidal functions, the separation theorem holds true. The reason: the intersection of two "tiles" is an integer polytope (three-dimensional variant of the Edmonds theorem [12] on the intersection of polymatroids).

This class is stable under convolution, but is not stable under summation. The main point is that the intersection of two halves of the octahedron being the quarter of the octahedron is not a g-polymatroid.

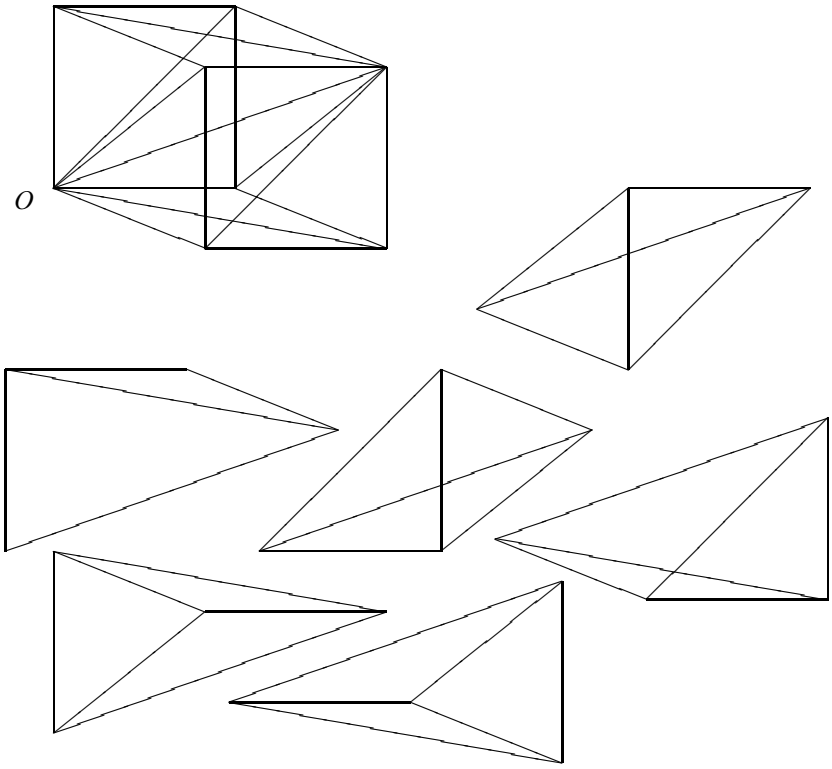

Figure 6. Standard triangulation of a cube

Dual Class of Functions: *The Class of* $*\mathbb{A}_3$*-Concave Functions*

Let us dice $\mathbb{R}^3$ by the hyperplanes $x = a, y = b, z = c, x - y = a', x - z = b', y - z = c'$ with integers $a, b, \ldots$. This dicing cuts the unit cube into 6 simplices (see Figure 6)

Now, we interpolate a function $f\colon \mathbb{Z}^3 \to \mathbb{R}$ to each simplex of this dicing. In order to get a concave function, the submodularity condition must hold. Specifically, a pair of adjoint simplices (in the same cube) gives the submodularity condition: for example, the simplices (O, Z, XZ, XYZ) and (O, Z, XZ, XYZ) are adjoint (the common wall is (O, Z, XYZ)), and the condition is $f(XZ) + f(YZ) \le f(Z) + f(XYZ)$.

Because there are adjoint simplices from adjoint cubes (Figure 7) we need more conditions.

This type of adjacency gives the following type of concavity conditions: for each basis vector e_i, $i = 1, 2, 3$, there holds

$$f(x) + f(x + e_i + \mathbf{1}) \le f(x + e_i) + f(x + \mathbf{1}).$$

The supermodularity together with these conditions characterize another class of functions, $*\mathbb{A}_3$-concave functions.

This class is stable under summation and possesses the separation property (intersection of a pair of tiles of the dicing is a tile of the dicing). However, this class of functions is not stable under convolution.

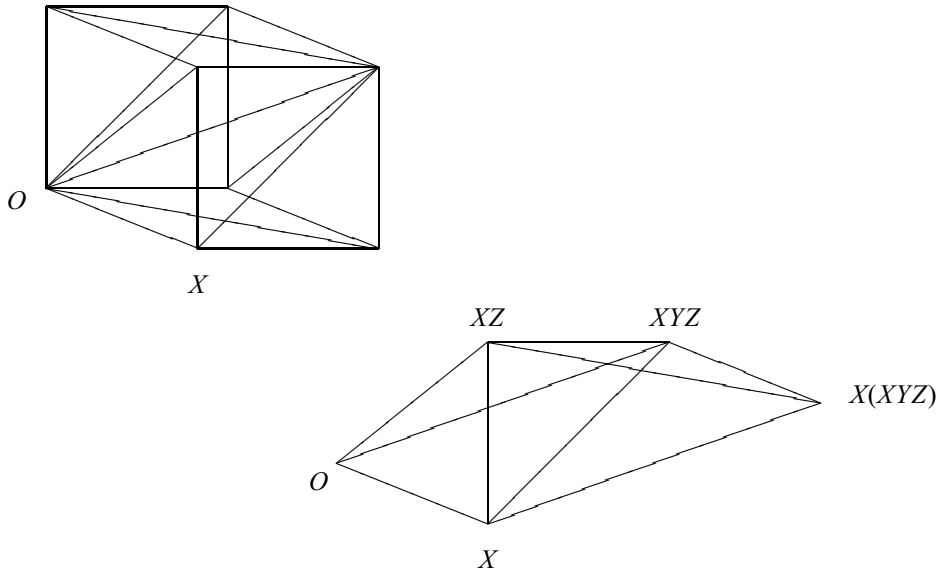

Figure 7. Two ajoint simplexes from standard triangulation of two adjoin cubes

An example of $*\mathbb{A}_3$-concave functions are separable functions. Less trivial examples come from quadratic functions. A function

$$f(x_1, \ldots, x_n) = \sum_{ij} a_{ij} x_i x_j$$

is an $*\mathbb{A}_n$-function if and only if the following two conditions are met:

(a) $a_{ij} \geq 0$ for $i \neq j$, and
(b) $\sum_j a_{ij} \leq 0$ for $i = 1, \ldots, n$.

In particular, the two-parameter function $f(x, y) = -ax^2 + 2bxy - cy^2$ is an $*\mathbb{A}_2$-function iff $0 \leq b \leq \min(a, c)$.

For integer-valued functions, the classes of $\mathbb{A}_3$-concave and $*\mathbb{A}_3$-concave functions are Fenchel dual.

Lecture 5

Higher Dimensions

The class of $*\mathbb{A}_n$-concave functions is characterized by the same inequalities as in the case of dimension three. That is, let us cut $\mathbb{R}^n$ by the hyperplanes $x_i = a_i$, $i = 1, \ldots, n$, $x_i - x_j = a_{ij}$, $i, j = 1, \ldots, n$, with integer a_i and a_{ij}. This dicing cuts each unit integer cube into $n!$ simplices (numerated by chains, or permutations). For a function $f\colon \mathbb{Z}^n \to \mathbb{R}$, we let $\tilde{f}\colon \mathbb{R}^n \to \mathbb{R}$ denote a function being obtained from the extrapolation of f by affinity on each cell of dicing.

Claim. *$\tilde{f}$ is concave iff f is supermodular, that is $f(x) + f(y) \leq f(x \vee y) + f(x \wedge y)$, where $\vee$ denotes the operation of taking the coordinate-wise maximum of vectors and $\wedge$ denotes for the coordinate-wise minimum; and, for each basis vector e_i, $i = 1, 2, \ldots, n$, at each point $x \in \mathbb{Z}^n$ there holds*

$$f(x) + f(x + e_i + \mathbf{1}) \leq f(x + e_i) + f(x + \mathbf{1}).$$

This class of functions is stable under summation, and possesses the separation property.

Here are some examples of such functions.

Let V be a finite dimensional vector space, $T\colon V \to V$ a nilpotent operator; $V_1, \ldots, V_n$ subspaces of V invariant under the action of T, that is $T\colon V_i \to V_i$, $i = 1, \ldots, n$.

Then the function $c = c(V, T; V_1, \ldots, V_n)$ defined by

$$c(k_1, \ldots, k_n) = \dim(V/T^{k_1}V_1 + \ldots T^{k_n}V_n),$$

is a monotone $*\mathbb{A}_n$-concave function. Contrary to the case $n = 2$, there exists $*\mathbb{A}_n$-concave functions not of this form (this follows from that there exists non-regular matroids (so-called the Vamos matroid [21])).

A function of the form $\varphi(\min(x_a, a \in A))$, where $A \subset [n]$, and φ is a concave function of one variable is $*\mathbb{A}_n$-concave. Being considered from the economic point of view as a utility function, is might be interpreted as a "package function," which embodies some sort of preference for complementarity between the goods.

Recall that a collection of sets $\mathcal{T} \subset [n]$ forms an *hierarchy*, if for any A and $A' \in \mathcal{T}$, the sets $A \cap A'$ and $A \cup A'$ are listed in the set $\{\varnothing, A, A'\}$.

Definition. A function f on *I agrees with a hierarchy $\mathcal{T}$* if, there exists a subcollection $\{A_t\}$ of $\mathcal{T}$, such that f equals the convolution of A_t-package-functions.

Theorem 7. *Let f be a function agreeing with a hierarchy $\mathcal{T}$. Then f is $*\mathbb{A}_n$-concave.*

A characterization of $*\mathbb{A}_n$-concave functions similar to usual concave functions.

Theorem 8. *A function $f\colon \mathbb{Z}^n \to \mathbb{R} \cup \{-\infty\}$ is $*\mathbb{A}_n$-concave if and only if, for any $p, q \in \mathbb{Z}^n$ and any crossing $\xi \in \sigma(q - p)$, the following holds*

$$f(p) + f(q) \leq f(p + \xi) + f(q - \xi).$$

Note that for the unimodular system $\mathbb{A}_n$ with respect to the standard basis, the crossings take the form $\pm\mathbf{1}_A$, $A \subset [n]$, and a crossing $\mathbf{1}_A$ belongs to $\sigma(q - p)$ iff the inequality $(q - p)(e_i - e_j) \geq 0$ implies either $i \in A$, or $i, j \notin A$.

$\mathbb{A}_n$-Concave Functions

A function $f\colon \mathbb{Z}^n \to \mathbb{R}$ is $\mathbb{A}_n$-concave if f has affinity areas of the form of $\mathbb{A}_n$-convex sets, that is (if we pick the standard basis of the Abelian group $\mathbb{Z}^n$) for any $p \in (\mathbb{R}^n)^*$, the set $\operatorname{Arg\,max}_{x \in \mathbb{Z}^n}(f(x) - p(x))$ takes the form of integer points of a generalized polymatroid.

This class of functions is stable under convolution and possesses the separation property and was studied in the book [5].

Here we propose several new characterizations of functions of this class.

We introduce for convenience reasons both an additional "fictitious" coordinate $x_0 := -\sum_i x_i$ and a "fictitious" basis vector $e_0 := 0$. By these means, we are able to write all the roots in a fully symmetric form, that is $e_i - e_j$, where i, j now belong to the extended item set $N' := [n] \cup \{0\}$.

Let us call tangent spaces to the octahedrons $\mathbb{A}_3$-*flats*, that is, a flat spanned by the vectors $\{e_i - e_j, e_k - e_l, e_i - e_l, e_k - e_j\}$, such that $|\{i, j, k, l\} \cap N'| = 4$. Then we have

Theorem 9. *A function f is an $\mathbb{A}_n$- concave function if and only if its restriction to any integer translation of any $\mathbb{A}_3$-flat is an $\mathbb{A}_3$-concave function.*

Let x and y be two integer points. Denote by $[x > y] := \{i \in N' \mid x_i > y_i\}$ and by $[x < y] := \{i \in N' \mid x_i < y_i\}$.

Theorem 10. *A function f is an $\mathbb{A}_n$- concave function if and only if, for any $x, y \in \mathbb{Z}^n$, and any $i \in [x > y]$, there exists $j \in [x < y]$ and there exists a root-path[2] which joins the points x and $y + e_i - e_j$, such that, for any vertex z of this path, we have*

$$f(y) + f(z) \leq f(y + e_i - e_j) + f(z - e_i + e_j).$$

How to Construct Such Functions

Let g be the indicator function of the positive orthant $\mathbb{Z}^n_+$ (i.e., $g(x) = 0$ for $x \in \mathbb{Z}^n_+$ and $g(x) = -\infty$ otherwise). Note that g is a PM-function (that is, a shortening for an $\mathbb{A}_n$-concave function with $\mathbb{A}_n$ being considered with respect to the standard basis). Consider now the convolution of some function f with g so defined. This is exactly the monotone extension of f. Thus the monotone extension of a PM-function is a PM-function. We now mention two particular cases.

Let f be a PM-function on the Boolean cube. Then the monotone extension of f on the positive orthant $\mathbb{Z}^n_+$ is a PM-function.

Let f be an arbitrary function on the set $\{0\} \cup \{1 \otimes i, i \in I\}$. Its monotone extension on the orthant $\mathbb{Z}^n_+$ is a PM-function.

The sum of two PM-functions, however, need not be a PM-function.

However, in one important case, summation does preserve polymatroidness, that is, when we consider discrete functions of a single variable. Then the cells of $\mathrm{co}(f)$ are "strips" of the form $a_i \leq x_i \leq b_i$. Now, the intersections of such strips with polymatroids are polymatroids as well. By induction, we have:

Proposition 9. *Let f be a discrete PM-function and g be a separable pseudoconcave function, then $f + g$ is a PM-function.*

We can make a similar statement when g depends on the single variable $x_0 = -\sum_i x_i$. With this remark in mind, we can now construct quasi-separable PM-functions by means of laminar families of subsets of $N := [n]$. Suppose we have two disjoint subsets A and B of N, and let f_A and f_B be, respectively, PM-functions of the variables in A and in B. Let ϕ be an auxiliary pseudoconcave function on $\mathbb{Z}$. Then the function

[2] We define *a root-path* to be any path in $\mathbb{Z}^n$ such that each edge of this path is parallel to a root.

$$f(x) = f_A(x_A) + f_B(x_B) + \phi(x(A \cup B))$$

is a PM-function, where x_A are the coordinates of x on A, and $x(C) = \sum_{i \in C} x_i$.

Let f be a function. Note that f' is a k-capacity constraint on f if f' is obtained from f as follows: $f'(x) = f(x)$ if $x(I) \leq k$ and $f'(x) = -\infty$ if $x(I) > k$. We can now state that if f is a discrete PM-function on the orthant on which we impose a k-capacity constraint, then f' is polymatroidal.

Economic Interpretation

Let $f : \mathbb{Z}^n \to \mathbb{R} \cup \{-\infty\}$ be a utility function of a buyer.

Definition. The function f satisfies the SWGS-property (the step-wise gross substitution) if for any $p \in (\mathbb{R}^n)^*$, for any $x \in \operatorname{Arg\,max}(f - p)$, and for any $i \in I$ either of the following two conditions hold:

(a) for any $\varepsilon \geq 0$, $x \in \operatorname{Arg\,max}(f - (p + \varepsilon \mathbf{1}_i)$,

(b) there exists $\varepsilon \geq 0$ and $y \in \operatorname{Arg\,max}(f - (p + \varepsilon\, \mathbf{1}_i)$ such that $y_i = x_i - 1$ and $y_{-i} \geq x_{-i}$.

The step-wise gross substitution condition can readily be given an economic interpretation: any decrease in the unitary demand for item i can always be compensated by an increase in the demand for the remaining items.

Proposition 10 ([22])**.** *Let f be a pseudoconcave function on the integer lattice $\mathbb{Z}^n$. The following properties are equivalent*:

1. *f is a PM-function*;
2. *f satisfies the SWGS-condition.*

Corollary. *PM-functions on the Boolean cube* (*ultradiscretization of valuated matroids*; *Dress and Wentzel* [4]) *are identical to functions with gross substitution* (*due to Kelso and Crawford* [7])*.*

Because the classes of integer valued $\mathbb{A}_n$-concave functions and $*\mathbb{A}_n$-concave functions are Fenchel conjugate classes and the later class is a subclass of the supermodular functions, the PM-functions are submodular.

Definition. A base polytope $B(b) \subset \mathbb{R}^n$ is said to be $\mathbb{A}$-polymatroid if b is a PM-function on the Boolean cube.

The images (under the moment map) of the Mirkovic–Vilonen cycles on the affine Grassmanians are $\mathbb{A}$-polymatroids indeed (see [23] and Lectures 4 and 5.)

Problem. To describe the cone (under the Minkowski summation) of an $\mathbb{A}$-polymatroid and its Hilbert basis. (This problems looks easier than the Lovász problem on extreme rays of the cone of submodular functions.)

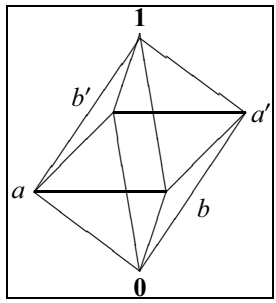

Figure 8. An elementary octahedron for octahedron recurrence

Algebraic Combinatorics

The main idea of the octahedron recurrence is rather transparent. Specifically, consider the octahedron with vertices $\mathbf{0}, a, a', b, b'$ and $\mathbf{1}$ (Figure 8). Let f be a real-valued function defined on the points $\mathbf{0}, a, a', b, b'$. Then we can *propagate* f to the point $\mathbf{1}$ by the following rule

$$f(\mathbf{1}) = \max(f(a) + f(a'), f(b) + f(b')) - f(\mathbf{0}).$$

Such a propagation is the *octahedron recurrence*.

We let $\Delta_n(OXYZ)$ denote the three-dimensional grid, constituted from the non-negative integer points (x, y, z), such that $x+y+z \leq n$ (in Figure 9 we draw $\Delta_3(XYZ)$). Using the octahedron recurrence, we can propagate initial data given at the *ground* $\Delta_n(OXY)$ and the *front wall* $\Delta_n(OXZ)$ to a function on the whole simplex $\Delta_n(OXYZ)$ and these initial data. We can set initial data at the *shadow wall* $\Delta_n(OYZ)$ and the *slope wall* $\Delta_n(XYZ)$ and get a function on the simplex using the octahedron recurrence with the reverse propagation vector $(1, -1, -1)$.

The fundamental property of the octahedron recurrence is that if the initial data (at the ground and the front wall) are discrete concave functions, then the corresponding polarized function on the grid $\Delta_n(OXYZ)$ is an $\mathbb{A}_3$-concave function (for details see [24]).

Other Classes of Discrete Concave Functions

Until now we considered functions related to a particular class of discrete convexity related to the unimodular system $\mathbb{A}_n$. However we may construct discrete convex analysis for any class of discrete convexity. Specifically, let $\mathcal{K}(\mathcal{U})$, where $\mathcal{U}$ is a pure system, be a class of discrete convexity. Then a function $f\colon M(\mathbb{Z}^n) \to \mathbb{R} \cup \{-\infty\}$ is $\mathcal{K}(\mathcal{U})$-*concave* if f is pseudoconcave and, for any $p \in V^*$, the set $D(f, p) := \operatorname{Arg\,max}(f - p)$ is $\mathcal{U}$-convex, that is, it belongs to the DC-class $\mathcal{K}(\mathcal{U})$.

The main properties of these functions are summarized in the following

Theorem 11. (a) *For a pure system $\mathcal{U}$, the $\mathcal{K}(\mathcal{U})$-concave/convex functions possess the separation property.*
(b) *For an S-class $\mathcal{U}$, the $\mathcal{K}(\mathcal{U})$-concave functions are stable under the convolution.*
(c) *For an I-class $\mathcal{U}$, the $\mathcal{K}(\mathcal{U})$-concave functions are stable under the summation.*
(d) *The classes of integer-valued $\mathcal{K}(\mathcal{U})$-concave and $\mathcal{K}(\mathcal{U}^{\perp})$-concave functions are Fenchel conjugate.*

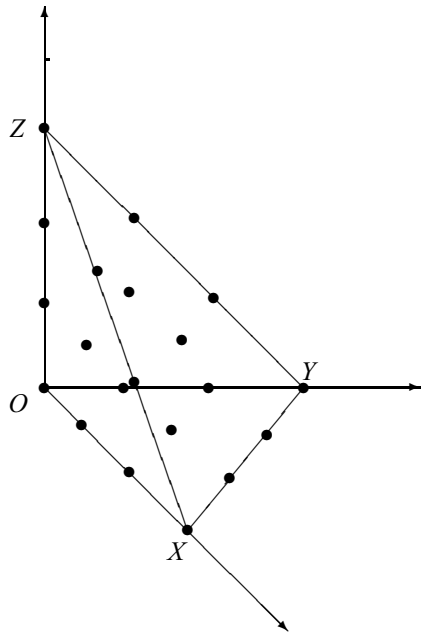

Figure 9. The three-dimensional grid $\Delta_n(OXYZ)$

References

[1] Frank, A. (1982) An algorithm for submodular functions on graphs. *Bonn Workshop on Combinatorial Optimization (Bonn, 1980)*, vol. 16 of *Ann. Discrete Math.*, pp. 97–120, North-Holland.

[2] Lovász, L. (1983) Submodular functions and convexity. *Mathematical Programming: The State of the Art*, Bonn, 1982, pp. 235–257, Springer, Berlin.

[3] Choquet, G. (1953–1954) Theory of capacities. *Ann. Inst. Fourier, Grenoble*, **5**, 131–295.

[4] Dress, A. W. M. and Wenzel, W. (1992) Perfect matroids. *Adv. Math.*, **91**, 158–208.

[5] Murota, K. (1996) Convexity and Steinitz's exchange property. *Adv. Math.*, **124**, 272–311.

[6] Danilov, V. I., Koshevoy, G. A., and Murota, K. (2001) Discrete convexity and equilibria in economies with indivisible goods and money. *Math. Social Sci.*, **41**, 251–273.

[7] Kelso, A. S., Jr. and Crawford, V. P. (1982) Job matching, coalition formation, and gross substitutes. *Econometrica*, **50**, 1483–1504.

[8] Danilov, V. I. and Koshevoy, G. A. (2004) Discrete convexity and unimodularity. I. *Adv. Math.*, **189**, 301–324.

[9] Murota, K. (2003) *Discrete Convex Analysis*. SIAM Monogr. Discrete Math. Appl., SIAM.

[10] Schrijver, A. (1986) *Theory of Linear and Integer Programming*. Wiley-Intersci. Ser. Discrete Math., Wiley.

[11] Frank, A. and Tardos, É. (1988) Generalized polymatroids and submodular flows. *Math. Programming*, **42**, 489–563.

[12] Edmonds, J. (1970) Submodular functions, matroids, and certain polyhedra. *Combinatorial Structures and their Applications*, Calgary, AB, 1969, pp. 69–87, Gordon and Breach, New York.

[13] McMullen, P. (1975) Space tiling zonotopes. *Mathematika*, **22**, 202–211.

[14] Erdahl, R. M. and Ryshkov, S. S. (1994) On lattice dicing. *European J. Combin.*, **15**, 459–481.

[15] Seymour, P. D. (1980) Decomposition of regular matroids. *J. Combin. Theory Ser. B*, **28**, 305–359.

[16] Danilov, V. I. and Koshevoy, G. A. (2003) Nilpotent operators and discretely concave functions. *Izv. Math.*, **67**, 1–15.

[17] Fulton, W. (2000) Eigenvalues, invariant factors, highest weights, and Schubert calculus. *Bull. Amer. Math. Soc. (N.S.)*, **37**, 209–249 (electronic).

[18] Klyachko, A. A. (1998) Stable bundles, representation theory and Hermitian operators. *Selecta Math. (N.S.)*, **4**, 419–445.
[19] Knutson, A. and Tao, T. (1999) The honeycomb model of $GL_n(\mathbb{C})$ tensor products. I. Proof of the saturation conjecture. *J. Amer. Math. Soc.*, **12**, 1055–1090.
[20] Danilov, V. I. and Koshevoy, G. A. (2003) Discrete convexity and Hermitian matrices. *Proc. Steklov Inst. Math.*, **2003**, 58–78.
[21] Oxley, J. G. (1992) *Matroid Theory*. Oxford Sci. Publ., Oxford Univ. Press, New York.
[22] Danilov, V. I., Koshevoy, G. A., and Lang, C. (2003) Gross substitution, discrete convexity, and submodularity. *Discrete Appl. Math.*, **131**, 283–298.
[23] Kamnitzer, J. (2005), Mirkovic–Vilonen cycles and polytopes. arXiv:math/0501365.
[24] Danilov, V. I. and Koshevoy, G. A. (2005), Arrays and the octahedron recurrence. arXiv:math.CO/0504299.

Combinatorial Optimization: Methods and Applications
V. Chvátal (Ed.)
IOS Press, 2011

doi:10.3233/978-1-60750-718-5-164

Branching on Split Disjunctions

Giacomo NANNICINI [a,1], Gérard CORNUÉJOLS [a,2], Miroslav KARAMANOV [b] and Leo LIBERTI [c,3]

[a] *Tepper School of Business, Carnegie Mellon University, USA*
[b] *Capacity and Operations Planning, Bank of America, USA*
[c] *LIX, École Polytechnique, France*

Abstract. Branch-and-Cut is the most commonly used algorithm for solving Integer and Mixed-Integer Linear Programs. In order to reduce the number of nodes that have to be enumerated before optimality of a solution can be proven, branching on general disjunctions (i.e., split disjunctions involving more than one variable, as opposed to branching on simple disjunctions defined on one variable only) was shown to be very effective on particular classes of instances, but not much work has been done to study general purpose methods of this kind. In this paper, we survey known results related to this line of research, and we study the relationship between branching and cutting from a split disjunction.

Keywords. Integer programming, Branch-and-Bound, Split disjunctions

Introduction

Solving Mixed-Integer Linear Programs (MILPs) is of great practical use in a number of applications, and efficient software exists for this purpose. One key ingredient is the Branch-and-Bound algorithm [1]. Branch-and-Bound has two main components: dividing a problem into subproblems, which is known as branching, and computing bounds on the objective function value at the subproblems. The idea is to recursively subdivide the initial problem into smaller problems, until the subproblems can be easily solved, and use bounds to eliminate as many as possible. Subproblems are typically stored in a tree structure, hence they are called *nodes* in the literature. The bounding phase is carried out by considering the Linear Programming (LP) relaxation of each node; in this paper, we focus on the branching phase.

Whenever the solution $\bar{x}$ to the LP associated with a node is fractional on a variable x_i that is required to take on integer values, a natural way of branching is to create two subproblems imposing the constraint $x_i \leq \lfloor \bar{x_i} \rfloor$ on one subproblem and $x_i \geq \lceil \bar{x_i} \rceil$ on the other. In this paper, we take a different approach whereby branching can occur on a general hyperplane with integer components π by imposing $\pi^\top x \leq \pi_0$ on one child and $\pi^\top x \geq \pi_0 + 1$ on the other.

[1] Corresponding Author: Giacomo Nannicini, Tepper School of Business, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA; Email: nannicin@andrew.cmu.edu. Supported by an IBM fellowship.
[2] Supported by NSF grant CMMI1024554 and ONR grant N00014-09-1-0033.
[3] Supported by ANR grant 07-JCJC-0151.

How do we choose π? We use the connections between branching and cutting from split disjunctions [2], Gomory Mixed-Integer (GMI) cuts [3] and intersection cuts [4]. GMI cuts arise as intersection cuts from a split disjunction, and provide a computationally inexpensive way of generating π. Therefore, we generate a pool of possible branching hyperplanes this way, and select one by using strong branching [5]. We also investigate the effect of modifying the hyperplanes by strenghtening the underlying GMI cut with a Reduce-and-Split like algorithm [6,7].

Computational experiments on MIPLIB instances show that this approach is effective in practice, and can significantly reduce the size of the enumeration tree; on average, the reduction in number of nodes is by more than a factor two on mixed-integer instances.

Extended versions of this work have appeared in [8,9]. In this paper, we give a unified treating of this topic. In Section 1, we give some preliminaries and survey the research carried out in this area. In Section 2 we introduce our notation and recall several known results that are useful for the subsequent parts of the paper. Section 3 studies the relationship between the integrality gap (i.e., the difference between the integer optimum and the relaxed optimum) closed by generating an intersection cut from a split disjunction, or branching on the same disjunction. In Section 4 we describe a branching scheme that is based on exploiting the disjunctions defining the GMI cuts read directly from an optimal simplex tableau; Section 5 modifies this scheme by adding a disjunction strenghtening step. In Section 6 we discuss the size of the coefficients of "good" disjunctions. Finally, Section 7 concludes the paper with a computational evaluation.

1. Preliminaries and Literature Review

In this paper we consider the Mixed Integer Linear Program in standard form:

$$\left.\begin{aligned} \min c^\top x \\ Ax = b \\ x \geq 0 \\ \forall j \in N_I\ x_j \in \mathbb{Z}, \end{aligned}\right\} \quad \mathcal{P}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ and $N_I \subset N = \{1, \ldots, n\}$. The LP relaxation of $\mathcal{P}$ is the linear program obtained by dropping the integrality constraints, and is denoted by $\overline{\mathcal{P}}$. We denote by P the set of feasible solutions of $\overline{\mathcal{P}}$, which is a polyhedron. The Branch-and-Bound algorithm makes an implicit use of the concept of disjunctions [10]: whenever the solution to the current LP relaxation is fractional, we divide the current problem $\mathcal{P}$ into two subproblems $\mathcal{P}_1$ and $\mathcal{P}_2$ such that the union of the feasible regions of $\mathcal{P}_1$ and $\mathcal{P}_2$ contains all feasible solutions to $\mathcal{P}$. Usually, this is done by choosing a fractional component $\bar{x}_i$ (for some $i \in N_I$) of the optimal solution $\bar{x}$ to the relaxation $\overline{\mathcal{P}}$, and adding the constraints $x_i \leq \lfloor \bar{x}_i \rfloor$ and $x_i \geq \lceil \bar{x}_i \rceil$ to $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively. Choosing which variable should be branched on at each step is of fundamental importance for the performance of Branch-and-Bound. We refer to [11] for a recent survey on this topic.

Here, we take a more general approach whereby branching can occur with respect to a direction $\pi \in \mathbb{R}^n$ by adding the constraints $\pi x \leq \beta_0$, $\pi x \geq \beta_1$ with $\beta_0 < \beta_1$ to $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively, as long as no feasible point of $\mathcal{P}$ is cut off. A natural way of generating such directions is to consider split disjunctions $D(\pi, \pi_0)$ of the form:

$$\pi^\top x \leq \pi_0 \quad \bigvee \quad \pi^\top x \geq \pi_0 + 1 \tag{1}$$

with $\pi \in \mathbb{Z}^n$, $\pi_0 \in \mathbb{Z}$, $\pi_i = 0 \; \forall i \notin N_I$. By integrality, every feasible solution to $\mathcal{P}$ satisfies any split disjunction. In other words, a split disjunction is defined by two parallel hyperplanes that have no integer point in the interior of the "strip" between them. In the branching literature, disjunctions involving only one variable are labeled *simple* or *elementary*, whereas those involving more than one variable are called *general*.

There are mainly two different categories of approaches to branching on general disjunctions that have been proposed in the MILP literature. The first category contains methods that try to identify "thin" directions of P ; the second category focuses on improving as much as possible the LP bound at the children nodes. [12] discusses both problems, which are shown to be strongly NP-hard in [13].

1.1. Branching on thin directions

The concept of thin direction requires the notion of *width* of a full-dimensional polyhedron P along a direction u, which is defined as $\max_{x,y \in P}(ux - uy)$. Thus, for a *pure* integer program associated with P, the *integer width* is defined as

$$\min_{\pi \in \mathbb{Z}^n \setminus \{0\}} \max_{x,y \in P} (\pi x - \pi y).$$

This definition naturally extends to the mixed integer case by considering integer directions $\pi \in \mathbb{Z}^n \setminus \{0\}$ with $\pi_j = 0$ for $j \notin N_I$.

The work of Lenstra [14] on solving integer programs in fixed dimension in polynomial time (see also [15,16]) is at the origin of the idea of branching on thin directions of P. The method works as follows. First, some thin directions of P are computed, using the lattice basis reduction algorithm by Lenstra, Lenstra and Lovász [17]. Then, the space is transformed so that these directions correspond to unit vectors, and the problem is solved by Branch-and-Bound in the new space. Thus, branching on single variables in the transformed space translates back to branching on general disjunctions in the original space. This method has proven successful for some particular instances where standard Branch-and-Bound fails because of the huge size of the enumeration tree, such as the Market Split instances [18], whose solution is discussed in [19]. Other examples are given in [20,21].

1.2. Branching for Maximum Bound Improvement

Another line of research which has been pursued is that of selecting a good general disjunction for branching at each node of the Branch-and-Bound tree, in order to improve as much as possible the bound at the children nodes. Owen and Mehrotra [22] proposed branching on split disjunctions with coefficients in $\{-1, 0, 1\}$ on the integer variables with fractional values at the current node. They generate all possible such disjunctions, and evaluate them using strong branching (i.e., solving the LPs associated with the children nodes to optimality), in order to select the one that gives the largest improvement of the dual bound. [8,9] follow this idea of generating disjunctions for maximum bound improvement, and try to do so by exploiting the relationship between split cuts [2] and split disjunctions for branching.

2. Split Disjunctions: Cutting and Branching

Given a split disjunction $D(\pi, \pi_0)$ of the form (1), a *split cut* for P is a cut which is valid (i.e., does not cut off any integral feasible solution) for both $P_1 = P \cap \{x : \pi^\top x \leq \pi_0\}$ and $P_2 = P \cap \{x : \pi^\top x \geq \pi_0 + 1\}$. Since $P_1 \cup P_2$ contains all integral feasible points of P, such a cut is valid for P as well. Split cuts were introduced in [2]. It is intuitive to see that there should be some kind of relationship between a split cut derived from $D(\pi, \pi_0)$ and the "strength" of the two polyhedra P_1, P_2; since the aim of branching is exactly that of creating two strong (but smaller) subproblems $\mathcal{P}_1, \mathcal{P}_2$ associated with P_1, P_2, we want to study this relationship. To do so, we first investigate some useful properties of split cuts that will lead to the formulas used in the rest of this paper.

Split cuts are disjunctive cuts [10], i.e., cutting planes which are valid for $\text{conv}(P_1 \cup P_2)$. A pictorial representation of such a cut is given in Figure 1. Is there an easier way of deriving split cuts without having to resort to disjunctive programming? [23] establishes a correspondence between split cuts and intersection cuts [4], showing that each split cut for P can be derived as an intersection cut from a split disjunction and a suitable basis of $\overline{\mathcal{P}}$. This allows for a geometric understanding of their derivation, and for closed form formulas.

We need some definitions. A *basis* for $\overline{\mathcal{P}}$ is an m-subset B of N such that the column submatrix of A induced by B is an invertible submatrix of A. Let $J := N \setminus B$ denote the index set of the nonbasic variables, $B_I = B \cap N_I$ the set of integer basic variables, $J_I = J \cap N_I$ the set of integer nonbasic variables, $J_C = J \setminus N_I$ the set of continuous nonbasic variables. Additionally, we denote by $\langle x \rangle$ the fractional part of x, i.e., $\langle x \rangle = x - \lfloor x \rfloor$. A further relaxation of the set P with respect to a basis B is obtained by removing the nonnegativity constraints on the basic variables. We denote it by $P(B)$:

$$P(B) := \{x \in \mathbb{R}^n : Ax = b \text{ and } x_j \geq 0 \text{ for } j \in J\}. \tag{2}$$

This set is a translate of a polyhedral cone: $P(B) = C + \bar{x}$, where $C = \{x \in \mathbb{R}^n : Ax = 0 \text{ and } x_j \geq 0 \text{ for } j \in J\}$ and $\bar{x}$ solves $\{x \in \mathbb{R}^n : Ax = b \text{ and } x_j = 0 \text{ for } j \in J\}$, i.e., $\bar{x}$ is the *basic solution* corresponding to the basis B. Typically, B will be the optimal basis of an LP relaxation of MILP, but not necessarily so (see, e.g., [24]). In this paper, B will be optimal for $\overline{\mathcal{P}}$. The cone C can be expressed also in terms of its extreme rays, r^j for $j \in J$: $P(B) = \text{Cone}(\{r^j\}_{j \in J}) + \bar{x}$, where $\text{Cone}(\{r^j\})$ denotes the polyhedral cone generated by vectors $\{r^j\}$. Looking at the simplex tableau associated with B written in the usual form:

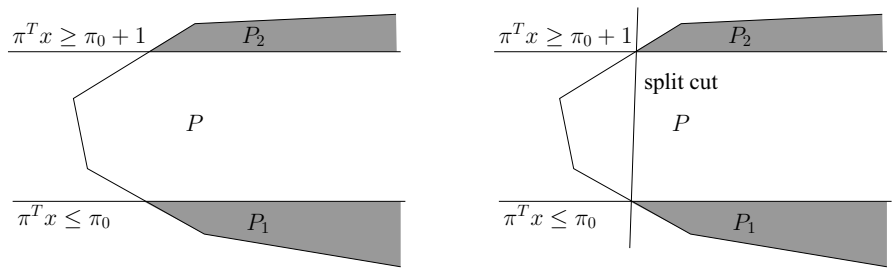

Figure 1. Deriving a split cut.

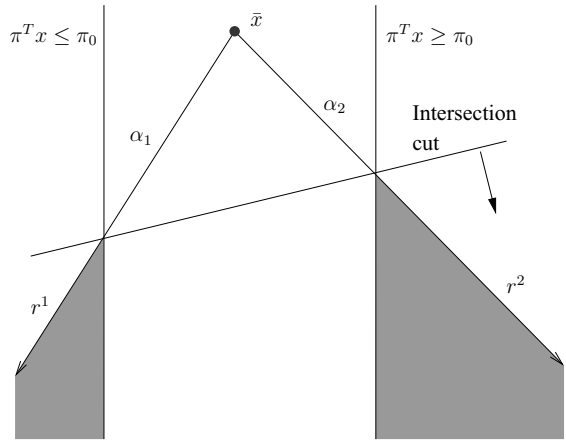

Figure 2. Deriving the intersection cut

$$x_i = \bar{x}_i - \sum_{j \in J} \bar{a}_{ij} x_j \quad \forall i \in B, \tag{3}$$

the extreme rays of $P(B)$ can be read directly as:

$$r_i^j = \begin{cases} -\bar{a}_{ij} & \text{if } i \in B \\ 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Observe that our interest is in split disjunctions that are violated by the current fractional solution $\bar{x}$. This is because we do not want $\bar{x}$ to be feasible for either P_1 or P_2, so that the solution to the LP relaxation is forced to "move" after branching. The same is true for cutting planes: the most interesting ones are typically those that cut off $\bar{x}$. How do we generate a split cut as an intersection cut? Given any disjunction $D(\pi, \pi_0)$ violated by $\bar{x}$, we can generate a split cut using $D(\pi, \pi_0)$ and $P(B)$ as exemplified in Figure 2. In particular, the intersection cut is a half-space bounded by the hyperplane passing through the intersection points of $D(\pi, \pi_0)$ with the extreme rays of $P(B)$.

In order to find the intersection points, for all $j \in J$ we compute the scalars:

$$\alpha_j(\pi, \pi_0) := \begin{cases} -\dfrac{\varepsilon(\pi, \pi_0)}{\pi^T r^j} & \text{if } \pi^T r^j < 0, \\ \dfrac{1 - \varepsilon(\pi, \pi_0)}{\pi^T r^j} & \text{if } \pi^T r^j > 0, \\ +\infty & \text{otherwise,} \end{cases} \tag{5}$$

where $\varepsilon(\pi, \pi_0) := \pi^T \bar{x} - \pi_0$ is the amount by which $\bar{x}$ violates the first term of the disjunction $D(\pi, \pi_0)$. The number $\alpha_j(\pi, \pi_0)$ for $j \in J$ is the smallest number α such that $\bar{x} + \alpha r^j$ satisfies the disjunction. In other words, $x^j = \bar{x} + \alpha_j(\pi, \pi_0) r^j$ lies on one of the disjunctive hyperplanes $\pi^T x = \pi_0$ and $\pi^T x = \pi_0 + 1$.

Now, the intersection cut associated with B and $D(\pi, \pi_0)$ supports the points x^j and is given by:

$$\sum_{j \in J} \frac{x_j}{\alpha_j(\pi, \pi_0)} \geq 1. \tag{6}$$

The Euclidean distance between $\bar{x}$ and this hyperplane is:

$$d(B, \pi, \pi_0) := \sqrt{\frac{1}{\sum_{j \in J} 1/(\alpha_j(\pi, \pi_0))^2}} \tag{7}$$

This quantity, called *distance cut off* or *depth*, was used as a measure of cut quality in [25].

The well-known GMI cuts from a basis B, which are included in virtually every Branch-and-Cut based software for solving MILPs, can be viewed as intersection cuts from a particular split disjunction [4]. They can be obtained as follows. We start from a disjunction on the integer basic variables:

$$\sum_{i \in B_I} \hat{\pi}_i x_i \leq \left\lfloor \sum_{i \in B_I} \hat{\pi}_i \bar{x_i} \right\rfloor \quad \bigvee \quad \sum_{i \in B_I} \hat{\pi}_i x_i \geq \left\lfloor \sum_{i \in B_I} \hat{\pi}_i \bar{x_i} \right\rfloor + 1, \tag{8}$$

with $\hat{\pi}_i \in \mathbb{Z}\ \forall i \in B_I$ and $\sum_{i \in B_I} \hat{\pi}_i \bar{x_i} \notin \mathbb{Z}$. Equation (8) is then strengthened on the nonbasic integer variables where the affected α_j are modified so that the distance cut off (7) is maximized. We obtain the following disjunction:

$$\pi_j = \begin{cases} \lfloor \sum_{i \in B_I} \hat{\pi}_i \bar{a_{ij}} \rfloor & \text{if } j \in J_I \text{ and } \langle \sum_{i \in B_I} \hat{\pi}_i \bar{a_{ij}} \rangle \leq \langle \sum_{i \in B_I} \hat{\pi}_i \bar{x_i} \rangle \\ \lceil \sum_{i \in B_I} \hat{\pi}_i \bar{a_{ij}} \rceil & \text{if } j \in J_I \text{ and } \langle \sum_{i \in B_I} \hat{\pi}_i \bar{a_{ij}} \rangle > \langle \sum_{i \in B_I} \hat{\pi}_i \bar{x_i} \rangle \\ \hat{\pi}_j & \text{if } j \in B_I \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

$$\pi_0 = \lfloor \pi^\top \bar{x} \rfloor.$$

Plugging Eq. (9) into Eq. (5) gives a GMI cut that cuts off $\bar{x}$. In the original cutting plane procedure of Gomory [3], Eq. (8) is an elementary disjunction. Notice that we have a closed form formula for this disjunction, therefore we can compute it very efficiently. This is the class of split disjunctions that will be employed in the remainder.

3. Gap Closed by Cutting and by Branching

A violated split disjunction can be used for generating an intersection cut but it can be used for branching as well. A good intersection cut cuts deeply into P and improves the LP bound at the children nodes. Our suggestion is that a split disjunction defining a deep cut is good for branching too.

Indeed, the improvement in the lower bound caused by branching on a split disjunction is no less than the improvement by the corresponding intersection cut. Let $\bar{x_1}$ and $\bar{x_2}$ be the LP relaxation optima of $\mathcal{P}_1$ and $\mathcal{P}_2$ (if $\mathcal{P}_k$ is infeasible then $\bar{x_k} = \infty$ for $k \in \{1, 2\}$).

Proposition 1. $\min_{j \in J} c^\top x^j \leq \min(c^\top \bar{x_1}, c^\top \bar{x_2})$.

Proof. We have $\bar{x}_1 = \arg\min\{c^\top x : c \in P_1\}$, $\bar{x}_2 = \arg\min\{c^\top x : c \in P_2\}$. Let $P_1^r = P(B) \cap \{x \in \mathbb{R}^n : \pi^\top x \leq \pi_0\}$ and $P_2^r = P(B) \cap \{x \in \mathbb{R}^n : \pi^\top x \geq \pi_0 + 1\}$. By the definition of the points $x^j \; \forall j \in J$, $\min_{j \in J}\{c^\top x^j\} = \min\{c^\top x : x \in P_1^r \cup P_2^r\}$. Since $P \subseteq P(B)$, $\min\{c^\top x : x \in P_1^r \cup P_2^r\} \leq \min\{c^\top x : x \in P_1 \cup P_2\} = \min(c^\top \bar{x}_1, c^\top \bar{x}_2)$, which completes the proof. □

Therefore, in order to generate children nodes that have tight LP relaxations, it makes sense to try to maximize the lower bound given in Proposition 1; as this quantity is bounded from below by the gap closed by the corresponding intersection cut, this explains our intuition. However, it can be shown with a small example that the bound on the gap closed by branching given by the corresponding intersection cut can be arbitrarily far from the real value.

Example 2. Consider the integer program:

$$\left.\begin{array}{r} \min -x_1 - x_2 \\ x_1 \leq 1.5 \\ x_2 \leq 1 \\ x_1/m - x_2 \geq 1.5/m - 1.25 \\ mx_1 - x_2 \leq 1.5m - 0.75 \\ x_1, x_2 \in \mathbb{Z}, \end{array}\right\} \quad \mathcal{P} \tag{10}$$

where $m > 1$ is a given parameter close to 1. The solution to the LP relaxation is (1.5, 1), with an objective value of −2.5. The intersection cut obtained from the disjunction $x_1 - x_2 \leq 0 \vee x_1 - x_2 \geq 1$ is $x_1 + x_2 \leq 2$, which gives an objective value of −2. Now suppose we branch on $x_1 - x_2 \leq 0 \vee x_1 - x_2 \geq 1$. We obtain two children $\mathcal{P}_1$ and $\mathcal{P}_2$, which are both feasible. One can verify that optimal solution to the LP relaxation of $\mathcal{P}_1$ is $((1.5 - 1.25m)/(1-m), (1.5-1.25m)/(1-m))$ with objective value $-2(1.5-1.25m)/(1-m)$, and the optimal solution to the LP relaxation of $\mathcal{P}_2$ is $((1.5m-1.75)/(m-1), (0.5m-0.75)/(m-1))$ with objective value $-(2m - 2.5)/(m - 1)$. Therefore, the gap closed by branching is:

$$\max\left\{-2\frac{1.5 - 1.25m}{1 - m}, -\frac{2m - 2.5}{m - 1}\right\} - 2.5,$$

which can be made arbitrarily large when m tends to 1 from above. At the same time, the intersection cut associated with the same disjunction closes a gap of 0.5 regardless of m. We give a picture of the situation for $m = 1.1$ in Figure 3.

4. Branching on Disjunctions Defining the GMI Cuts

We need a procedure for selecting promising split disjunctions for branching. As we discussed in the introduction, optimizing over the set of all split disjunctions is strongly NP-hard [13]. [9] simply suggests to concentrate on a finite class of general disjunctions generated directly from the current optimal basis — the set $\mathcal{G}$ of split disjunctions defining the GMI cuts that can be read from the simplex tableau. The GMI cuts as defined by Gomory [3] arise from simple disjunctions (8) with $\hat{\pi} = e_i$ for $i \in B_I$ where $\bar{x}_i \notin \mathbb{Z}$. These cuts have been shown to be very effective in practice [26]. The reasons for this choice are the following. First, the set $\mathcal{G}$ is not only finite but relatively small. Its cardinality at

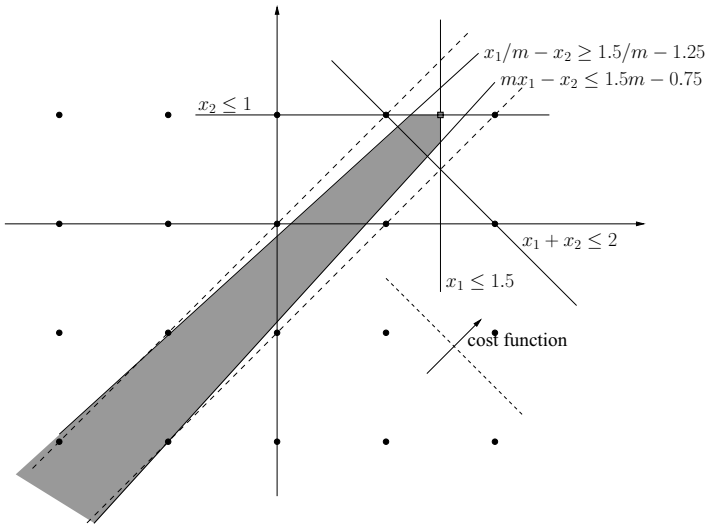

Figure 3. Representation of Example 2 for $m = 1.1$.

a given node of the Branch-and-Bound tree equals the number of integer variables with fractional values in the current basic solution. Second, these disjunctions are fast to obtain. They can be read from the current tableau with a closed form formula (9). Third, as we explained at the end of Section 2, these disjunctions can be viewed as strengthened simple disjunctions (with respect to the cut depth) which suggests that they could perform better than the elementary disjunctions.

The branching procedure proposed in [9] is as follows. Consider the set of all GMI disjunctions arising from elementary disjunctions for a specific basic solution, and select a subset $\mathcal{S}$ of it, containing the most promising disjunctions according to the chosen criterion for comparison. The distance (7) cut off by the underlying intersection cut, is used as a criterion for selecting promising disjunctions, picking those with the largest distance. The cardinality of $\mathcal{S}$ is limited to a parameter k, which can be used to manage the computational effort at different levels, e.g. a larger k can be used close to the root where branching decisions are more important and a smaller k in the deep levels. Finally, in view of Example 2, we apply strong branching to the disjunctions in $\mathcal{S}$, in the spirit of [5,22], to evaluate the true impact of each disjunction. Note that the computational complexity of this algorithm is dominated by the strong branching phase.

Once strong branching is performed and we know the objective function improvement at the children nodes $c^\top \bar{x}_1, c^\top \bar{x}_2$, we use

$$\gamma \min(c^\top \bar{x}_1, c^\top \bar{x}_2) + (1 - \gamma) \max(c^\top \bar{x}_1, c^\top \bar{x}_2), \tag{11}$$

with $0 \leq \gamma \leq 1$, as a measure of quality of a disjunction, attempting to increase the LP bound. This approach is not new: for instance, [11] proposes $\gamma = \frac{5}{6}$ in the context of branching on elementary disjunctions.

5. Strengthening the GMI Disjunctions

GMI cuts derived from elementary disjunctions are very strong in practice; but can we do better? [6,7] experiment with cutting planes derived as GMI cuts from split disjunctions, with good results. In our framework, their procedure can be seen as a method for finding a disjunction (8) that gives rise to an intersection cut with better cut coefficients on the continuous variables. The starting disjunction $\hat{\pi}$ is then plugged into Eq. (9) as usual. Clearly, this approach is computationally more expensive: the elementary disjunctions of Section 4 can be read from the tableau with no additional cost, but finding a strong split disjunction of the form (8) is not as simple, as there is an infinite number of them.

[8] proposes the following approach, which has also been modified and enhanced for cutting plane generation in [7]. The motivating idea traces back to [6]. We look at the expression of α_j (5) for the intersection cut derived from Eq. (9):

$$\alpha_j \begin{cases} \max\left(\dfrac{\langle\sum_{i\in B_I}\hat{\pi}_i\bar{x}_i\rangle}{\langle\sum_{i\in B_I}\hat{\pi}_i\bar{a}_{ij}\rangle}, \dfrac{1-\langle\sum_{i\in B_I}\hat{\pi}_i\bar{x}_i\rangle}{1-\langle\sum_{i\in B_I}\hat{\pi}_i\bar{a}_{ij}\rangle}\right) & \text{if } j \in J_I \\ \max\left(\dfrac{\langle\sum_{i\in B_I}\hat{\pi}_i\bar{x}_i\rangle}{\sum_{i\in B_I}\hat{\pi}_i\bar{a}_{ij}}, \dfrac{1-\langle\sum_{i\in B_I}\hat{\pi}_i\bar{x}_i\rangle}{-\sum_{i\in B_I}\hat{\pi}_i\bar{a}_{ij}}\right) & \text{if } j \in J_C \end{cases} \tag{12}$$

where $\alpha_j = \infty$ if its denominator is zero. A larger α_j means a smaller cut coefficient in Eq. (6), hence a stronger cut, as can be seen from Eq. 7; and by Proposition 1, we argue that a disjunction with large α_j will be strong for branching as well. Therefore, we study a method for increasing α_j by acting on $\hat{\pi}$.

It seems difficult to optimize α_j for $j \in J_I$ because both terms of the fraction are nonlinear. Furthermore, for $j \in N_I$, α_j is always at least 1, independent of the choice of $\hat{\pi}$. For $j \in J_C$, α_j can be smaller than 1, therefore we concentrate on trying to improve these α_j. From Eq. (12) we see that the denominator of α_j for $j \in J_C$ is a linear function of $\hat{\pi}$, whereas the numerator is a nonlinear function of $\hat{\pi}$ and is always between 0 and 1. For this reason we attempt to minimize the denominator, i.e., $\sum_{i\in B_I}\hat{\pi}_i\bar{a}_{ij}$ for $j \in J_C$, over integral vectors $\hat{\pi}$. More specifically, we would like to minimize $\|\tilde{d}\|$, where

$$\tilde{d} = \left(\sum_{i\in B_I}\hat{\pi}_i\bar{a}_{ij}\right)_{j\in J_C}. \tag{13}$$

Since we try to improve the disjunction by looking at the cut coefficients on the continuous variables, the method described in this section is only suitable for mixed-integer instances.

Apply a permutation to the simplex tableau in order to obtain $B_I = \{1, \ldots, |B_I|\}$, $J_C = \{1, \ldots, |J_C|\}$, and define the matrix $D \in \mathbb{R}^{|B_I|\times|J_C|}$, $d_{ij} = \bar{a}_{ij}$. Minimizing $\|\tilde{d}\|$ can be written as

$$\min_{\hat{\pi}\in\mathbb{Z}^{|B_I|}\setminus\{0\}} \left\|\sum_{i\in B_I}\hat{\pi}_i d_i\right\|. \tag{14}$$

This is a shortest vector problem in the additive group generated by the rows of D. If these rows are linearly independent, the group defines a lattice, and we have the classical shortest vector problem in a lattice, which is NP-hard under randomized reductions [27].

[6] proposes a heuristic for Eq. (14) based on a reduction algorithm which cycles through the rows of D and, for each such row d_k, considers whether summing an integer multiple of some other row yields a reduction of $\|d_k\|$. If this is the case, the matrix D is updated by replacing d_k with the shorter vector. Note, however, that this method only considers two rows at a time.

The idea of [8] is to use, for each row d_k of D, a subset $R_k \subset B_I$ of the rows of the simplex tableau with $d_k \in R_k$, in order to reduce $\|d_k\|$ as much as possible with a linear combination with integer coefficients of d_k and d_i for all $i \in R_k \setminus \{k\}$. This is done by defining, for each row d_k that we want to reduce, the convex minimization problem:

$$\min_{\hat{\pi}^k \in \mathbb{R}^{|R_k|}, \hat{\pi}^k_k = 1} \left\| \sum_{i \in R_k} \hat{\pi}^k_i d_i \right\|, \tag{15}$$

and then rounding the coefficients $\hat{\pi}^k_i$ to the nearest integer $\lfloor \hat{\pi}^k_i \rceil$. There are several reasons for imposing $\hat{\pi}^k_k = 1$. One reason is that not only do we want to find a short vector, but it is also important to find a vector $\hat{\pi}^k$ with small norm: in the space $B \cap N_I$, the distance between the two hyperplanes that define a split disjunction $D(\pi, \pi_0)$ is related to the norm of $\hat{\pi}$: in this space, disjunctions that cut off a larger volume have a small $\|\hat{\pi}\|$. We will come back to this issue in Section 6. Another reason is that we must avoid the zero vector as a solution. Yet another is to get different optimization problems for $k = 1, \ldots, |B_I|$, thus increasing the chance of obtaining different branching directions. Vanishing the partial derivatives of $\left\| \sum_{i \in R_k} \hat{\pi}^k_i d_i \right\|$ with respect to $\hat{\pi}^k_i$ for all i, we obtain an $|R_k| \times |R_k|$ linear system that yields the optimal (continuous) solution.

Once these linear systems are solved and we have the optimal coefficients $\hat{\pi}^k \in \mathbb{R}^{|R_k|}$ for all $k \in \{1, \ldots, |B_I|\}$, we round them to the nearest integer. Then, we consider the norm of $\sum_{i \in R_k} \lfloor \hat{\pi}^k_i \rceil d_i$. If $\left\| \sum_{i \in R_k} \lfloor \hat{\pi}^k_i \rceil d_i \right\| < \|d_k\|$, then we have an improvement with respect to the original row of the simplex tableau; in this case, we use

$$\sum_{i \in R_k} \hat{\pi}^k_i x_i = \sum_{i \in R_k} \hat{\pi}^k_i \bar{x}_i - \sum_{j \in J} \sum_{i \in R_k} \hat{\pi}^k_i \bar{a}_{ij} x_j, \tag{16}$$

instead of row $\bar{a}_k$ in order to compute a GMI disjunction, and consider the possibly improved disjunction for branching.

It is natural to ask how to choose $R_k \subset B_I$. Although using $R_k = B_I$ is possible, in that case two problems arise: first, the size of the linear systems may become too large, and second, if we add up too many rows then the coefficients on the variables with indices in $J \cap N_I$ may deteriorate. In particular, we may get more nonzero coefficients. Thus, we do the following. We fix a maximum cardinality $M_{|R_k|}$; if $M_{|R_k|} \geq |B_I|$, we set $R_k = B_I$. Otherwise, for each row k that we want to reduce, we sort the remaining rows by ascending number of nonzero coefficients on the variables with indices in $\{i \in J \cap N_I \mid \bar{a}_{ki} = 0\}$, and select the first $M_{|R_k|}$ indices as those in R_k. The reason for this choice is that $\bar{a}_{kj} = 0$ implies $\alpha_j = \infty$, i.e., the cut is strong on that variable. Therefore, we would like those coefficients that are 0 in row $\bar{a}_k$ to be left unmodified when we compute $\sum_{j \in R_k} \lfloor \hat{\pi}^k_j \rceil \bar{a}_j$.

6. On Non-Dominated Disjunctions

Although solving the shortest vector problem (14) is important for finding a deep cut, it is not the only consideration when trying to find a good branching direction. In the space $B \cap N_I$, the distance between the two hyperplanes that define a split disjunction $D(\pi, \pi_0)$ is equal to $1/\|\lambda\|$ as can be seen from Eq. (9). Therefore, in this space, disjunctions that cut off a larger volume have a small $\|\lambda\|$. We illustrate this with an example.

Example 3. Consider the following tableau, where x_1, x_2 are binary variables and y_1, y_2 are continuous:

$$\begin{cases} x_1 = \frac{1}{3} + 98y_1 + y_2 \\ x_2 = \frac{1}{3} - 99y_1 - 1.01y_2. \end{cases} \tag{17}$$

The solution to the shortest vector problem (14) is given by the integer multipliers $\lambda_1 = 99$, $\lambda_2 = 98$ which yield the shortest vector in the lattice $(0, 0.02)$ and the disjunction $99x_1 + 98x_2 \leq 65 \vee 99x_1 + 98x_2 \geq 66$. The heuristic method of Section 5 computes the continuous multipliers $\lambda_1 = 1$, $\lambda_2 = 98/99$ which are rounded to $\lambda_1 = 1$, $\lambda_2 = 1$, that correspond to the disjunction $x_1 + x_2 \leq 0 \vee x_1 + x_2 \geq 1$. It is easy to verify that the distance between these two hyperplanes is roughly ten times larger than in the first case. Therefore, in the unit square, the disjunction obtained through the heuristic method dominates the one computed through the exact solution of the shortest vector problem. Figure 4 gives a picture of this.

It is clear from Example 3 why disjunctions with small coefficients are likely to perform better. It is intuitive to think that, at least in the unit hypercube, the coefficients of "good" disjunctions will be small. However, this is not true in general. We formalize our statament.

For a polyhedron P, we say that the split disjunction $D(\pi^1, \pi^1_0)$ *dominates* $D(\pi^2, \pi^2_0)$ if $P \cap \{{\pi^1}^\top x \leq \pi^1_0\} \subseteq P \cap \{{\pi^2}^\top x \leq \pi^2_0\}$ and $P \cap \{{\pi^1}^\top x \geq \pi^1_0 + 1\} \subseteq P \cap \{{\pi^2}^\top x \geq \pi^2_0 + 1\}$, with

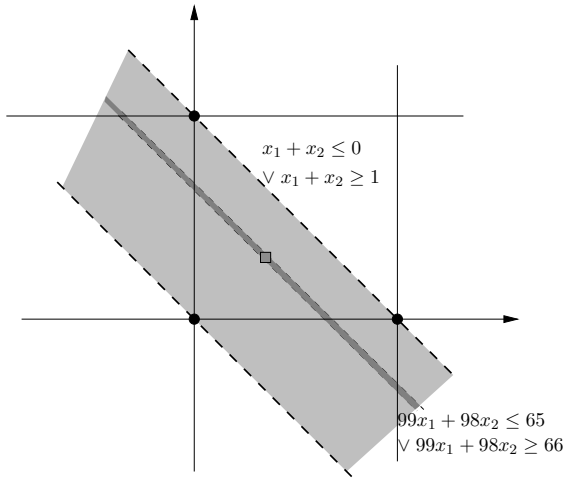

Figure 4. Representation of the disjunctions discussed in Example 3.

at least one of the two inclusions being strict. The dominating disjunction is obviously to be preferred to the dominated one for branching, as it induces the same partition of the feasible integer points, while generating smaller feasible regions for the two children. Thus, we are interested in finding nondominated disjunctions only. Do the coefficients of nondominated disjunctions have a "nice" characterization, so that we can restrict our search to disjunctions with small norm? Unfortunately, the answer is negative in general. Even by restricting our attention to 0/1 polytopes, no polynomial bound (in the dimension n) can be given on the size of the coefficients of nondominated disjunctions.

Proposition 4. *The size of the coefficients of nondominated disjunctions for* 0/1 *polytopes of dimension n cannot be polynomially bounded in n.*

Proof. It is known [28] that the largest integer coefficient in the facet description of a full-dimensional 0/1 polytope can be exponential in n. Let $a^\top x \geq b$ be the hyperplane, which we can assume to have all integer coefficients, describing such a facet with a large coefficient. Consider the polytope defined by $P = \{x \in [0,1]^n \mid a^\top x \geq b, a^\top \leq b + 0.5\}$. The disjunction $D(\pi, \pi_0)$ with $\pi = a, \pi_0 = b$ is nondominated, and in fact gives the convex hull of the integer points in one branching step. However, its largest coefficient has size exponential in n. □

Therefore, even though in low dimension nondominated disjunctions have small integer coefficients, in general there is no hope of finding a nice characterization of their coefficients. The method described in Section 5 tries to generate disjunctions with small coefficients heuristically, following the intuition of Example 3.

7. Computational Experiments

The ideas proposed in [8,9] were tested in a Branch-and-Bound framework implemented on top of Cplex [29]. The test set consists of all instances in `MIPLIB2.0`, `MIPLIB3` and `MIPLIB2003`, excluding those that can be solved in less than 50 nodes by branching on simple disjunctions, and those for which less than 50 nodes can be processed in an hour. Also removed were the instances with zero integrality gap, which leaves 84 instances.

We report tests with three branching algorithms:

- Branching on single variables (Simple Disjunctions, SD);
- Branching on the disjunctions defining the GMI cuts at the optimal LP basis (GMI Disjunctions, GD);
- Branching on the disjunctions defining the GMI cuts after the strengthening procedure described in Section 5 (Improved GMI Disjunctions, IGD).

In order to evaluate the effect of branching on split disjunctions, we focus primarily on the integrality gap closed by branching. An additional important factor is the number of infeasible children which are created by branching: in fact, if one of the two sides of the branching disjunction is infeasible, the number of nodes in the enumeration tree does not grow. This can be seen as adding a cutting plane (i.e., the feasible side of the disjunction) to the current node. In case such a disjunction a discovered, it is always preferred to the ones that create two children. Note that if both sides of the disjunction are infeasible, the node is infeasible.

7.1. Branching for Eight Levels

In this experiment, we branch at the top eight levels of the Branch-and-Bound tree, and compare the resulting gap closed. At each node, for the SD algorithm we consider all fractional integer variables for branching, whereas for GD we consider all simple GMI disjunctions. Note that the number of candidate branching objects is the same for both SD and GD. We set $\gamma = \frac{5}{6}$ in Eq. (11) as suggested by [11], trying to increase the LP bound in both children nodes. In this experiment, GD performs better, mainly due to the larger gap closed by branching on split disjunctions. We observe an interesting secondary effect: branching on GMI disjunctions tends to produce more infeasible children, which additionally decreases the amount of enumeration. We record this phenomenon by counting the number of active nodes at the ninth level.

Table 1 contains a summary of the results. We report average values, and the number of times that one method is better than the other according to the comparison criterion.

In terms of amount of gap closed, SD dominates in 20 cases, GD in 48 cases out of 84. The average gap closed by SD and GD is 32.1% and 41.7%, respectively. The difference in the average gap closed is 9.6%. It is statistically significantly larger than zero with 99% confidence, according to a one-sided paired t-test (p-value = 0.0021). These results support our observation that GD closes more gap.

A graphical representation of the gap closed by SD and GD is shown in Figure 5.A. In the figure, dots correspond to test instances. The gap closed by SD is shown on the abscissa while that closed by GD is shown on the ordinate. The diagonal line represents equality in the gap closed by both methods. We observe that most points lie in the upper-left triangle, corresponding to "GD outperforms SD." Furthermore, most of the points that lie in the lower-right triangle are close to the diagonal line — there are few cases in which SD outperforms GD significantly.

It is interesting to observe that GD typically produces a smaller number of active nodes at the ninth level. On this criterion, SD performs better in 16 cases while GD does so in 53 cases. Out of the maximum possible 256 nodes at level nine, SD generates 113 while GD generates 65, on average. A statistical t-test rejects the null hypothesis "GD produces at least as many active nodes at level nine as SD" at 99.9% level of confidence

Table 1. Comparison of SD and GD after eight levels of branching. Branch-and-Bound.

Percentage gap closed average (# better)	
Simple disjunctions (SD):	32.1% (20)
GMI disjunctions (GD):	41.7% (48)

Active nodes at level 9 average (# better)	
Simple disjunctions (SD):	114.6 (16)
GMI disjunctions (GD):	66.7 (53)

Gap closed and active nodes together # better	
Simple disjunctions (SD):	6
GMI disjunctions (GD):	45

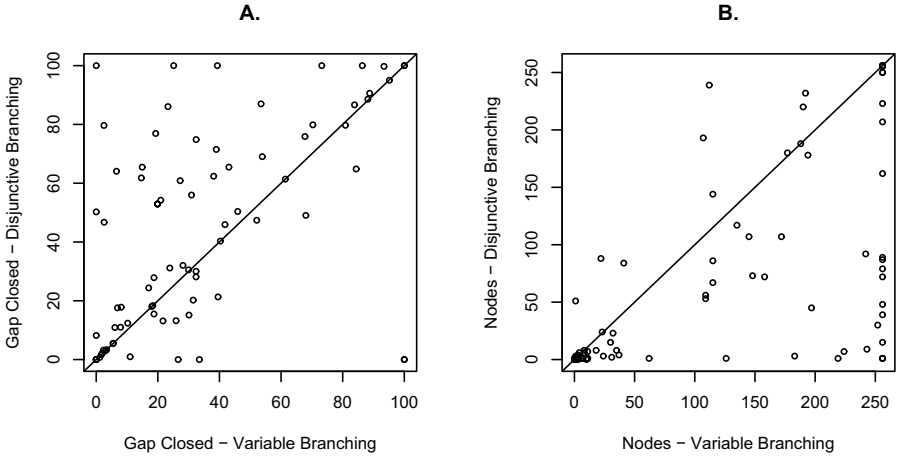

Figure 5. A. Gap closed (in percentage) after eight levels of branching: GD vs. SD. B. Number of active nodes after eight levels of branching: GD vs. SD. Every data point represents a test instance.

(p-value = 1.30e − 6). This indicates that the number of active nodes created by GD is significantly smaller.

The difference in the performance is best seen graphically. In Figure 5.B, we plot the number of active nodes at level nine produced by GD vs. that produced by SD. Not only do most of the points lie below the equality line but many of them reside in the bottom-right corner, corresponding to a significant difference in the number of nodes. On the other hand, out of the 16 instances for which SD outperforms GD, only eight lie visibly far from the equality line.

The effect of a smaller number of active nodes is important not by itself but in combination with improvement in the gap. Combining both criteria, we count the cases in which an algorithm strictly dominates in one of the criteria and performs at least as well in the other criterion. SD is better than GD in only 6 cases, while GD outperforms SD in 45 cases out of 84.

The reason for the smaller number of active nodes is that GD often generates disjunctions that produce only one feasible child. For some instances, this happens at most nodes of the branching tree, resulting in only a few nodes at level nine. Although SD generates many infeasible children, GD generates even more. Sometimes, this is combined with an impressive improvement of the gap closed over SD.

The combination of a larger improvement in the gap and a smaller number of active nodes is a very desirable effect and it deserves more attention. Branching on a disjunction that generates only one feasible child is equivalent to adding a single cut to the formulation. One may argue that this cut would be added by a branch-and-cut algorithm anyway. This is true in some cases but in others the disjunction inequality is stronger than the corresponding GMI cut. Figure 6 is an example. The cut generation procedure considers the polyhedral cone pointed at $\bar{x}$, relaxing some of the constraints defining P, and generates the intersection cut $\beta^T x \leq \beta_0$. But it cannot detect the fact that one of the feasible sets of the children is empty. (Here, $P \cap \{x \in \mathbb{R}^n : \pi^T x \leq \pi_0\}$.) When branching on $D(\pi, \pi_0)$, we essentially add the cut $\pi^T x \geq \pi_0 + 1$, which is stronger than $\beta^T x \leq \beta_0$.

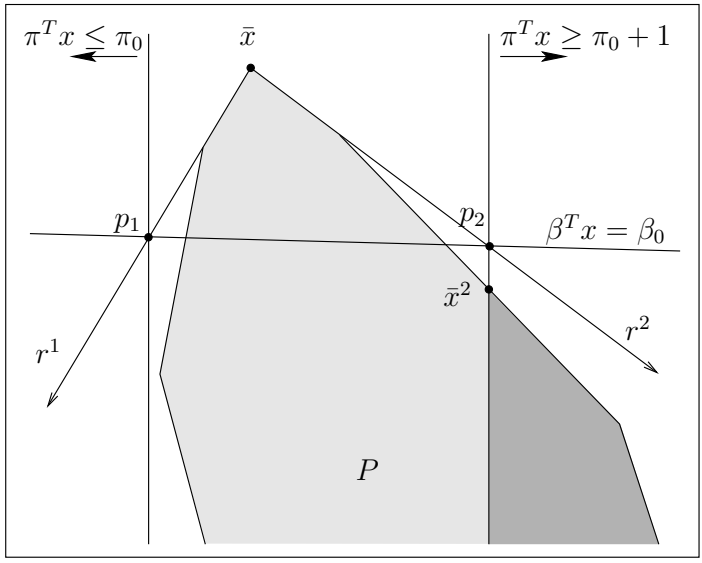

Figure 6. Disjunction with only one feasible child.

Consequently, branching on a split disjunction that generates only one child can be viewed as strengthening the underlying intersection cut. Thus, branching on a split disjunction cannot be substituted by adding the corresponding intersection cut even when one of the disjunctive sets is empty. When both disjunctive sets are nonempty, branching on a split disjunction can still close more gap than the corresponding cut, as we showed in Section 3.

We do not consider branching on split disjunctions a substitute for cutting planes. The procedure comes into play when Branch-and-Cut decides to start branching. It is important to note that the observed good effects of branching on split disjunctions are not neutralized by adding cuts. We repeat the above experiment in a Cut-and-Branch framework where we add ten rounds of GMI cuts, MIR cuts, and knapsack cover cuts. As expected, aggressive cut generation closes a significant amount of gap (63% on average), leaving less work for the branching phase. As a result, the amount of gap closed by branching on the top eight levels is smaller and the difference between the two methods is smaller. Nevertheless, the mutual relation in performance is preserved, as seen in Table 2.

7.2. Effect of the Disjunction Strengthening Procedure

In this section we want to evaluate the impact of the disjunction improvement procedure on the branching phase. We have already seen that GD is able to outperform SD in several respects. We want to see if the same holds true for IGD. Therefore, we design a similar experiment: we branch for 1000 nodes, and compare the integrality gap closed by each method (or the number of nodes, for instances solved to optimality in less than 1000 nodes). In this experiment, generating fewer feasible nodes is clearly an advantage, as it allows to progress further in the tree. Note that IGD can only be applied on mixed-integer instances, because the disjunction strengthening procedure requires the presence

Table 2. Comparison of SD and GD after eight levels of branching. Cut-and-branch.

Percentage gap closed average (# better)	
Simple disjunctions (SD):	5.6% (11)
GMI disjunctions (GD):	7.4% (52)
Active nodes at level 9 average (# better)	
Simple disjunctions (SD):	107.6 (23)
GMI disjunctions (GD):	81.6 (44)
Gap closed and active nodes together # better	
Simple disjunctions (SD):	6
GMI disjunctions (GD):	39

of continuous variable. Thus, for this experiment the test set consists of the 57 instances with more than one continuous variable only.

Since we are focusing on closing more integrality gap, in this experiment we set $\gamma = 1$ in Eq. (11). Besides, to speed up the computations, we do not apply strong branching to all possible branching disjunctions, but only to the 10 most promising ones. This setting is meant to mimick more closely what is done in commercial software, since strong branching can be very expensive. This allows us to better evaluate the computational overhead introduced by branching on split disjunctions. The most promising disjunctions are chosen as the 10 variables with larges fractional variable (for SD), or as the split disjunctions with largest distance cut off by the corresponding intersection cut (for GD and IGD).

For IGD, after some preliminary testing, we decided to set $M_{|R_k|} = 50$, i.e., we combine at most 50 rows during the disjunction strenghtening phase.

Table 3 shows that the increase in the gap closed per node by branching on GMI disjunctions is large compared to branching on single variables. Besides, the IGD method seems to be on average superior in all respects to the two other methods, as it closes more gap for the unsolved instances under 1000 nodes, and requires less nodes for the solved instances. This is also evident if we compare the number of instances where each method closes at least the same absolute gap as the other two methods: IGD ranks first with 36 instances over 57.

On the instances solved by all methods, SD is roughly twice as fast as GD and IGD. Moreover, if we consider only the instances not solved by any method (i.e., all branching algorithms solve 1000 nodes without reaching optimality) we obtain the following average times:

- SD: 32.59 seconds;
- GD: 150.61 seconds;
- IGD: 176.78 seconds.

This suggests that branching on split disjunctions introduces a significant computational overhead at each node with respect to branching on simple disjunctions. The average

Table 3. Results on mixed-integer instances after 1000 solved nodes

Number of solved instances	
Simple disjunctions (SD):	15
GMI disjunctions (GD):	20
Improved GMI disjunctions (IGD):	20

Average number of nodes on instances solved by all methods	
Simple disjunctions (SD):	125.6
GMI disjunctions (GD):	98.1
Improved GMI disjunctions (IGD):	75.3

Average CPU time [sec] on instances solved by all methods	
Simple disjunctions (SD):	2.53
GMI disjunctions (GD):	5.23
Improved GMI disjunctions (IGD):	4.79

Average gap closed on instances not solved by any method	
Simple disjunctions (SD):	9.02%
GMI disjunctions (GD):	12.99%
Improved GMI disjunctions (IGD):	13.30%

Number of instances with largest closed gap (at least as much as the other methods)	
Simple disjunctions (SD):	34
GMI disjunctions (GD):	33
Improved GMI disjunctions (IGD):	36

time spent per node by the three methods, recorded as the geometric mean of the average time spent per node over all the instances, is as follows:

- SD: 0.02 seconds;
- GD: 0.08 seconds;
- IGD: 0.10 seconds.

Therefore, the most evident drawback of branching on split disjunctions is that it is slower than using simple disjunctions. It is slower in several respects: the first reason is that the computations at each node take longer. This is because we have to compute the distance cut off by the GMI cut associated with each row of the simplex tableau, and the reduction step proposed in Section 5 involves the solution of an $M_{|R_k|} \times M_{|R_k|}$ linear system for each row which is improved, where we chose $M_{|R_k|} = 50$. All these computations are carried out several times, thus the overhead per node with respect to branching on simple disjunctions is significant. Additionally, generating the GMI disjunctions requires the computation of the optimal simplex tableau, which is not necessary (and is typically not carried out) when branching on single variables. The second reason is that, by branching on GMI disjunctions, we add one (or more) rows to the formulation of children nodes,

which may result in a slowdown of the LP solution process. On the other hand, branching on simple disjunctions involves only a change in the bounds of some variables, thus the size of the LP does not increase.

In summary, computational experience with branching on split disjunctions shows that the size of the enumeration tree can be reduced by a factor of two or more on average. This is not quite sufficient to compensate for the increased computing time per node. A possibility for overcoming this drawback is to combine branching on single variables and on split disjunctions, using the latter disjunctions only when the gap closed is significantly greater.

References

[1] Land, A. H. and Doig, A. G. (1960) An automatic method of solving discrete programming problems. *Econometrica*, **28**, 497–520.

[2] Cook, W., Kannan, R., and Schrijver, A. (1990) Chvátal closures for mixed integer programming problems. *Mathematical Programming, Series A*, **47**, 155–174.

[3] Gomory, R. E. (1960) An algorithm for the mixed integer problem. Tech. Rep. RM-2597, RAND Corporation, Santa Monica, CA.

[4] Balas, E. (1971) Intersection cuts—a new type of cutting planes for integer programming. *Operations Research*, **19**, 19–39.

[5] Applegate, D., Bixby, R. E., Chvátal, V., and Cook, W. (1995) Finding cuts in the TSP. Tech. Rep. 95-06, DIMACS.

[6] Andersen, K., Cornuéjols, G., and Li, Y. (2005) Reduce-and-split cuts: Improving the performance of mixed-integer Gomory cuts. *Management Science*, **51**, 1720–1732.

[7] Cornuéjols, G. and Nannicini, G. (2010) Reduce-and-split revisited: Efficient generation of split cuts for mixed-integer linear programs. Tech. rep., Tepper School of Business, Carnegie Mellon University.

[8] Cornuéjols, G., Liberti, L., and Nannicini, G. (2010) Improved strategies for branching on general disjunctions. *Mathematical Programming, Series A*, to appear.

[9] Karamanov, M. and Cornuéjols, G. (2010) Branching on general disjunctions. *Mathematical Programming,*, to appear.

[10] Balas, E. (1979) Disjunctive programming. *Annals of Discrete Mathematics*, **5**, 3–51.

[11] Achterberg, T., Koch, T., and Martin, A. (2005) Branching rules revisited. *Operations Research Letters*, **33**, 42–54.

[12] Mahajan, A. and Ralphs, T. K. (2009) Experiments with branching using general disjunctions. Chinneck, J. W., Kristjansson, B., and Saltzman, M. J. (eds.), *Operations Research and Cyber-Infrastructure*, vol. 47 of *Operations Research/Computer Science Interfaces Series*, pp. 101–118, Springer, New York.

[13] Mahajan, A. and Ralphs, T. (2010) On the complexity of selecting disjunctions in integer programming. *SIAM Journal on Optimization*, **20**, 2181–2198.

[14] Lenstra, H. W., Jr. (1983) Integer programming with a fixed number of variables. *Mathematics of Operations Research*, **8**, 538–548.

[15] Grötschel, M., Lovász, L., and Schrijver, A. (1984) Geometric methods in combinatorial optimization. *Progress in Combinatorial Optimization*, (Waterloo, ON, 1982), pp. 167–183, Academic Press, Toronto, ON.

[16] Lovász, L. and Scarf, H. E. (1992) The generalized basis reduction algorithm. *Mathematics of Operations Research*, **17**, 751–764.

[17] Lenstra, A. K., Lenstra, H. W., Jr., and Lovász, L. (1982) Factoring polynomials with rational coefficients. *Mathematische Annalen*, **261**, 515–534.

[18] Cornuéjols, G. and Dawande, M. (1998) A class of hard small 0-1 programs. *Integer Programming and Combinatorial Optimization*, (Houston, TX, 1998), vol. 1412 of *Lecture Notes in Computer Science*, pp. 284–293, Springer, Berlin.

[19] Aardal, K., Bixby, R. E., Hurkens, C. A. J., Lenstra, A. K., and Smeltink, J. W. (2000) Market split and basis reduction: towards a solution of the Cornuéjols–Dawande instances. *INFORMS Journal on Computing*, **12**, 192–202.

[20] Krishnamoorthy, B. and Pataki, G. (2009) Column basis reduction and decomposable knapsack problems. *Discrete Optimization*, **6**, 242–270.
[21] Mehrotra, S. and Li, Z. (2004) On generalized ranching method for mixed integer programming. Tech. rep., Northwestern University, Evanston, IL.
[22] Owen, J. H. and Mehrotra, S. (2001) Experimental results on using general disjunctions in branch-and-bound for general-integer linear programs. *Computational Optimization and Applications*, **20**, 159–170.
[23] Andersen, K., Cornuéjols, G., and Li, Y. (2005) Split closure and intersection cuts. *Mathematical Programming, Series A*, **102**, 457–493.
[24] Balas, E. and Perregaard, M. (2003) A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer Gomory cuts for 0-1 programming. *Mathematical Programming, Series A*, **94**, 221–245.
[25] Balas, E., Ceria, S., and Cornuéjols, G. (1996) Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, **42**, 1229–1246.
[26] Balas, E., Ceria, S., Cornuéjols, G., and Natraj, N. (1996) Gomory cuts revisited. *Operations Research Letters*, **19**, 1–9.
[27] Ajtai, M. (1998) The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract). *STOC '98: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, (Dallas, TX, 1998), pp. 10–19, ACM, New York, NY.
[28] Ziegler, G. M. (2000) Lectures on 0/1-polytopes. Kalai, G. and Ziegler, G. M. (eds.), *Polytopes—Combinatorics and Computation*, (Oberwolfach, 1997), vol. 29 of *DMV Seminar*, pp. 1–41, Birkhäuser, Basel.
[29] ILOG S.A., Gentilly (2002) *ILOG CPLEX 8.0 User's Manual*.

Combinatorial Optimization: Methods and Applications
V. Chvátal (Ed.)
IOS Press, 2011

doi:10.3233/978-1-60750-718-5-183

Convex Discrete Optimization

Shmuel ONN
Technion, Israel

Abstract. We develop an algorithmic theory of convex optimization over discrete sets. Using a combination of algebraic and geometric tools we are able to provide polynomial time algorithms for solving broad classes of convex combinatorial optimization problems and convex integer programming problems in variable dimension. We discuss some of the many applications of this theory including to quadratic programming, matroids, bin packing and cutting-stock problems, vector partitioning and clustering, multiway transportation problems, and privacy and confidential statistical data disclosure. Highlights of our work include a strongly polynomial time algorithm for convex and linear combinatorial optimization over any family presented by a membership oracle when the underlying polytope has few edge-directions; a new theory of so-termed n-fold integer programming, yielding polynomial time solution of important and natural classes of convex and linear integer programming problems in variable dimension; and a complete complexity classification of high dimensional transportation problems, with practical applications to fundamental problems in privacy and confidential statistical data disclosure.

Keywords. Integer programming, combinatorial optimization, discrete optimization, nonlinear optimization, convex optimization, multi criteria optimization, polynomial time, transportation problem, multiindex transportation problem, matroid, spanning tree, partitioning, clustering, polytope, zonotope, edge direction, test set, augmentation, Graver base, Graver complexity, n-fold integer programming, contingency table, statistical table, multiway table, margin, privacy in databases, disclosure control, data security

1. Introduction

The general linear discrete optimization problem can be posed as follows.

Linear Discrete Optimization. Given a set $S \subseteq \mathbb{Z}^n$ of integer points and an integer vector $w \in \mathbb{Z}^n$, find an $x \in S$ maximizing the standard inner product $wx := \sum_{i=1}^n w_i x_i$.

The algorithmic complexity of this problem, which includes *integer programming* and *combinatorial optimization* as special cases, depends on the presentation of the set S of feasible points. In integer programming, this set is presented as the set of integer points satisfying a given system of linear inequalities, which in standard form is given by

$$S = \{x \in \mathbb{N}^n : Ax = b\},$$

where $\mathbb{N}$ stands for the nonnegative integers, $A \in \mathbb{Z}^{m \times n}$ is an $m \times n$ integer matrix, and $b \in \mathbb{Z}^m$ is an integer vector. The input for the problem then consists of A, b, w. In combinatorial optimization, $S \subseteq \{0, 1\}^n$ is a set of $\{0, 1\}$-vectors, often interpreted as a family of

subsets of a ground set $N := \{1, \ldots, n\}$, where each $x \in S$ is the indicator of its support $\mathrm{supp}(x) \subseteq N$. The set S is presented implicitly and compactly, say as the set of indicators of subsets of edges in a graph G satisfying a given combinatorial property (such as being a matching, a forest, and so on), in which case the input is G, w. Alternatively, S is given by an oracle, such as a *membership oracle* which, queried on $x \in \{0,1\}^n$, asserts whether or not $x \in S$, in which case the algorithmic complexity also includes a count of the number of oracle queries needed to solve the problem.

Here we study the following broad generalization of linear discrete optimization.

Convex Discrete Optimization. Given a set $S \subseteq \mathbb{Z}^n$, vectors $w_1, \ldots, w_d \in \mathbb{Z}^n$, and a convex functional $c\colon \mathbb{R}^d \to \mathbb{R}$, find an $x \in S$ maximizing $c(w_1 x, \ldots, w_d x)$.

This problem can be interpreted as *multi-objective* linear discrete optimization: given d linear functionals $w_1 x, \ldots, w_d x$ representing the values of points $x \in S$ under d criteria, the goal is to maximize their "convex balancing" defined by $c(w_1 x, \ldots, w_d x)$. In fact, we have a hierarchy of problems of increasing generality and complexity, parameterized by the number d of linear functionals: at the bottom lies the linear discrete optimization problem, recovered as the special case of $d = 1$ and c the identity on $\mathbb{R}$; and at the top lies the problem of maximizing an arbitrary convex functional over the feasible set S, arising with $d = n$ and with $w_i = \mathbf{1}_i$ the ith standard unit vector in $\mathbb{R}^n$ for all i.

The algorithmic complexity of the convex discrete optimization problem depends on the presentation of the set S of feasible points as in the linear case, as well as on the presentation of the convex functional c. When S is presented as the set of integer points satisfying a given system of linear inequalities we also refer to the problem as *convex integer programming*, and when $S \subseteq \{0,1\}^n$ and is presented implicitly or by an oracle we also refer to the problem as *convex combinatorial optimization*. As for the convex functional c, we will assume throughout that it is presented by a *comparison oracle* that, queried on $x, y \in \mathbb{R}^d$, asserts whether or not $c(x) \leq c(y)$. This is a very broad presentation that reveals little information on the function, making the problem, on the one hand, very expressive and applicable, but on the other hand, very hard to solve.

There is a massive body of knowledge on the complexity of linear discrete optimization — in particular (linear) integer programming [1] and (linear) combinatorial optimization [2]. The purpose of this monograph is to provide the first comprehensive unified treatment of the extended convex discrete optimization problem. The monograph follows the outline of five lectures given by the author in the Séminaire de mathématiques supérieures Series, Université de Montréal, during June 2006. Colorful slides of theses lectures are available online at [3] and can be used as a visual supplement to this monograph. The monograph has been written under the support of the ISF — Israel Science Foundation. The theory developed here is based on and is a culmination of several recent papers including [4–17] written in collaboration with several colleagues — Eric Babson, Jesus De Loera, Komei Fukuda, Raymond Hemmecke, Frank Hwang, Vera Rosta, Uriel Rothblum, Leonard Schulman, Bernd Sturmfels, Rekha Thomas, and Robert Weismantel. By developing and using a combination of geometric and algebraic tools, we are able to provide polynomial time algorithms for several broad classes of convex discrete optimization problems. We also discuss in detail some of the many applications of our theory, including to quadratic programming, matroids, bin packing and cutting-stock problems, vector partitioning and clustering, multiway transportation problems, and privacy and confidential statistical data disclosure.

We hope that this monograph will, on the one hand, allow users of discrete optimization to enjoy the new powerful modelling and expressive capability of convex discrete optimization along with its broad polynomial time solvability, and on the other hand, stimulate more research on this new and fascinating class of problems, their complexity, and the study of various relaxations, bounds, and approximations for such problems.

1.1. Limitations

Convex discrete optimization is generally intractable even for small fixed d, since already for $d = 1$ it includes linear integer programming which is NP-hard. When d is a variable part of the input, even very simple special cases are NP-hard, such as the following problem, so-called *positive semi-definite quadratic binary programming*,

$$\max\{(w_1x)^2 + \cdots + (w_nx)^2 : x \in \mathbb{N}^n,\ x_i \leq 1, i = 1, \ldots, n\}.$$

Therefore, throughout this monograph we will assume that d is fixed (but arbitrary).

As explained above, we also assume throughout that the convex functional c which constitutes part of the data for the convex discrete optimization problem is presented by a comparison oracle. Under such broad presentation, the problem is generally very hard. In particular, if the feasible set is $S := \{x \in \mathbb{N}^n : Ax = b\}$ and the underlying polyhedron $P := \{x \in \mathbb{R}^n_+ : Ax = b\}$ is *unbounded*, then the problem is inaccessible even in one variable with no equation constraints. Indeed, consider the following family of univariate convex integer programs with convex functions parameterized by $-\infty < u \leq \infty$,

$$\max\{c_u(x) : x \in \mathbb{N}\}, \qquad c_u(x) := \begin{cases} -x, & \text{if } x < u; \\ x - 2u, & \text{if } x \geq u. \end{cases}$$

Consider any algorithm attempting to solve the problem and let u be the maximum value of x in all queries to the oracle of c. Then the algorithm can not distinguish between the problem with c_u, whose objective function is unbounded, and the problem with c_∞, whose optimal objective value is 0. Thus, convex discrete optimization (with an oracle presented functional) over an *infinite* set $S \subset \mathbb{Z}^n$ is quite hopeless. Therefore, an algorithm that solves the convex discrete optimization problem will either return an optimal solution, or assert that the problem is infeasible, or assert that the underlying polyhedron is unbounded. In fact, in most applications, such as in combinatorial optimization with $S \subseteq \{0, 1\}^n$ or integer programming with $S := \{x \in \mathbb{Z}^n : Ax = b, l \leq x \leq u\}$ and $l, u \in \mathbb{Z}^n$, the set S is finite and the problem of unboundedness does not arise.

1.2. Outline and Overview of Main Results and Applications

We now outline the structure of this monograph and provide a brief overview of what we consider to be our main results and main applications. The precise relevant definitions and statements of the theorems and corollaries mentioned here are provided in the relevant sections in the monograph body. As mentioned above, most of these results are adaptations or extensions of results from one of the papers [4–17]. The monograph gives many more applications and results that may turn out to be useful in future development of the theory of convex discrete optimization.

The rest of the monograph consists of five sections. While the results evolve from one section to the next, it is quite easy to read the sections independently of each other (while just browsing now and then for relevant definitions and results). Specifically, Section 3 uses definitions and the main result of Section 2; Section 5 uses definitions and results from Sections 2 and 4; and Section 6 uses the main results of Sections 4 and 5.

In Section 2 we show how to reduce the convex discrete optimization problem over $S \subset \mathbb{Z}^n$ to strongly polynomially many linear discrete optimization counterparts over S, provided that the convex hull $\mathrm{conv}(S)$ satisfies a suitable geometric condition, as follows.

Theorem 2.4. *For every fixed d, the convex discrete optimization problem over any finite $S \subset \mathbb{Z}^n$ presented by a linear discrete optimization oracle and endowed with a set covering all edge-directions of* $\mathrm{conv}(S)$, *can be solved in strongly polynomial time.*

This result will be incorporated in the polynomial time algorithms for convex combinatorial optimization and convex integer programming to be developed in §3 and §5.

In Section 3 we discuss convex combinatorial optimization. The main result is that convex combinatorial optimization over a set $S \subseteq \{0,1\}^n$ presented by a membership oracle can be solved in strongly polynomial time provided it is endowed with a set covering all edge-directions of $\mathrm{conv}(S)$. In particular, the standard linear combinatorial optimization problem over S can be solved in strongly polynomial time as well.

Theorem 3.5. *For every fixed d, the convex combinatorial optimization problem over any $S \subseteq \{0,1\}^n$ presented by a membership oracle and endowed with a set covering all edge-directions of the polytope* $\mathrm{conv}(S)$, *can be solved in strongly polynomial time.*

An important application of Theorem 3.5 concerns convex matroid optimization.

Corollary 3.11. *For every fixed d, convex combinatorial optimization over the family of bases of a matroid presented by membership oracle is strongly polynomial time solvable.*

In Section 4 we develop the theory of linear *n-fold integer programming*. As a consequence of this theory we are able to solve a broad class of linear integer programming problems in variable dimension in polynomial time, in contrast with the general intractability of linear integer programming. The main theorem here may seem a bit technical at a first glance, but is really very natural and has many applications discussed in detail in §4, §5 and §6. To state it we need a definition. Given an $(r+s)\times t$ matrix A, let A_1 be its $r\times t$ sub-matrix consisting of the first r rows and let A_2 be its $s\times t$ sub-matrix consisting of the last s rows. We refer to A explicitly as $(r+s)\times t$ *matrix*, since the definition below depends also on r and s and not only on the entries of A. The *n-fold matrix* of an $(r+s)\times t$ matrix A is then defined to be the following $(r+ns)\times nt$ matrix,

$$A^{(n)} := (\mathbf{1}_n \otimes A_1) \oplus (I_n \otimes A_2) = \begin{pmatrix} A_1 & A_1 & A_1 & \cdots & A_1 \\ A_2 & 0 & 0 & \cdots & 0 \\ 0 & A_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & A_2 \end{pmatrix}.$$

Given now any $n \in \mathbb{N}$, lower and upper bounds $l, u \in \mathbb{Z}_{\infty}^{nt}$ with $\mathbb{Z}_{\infty} := \mathbb{Z} \uplus \{\pm\infty\}$, right-hand side $b \in \mathbb{Z}^{r+ns}$, and linear functional wx with $w \in \mathbb{Z}^{nt}$, the corresponding linear n-fold integer programming problem is the following program in variable dimension nt,

$$\max\{wx : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, l \leq x \leq u\}.$$

The main theorem of §4 asserts that such integer programs are polynomial time solvable.

Theorem 4.11. *For every fixed $(r + s) \times t$ integer matrix A, the linear n-fold integer programming problem with any n, l, u, b, and w can be solved in polynomial time.*

Theorem 4.11 has very important applications to high-dimensional transportation problems which are discussed in §4.5.1 and in more detail in §6. Another major application concerns bin packing problems, where items of several types are to be packed into bins so as to maximize packing utility subject to weight constraints. This includes as a special case the classical cutting-stock problem of [18]. These are discussed in detail in §4.5.2.

Corollary 4.15. *For every fixed number t of types and type weights $v_1, \ldots, v_t$, the corresponding integer bin packing and cutting-stock problems are polynomial time solvable.*

In Section 5 we discuss convex integer programming, where the feasible set S is presented as the set of integer points satisfying a given system of linear inequalities. In particular, we consider convex integer programming over n-fold systems for any fixed (but arbitrary) $(r + s) \times t$ matrix A, where, given $n \in \mathbb{N}$, vectors $l, u \in \mathbb{Z}_{\infty}^{nt}$, $b \in \mathbb{Z}^{r+ns}$ and $w_1, \ldots, w_d \in \mathbb{Z}^{nt}$, and convex functional $c\colon \mathbb{R}^d \to \mathbb{R}$, the problem is

$$\max\{c(w_1x, \ldots, w_dx) : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, l \leq x \leq u\}.$$

The main theorem of §5 is the following extension of Theorem 4.11, asserting that convex integer programming over n-fold systems is polynomial time solvable as well.

Theorem 5.5. *For every fixed d and $(r + s) \times t$ integer matrix A, convex n-fold integer programming with any $n, l, u, b, w_1, \ldots, w_d$, and c can be solved in polynomial time.*

Theorem 5.5 broadly extends the class of objective functions that can be efficiently maximized over n-fold systems. Thus, all applications discussed in §4.5 automatically extend accordingly. These include convex high-dimensional transportation problems and convex bin packing and cutting-stock problems, which are discussed in detail in §5.4.1 and §6.

Another important application of Theorem 5.5 concerns vector partitioning problems which have applications in many areas including load balancing, circuit layout, ranking, cluster analysis, inventory, and reliability, see, e.g., [11, 12, 16, 19, 20] and the references therein. The problem is to partition n items among p players so as to maximize social utility. With each item is associated a k-dimensional vector representing its utility under k criteria. The social utility of a partition is a convex function of the sums of vectors of items that each player receives. In the constrained version of the problem, there are also restrictions on the number of items each player can receive. We have the following consequence of Theorem 5.5; more details on this application are in §5.4.2.

Corollary 5.10. *For every fixed number p of players and number k of criteria, the constrained and unconstrained vector partition problems with any item vectors, convex utility, and constraints on the number of item per player, are polynomial time solvable.*

In the last Section 6 we discuss multiway (high-dimensional) transportation problems and secure statistical data disclosure. Multiway transportation problems form a very important class of discrete optimization problems and have been used and studied extensively in the operations research and mathematical programming literature, as well as in the statistics literature in the context of secure statistical data disclosure and management by public agencies, see, e.g., [21–30] and the references therein. The feasible points in a transportation problem are the multiway tables ("contingency tables" in statistics) such that the sums of entries over some of their lower dimensional sub-tables such as lines or planes ("margins" in statistics) are specified. We completely settle the algorithmic complexity of treating multiway tables and discuss the applications to transportation problems and secure statistical data disclosure, as follows.

In §6.2 we show that "short" 3-way transportation problems, over $r \times c \times 3$ tables with variable number r of rows and variable number c of columns but fixed small number 3 of layers (hence "short"), are *universal* in that *every* integer programming problem is such a problem (see §6.2 for the precise stronger statement and for more details).

Theorem 6.1. *Every linear integer programming problem* $\max\{cy : y \in \mathbb{N}^n : Ay = b\}$ *is polynomial time representable as a short* 3*-way line-sum transportation problem*

$$\max\Big\{wx : x \in \mathbb{N}^{r\times c\times 3} : \sum_i x_{i,j,k} = z_{j,k}, \sum_j x_{i,j,k} = v_{i,k}, \sum_k x_{i,j,k} = u_{i,j}\Big\}.$$

In §6.3 we discuss k-way transportation problems of any dimension k. We provide the first polynomial time algorithm for convex and linear "long" $(k+1)$-way transportation problems, over $m_1 \times \cdots \times m_k \times n$ tables, with k and $m_1, \ldots, m_k$ fixed (but arbitrary), and variable number n of layers (hence "long"). This is best possible in view of Theorem 6.1. Our algorithm works for any *hierarchical collection of margins*: this captures common margin collections such as all line-sums, all plane-sums, and more generally all h-flat sums for any $0 \le h \le k$ (see §6.1 for more details). We point out that even for the very special case of linear integer transportation over $3 \times 3 \times n$ tables with specified line-sums, our polynomial time algorithm is the only one known. We prove the following statement.

Corollary 6.4. *For every fixed* $d, k, m_1, \ldots, m_k$ *and family* $\mathcal{F}$ *of subsets of* $\{1, \ldots, k+1\}$ *specifying a hierarchical collection of margins, the convex* (*and in particular linear*) *long transportation problem over* $m_1 \times \cdots \times m_k \times n$ *tables is polynomial time solvable.*

In our last subsection §6.4 we discuss an important application concerning privacy in statistical databases. It is a common practice in the disclosure of a multiway table containing sensitive data to release some table margins rather than the table itself. Once the margins are released, the security of any specific entry of the table is related to the set of possible values that can occur in that entry in any table having the same margins as those of the source table in the data base. In particular, if this set consists of a unique value, that of the source table, then this entry can be exposed and security can be violated. We show that for multiway tables where one category is significantly richer than the others, that is, when each sample point can take many values in one category and only

few values in the other categories, it is possible to check entry-uniqueness in polynomial time, allowing disclosing agencies to make learned decisions on secure disclosure.

Corollary 6.6. *For every fixed $k, m_1, \ldots, m_k$ and family $\mathcal{F}$ of subsets of $\{1, \ldots, k+1\}$ specifying a hierarchical collection of margins to be disclosed, it can be decided in polynomial time whether any specified entry $x_{i_1,\ldots,i_{k+1}}$ is the same in all long $m_1 \times \cdots \times m_k \times n$ tables with the disclosed margins, and hence at risk of exposure.*

1.3. Terminology and Complexity

We use $\mathbb{R}$ for the reals, $\mathbb{R}_+$ for the nonnegative reals, $\mathbb{Z}$ for the integers, and $\mathbb{N}$ for the nonnegative integers. The sign of a real number r is denoted by $\text{sign}(r) \in \{0, -1, 1\}$ and its absolute value is denoted by $|r|$. The ith standard unit vector in $\mathbb{R}^n$ is denoted by $\mathbf{1}_i$. The *support* of $x \in \mathbb{R}^n$ is the index set $\text{supp}(x) := \{i : x_i \neq 0\}$ of nonzero entries of x. The *indicator* of a subset $I \subseteq \{1, \ldots, n\}$ is the vector $\mathbf{1}_I := \sum_{i \in I} \mathbf{1}_i$, so $\text{supp}(\mathbf{1}_I) = I$. When several vectors are indexed by subscripts, $w_1, \ldots, w_d \in \mathbb{R}^n$, their entries are indicated by pairs of subscripts, $w_i = (w_{i,1}, \ldots, w_{i,n})$. When vectors are indexed by superscripts, $x^1, \ldots, x^k \in \mathbb{R}^n$, their entries are indicated by subscripts, $x^i = (x^i_1, \ldots, x^i_n)$. The integer lattice $\mathbb{Z}^n$ is naturally embedded in $\mathbb{R}^n$. The space $\mathbb{R}^n$ is endowed with the standard inner product which, for $w, x \in \mathbb{R}^n$, is given by $wx := \sum_{i=1}^n w_i x_i$. Vectors w in $\mathbb{R}^n$ will also be regarded as linear functionals on $\mathbb{R}^n$ via the inner product wx. Thus, we refer to elements of $\mathbb{R}^n$ as points, vectors, or linear functionals, as will be appropriate from the context. The *convex hull* of a set $S \subseteq \mathbb{R}^n$ is denoted by $\text{conv}(S)$ and the set of *vertices* of a polyhedron $P \subseteq \mathbb{R}^n$ is denoted by $\text{vert}(P)$. In linear discrete optimization over $S \subseteq \mathbb{Z}^n$, the *facets* of $\text{conv}(S)$ play an important role, see Chvátal [31] and the references therein for earlier work, and Grötschel, Lovász and Schrijver [2, 32] for the later culmination in the equivalence of separation and linear optimization via the ellipsoid method of Yudin and Nemirovskii [33]. As will turn out in §2, in convex discrete optimization over S, the *edges* of $\text{conv}(S)$ play an important role (most significantly in a way which is *not* related to the Hirsch conjecture discussed in [34]). We therefore use extensively convex polytopes, for which we follow the terminology of [35, 36].

We often assume that the feasible set $S \subseteq \mathbb{Z}^n$ is finite. We then define its *radius* to be its l_∞ radius $\rho(S) := \max\{\|x\|_\infty : x \in S\}$ where, as usual, $\|x\|_\infty := \max_{i=1}^n |x_i|$. In other words, $\rho(S)$ is the smallest $\rho \in \mathbb{N}$ such that S is contained in the cube $[-\rho, \rho]^n$.

Our algorithms are applied to rational data only, and the time complexity is as in the standard Turing machine model, see, e.g., [1, 37, 38]. The input typically consists of rational (usually integer) numbers, vectors, matrices, and finite sets of such objects. The *binary length* of an integer number $z \in \mathbb{Z}$ is defined to be the number of bits in its binary representation, $\langle z \rangle := 1 + \lceil \log_2(|z| + 1) \rceil$ (with the extra bit for the sign). The length of a rational number presented as a fraction $r = p/q$ with $p, q \in \mathbb{Z}$ is $\langle r \rangle := \langle p \rangle + \langle q \rangle$. The length of an $m \times n$ matrix A (and in particular of a vector) is the sum $\langle A \rangle := \sum_{i,j} \langle a_{i,j} \rangle$ of the lengths of its entries. Note that the length of A is no smaller than the number of entries, $\langle A \rangle \geq mn$. Therefore, when A is, say, part of an input to an algorithm, with m, n variable, the length $\langle A \rangle$ already incorporates mn, and so we will typically not account additionally for m, n directly. But sometimes, especially in results related to n-fold integer programming, we will also emphasize n as part of the input length. Similarly, the length of a finite set E of numbers, vectors or matrices is the sum of lengths of its elements and hence, since $\langle E \rangle \geq |E|$, automatically accounts for its cardinality.

Some input numbers affect the running time of some algorithms through their unary presentation, resulting in so-called "pseudo polynomial" running time. The *unary length* of an integer number $z \in \mathbb{Z}$ is the number $|z|+1$ of bits in its unary representation (again, an extra bit for the sign). The unary length of a rational number, vector, matrix, or finite set of such objects are defined again as the sums of lengths of their numerical constituents, and is again no smaller than the number of such numerical constituents.

When studying convex and linear integer programming in §4 and §5 we sometimes have lower and upper bound vectors l, u with entries in $\mathbb{Z}_\infty := \mathbb{Z} \uplus \{\pm\infty\}$. Both binary and unary lengths of a $\pm\infty$ entry are constant, say 3 by encoding $\pm\infty := \pm$"00."

To make the input encoding precise, we introduce the following notation. In every algorithmic statement we describe explicitly the input encoding, by listing in square brackets all input objects affecting the running time. Unary encoded objects are listed directly whereas binary encoded objects are listed in terms of their length. For example, as is often the case, if the input of an algorithm consists of binary encoded vectors (linear functionals) $w_1, \ldots, w_d \in \mathbb{Z}^n$ and unary encoded integer $\rho \in \mathbb{N}$ (bounding the radius $\rho(S)$ of the feasible set) then we will indicate that the input is *encoded as* $[\rho, \langle w_1, \ldots, w_d \rangle]$.

Some of our algorithms are strongly polynomial time in the sense of [39]. For this, part of the input is regarded as "special." An algorithm is then *strongly polynomial time* if it is polynomial time in the usual Turing sense with respect to all input, and in addition, the number of arithmetic operations (additions, subtractions, multiplications, divisions, and comparisons) it performs is polynomial in the special part of the input. To make this precise, we extend our input encoding notation above by splitting the square bracketed expression indicating the input encoding into a "left" side and a "right" side, separated by semicolon, where the entire input is described on the right and the special part of the input on the left. For example, Theorem 2.4, asserting that the algorithm underlying it is strongly polynomial with data *encoded as* $[n, |E|; \langle \rho(S), w_1, \ldots, w_d, E \rangle]$, where $\rho(S) \in \mathbb{N}$, $w_1, \ldots, w_d \in \mathbb{Z}^n$ and $E \subset \mathbb{Z}^n$, means that the running time is polynomial in the binary length of $\rho(S), w_1, \ldots, w_d$, and E, and the number of arithmetic operations is polynomial in n and the cardinality $|E|$, which constitute the special part of the input.

Often, as in [2], part of the input is presented by oracles. Then the running time and the number of arithmetic operations count also the number of oracle queries. An oracle algorithm is *polynomial time* if its running time, including the number of oracle queries, and the manipulations of numbers, some of which are answers to oracle queries, is polynomial in the length of the input encoding. An oracle algorithm is *strongly polynomial time* (with specified input encoding as above), if it is polynomial time in the entire input (on the "right"), and in addition, the number of arithmetic operations it performs (including oracle queries) is polynomial in the special part of the input (on the "left").

2. Reducing Convex to Linear Discrete Optimization

In this section we show that when suitable auxiliary geometric information about the convex hull $\mathrm{conv}(S)$ of a finite set $S \subseteq \mathbb{Z}^n$ is available, the convex discrete optimization problem over S can be reduced to the solution of strongly polynomially many linear discrete optimization counterparts over S. This result will be incorporated into the polynomial time algorithms developed in §3 and §5 for convex combinatorial optimization and convex integer programming respectively. In §2.1 we provide some preliminaries on

edge-directions and zonotopes. In §2.2 we prove the reduction which is the main result of this section. In §2.3 we prove a pseudo polynomial reduction for any finite set.

2.1. Edge-Directions and Zonotopes

We begin with some terminology and facts that play an important role in the sequel. A *direction* of an edge (1-dimensional face) $e = [u, v]$ of a polytope P is any nonzero scalar multiple of $u - v$. A set of vectors E *covers all edge-directions of* P if it contains a direction of each edge of P. The *normal cone* of a polytope $P \subset \mathbb{R}^n$ at its face F is the (relatively open) cone C_P^F of those linear functionals $h \in \mathbb{R}^n$ which are maximized over P precisely at points of F. A polytope Z is a *refinement* of a polytope P if the normal cone of every vertex of Z is contained in the normal cone of some vertex of P. If Z refines P then, moreover, the closure of each normal cone of P is the union of closures of normal cones of Z. The *zonotope* generated by a set of vectors $E = \{e_1, \ldots, e_m\}$ in $\mathbb{R}^d$ is the following polytope, which is the projection by E of the cube $[-1, 1]^m$ into $\mathbb{R}^d$,

$$Z := \text{zone}(E) := \text{conv}\Big\{\sum_{i=1}^{m} \lambda_i e_i : \lambda_i = \pm 1\Big\} \subset \mathbb{R}^d.$$

The following fact goes back to Minkowski, see [35].

Lemma 2.1. *Let P be a polytope and let E be a finite set that covers all edge-directions of P. Then the zonotope $Z := \text{zone}(E)$ generated by E is a refinement of P.*

Proof. Consider any vertex u of Z. Then $u = \sum_{e \in E} \lambda_e e$ for suitable $\lambda_e = \pm 1$. Thus, the normal cone C_Z^u consists of those h satisfying $h\lambda_e e > 0$ for all e. Pick any $\hat{h} \in C_Z^u$ and let v be a vertex of P at which $\hat{h}$ is maximized over P. Consider any edge $[v, w]$ of P. Then $v - w = \alpha_e e$ for some scalar $\alpha_e \neq 0$ and some $e \in E$, and $0 \leq \hat{h}(v - w) = \hat{h}\alpha_e e$, implying $\alpha_e \lambda_e > 0$. It follows that every $h \in C_Z^u$ satisfies $h(v - w) > 0$ for every edge of P containing v. Therefore h is maximized over P uniquely at v and hence is in the cone C_P^v of P at v. This shows $C_Z^u \subseteq C_P^v$. Since u was arbitrary, it follows that the normal cone of every vertex of Z is contained in the normal cone of some vertex of P. □

The next lemma provides bounds on the number of vertices of any zonotope and on the algorithmic complexity of constructing its vertices, each vertex along with a linear functional maximized over the zonotope uniquely at that vertex. The bound on the number of vertices has been rediscovered many times over the years. An early reference is [40], stated in the dual form of 2-partitions. A more general treatment is [41]. Recent extensions to p-partitions for any p are in [12, 42], and to Minkowski sums of arbitrary polytopes are in [43]. Interestingly, already in [40], back in 1967, the question was raised about the algorithmic complexity of the problem; this is now settled in [44, 45] (the latter reference correcting the former). We state the precise bounds on the number of vertices and arithmetic complexity, but will need later only that for any fixed d the bounds are polynomial in the number of generators. Therefore, below we only outline a proof that the bounds are polynomial. Complete details are in the above references.

Lemma 2.2. *The number of vertices of any zonotope $Z := \text{zone}(E)$ generated by a set E of m vectors in $\mathbb{R}^d$ is at most $2\sum_{k=0}^{d-1} \binom{m-1}{k}$. For every fixed d, there is a strongly polynomial*

time algorithm that, given $E \subset \mathbb{Z}^d$, encoded as $[m := |E|; \langle E \rangle]$, outputs every vertex v of $Z := \mathrm{zone}(E)$ along with a linear functional $h_v \in \mathbb{Z}^d$ maximized over Z uniquely at v, using $O(m^{d-1})$ arithmetics operations for $d \geq 3$ and $O(m^d)$ for $d \leq 2$.

Proof. We only outline a proof that, for every fixed d, the polynomial bounds $O(m^{d-1})$ on the number of vertices and $O(m^d)$ on the arithmetic complexity hold. We assume that E linearly spans $\mathbb{R}^d$ (else the dimension can be reduced) and is generic, that is, no d points of E lie on a linear hyperplane (one containing the origin). In particular, $0 \notin E$. The same bound for arbitrary E then follows using a perturbation argument (cf. [12]).

Each oriented linear hyperplane $H = \{x \in \mathbb{R}^d : hx = 0\}$ with $h \in \mathbb{R}^d$ nonzero induces a partition of E by $E = H^- \uplus H^0 \uplus H^+$, with $H^- := \{e \in E : he < 0\}$, $E^0 := E \cap H$, and $H^+ := \{e \in E : he > 0\}$. The vertices of $Z = \mathrm{zone}(E)$ are in bijection with ordered 2-partitions of E induced by such hyperplanes that avoid E. Indeed, if $E = H^- \uplus H^+$ then the linear functional $h_v := h$ defining H is maximized over Z uniquely at the vertex $v := \sum\{e : e \in H^+\} - \sum\{e : e \in H^-\}$ of Z.

We now show how to enumerate all such 2-partitions and hence vertices of Z. Let M be any of the $\binom{m}{d-1}$ subsets of E of size $d-1$. Since E is generic, M is linearly independent and spans a unique linear hyperplane $\mathrm{lin}(M)$. Let $\widehat{H} = \{x \in \mathbb{R}^d : \hat{h}x = 0\}$ be one of the two orientations of the hyperplane $\mathrm{lin}(M)$. Note that $\widehat{H}^0 = M$. Finally, let L be any of the 2^{d-1} subsets of M. Since M is linearly independent, there is a $g \in \mathbb{R}^d$ which linearly separates L from $M \setminus L$, namely, satisfies $gx < 0$ for all $x \in L$ and $gx > 0$ for all $x \in M \setminus L$. Furthermore, there is a sufficiently small $\epsilon > 0$ such that the oriented hyperplane $H := \{x \in \mathbb{R}^d : hx = 0\}$ defined by $h := \hat{h} + \epsilon g$ avoids E and the 2-partition induced by H satisfies $H^- = \widehat{H}^- \uplus L$ and $H^+ = \widehat{H}^+ \uplus (M \setminus L)$. The corresponding vertex of Z is $v := \sum\{e : e \in H^+\} - \sum\{e : e \in H^-\}$ and the corresponding linear functional which is maximized over Z uniquely at v is $h_v := h = \hat{h} + \epsilon g$.

We claim that any ordered 2-partition arises that way from some M, some orientation $\widehat{H}$ of $\mathrm{lin}(M)$, and some L. Indeed, consider any oriented linear hyperplane $\widetilde{H}$ avoiding E. It can be perturbed to a suitable oriented $\widehat{H}$ that touches precisely $d-1$ points of E. Put $M := \widehat{H}^0$ so that $\widehat{H}$ coincides with one of the two orientations of the hyperplane $\mathrm{lin}(M)$ spanned by M, and put $L := \widetilde{H}^- \cap M$. Let H be an oriented hyperplane obtained from M, $\widehat{H}$ and L by the above procedure. Then the ordered 2-partition $E = H^- \uplus H^+$ induced by H coincides with the ordered 2-partition $E = \widetilde{H}^- \uplus \widetilde{H}^+$ induced by $\widetilde{H}$.

Since there are $\binom{m}{d-1}$ many $(d-1)$-subsets $M \subseteq E$, two orientations $\widehat{H}$ of $\mathrm{lin}(M)$, and 2^{d-1} subsets $L \subseteq M$, and d is fixed, the total number of 2-partitions and hence also the total number of vertices of Z obey the upper bound $2^d\binom{m}{d-1} = O(m^{d-1})$. Furthermore, for each choice of M, $\widehat{H}$ and L, the linear functional $\hat{h}$ defining $\widehat{H}$, as well as g, ϵ, $h_v = h = \hat{h} + \epsilon g$, and the vertex $v = \sum\{e : e \in H^+\} - \sum\{e : e \in H^-\}$ of Z at which h_v is uniquely maximized over Z, can all be computed using $O(m)$ arithmetic operations. This shows the claimed bound $O(m^d)$ on the arithmetic complexity. □

We conclude with a simple fact about edge-directions of projections of polytopes.

Lemma 2.3. *If E covers all edge-directions of a polytope P, and $Q := \omega(P)$ is the image of P under a linear map $\omega\colon \mathbb{R}^n \to \mathbb{R}^d$, then $\omega(E)$ covers all edge-directions of Q.*

Proof. Let f be a direction of an edge $[x, y]$ of Q. Consider the face $F := \omega^{-1}([x, y])$ of P. Let V be the set of vertices of F and let $U = \{u \in V : \omega(u) = x\}$. Then for some $u \in U$

and $v \in V \setminus U$, there must be an edge $[u, v]$ of F, and hence of P. Then $\omega(v) \in (x, y]$ hence $\omega(v) = x + \alpha f$ for some $\alpha \neq 0$. Therefore, with $e := (1/\alpha)(v - u)$, a direction of the edge $[u, v]$ of P, we find that $f = (1/\alpha)(\omega(v) - \omega(u)) = \omega(e) \in \omega(E)$. □

2.2. Strongly Polynomial Reduction of Convex to Linear Discrete Optimization

A *linear discrete optimization oracle* for a set $S \subseteq \mathbb{Z}^n$ is one that, queried on $w \in \mathbb{Z}^n$, either returns an optimal solution to the linear discrete optimization problem over S, that is, an $x^* \in S$ satisfying $wx^* = \max\{wx : x \in S\}$, or asserts that none exists, that is, either the problem is infeasible or the objective function is unbounded. We now show that a set E covering all edge-directions of the polytope $\mathrm{conv}(S)$ underlying a convex discrete optimization problem over a finite set $S \subset \mathbb{Z}^n$ allows to solve it by solving polynomially many linear discrete optimization counterparts over S. The following theorem extends and unifies the corresponding reductions in [15] and [10] for convex combinatorial optimization and convex integer programming respectively. Recall from §1.3 that the *radius* of a finite set $S \subset \mathbb{Z}^n$ is defined to be $\rho(S) := \max\{|x_i| : x \in S, i = 1, \ldots, n\}$.

Theorem 2.4. *For every fixed d there is a strongly polynomial time algorithm that, given finite set $S \subset \mathbb{Z}^n$ presented by a linear discrete optimization oracle, integer vectors $w_1, \ldots, w_d \in \mathbb{Z}^n$, set $E \subset \mathbb{Z}^n$ covering all edge-directions of* $\mathrm{conv}(S)$*, and convex functional $c\colon \mathbb{R}^d \to \mathbb{R}$ presented by a comparison oracle, encoded as $[n, |E|; \langle \rho(S), w_1, \ldots, w_d, E\rangle]$, solves the convex discrete optimization problem*

$$\max\{c(w_1 x, \ldots, w_d x) : x \in S\}.$$

Proof. First, query the linear discrete optimization oracle presenting S on the trivial linear functional $w = 0$. If the oracle asserts that there is no optimal solution then S is empty so terminate the algorithm asserting that no optimal solution exists to the convex discrete optimization problem either. So assume the problem is feasible. Let $P := \mathrm{conv}(S) \subset \mathbb{R}^n$ and $Q := \{(w_1 x, \ldots, w_d x) : x \in P\} \subset \mathbb{R}^d$. Then Q is a projection of P, and hence by Lemma 2.3 the projection $D := \{(w_1 e, \ldots, w_d e) : e \in E\}$ of the set E is a set covering all edge-directions of Q. Let $Z := \mathrm{zone}(D) \subset \mathbb{R}^d$ be the zonotope generated by D. Since d is fixed, by Lemma 2.2 we can produce in strongly polynomial time all vertices of Z, every vertex v along with a linear functional $h_v \in \mathbb{Z}^d$ maximized over Z uniquely at v. For each of these polynomially many h_v, repeat the following procedure. Define a vector $g_v \in \mathbb{Z}^n$ by $g_{v,j} := \sum_{i=1}^d w_{i,j} h_{v,i}$ for $j = 1, \ldots, n$. Now query the linear discrete optimization oracle presenting S on the linear functional $w := g_v \in \mathbb{Z}^n$. Let $x_v \in S$ be the optimal solution obtained from the oracle, and let $z_v := (w_1 x_v, \ldots, w_d x_v) \in Q$ be its projection. Since $P = \mathrm{conv}(S)$, we have that x_v is also a maximizer of g_v over P. Since for every $x \in P$ and its projection $z := (w_1 x, \ldots, w_d x) \in Q$ we have $h_v z = g_v x$, we conclude that z_v is a maximizer of h_v over Q. Now we claim that each vertex u of Q equals some z_v. Indeed, since Z is a refinement of Q by Lemma 2.1, it follows that there is some vertex v of Z such that h_v is maximized over Q uniquely at u, and therefore $u = z_v$. Since $c(w_1 x, \ldots, w_d x)$ is convex on $\mathbb{R}^n$ and c is convex on $\mathbb{R}^d$, we find that

$$\begin{aligned}\max_{x \in S} c(w_1 x, \ldots, w_d x) &= \max_{x \in P} c(w_1 x, \ldots, w_d x) = \max_{z \in Q} c(z) \\ &= \max\{c(u) : u \text{ vertex of } Q\} = \max\{c(z_v) : v \text{ vertex of } Z\}.\end{aligned}$$

Using the comparison oracle of c, find a vertex v of Z attaining maximum value $c(z_v)$, and output $x_v \in S$, an optimal solution to the convex discrete optimization problem. □

2.3. Pseudo Polynomial Reduction when Edge-Directions are not Available

Theorem 2.4 reduces convex discrete optimization to polynomially many linear discrete optimization counterparts when a set covering all edge-directions of the underlying polytope is available. However, often such a set is not available (see, e.g., [46] for the important case of bipartite matching). We now show how to reduce convex discrete optimization to many linear discrete optimization counterparts when a set covering all edge-directions is not offhand available. In the absence of such a set, the problem is much harder, and the algorithm below is polynomially bounded only in the unary length of the radius $\rho(S)$ and of the linear functionals $w_1, \ldots, w_d$, rather than in their binary length $\langle \rho(S), w_1, \ldots, w_d \rangle$ as in the algorithm of Theorem 2.4. Moreover, an upper bound $\rho \geq \rho(S)$ on the radius of S is required to be given explicitly in advance as part of the input.

Theorem 2.5. *For every fixed d there is a polynomial time algorithm that, given finite set $S \subseteq \mathbb{Z}^n$ presented by a linear discrete optimization oracle, integer $\rho \geq \rho(S)$, vectors $w_1, \ldots, w_d \in \mathbb{Z}^n$, and convex functional $c\colon \mathbb{R}^d \to \mathbb{R}$ presented by a comparison oracle, encoded as $[\rho, w_1, \ldots, w_d]$, solves the convex discrete optimization problem*

$$\max\{c(w_1 x, \ldots, w_d x) : \ x \in S\}.$$

Proof. Let $P := \mathrm{conv}(S) \subset \mathbb{R}^n$, let $T := \{(w_1 x, \ldots, w_d x) : x \in S\}$ be the projection of S by $w_1, \ldots, w_d$, and let $Q := \mathrm{conv}(T) \subset \mathbb{R}^d$ be the corresponding projection of P. Let $r := n\rho \max_{i=1}^{d} \|w_i\|_\infty$ and let $G := \{-r, \ldots, -1, 0, 1, \ldots, r\}^d$. Then $T \subseteq G$ and the number $(2r+1)^d$ of points of G is polynomially bounded in the input as encoded.

Let $D := \{u - v : u, v \in G, u \neq v\}$ be the set of differences of pairs of distinct point of G. It covers all edge-directions of Q since $\mathrm{vert}(Q) \subseteq T \subseteq G$. Moreover, the number of points of D is less than $(2r+1)^{2d}$ and hence polynomial in the input. Now invoke the algorithm of Theorem 2.4: while the algorithm requires a set E covering all edge-directions of P, it needs E only to compute a set D covering all edge-directions of the projection Q (see proof of Theorem 2.4), which here is computed directly. □

3. Convex Combinatorial Optimization and More

In this section we discuss convex combinatorial optimization. The main result is that convex combinatorial optimization over a set $S \subseteq \{0,1\}^n$ presented by a membership oracle can be solved in strongly polynomial time provided it is endowed with a set covering all edge-directions of $\mathrm{conv}(S)$. In particular, the standard linear combinatorial optimization problem over S can be solved in strongly polynomial time as well. In §3.1 we provide some preparatory statements involving various oracle presentation of the feasible set S. In §3.2 we combine these preparatory statements with Theorem 2.4 and prove the main result of this section. An extension to arbitrary finite sets $S \subset \mathbb{Z}^n$ endowed with edge-directions is established in §3.3. We conclude with some applications in §3.4.

As noted in the introduction, when S is contained in $\{0,1\}^n$ we refer to discrete optimization over S also as *combinatorial optimization* over S, to emphasize that S typically represents a family $\mathcal{F} \subseteq 2^N$ of subsets of a ground set $N := \{1,\ldots,n\}$ possessing some combinatorial property of interest (for instance, the family of bases of a matroid over N, see §3.4.2). The convex combinatorial optimization problem then also has the following interpretation (taken in [13,15]). We are given a weighting $\omega\colon N \to \mathbb{Z}^d$ of elements of the ground set by d-dimensional integer vectors. We interpret the weight vector $\omega(j) \in \mathbb{Z}^d$ of element j as representing its value under d criteria (e.g., if N is the set of edges in a network then such criteria may include profit, reliability, flow velocity, etc.). The weight of a subset $F \subseteq N$ is the sum $\omega(F) := \sum_{j\in F} \omega(j)$ of weights of its elements, representing the total value of F under the d criteria. Now, given a convex functional $c\colon \mathbb{R}^d \to \mathbb{R}$, the objective function value of $F \subseteq N$ is the "convex balancing" $c(\omega(F))$ of the values of the weight vector of F. The convex combinatorial optimization problem is to find a family member $F \in \mathcal{F}$ maximizing $c(\omega(F))$. The usual linear combinatorial optimization problem over $\mathcal{F}$ is the special case of $d = 1$ and c the identity on $\mathbb{R}$. To cast a problem of that form in our usual setup just let $S := \{\mathbf{1}_F : F \in \mathcal{F}\} \subseteq \{0,1\}^n$ be the set of indicators of members of $\mathcal{F}$ and define weight vectors $w_1,\ldots,w_d \in \mathbb{Z}^n$ by $w_{i,j} := \omega(j)_i$ for $i = 1,\ldots,d$ and $j = 1,\ldots,n$.

3.1. From Membership to Linear Optimization

A *membership oracle* for a set $S \subseteq \mathbb{Z}^n$ is one that, queried on $x \in \mathbb{Z}^n$, asserts whether or not $x \in S$. An *augmentation oracle* for S is one that, queried on $x \in S$ and $w \in \mathbb{Z}^n$, either returns an $\hat{x} \in S$ with $w\hat{x} > wx$, i.e. a better point of S, or asserts that none exists, i.e. x is optimal for the linear discrete optimization problem over S.

A membership oracle presentation of S is very broad and available in all reasonable applications, but reveals little information on S, making it hard to use. However, as we now show, the edge-directions of $\mathrm{conv}(S)$ allow to convert membership to augmentation.

Lemma 3.1. *There is a strongly polynomial time algorithm that, given set $S \subseteq \{0,1\}^n$ presented by a membership oracle, $x \in S$, $w \in \mathbb{Z}^n$, and set $E \subset \mathbb{Z}^n$ covering all edge-directions of the polytope* $\mathrm{conv}(S)$, *encoded as* $[n, |E|; \langle x, w, E\rangle]$, *either returns a better point $\hat{x} \in S$, that is, one satisfying $w\hat{x} > wx$, or asserts that none exists.*

Proof. Each edge of $P := \mathrm{conv}(S)$ is the difference of two $\{0,1\}$-vectors. Therefore, each edge direction of P is, up to scaling, a $\{-1,0,1\}$-vector. Thus, scaling $e := (1/\|e\|_\infty)e$ and $e := -e$ if necessary, we may and will assume that $e \in \{-1,0,1\}^n$ and $we \geq 0$ for all $e \in E$. Now, using the membership oracle, check if there is an $e \in E$ such that $x + e \in S$ and $we > 0$. If there is such an e then output $\hat{x} := x + e$ which is a better point, whereas if there is no such e then terminate asserting that no better point exists.

Clearly, if the algorithm outputs an $\hat{x}$ then it is indeed a better point. Conversely, suppose x is not a maximizer of w over S. Since $S \subseteq \{0,1\}^n$, the point x is a vertex of P. Since x is not a maximizer of w, there is an edge $[x,\hat{x}]$ of P with $\hat{x}$ a vertex satisfying $w\hat{x} > wx$. But then $e := \hat{x} - x$ is the one $\{-1,0,1\}$ edge-direction of $[x,\hat{x}]$ with $we \geq 0$ and hence $e \in E$. Thus, the algorithm will find and output $\hat{x} = x + e$ as it should. □

An augmentation oracle presentation of a finite S allows to solve the linear discrete optimization problem $\max\{wx : x \in S\}$ over S by starting from any feasible $x \in S$ and

repeatedly augmenting it until an optimal solution $x^* \in S$ is reached. The next lemma bounds the running time needed to reach optimality using this procedure. While the running time is polynomial in the binary length of the linear functional w and the initial point x, it is more sensitive to the radius $\rho(S)$ of the feasible set S, and is polynomial only in its unary length. The lemma is an adaptation of a result of [47,48] (stated therein for $\{0,1\}$-sets), which makes use of bit-scaling ideas going back to [49].

Lemma 3.2. *There is a polynomial time algorithm that, given finite set $S \subset \mathbb{Z}^n$ presented by an augmentation oracle, $x \in S$, and $w \in \mathbb{Z}^n$, encoded as $[\rho(S), \langle x, w\rangle]$, provides an optimal solution $x^* \in S$ to the linear discrete optimization problem* $\max\{wz : z \in S\}$.

Proof. Let $k := \max_{j=1}^{n} \lceil \log_2(|w_j| + 1) \rceil$ and note that $k \leq \langle w \rangle$. For $i = 0, \ldots, k$ define a linear functional $u_i = (u_{i,1}, \ldots, u_{i,n}) \in \mathbb{Z}^n$ by $u_{i,j} := \operatorname{sign}(w_j) \lfloor 2^{i-k} |w_j| \rfloor$ for $j = 1, \ldots, n$. Then $u_0 = 0$, $u_k = w$, and $u_i - 2u_{i-1} \in \{-1, 0, 1\}^n$ for all $i = 1, \ldots, k$.

We now describe how to construct a sequence of points $y_0, y_1, \ldots, y_k \in S$ such that y_i is an optimal solution to $\max\{u_i y : y \in S\}$ for all i. First note that all points of S are optimal for $u_0 = 0$ and hence we can take $y_0 := x$ to be the point of S given as part of the input. We now explain how to determine y_i from y_{i-1} for $i = 1, \ldots, k$. Suppose y_{i-1} has been determined. Set $\tilde{y} := y_{i-1}$. Query the augmentation oracle on $\tilde{y} \in S$ and u_i; if the oracle returns a better point $\hat{y}$ then set $\tilde{y} := \hat{y}$ and repeat, whereas if it asserts that there is no better point then the optimal solution for u_i is read off to be $y_i := \tilde{y}$. We now bound the number of calls to the oracle. Each time the oracle is queried on $\tilde{y}$ and u_i and returns a better point $\hat{y}$, the improvement is by at least one, i.e., $u_i(\hat{y} - \tilde{y}) \geq 1$; this is so because $u_i, \tilde{y}$ and $\hat{y}$ are integer. Thus, the number of necessary augmentations from y_{i-1} to y_i is at most the total improvement, which we claim satisfies

$$u_i(y_i - y_{i-1}) = (u_i - 2u_{i-1})(y_i - y_{i-1}) + 2u_{i-1}(y_i - y_{i-1}) \leq 2n\rho + 0 = 2n\rho,$$

where $\rho := \rho(S)$. Indeed, $u_i - 2u_{i-1} \in \{-1, 0, 1\}^n$ and $y_i, y_{i-1} \in S \subset [-\rho, \rho]^n$ imply $(u_i - 2u_{i-1})(y_i - y_{i-1}) \leq 2n\rho$; and y_{i-1} optimal for u_{i-1} gives $u_{i-1}(y_i - y_{i-1}) \leq 0$.

Thus, after a total number of at most $2n\rho k$ calls to the oracle we obtain y_k which is optimal for u_k. Since $w = u_k$ we can output $x^* := y_k$ as the desired optimal solution to the linear discrete optimization problem. Clearly the number $2n\rho k$ of calls to the oracle, as well as the number of arithmetic operations and binary length of numbers occurring during the algorithm, are polynomial in $\rho(S), \langle x, w\rangle$. This completes the proof. □

We conclude this preparatory subsection by recording the following result of [50] which incorporates the heavy simultaneous Diophantine approximation of [51].

Proposition 3.3. *There is a strongly polynomial time algorithm that, given $w \in \mathbb{Z}^n$, encoded as $[n; \langle w \rangle]$, produces $\widehat{w} \in \mathbb{Z}^n$, whose binary length $\langle \widehat{w} \rangle$ is polynomially bounded in n and independent of w, and with* $\operatorname{sign}(\widehat{w}z) = \operatorname{sign}(wz)$ *for every $z \in \{-1, 0, 1\}^n$.*

3.2. Linear and Convex Combinatorial Optimization in Strongly Polynomial Time

Combining the preparatory statements of §3.1 with Theorem 2.4, we can now solve the convex combinatorial optimization over a set $S \subseteq \{0,1\}^n$ presented by a membership oracle and endowed with a set covering all edge-directions of $\operatorname{conv}(S)$ in strongly polynomial time. We start with the special case of linear combinatorial optimization.

Theorem 3.4. *There is a strongly polynomial time algorithm that, given set* $S \subseteq \{0,1\}^n$ *presented by a membership oracle,* $x \in S$, $w \in \mathbb{Z}^n$, *and set* $E \subset \mathbb{Z}^n$ *covering all edge-directions of the polytope* $\mathrm{conv}(S)$, *encoded as* $[n, |E|; \langle x, w, E\rangle]$, *provides an optimal solution* $x^* \in S$ *to the linear combinatorial optimization problem* $\max\{wz : z \in S\}$.

Proof. First, an augmentation oracle for S can be simulated using the membership oracle, in strongly polynomial time, by applying the algorithm of Lemma 3.1

Next, using the simulated augmentation oracle for S, we can now do linear optimization over S in strongly polynomial time as follows. First, apply to w the algorithm of Proposition 3.3 and obtain $\widehat{w} \in \mathbb{Z}^n$ whose binary length $\langle\widehat{w}\rangle$ is polynomially bounded in n, which satisfies $\mathrm{sign}(\widehat{w}z) = \mathrm{sign}(wz)$ for every $z \in \{-1,0,1\}^n$. Since $S \subseteq \{0,1\}^n$, it is finite and has radius $\rho(S) = 1$. Now apply the algorithm of Lemma 3.2 to S, x and $\widehat{w}$, and obtain a maximizer x^* of $\widehat{w}$ over S. For every $y \in \{0,1\}^n$ we then have $x^* - y \in \{-1,0,1\}^n$ and hence $\mathrm{sign}(w(x^* - y)) = \mathrm{sign}(\widehat{w}(x^* - y))$. So x^* is also a maximizer of w over S and hence an optimal solution to the given linear combinatorial optimization problem. Now, $\rho(S) = 1$, $\langle\widehat{w}\rangle$ is polynomial in n, and $x \in \{0,1\}^n$ and hence $\langle x\rangle$ is linear in n. Thus, the entire length of the input $[\rho(S), \langle x, \widehat{w}\rangle]$ to the polynomial-time algorithm of Lemma 3.2 is polynomial in n, and so its running time is in fact strongly polynomial on that input. □

Combining Theorems 2.4 and 3.4 we recover at once the following result of [15].

Theorem 3.5. *For every fixed* d *there is a strongly polynomial time algorithm that, given set* $S \subseteq \{0,1\}^n$ *presented by a membership oracle,* $x \in S$, *vectors* $w_1, \ldots, w_d \in \mathbb{Z}^n$, *set* $E \subset \mathbb{Z}^n$ *covering all edge-directions of the polytope* $\mathrm{conv}(S)$, *and convex functional* $c\colon \mathbb{R}^d \to \mathbb{R}$ *presented by a comparison oracle, encoded as* $[n, |E|; \langle x, w_1, \ldots, w_d, E\rangle]$, *provides an optimal solution* $x^* \in S$ *to the convex combinatorial optimization problem*

$$\max\{c(w_1 z, \ldots, w_d z) : z \in S\}.$$

Proof. Since S is nonempty, a linear discrete optimization oracle for S can be simulated in strongly polynomial time by the algorithm of Theorem 3.4. Using this simulated oracle, we can apply the algorithm of Theorem 2.4 and solve the given convex combinatorial optimization problem in strongly polynomial time. □

3.3. Linear and Convex Discrete Optimization over any Set in Pseudo Polynomial Time

In §3.2 above we developed strongly polynomial time algorithms for linear and convex discrete optimization over $\{0,1\}$-sets. We now provide extensions of these algorithms to arbitrary finite sets $S \subset \mathbb{Z}^n$. As can be expected, the algorithms become slower.

We start by recording the following fundamental result of Khachiyan [52] asserting that linear programming is polynomial time solvable via the ellipsoid method [33]. This result will be used below as well as several more times later in the monograph.

Proposition 3.6. *There is a polynomial time algorithm that, given* $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$, *and* $w \in \mathbb{Z}^n$, *encoded as* $[\langle A, b, w\rangle]$, *either asserts that* $P := \{x \in \mathbb{R}^n : Ax \le b\}$ *is empty, or asserts that the linear functional* wx *is unbounded over* P, *or provides a vertex* $v \in \mathrm{vert}(P)$ *which is an optimal solution to the linear program* $\max\{wx : x \in P\}$.

The following analog of Lemma 3.1 shows how to covert membership to augmentation in polynomial time, albeit, no longer in strongly polynomial time. Here, both the given initial point x and the returned better point $\hat{x}$ if any, are *vertices* of $\mathrm{conv}(S)$.

Lemma 3.7. *There is a polynomial time algorithm that, given finite set $S \subset \mathbb{Z}^n$ presented by a membership oracle, vertex x of the polytope $\mathrm{conv}(S)$, $w \in \mathbb{Z}^n$, and set $E \subset \mathbb{Z}^n$ covering all edge-directions of $\mathrm{conv}(S)$, encoded as $[\rho(S), \langle x, w, E\rangle]$, either returns a better vertex $\hat{x}$ of $\mathrm{conv}(S)$, that is, one satisfying $w\hat{x} > wx$, or asserts that none exists.*

Proof. Dividing each vector $e \in E$ by the greatest common divisor of its entries and setting $e := -e$ if necessary, we can and will assume that each e is *primitive*, that is, its entries are relatively prime integers, and $we \geq 0$. Using the membership oracle, construct the subset $F \subseteq E$ of those $e \in E$ for which $x + re \in S$ for some $r \in \{1, \ldots, 2\rho(S)\}$. Let $G \subseteq F$ be the subset of those $f \in F$ for which $wf > 0$. If G is empty then terminate asserting that there is no better vertex. Otherwise, consider the convex cone $\mathrm{cone}(F)$ generated by F. It is clear that x is incident on an edge of $\mathrm{conv}(S)$ in direction f if and only if f is an extreme ray of $\mathrm{cone}(F)$. Moreover, since $G = \{f \in F : wf > 0\}$ is nonempty, there must be an extreme ray of $\mathrm{cone}(F)$ which lies in G. Now $f \in F$ is an extreme ray of $\mathrm{cone}(F)$ if and only if there do not exist nonnegative λ_e, $e \in F \setminus \{f\}$, such that $f = \sum_{e \neq f} \lambda_e e$; this can be checked in polynomial time using linear programming. Applying this procedure to each $f \in G$, identify an extreme ray $g \in G$. Now, using the membership oracle, determine the largest $r \in \{1, \ldots, 2\rho(S)\}$ for which $x + rg \in S$. Output $\hat{x} := x + rg$ which is a better vertex of $\mathrm{conv}(S)$. □

We now prove the extensions of Theorems 3.4 and 3.5 to arbitrary, not necessarily $\{0, 1\}$-valued, finite sets. While the running time remains polynomial in the binary length of the weights $w_1, \ldots, w_d$ and the set of edge-directions E, it is more sensitive to the radius $\rho(S)$ of the feasible set S, and is polynomial only in its unary length. Here, the initial feasible point and the optimal solution output by the algorithms are vertices of $\mathrm{conv}(S)$. Again, we start with the special case of linear combinatorial optimization.

Theorem 3.8. *There is a polynomial time algorithm that, given finite $S \subset \mathbb{Z}^n$ presented by a membership oracle, vertex x of the polytope $\mathrm{conv}(S)$, $w \in \mathbb{Z}^n$, and set $E \subset \mathbb{Z}^n$ covering all edge-directions of $\mathrm{conv}(S)$, encoded as $[\rho(S), \langle x, w, E\rangle]$, provides an optimal solution $x^* \in S$ to the linear discrete optimization problem $\max\{wz : z \in S\}$.*

Proof. Apply the algorithm of Lemma 3.2 to the given data. Consider any query $x' \in S$, $w' \in \mathbb{Z}^n$ made by that algorithm to an augmentation oracle for S. To answer it, apply the algorithm of Lemma 3.7 to x' and w'. Since the first query made by the algorithm of Lemma 3.2 is on the given input vertex $x' := x$, and any consequent query is on a point $x' := \hat{x}$ which was the reply of the augmentation oracle to the previous query (see proof of Lemma 3.2), we see that the algorithm of Lemma 3.7 will always be asked on a vertex of S and reply with another. Thus, the algorithm of Lemma 3.7 can answer all augmentation queries and enables the polynomial time solution of the given problem. □

Theorem 3.9. *For every fixed d there is a polynomial time algorithm that, given finite set $S \subseteq \mathbb{Z}^n$ presented by membership oracle, vertex x of $\mathrm{conv}(S)$, vectors $w_1, \ldots, w_d \in \mathbb{Z}^n$ set $E \subset \mathbb{Z}^n$ covering all edge-directions of the polytope $\mathrm{conv}(S)$, and convex functional*

$c\colon \mathbb{R}^d \to \mathbb{R}$ presented by a comparison oracle, encoded as $[\rho(S), \langle x, w_1, \ldots, w_d, E\rangle]$, provides an optimal solution $x^* \in S$ to the convex combinatorial optimization problem

$$\max\{c(w_1 z, \ldots, w_d z) : z \in S\}.$$

Proof. Since S is nonempty, a linear discrete optimization oracle for S can be simulated in polynomial time by the algorithm of Theorem 3.8. Using this simulated oracle, we can apply the algorithm of Theorem 2.4 and solve the given problem in polynomial time. □

3.4. Some Applications

3.4.1. Positive Semidefinite Quadratic Binary Programming

The quadratic binary programming problem is the following: given an $n \times n$ matrix M, find a vector $x \in \{0,1\}^n$ maximizing the quadratic form $x^T Mx$ induced by M. We consider here the instance where M is positive semidefinite, in which case it can be assumed to be presented as $M = W^T W$ with W a given $d \times n$ matrix. Already this restricted version is very broad: if the rank d of W and M is variable then, as mentioned in the introduction, the problem is NP-hard. We now show that, for fixed d, Theorem 3.5 implies at once that the problem is strongly polynomial time solvable (see also [53]).

Corollary 3.10. *For every fixed d there is a strongly polynomial time algorithm that given $W \in \mathbb{Z}^{d\times n}$, encoded as $[n; \langle W\rangle]$, finds $x^* \in \{0,1\}^n$ maximizing the form $x^T W^T W x$.*

Proof. Let $S := \{0,1\}^n$ and let $E := \{\mathbf{1}_1, \ldots, \mathbf{1}_n\}$ be the set of unit vectors in $\mathbb{R}^n$. Then $P := \mathrm{conv}(S)$ is just the n-cube $[0,1]^n$ and hence E covers all edge-directions of P. A membership oracle for S is easily and efficiently realizable and $x := 0 \in S$ is an initial point. Also, $|E|$ and $\langle E\rangle$ are polynomial in n, and E is easily and efficiently computable.

Now, for $i = 1, \ldots, d$ define $w_i \in \mathbb{Z}^n$ to be the ith row of the matrix W, that is, $w_{i,j} := W_{i,j}$ for all i, j. Finally, let $c\colon \mathbb{R}^d \to \mathbb{R}$ be the squared l_2 norm given by $c(y) := \|y\|_2^2 := \sum_{i=1}^d y_i^2$, and note that the comparison of $c(y)$ and $c(z)$ can be done for $y, z \in \mathbb{Z}^d$ in time polynomial in $\langle y, z\rangle$ using a constant number of arithmetic operations, providing a strongly polynomial time realization of a comparison oracle for c.

This translates the given quadratic programming problem into a convex combinatorial optimization problem over S, which can be solved in strongly polynomial time by applying the algorithm of Theorem 3.5 to S, $x = 0$, $w_1, \ldots, w_d$, E, and c. □

3.4.2. Matroids and Maximum Norm Spanning Trees

Optimization problems over matroids form a fundamental class of combinatorial optimization problems. Here we discuss matroid bases, but everything works for independent sets as well. Recall that a family $\mathcal{B}$ of subsets of $\{1, \ldots, n\}$ is the family of *bases* of a *matroid* if all members of $\mathcal{B}$ have the same cardinality, called the *rank* of the matroid, and for every $B, B' \in \mathcal{B}$ and $i \in B \setminus B'$ there is a $j \in B'$ such that $B \setminus \{i\} \cup \{j\} \in \mathcal{B}$. Useful models include the *graphic matroid* of a graph G with edge set $\{1, \ldots, n\}$ and $\mathcal{B}$ the family of spanning forests of G, and the *linear matroid* of an $m \times n$ matrix A with $\mathcal{B}$ the family of sets of indices of maximal linearly independent subsets of columns of A.

It is well known that linear combinatorial optimization over matroids can be solved by the fast greedy algorithm [54]. We now show that, as a consequence of Theorem 3.5,

convex combinatorial optimization over a matroid presented by a membership oracle can be solved in strongly polynomial time as well (see also [13, 55]). We state the result for bases, but the analogous statement for independent sets hold as well. We say that $S \subseteq \{0,1\}^n$ is the *set of bases of a matroid* if it is the set of indicators of the family $\mathcal{B}$ of bases of some matroid, in which case we call $\mathrm{conv}(S)$ the *matroid base polytope*.

Corollary 3.11. *For every fixed d there is a strongly polynomial time algorithm that, given set $S \subseteq \{0,1\}^n$ of bases of a matroid presented by a membership oracle, $x \in S$, $w_1, \ldots, w_d \in \mathbb{Z}^n$, and convex functional $c\colon \mathbb{R}^d \to \mathbb{R}$ presented by a comparison oracle, encoded as $[n; \langle x, w_1, \ldots, w_d\rangle]$, solves the convex matroid optimization problem*

$$\max\{c(w_1 z, \ldots, w_d z) : z \in S\}.$$

Proof. Let $E := \{\mathbf{1}_i - \mathbf{1}_j : 1 \le i < j \le n\}$ be the set of differences of pairs of unit vectors in $\mathbb{R}^n$. We claim that E covers all edge-directions of the matroid base polytope $P := \mathrm{conv}(S)$. Consider any edge $e = [y, y']$ of P with $y, y' \in S$ and let $B := \mathrm{supp}(y)$ and $B' := \mathrm{supp}(y')$ be the corresponding bases. Let $h \in \mathbb{R}^n$ be a linear functional uniquely maximized over P at e. If $B \setminus B' = \{i\}$ is a singleton then $B' \setminus B = \{j\}$ is a singleton as well in which case $y - y' = \mathbf{1}_i - \mathbf{1}_j$ and we are done. Suppose then, indirectly, that it is not, and pick an element i in the symmetric difference $B \Delta B' := (B \setminus B') \cup (B' \setminus B)$ of minimum value h_i. Without loss of generality assume $i \in B \setminus B'$. Then there is a $j \in B' \setminus B$ such that $B'' := B \setminus \{i\} \cup \{j\}$ is also a basis. Let $y'' \in S$ be the indicator of B''. Now $|B \Delta B'| > 2$ implies that B'' is neither B nor B'. By the choice of i we have $hy'' = hy - h_i + h_j \ge hy$. So y'' is also a maximizer of h over P and hence $y'' \in e$. But no $\{0,1\}$-vector is a convex combination of others, a contradiction.

Now, $|E| = \binom{n}{2}$ and $E \subset \{-1, 0, 1\}^n$ imply that $|E|$ and $\langle E \rangle$ are polynomial in n. Moreover, E can be easily computed in strongly polynomial time. Therefore, applying the algorithm of Theorem 3.5 to the given data and the set E, the convex discrete optimization problem over S can be solved in strongly polynomial time. □

One important application of Corollary 3.11 is a polynomial time algorithm for computing the *universal Gröbner basis* of any system of polynomials having a finite set of common zeros in fixed (but arbitrary) number of variables, as well as the construction of the *state polyhedron* of any member of the *Hilbert scheme*, see [4, 17]. Other important applications are in the field of *algebraic statistics* [56], in particular for *optimal experimental design*. These applications are beyond our scope here and will be discussed elsewhere.

Here is another concrete example of a convex matroid optimization application.

Example 3.12 (Maximum Norm Spanning Tree)**.** Fix any positive integer d. Let $\|\cdot\|_p\colon \mathbb{R}^d \to \mathbb{R}$ be the l_p norm given by $\|x\|_p := (\sum_{i=1}^d |x_i|^p)^{1/p}$ for $1 \le p < \infty$ and $\|x\|_\infty := \max_{i=1}^d |x_i|$. Let G be a connected graph with edge set $N := \{1, \ldots, n\}$. For $j = 1, \ldots, n$ let $u_j \in \mathbb{Z}^d$ be a weight vector representing the values of edge j under some d criteria. The weight of a subset $T \subseteq N$ is the sum $\sum_{j \in T} u_j$ representing the total values of T under the d criteria. The problem is to find a spanning tree T of G whose weight has maximum l_p norm, that is, a spanning tree T maximizing $\|\sum_{j \in T} u_j\|_p$.

Define $w_1, \ldots, w_d \in \mathbb{Z}^n$ by $w_{i,j} := u_{j,i}$ for $i = 1, \ldots, d$, $j = 1, \ldots, n$. Let $S \subseteq \{0,1\}^n$ be the set of indicators of spanning trees of G. Then, in time polynomial in n, a

membership oracle for S is realizable, and an initial $x \in S$ is obtainable as the indicator of any greedily constructible spanning tree T. Finally, define the convex functional $c := \|\cdot\|_p$. Then for most common values $p = 1, 2, \infty$, and in fact for any $p \in \mathbb{N}$, the comparison of $c(y)$ and $c(z)$ can be done for $y, z \in \mathbb{Z}^d$ in time polynomial in $\langle y, z, p\rangle$ by computing and comparing the integer valued pth powers $\|y\|_p^p$ and $\|z\|_p^p$. Thus, by Corollary 3.11, this problem is solvable in time polynomial in $\langle u_1, \ldots, u_n, p\rangle$.

4. Linear *n*-fold Integer Programming

In this section we develop a theory of linear *n-fold integer programming*, which leads to the polynomial time solution of broad classes of linear integer programming problems in variable dimension. This will be extended to convex n-fold integer programming in §5.

In §4.1 we describe an adaptation of a result of [57] involving an oriented version of the augmentation oracle of §3.1. In §4.2 we discuss Graver bases and their application to linear integer programming. In §4.3 we show that Graver bases of n-fold matrices can be computed efficiently. In §4.4 we combine the preparatory statements from §4.1, §4.2, and §4.3, and prove the main result of this section, asserting that linear n-fold integer programming is polynomial time solvable. We conclude with some applications in §4.5.

Here and in §5 we concentrate on discrete optimization problems over a set S presented as the set of integer points satisfying an explicitly given system of linear inequalities. Without loss of generality we may and will assume that S is given either in standard form $S := \{x \in \mathbb{N}^n : Ax = b\}$ where $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$, or in the form

$$S := \{x \in \mathbb{Z}^n : Ax = b, l \le x \le u\}$$

where $l, u \in \mathbb{Z}_\infty^n$ and $\mathbb{Z}_\infty = \mathbb{Z} \uplus \{\pm\infty\}$, where some of the variables are bounded below or above and some are unbounded. Thus, S is no longer presented by an oracle, but by the explicit data A, b and possibly l, u. In this setup we refer to discrete optimization over S also as *integer programming* over S. As usual, an algorithm solving the problem must either provide an $x \in S$ maximizing wx over S, or assert that none exists (either because S is empty or because the objective function is unbounded over the underlying polyhedron). We will sometimes assume that an initial point $x \in S$ is given, in which case b will be computed as $b := Ax$ and not be part of the input.

4.1. Oriented Augmentation and Linear Optimization

We have seen in §3.1 that an augmentation oracle presentation of a finite set $S \subset \mathbb{Z}^n$ enables to solve the linear discrete optimization problem over S. However, the running time of the algorithm of Lemma 3.2 which demonstrated this, was polynomial in the unary length of the radius $\rho(S)$ of the feasible set rather than in its binary length.

In this subsection we discuss a recent result of [57] and show that, when S is presented by a suitable stronger oriented version of the augmentation oracle, the linear optimization problem can be solved by a much faster algorithm, whose running time is in fact polynomial in the binary length $\langle\rho(S)\rangle$. The key idea behind this algorithm is that it gives preference to augmentations along interior points of $\mathrm{conv}(S)$ staying far off its boundary. It is inspired by and extends the combinatorial interior point algorithm of [58].

For any vector $g \in \mathbb{R}^n$, let $g^+, g^- \in \mathbb{R}^n_+$ denote its *positive* and *negative* parts, defined by $g^+_j := \max\{g_j, 0\}$ and $g^-_j := -\min\{g_j, 0\}$ for $j = 1, \ldots, n$. Note that both g^+, g^- are nonnegative, $\mathrm{supp}(g) = \mathrm{supp}(g^+) \uplus \mathrm{supp}(g^-)$, and $g = g^+ - g^-$.

An *oriented augmentation oracle* for a set $S \subset \mathbb{Z}^n$ is one that, queried on $x \in S$ and $w_+, w_- \in \mathbb{Z}^n$, either returns an *augmenting vector* $g \in \mathbb{Z}^n$, defined to be one satisfying $x + g \in S$ and $w_+ g^+ - w_- g^- > 0$, or asserts that none exists.

Note that this oracle involves *two* linear functionals $w_+, w_- \in \mathbb{Z}^n$ rather than one (w_+, w_- are two distinct independent vectors and *not* the positive and negative parts of one vector). The conditions on an augmenting vector g indicate that it is a feasible direction and has positive value under the nonlinear objective function determined by w_+, w_-. Note that this oracle is indeed stronger than the augmentation oracle of §3.1: to answer a query $x \in S$, $w \in \mathbb{Z}^n$ to the latter, set $w_+ := w_- := w$, thereby obtaining $w_+ g^+ - w_- g^- = wg$ for all g, and query the former on x, w_+, w_-; if it replies with an augmenting vector g then reply with the better point $\hat{x} := x + g$, whereas if it asserts that no g exists then assert that no better point exists.

The following lemma is an adaptation of the result of [57] concerning sets of the form $S := \{x \in \mathbb{Z}^n : Ax = b, 0 \le x \le u\}$ of nonnegative integer points satisfying equations and upper bounds. However, the pair A, b is neither explicitly needed nor does it affect the running time of the algorithm underlying the lemma. It suffices that S is of that form. Moreover, an arbitrary lower bound vector l rather than 0 can be included. So it suffices to assume that S coincides with the intersection of its affine hull and the set of integer points in a box, that is, $S = \mathrm{aff}(S) \cap \{x \in \mathbb{Z}^n : l \le x \le u\}$ where $l, u \in \mathbb{Z}^n$. We now describe and prove the algorithm of [57] adjusted to any lower and upper bounds l, u.

Lemma 4.1. *There is a polynomial time algorithm that, given vectors $l, u \in \mathbb{Z}^n$, set $S \subset \mathbb{Z}^n$ satisfying $S = \mathrm{aff}(S) \cap \{z \in \mathbb{Z}^n : l \le z \le u\}$ and presented by an oriented augmentation oracle, $x \in S$, and $w \in \mathbb{Z}^n$, encoded as $[\langle l, u, x, w\rangle]$, provides an optimal solution $x^* \in S$ to the linear discrete optimization problem* $\max\{wz : z \in S\}$.

Proof. We start with some strengthening adjustments to the oriented augmentation oracle. Let $\rho := \max\{\|l\|_\infty, \|u\|_\infty\}$ be an upper bound on the radius of S. Then any augmenting vector g obtained from the oriented augmentation oracle when queried on $y \in S$ and $w_+, w_- \in \mathbb{Z}^n$, can be made in polynomial time to be *exhaustive*, that is, to satisfy $y + 2g \notin S$ (which means that no longer augmenting step in direction g can be taken). Indeed, using binary search, find the largest $r \in \{1, \ldots, 2\rho\}$ for which $l \le y + rg \le u$; then $S = \mathrm{aff}(S) \cap \{z \in \mathbb{Z}^n : l \le z \le u\}$ implies $y + rg \in S$ and hence we can replace $g := rg$. So from here on we will assume that if there is an augmenting vector then the oracle returns an exhaustive one. Second, let $\mathbb{R}_\infty := \mathbb{R} \uplus \{\pm\infty\}$ and for any vector $v \in \mathbb{R}^n$ let $v^{-1} \in \mathbb{R}^n_\infty$ denote its entry-wise reciprocal defined by $v_i^{-1} := 1/v_i$ if $v_i \ne 0$ and $v_i^{-1} := \infty$ if $v_i = 0$. For any $y \in S$, the vectors $(y - l)^{-1}$ and $(u - y)^{-1}$ are the reciprocals of the "entry-wise distance" of y from the given lower and upper bounds. The algorithm will query the oracle on triples y, w_+, w_- with $w_+ := w - \mu(u - y)^{-1}$ and $w_- := w + \mu(y - l)^{-1}$ where μ is a suitable positive scalar and w is the input linear functional. The fact that such w_+, w_- may have infinite entries does not cause any problem: indeed, if g is an augmenting vector then $y + g \in S$ implies that $g^+_i = 0$ whenever $y_i = u_i$ and $g^-_i = 0$ whenever $l_i = y_i$, so each infinite entry in w_+ or w_- occurring in the expression $w_+ g^+ - w_- g^-$ is multiplied by 0 and hence zeroed out.

The algorithm proceeds in phases. Each phase i starts with a feasible point $y_{i-1} \in S$ and performs repeated augmentations using the oriented augmentation oracle, terminating with a new feasible point $y_i \in S$ when no further augmentations are possible. The queries to the oracle make use of a positive scalar parameters μ_i fixed throughout the phase. The first phase (i=1) starts with the input point $y_0 := x$ and sets $\mu_1 := \rho \|w\|_\infty$. Each further phase $i \geq 2$ starts with the point y_{i-1} obtained from the previous phase and sets the parameter value $\mu_i := \frac{1}{2}\mu_{i-1}$ to be half its value in the previous phase. The algorithm terminates at the end of the first phase i for which $\mu_i < 1/n$, and outputs $x^* := y_i$. Thus, the number of phases is at most $\lceil \log_2(2n\rho\|w\|_\infty) \rceil$ and hence polynomial in $\langle l, u, w \rangle$.

We now describe the ith phase which determines y_i from y_{i-1}. Set $\mu_i := \frac{1}{2}\mu_{i-1}$ and $\hat{y} := y_{i-1}$. Iterate the following: query the strengthened oriented augmentation oracle on $\hat{y}$, $w_+ := w - \mu_i(u - \hat{y})^{-1}$, and $w_- := w + \mu_i(\hat{y} - l)^{-1}$; if the oracle returns an exhaustive augmenting vector g then set $\hat{y} := \hat{y} + g$ and repeat, whereas if it asserts that there is no augmenting vector then set $y_i := \hat{y}$ and complete the phase. If $\mu_i \geq 1/n$ then proceed to the $(i+1)$th phase, else output $x^* := y_i$ and terminate the algorithm.

It remains to show that the output of the algorithm is indeed an optimal solution and that the number of iterations (and hence calls to the oracle) in each phase is polynomial in the input. For this we need the following facts, the easy proofs of which are omitted:

1. For every feasible $y \in S$ and direction g with $y + g \in S$ also feasible, we have

$$(u - y)^{-1}g^+ + (y - l)^{-1}g^- \leq n.$$

2. For every $y \in S$ and direction g with $y + g \in S$ but $y + 2g \notin S$, we have

$$(u - y)^{-1}g^+ + (y - l)^{-1}g^- > \tfrac{1}{2}.$$

3. For every feasible $y \in S$, direction g with $y + g \in S$ also feasible, and $\mu > 0$, setting $w_+ := w - \mu(u - y)^{-1}$ and $w_- := w + \mu(y - l)^{-1}$ we have

$$w_+g^+ - w_-g^- = wg - \mu((u - y)^{-1}g^+ + (y - l)^{-1}g^-).$$

Now, consider the last phase i with $\mu_i < 1/n$, let $x^* := y_i := \hat{y}$ be the output of the algorithm at the end of this phase, and let $\hat{x} \in S$ be any optimal solution. Now, the phase is completed when the oracle, queried on the triple $\hat{y}$, $w_+ = w - \mu_i(u - \hat{y})^{-1}$, and $w_- = w + \mu_i(\hat{y} - l)^{-1}$, asserts that there is no augmenting vector. In particular, setting $g := \hat{x} - \hat{y}$, we find $w_+g^+ - w_-g^- \leq 0$ and hence, by facts 1 and 3 above,

$$w\hat{x} - wx^* = wg \leq \mu_i((u - \hat{y})^{-1}g^+ + (\hat{y} - l)^{-1}g^-) < \frac{1}{n}n = 1.$$

Since $w\hat{x}$ and wx^* are integer, this implies that in fact $w\hat{x} - wx^* \leq 0$ and hence the output x^* of the algorithm is indeed an optimal solution to the given optimization problem.

Next we bound the number of iterations in each phase i starting from $y_{i-1} \in S$. Let again $\hat{x} \in S$ be any optimal solution. Consider any iteration in that phase, where the oracle is queried on $\hat{y}$, $w_+ = w - \mu_i(u - \hat{y})^{-1}$, and $w_- = w + \mu_i(\hat{y} - l)^{-1}$, and returns an exhaustive augmenting vector g. We will now show that

$$w(\hat{y} + g) - w\hat{y} \geq \frac{1}{4n}(w\hat{x} - wy_{i-1}), \tag{1}$$

that is, the increment in the objective value from $\hat{y}$ to the augmented point $\hat{y}+g$ is at least $1/(4n)$ times the difference between the optimal objective value $w\hat{x}$ and the objective value wy_{i-1} of the point y_{i-1} at the beginning of phase i. This shows that at most $4n$ such increments (and hence iterations) can occur in the phase before it is completed.

To establish Eq. (1), we show that $wg \geq \frac{1}{2}\mu_i$ and $w\hat{x} - wy_{i-1} \leq 2n\mu_i$. For the first inequality, note that g is an exhaustive augmenting vector and so $w_+g^+ - w_-g^- > 0$ and $\hat{y}+2g \notin S$ and hence, by facts 2 and 3, $wg > \mu_i((u-\hat{y})^{-1}g^+ + (\hat{y}-l)^{-1}g^-) > \frac{1}{2}\mu_i$. We proceed with the second inequality. If $i=1$ (first phase) then this indeed holds since $w\hat{x} - wy_0 \leq 2n\rho\|w\|_\infty = 2n\mu_1$. If $i \geq 2$, let $\widetilde{w}_+ := w - \mu_{i-1}(u-y_{i-1})^{-1}$ and $\widetilde{w}_- := w + \mu_{i-1}(y_{i-1}-l)^{-1}$. The $(i-1)$th phase was completed when the oracle, queried on the triple y_{i-1}, $\widetilde{w}_+$, and $\widetilde{w}_-$, asserted that there is no augmenting vector. In particular, for $\tilde{g} := \hat{x} - y_{i-1}$, we find $\widetilde{w}_+\tilde{g}^+ - \widetilde{w}_-\tilde{g}^- \leq 0$ and so, by facts 1 and 3,

$$w\hat{x} - wy_{i-1} = w\tilde{g} \leq \mu_{i-1}((u-y_{i-1})^{-1}\tilde{g}^+ + (y_{i-1}-l)^{-1}\tilde{g}^-)) \leq \mu_{i-1}n = 2n\mu_i. \qquad \square$$

4.2. Graver Bases and Linear Integer Programming

We now come to the definition of a fundamental object introduced by Graver in [59]. The *Graver basis* of an integer matrix A is a canonical finite set $\mathcal{G}(A)$ that can be defined as follows. Define a partial order $\sqsubseteq$ on $\mathbb{Z}^n$ which extends the coordinate-wise order $\leq$ on $\mathbb{N}^n$ as follows: for two vectors $u, v \in \mathbb{Z}^n$ put $u \sqsubseteq v$ and say that u is *conformal* to v if $|u_i| \leq |v_i|$ and $u_iv_i \geq 0$ for $i = 1, \ldots, n$, that is, u and v lie in the same orthant of $\mathbb{R}^n$ and each component of u is bounded by the corresponding component of v in absolute value. It is not hard to see that $\sqsubseteq$ is a *well* partial ordering (this is basically Dickson's lemma) and hence every subset of $\mathbb{Z}^n$ has finitely-many $\sqsubseteq$-minimal elements. Let $\mathcal{L}(A) := \{x \in \mathbb{Z}^n : Ax = 0\}$ be the lattice of linear integer dependencies on A. The *Graver basis* of A is defined to be the set $\mathcal{G}(A)$ of all $\sqsubseteq$-minimal vectors in $\mathcal{L}(A) \setminus \{0\}$.

Note that if A is an $m \times n$ matrix then its Graver basis consist of vectors in $\mathbb{Z}^n$. We sometimes write $\mathcal{G}(A)$ as a suitable $|\mathcal{G}(A)| \times n$ matrix whose rows are the Graver basis elements. The Graver basis is centrally symmetric ($g \in \mathcal{G}(A)$ implies $-g \in \mathcal{G}(A)$); thus, when listing a Graver basis we will typically give one of each antipodal pair and prefix the set (or matrix) by $\pm$. Any element of the Graver basis is primitive (its entries are relatively prime integers). Every circuit of A (nonzero primitive minimal support element of $\mathcal{L}(A)$) is in $\mathcal{G}(A)$; in fact, if A is totally unimodular then $\mathcal{G}(A)$ coincides with the set of circuits (see §5.1 in the sequel for more details on this). However, in general $\mathcal{G}(A)$ is much larger. For more details on Graver bases and their connection to Gröbner bases see Sturmfels [60] and for the currently fastest procedure for computing them see [61, 62].

Here is a quick simple example; we will see more structured and complex examples later on. Consider the 1×3 matrix $A := (1, 2, 1)$. Then its Graver basis can be shown to be the set $\mathcal{G}(A) = \pm\{(2,-1,0), (0,-1,2), (1,0,-1), (1,-1,1)\}$. The first three elements (and their antipodes) are the circuits of A; already in this small example noncircuits appear as well: the fourth element (and its antipode) is a primitive linear integer dependency whose support is not minimal.

We now show that when we do have access to the Graver basis, it can be used to solve linear integer programming. We will extend this in §5, where we show that the Graver basis enables to solve convex integer programming as well. In §4.3 we will show that there are important classes of matrices for which the Graver basis is indeed accessible.

First, we need a simple property of Graver bases. A finite sum $u := \sum_i v_i$ of vectors $v_i \in \mathbb{R}^n$ is *conformal* if each summand is conformal to the sum, that is, $v_i \sqsubseteq u$ for all i.

Lemma 4.2. *Let A be any integer matrix. Then any $h \in \mathcal{L}(A) \setminus \{0\}$ can be written as a conformal sum $h := \sum g_i$ of (not necessarily distinct) Graver basis elements $g_i \in \mathcal{G}(A)$.*

Proof. By induction on the well partial order $\sqsubseteq$. Recall that $\mathcal{G}(A)$ is the set of $\sqsubseteq$-minimal elements in $\mathcal{L}(A) \setminus \{0\}$. Consider any $h \in \mathcal{L}(A) \setminus \{0\}$. If it is $\sqsubseteq$-minimal then $h \in \mathcal{G}(A)$ and we are done. Otherwise, there is a $h' \in \mathcal{G}(A)$ such that $h' \sqsubset h$. Set $h'' := h - h'$. Then $h'' \in \mathcal{L}(A) \setminus \{0\}$ and $h'' \sqsubset h$, so by induction there is a conformal sum $h'' = \sum_i g_i$ with $g_i \in \mathcal{G}(A)$ for all i. Now $h = h' + \sum_i g_i$ is the desired conformal sum of h. □

The next lemma shows the usefulness of Graver bases for oriented augmentation.

Lemma 4.3. *Let A be an $m \times n$ integer matrix with Graver basis $\mathcal{G}(A)$ and let $l, u \in \mathbb{Z}^n_\infty$, $w_+, w_- \in \mathbb{Z}^n$, and $b \in \mathbb{Z}^m$. Suppose $x \in T := \{y \in \mathbb{Z}^n : Ay = b, l \le y \le u\}$. Then for every $g \in \mathbb{Z}^n$ which satisfies $x + g \in T$ and $w_+g^+ - w_-g^- > 0$ there exists an element $\hat{g} \in \mathcal{G}(A)$ with $\hat{g} \sqsubseteq g$ which also satisfies $x + \hat{g} \in T$ and $w_+\hat{g}^+ - w_-\hat{g}^- > 0$.*

Proof. Suppose $g \in \mathbb{Z}^n$ satisfies the requirements. Then $Ag = A(x + g) - Ax = b - b = 0$ since $x, x + g \in T$. Thus, $g \in \mathcal{L}(A) \setminus \{0\}$ and hence, by Lemma 4.2, there is a conformal sum $g = \sum_i h_i$ with $h_i \in \mathcal{G}(A)$ for all i. Now, $h_i \sqsubseteq g$ is equivalent to $h_i^+ \le g^+$ and $h_i^- \le g^-$, so the conformal sum $g = \sum_i h_i$ gives corresponding sums of the positive and negative parts $g^+ = \sum_i h_i^+$ and $g^- = \sum_i h_i^-$. Therefore we obtain

$$0 < w_+g^+ - w_-g^- = w_+ \sum_i h_i^+ - w_- \sum_i h_i^- = \sum_i (w_+h_i^+ - w_-h_i^-)$$

which implies that there is some h_i in this sum with $w_+h_i^+ - w_-h_i^- > 0$. Now, $h_i \in \mathcal{G}(A)$ implies $A(x + h_i) = Ax = b$. Also, $l \le x, x + g \le u$ and $h_i \sqsubseteq g$ imply that $l \le x + h_i \le u$. So $x + h_i \in T$. Therefore the vector $\hat{g} := h_i$ satisfies the claim. □

We can now show that the Graver basis enables to solve linear integer programming in polynomial time provided an initial feasible point is available.

Theorem 4.4. *There is a polynomial time algorithm that, given $A \in \mathbb{Z}^{m \times n}$, its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}^n_\infty$, $x, w \in \mathbb{Z}^n$ with $l \le x \le u$, encoded as $[\langle A, \mathcal{G}(A), l, u, x, w\rangle]$, solves the linear integer program $\max\{wz : z \in \mathbb{Z}^n, Az = b, l \le z \le u\}$ with $b := Ax$.*

Proof. First, note that the objective function of the integer program is unbounded if and only if the objective function of its relaxation $\max\{wy : y \in \mathbb{R}^n, Ay = b, l \le y \le u\}$ is unbounded, which can be checked in polynomial time using linear programming. If it is unbounded then assert that there is no optimal solution and terminate the algorithm.

Assume then that the objective is bounded. Then, since the program is feasible, it has an optimal solution. Furthermore, (as basically follows from Cramer's rule, see, e.g., [1, Theorem 17.1]) it has an optimal x^* satisfying $|x^*_j| \le \rho$ for all j, where ρ is an easily computable integer upper bound whose binary length $\langle\rho\rangle$ is polynomially bounded in $\langle A, l, u, x\rangle$. For instance, $\rho := (n + 1)(n + 1)!r^{n+1}$ will do, with r the maximum among $\max_i|\sum_j A_{i,j}x_j|$, $\max_{i,j}|A_{i,j}|$, $\max\{|l_j| : |l_j| < \infty\}$, and $\max\{|u_j| : |u_j| < \infty\}$.

Let $T := \{y \in \mathbb{Z}^n : Ay = b, l \leq y \leq u\}$ and $S := T \cap [-\rho, \rho]^n$. Then our linear integer programming problem now reduces to linear discrete optimization over S. Now, an oriented augmentation oracle for S can be simulated in polynomial time using the given Graver basis $\mathcal{G}(A)$ as follows: given a query $y \in S$ and $w_+, w_- \in \mathbb{Z}^n$, search for $g \in \mathcal{G}(A)$ which satisfies $w_+g^+ - w_-g^- > 0$ and $y + g \in S$; if there is such a g then return it as an augmenting vector, whereas if there is no such g then assert that no augmenting vector exists. Clearly, if this simulated oracle returns a vector g then it is an augmenting vector. On the other hand, if there exists an augmenting vector g then $y + g \in S \subseteq T$ and $w_+g^+ - w_-g^- > 0$ imply by Lemma 4.3 that there is also a $\hat{g} \in \mathcal{G}(A)$ with $\hat{g} \sqsubseteq g$ such that $w_+\hat{g}^+ - w_-\hat{g}^- > 0$ and $y + \hat{g} \in T$. Since $y, y + g \in S$ and $\hat{g} \sqsubseteq g$, we find that $y + \hat{g} \in S$ as well. Therefore the Graver basis contains an augmenting vector and hence the simulated oracle will find and output one.

Define $\hat{l}, \hat{u} \in \mathbb{Z}^n$ by $\hat{l}_j := \max(l_j, -\rho)$, $\hat{u}_j := \min(u_j, \rho)$, $j = 1, \ldots, n$. Then it is easy to see that $S = \text{aff}(S) \cap \{y \in \mathbb{Z}^n : \hat{l} \leq y \leq \hat{u}\}$. Now apply the algorithm of Lemma 4.1 to $\hat{l}, \hat{u}, S, x$, and w, using the above simulated oriented augmentation oracle for S, and obtain in polynomial time a vector $x^* \in S$ which is optimal to the linear discrete optimization problem over S and hence to the given linear integer program. □

As a special case of Theorem 4.4 we recover the following result of [9] concerning linear integer programming in standard form when the Graver basis is available.

Theorem 4.5. *There is a polynomial time algorithm that, given matrix $A \in \mathbb{Z}^{m \times n}$, its Graver basis $\mathcal{G}(A)$, $x \in \mathbb{N}^n$, and $w \in \mathbb{Z}^n$, encoded as $[\langle A, \mathcal{G}(A), x, w\rangle]$, solves the linear integer programming problem* $\max\{wz : z \in \mathbb{N}^n, Az = b\}$ *where $b := Ax$.*

4.3. Graver Bases of n-fold Matrices

As mentioned above, the Graver basis $\mathcal{G}(A)$ of an integer matrix A contains all circuits of A and typically many more elements. While the number of circuits is already typically exponential and can be as large as $\binom{n}{m+1}$, the number of Graver basis elements is usually even larger and depends also on the entries of A and not only on its dimensions m, n. So unfortunately it is typically very hard to compute $\mathcal{G}(A)$. However, we now show that for the important and useful broad class of n-fold matrices, the Graver basis is better behaved and can be computed in polynomial time. Recall the following definition from the introduction. Given an $(r + s) \times t$ matrix A, let A_1 be its $r \times t$ sub-matrix consisting of the first r rows and let A_2 be its $s \times t$ sub-matrix consisting of the last s rows. We refer to A explicitly as $(r + s) \times t$ *matrix*, since the definition below depends also on r and s and not only on the entries of A. The *n-fold matrix* of an $(r + s) \times t$ matrix A is then defined to be the following $(r + ns) \times nt$ matrix,

$$A^{(n)} := (\mathbf{1}_n \otimes A_1) \oplus (I_n \otimes A_2) = \begin{pmatrix} A_1 & A_1 & A_1 & \cdots & A_1 \\ A_2 & 0 & 0 & \cdots & 0 \\ 0 & A_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & A_2 \end{pmatrix}.$$

We now discuss a recent result of [63], which originates in [21], and its extension in [64], on the stabilization of Graver bases of n-fold matrices. Consider vectors $x =$

$(x^1,\ldots,x^n)$ with $x^k \in \mathbb{Z}^t$ for $k = 1,\ldots,n$. The *type* of x is the number $|\{k : x^k \neq 0\}|$ of nonzero components $x^k \in \mathbb{Z}^t$ of x. The *Graver complexity* of an $(r+s)\times t$ matrix, denoted $c(A)$, is defined to be the smallest $c \in \mathbb{N} \uplus \{\infty\}$ such that for all n, the Graver basis of $A^{(n)}$ consists of vectors of type at most $c(A)$. We provide the proof of the following result of [63, 64] stating that the Graver complexity is always finite.

Lemma 4.6. *The Graver complexity $c(A)$ of any $(r+s)\times t$ integer matrix A is finite.*

Proof. Call an element $x = (x^1,\ldots,x^n)$ in the Graver basis of some $A^{(n)}$ *pure* if $x^i \in \mathcal{G}(A_2)$ for all i. Note that the type of a pure $x \in \mathcal{G}(A^{(n)})$ is n. First, we claim that if there is an element of type m in some $\mathcal{G}(A^{(l)})$ then for some $n \geq m$ there is a pure element in $\mathcal{G}(A^{(n)})$, and so it will suffice to bound the type of pure elements. Suppose there is an element of type m in some $\mathcal{G}(A^{(l)})$. Then its restriction to its m nonzero components is an element $x = (x^1,\ldots,x^m)$ in $\mathcal{G}(A^{(m)})$. Let $x^i = \sum_{j=1}^{k_i} g_{i,j}$ be a conformal decomposition of x^i with $g_{i,j} \in \mathcal{G}(A_2)$ for all i, j, and let $n := k_1 + \cdots + k_m \geq m$. Then $g := (g_{1,1},\ldots,g_{m,k_m})$ is in $\mathcal{G}(A^{(n)})$, else there would be $\hat{g} \sqsubset g$ in $\mathcal{G}(A^{(n)})$ in which case the nonzero $\hat{x}$ with $\hat{x}^i := \sum_{j=1}^{k_i} \hat{g}_{i,j}$ for all i would satisfy $\hat{x} \sqsubset x$ and $\hat{x} \in \mathcal{L}(A^{(m)})$, contradicting $x \in \mathcal{G}(A^{(m)})$. Thus g is a pure element of type $n \geq m$, proving the claim.

We proceed to bound the type of pure elements. Let $\mathcal{G}(A_2) = \{g_1,\ldots,g_m\}$ be the Graver basis of A_2 and let G_2 be the $t\times m$ matrix whose columns are the g_i. Suppose $x = (x^1,\ldots,x^n) \in \mathcal{G}(A^{(n)})$ is pure for some n. Let $v \in \mathbb{N}^m$ be the vector with $v_i := |\{k : x^k = g_i\}|$ counting the number of g_i components of x for each i. Then $\sum_{i=1}^m v_i$ is equal to the type n of x. Next, note that $A_1G_2v = A_1(\sum_{k=1}^n x^k) = 0$ and hence $v \in \mathcal{L}(A_1G_2)$. We claim that, moreover, $v \in \mathcal{G}(A_1G_2)$. Suppose indirectly not. Then there is $\hat{v} \in \mathcal{G}(A_1G_2)$ with $\hat{v} \sqsubset v$, and it is easy to obtain a nonzero $\hat{x} \sqsubset x$ from x by zeroing out some components so that $\hat{v}_i = |\{k : \hat{x}^k = g_i\}|$ for all i. Then $A_1(\sum_{k=1}^n \hat{x}^k) = A_1G_2\hat{v} = 0$ and hence $\hat{x} \in \mathcal{L}(A^{(n)})$, contradicting $x \in \mathcal{G}(A^{(n)})$.

So the type of any pure element, and hence the Graver complexity of A, is at most the largest value $\sum_{i=1}^m v_i$ of any nonnegative element v of the Graver basis $\mathcal{G}(A_1G_2)$. □

Using Lemma 4.6 we now show how to compute $\mathcal{G}(A^{(n)})$ in polynomial time.

Theorem 4.7. *For every fixed $(r+s)\times t$ integer matrix A there is a strongly polynomial time algorithm that, given $n \in \mathbb{N}$, encoded as $[n;n]$, computes the Graver basis $\mathcal{G}(A^{(n)})$ of the n-fold matrix $A^{(n)}$. In particular, the cardinality $|\mathcal{G}(A^{(n)})|$ and binary length $\langle\mathcal{G}(A^{(n)})\rangle$ of the Graver basis of the n-fold matrix are polynomially bounded in n.*

Proof. Let $c := c(A)$ be the Graver complexity of A and consider any $n \geq c$. We show that the Graver basis of $A^{(n)}$ is the union of $\binom{n}{c}$ suitably embedded copies of the Graver basis of $A^{(c)}$. For every c indices $1 \leq k_1 < \cdots < k_c \leq n$ define a map $\phi_{k_1,\ldots,k_c}$ from $\mathbb{Z}^{ct}$ to $\mathbb{Z}^{nt}$ sending $x = (x^1,\ldots,x^c)$ to $y = (y^1,\ldots,y^n)$ with $y^{k_i} := x^i$ for $i = 1,\ldots,c$ and $y^k := 0$ for $k \notin \{k_1,\ldots,k_c\}$. We claim that $\mathcal{G}(A^{(n)})$ is the union of the images of $\mathcal{G}(A^{(c)})$ under the $\binom{n}{c}$ maps $\phi_{k_1,\ldots,k_c}$ for all $1 \leq k_1 < \cdots < k_c \leq n$, that is,

$$\mathcal{G}(A^{(n)}) = \bigcup_{1\leq k_1<\cdots<k_c\leq n} \phi_{k_1,\ldots,k_c}(\mathcal{G}(A^{(c)})). \tag{2}$$

If $x = (x^1,\ldots,x^c) \in \mathcal{G}(A^{(c)})$ then x is a $\sqsubseteq$-minimal nonzero element of $\mathcal{L}(A^{(c)})$, implying that $\phi_{k_1,\ldots,k_c}(x)$ is a $\sqsubseteq$-minimal nonzero element of $\mathcal{L}(A^{(n)})$ and therefore we have

$\phi_{k_1,\ldots,k_c}(x) \in \mathcal{G}(A^{(n)})$. So the right-hand side of Eq. (2) is contained in the left-hand side. Conversely, consider any $y \in \mathcal{G}(A^{(n)})$. Then, by Lemma 4.6, the type of y is at most c, so there are indices $1 \leq k_1 < \cdots < k_c \leq n$ such that all nonzero components of y are among those of the reduced vector $x := (y^{k_1}, \ldots, y^{k_c})$ and therefore $y = \phi_{k_1,\ldots,k_c}(x)$. Now, $y \in \mathcal{G}(A^{(n)})$ implies that y is a $\sqsubseteq$-minimal nonzero element of $\mathcal{L}(A^{(n)})$ and hence x is a $\sqsubseteq$-minimal nonzero element of $\mathcal{L}(A^{(c)})$. Therefore $x \in \mathcal{G}(A^{(c)})$ and $y \in \phi_{k_1,\ldots,k_g}(\mathcal{G}(A^{(c)}))$. So the left-hand side of (2) is contained in the right-hand side.

Since A is fixed we have that $c = c(A)$ and $\mathcal{G}(A^{(c)})$ are constant. Then Eq. (2) implies that $|\mathcal{G}(A^{(n)})| \leq \binom{n}{c}|\mathcal{G}(A^{(c)})| = O(n^c)$. Moreover, every element of $\mathcal{G}(A^{(n)})$ is an nt-dimensional vector $\phi_{k_1,\ldots,k_c}(x)$ obtained by appending zero components to some $x \in \mathcal{G}(A^{(c)})$ and hence has linear binary length $O(n)$. So the binary length of the entire Graver basis $\mathcal{G}(A^{(n)})$ is $O(n^{c+1})$. Thus, the $\binom{n}{c} = O(n^c)$ images $\phi_{k_1,\ldots,k_c}(\mathcal{G}(A^{(c)}))$ and their union $\mathcal{G}(A^{(n)})$ can be computed in strongly polynomial time, as claimed. □

Example 4.8. Consider the $(2 + 1) \times 2$ matrix A with $A_1 := I_2$ the 2×2 identity and $A_2 := (1, 1)$. Then $\mathcal{G}(A_2) = \pm(1, -1)$ and $\mathcal{G}(A_1G_2) = \pm(1, 1)$ from which the Graver complexity of A can be concluded to be $c(A) = 2$ (see the proof of Lemma 4.6). The 2-fold matrix of A and its Graver basis, consisting of two antipodal vectors only, are

$$A^{(2)} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \qquad \mathcal{G}(A^{(2)}) = \pm \begin{pmatrix} 1 & -1 & -1 & 1 \end{pmatrix}.$$

By Theorem 4.7, the Graver basis of the 4-fold matrix $A^{(4)}$ is computed to be the union of the images of the $6 = \binom{4}{2}$ maps $\phi_{k_1,k_2} : \mathbb{Z}^{2\cdot 2} \to \mathbb{Z}^{4\cdot 2}$ for $1 \leq k_1 < k_2 \leq 4$, getting

$$A^{(4)} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}, \qquad \mathcal{G}(A^{(4)}) = \pm \begin{pmatrix} 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \end{pmatrix}.$$

4.4. Linear n-fold Integer Programming in Polynomial Time

We now proceed to provide a polynomial time algorithm for linear integer programming over n-fold matrices. First, combining the results of §4.2 and §4.3, we get at once the following polynomial time algorithm for converting any feasible point to an optimal one.

Lemma 4.9. *For every fixed $(r + s) \times t$ integer matrix A there is a polynomial time algorithm that, given $n \in \mathbb{N}$, $l, u \in \mathbb{Z}^{nt}_{\infty}$, $x, w \in \mathbb{Z}^{nt}$ satisfying $l \leq x \leq u$, encoded as $[\langle l, u, x, w\rangle]$, solves the linear n-fold integer programming problem with $b := A^{(n)}x$,*

$$\max\{wz : z \in \mathbb{Z}^{nt}, A^{(n)}z = b, l \leq z \leq u\}.$$

Proof. First, apply the polynomial time algorithm of Theorem 4.7 and compute the Graver basis $\mathcal{G}(A^{(n)})$ of the n-fold matrix $A^{(n)}$. Then apply the polynomial time algorithm of Theorem 4.4 to the data $A^{(n)}, \mathcal{G}(A^{(n)}), l, u, x$ and w. □

Next we show that an initial feasible point can also be found in polynomial time.

Lemma 4.10. *For every fixed $(r+s)\times t$ integer matrix A there is a polynomial time algorithm that, given $n\in\mathbb{N}$, $l,u\in\mathbb{Z}^{nt}_{\infty}$, and $b\in\mathbb{Z}^{r+ns}$, encoded as $[\langle l,u,b\rangle]$, either finds an $x\in\mathbb{Z}^{nt}$ satisfying $l\leq x\leq u$ and $A^{(n)}x=b$ or asserts that none exists.*

Proof. If $l\neq u$ then assert that there is no feasible point and terminate the algorithm. Assume then that $l\leq u$ and determine some $x\in\mathbb{Z}^{nt}$ with $l\leq x\leq u$ and $\langle x\rangle\leq\langle l,u\rangle$. Now, introduce $n(2r+2s)$ auxiliary variables to the given n-fold integer program and denote by $\hat{x}$ the resulting vector of $n(t+2r+2s)$ variables. Suitably extend the lower and upper bound vectors to $\hat{l},\hat{u}$ by setting $\hat{l}_j:=0$ and $\hat{u}_j:=\infty$ for each auxiliary variable $\hat{x}_j$. Consider the auxiliary integer program of finding an integer vector $\hat{x}$ that minimizes the sum of auxiliary variables subject to the lower and upper bounds $\hat{l}\leq\hat{x}\leq\hat{u}$ and the following system of equations, with I_r and I_s the $r\times r$ and $s\times s$ identity matrices,

$$\begin{pmatrix} A_1 & I_r & -I_r & 0 & 0 & A_1 & I_r & -I_r & 0 & 0 & \cdots & A_1 & I_r & -I_r & 0 & 0\\ A_2 & 0 & 0 & I_s & -I_s & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & A_2 & 0 & 0 & I_s & -I_s & \cdots & 0 & 0 & 0 & 0 & 0\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & A_2 & 0 & 0 & I_s & -I_s \end{pmatrix}\hat{x}=b.$$

This is again an n-fold integer program, with an $(r+s)\times(t+2r+2s)$ matrix $\hat{A}$, where $\hat{A}_1=(A_1,I_r,-I_r,0,0)$ and $\hat{A}_2=(A_2,0,0,I_s,-I_s)$. Since A is fixed, so is $\hat{A}$. It is now easy to extend the vector $x\in\mathbb{Z}^{nt}$ determined above to a feasible point $\hat{x}$ of the auxiliary program. Indeed, put $\hat{b}:=b-A^{(n)}x\in\mathbb{Z}^{r+ns}$; now, for $i=1,\ldots,r+ns$, simply choose an auxiliary variable $\hat{x}_j$ appearing only in the ith equation, whose coefficient equals the sign $\mathrm{sign}(\hat{b}_i)$ of the corresponding entry of $\hat{b}$, and set $\hat{x}_j:=|\hat{b}_i|$. Define $\widehat{w}\in\mathbb{Z}^{n(t+2r+2s)}$ by setting $\widehat{w}:=0$ for each original variable and $\widehat{w}:=-1$ for each auxiliary variable, so that maximizing $\widehat{w}\hat{x}$ is equivalent to minimizing the sum of auxiliary variables. Now solve the auxiliary linear integer program in polynomial time by applying the algorithm of Lemma 4.9 corresponding to $\hat{A}$ to the data $n,\hat{l},\hat{u},\hat{x}$, and $\widehat{w}$. Since the auxiliary objective $\widehat{w}\hat{x}$ is bounded above by zero, the algorithm will output an optimal solution $\hat{x}^*$. If the optimal objective value is negative, then the original n-fold program is infeasible, whereas if the optimal value is zero, then the restriction of $\hat{x}^*$ to the original variables is a feasible point x^* of the original integer program. □

Combining Lemmas 4.9 and 4.10 we get at once the main result of this section.

Theorem 4.11. *For every fixed $(r+s)\times t$ integer matrix A there is a polynomial time algorithm that, given n, lower and upper bounds $l,u\in\mathbb{Z}^{nt}_{\infty}$, $w\in\mathbb{Z}^{nt}$, and $b\in\mathbb{Z}^{r+ns}$, encoded as $[\langle l,u,w,b\rangle]$, solves the following linear n-fold integer programming problem,*

$$\max\{wx : x\in\mathbb{Z}^{nt}, A^{(n)}x=b, l\leq x\leq u\}.$$

Again, as a special case of Theorem 4.11 we recover the following result of [9] concerning linear integer programming in standard form over n-fold matrices.

Theorem 4.12. *For every fixed $(r+s)\times t$ integer matrix A there is a polynomial time algorithm that, given n, linear functional $w\in\mathbb{Z}^{nt}$, and right-hand side $b\in\mathbb{Z}^{r+ns}$, encoded as $[\langle w,b\rangle]$, solves the following linear n-fold integer program in standard form,*

$$\max\{wx : x\in\mathbb{N}^{nt}, A^{(n)}x=b\}.$$

4.5. Some Applications

4.5.1. Three-Way Line-Sum Transportation Problems

Transportation problems form a very important class of discrete optimization problems studied extensively in the operations research and mathematical programming literature, see, e.g., [22, 26–30] and the references therein. We will discuss this class of problem and its applications to secure statistical data disclosure in more detail in §6.

It is well known that 2-way transportation problems are polynomial time solvable, since they can be encoded as linear integer programs over totally unimodular systems. However, already 3-way transportation problem are much more complicated. Consider the following 3-way transportation problem over $p\times q\times n$ tables with all line-sums fixed,

$$\max\Big\{wx : x\in\mathbb{N}^{p\times q\times n}, \sum_i x_{i,j,k}=z_{j,k}, \sum_j x_{i,j,k}=v_{i,k}, \sum_k x_{i,j,k}=u_{i,j}\Big\}.$$

The data for the problem consist of given integer numbers (lines-sums) $u_{i,j}$, $v_{i,k}$, $z_{j,k}$ for $i=1,\ldots,p$, $j=1,\ldots,q$, $k=1,\ldots,n$, and a linear functional given by a $p\times q\times n$ integer array w representing the transportation profit per unit on each cell. The problem is to find a transportation, that is, a $p\times q\times n$ nonnegative integer table x satisfying the line sum constraints, which attains maximum profit $wx=\sum_{i=1}^{p}\sum_{j=1}^{q}\sum_{k=1}^{n}w_{i,j,k}x_{i,j,k}$.

When at least two of the table sides, say p,q, are variable part of the input, and even when the third side is fixed and as small as $n=3$, this problem is already *universal* for integer programming in a very strong sense [6, 8], and in particular is NP-hard [5]; this will be discussed in detail and proved in §6. We now show that in contrast, when two sides, say p,q, are fixed (but arbitrary), and one side n is variable, then the 3-way transportation problem over such *long* tables is an n-fold integer programming problem and therefore, as a consequence of Theorem 4.12, can be solved is polynomial time.

Corollary 4.13. *For every fixed p and q there is a polynomial time algorithm that, given n, integer profit array $w\in\mathbb{Z}^{p\times q\times n}$, and line-sums $u\in\mathbb{Z}^{p\times q}$, $v\in\mathbb{Z}^{p\times n}$ and $z\in\mathbb{Z}^{q\times n}$, encoded as $[\langle w,u,v,z\rangle]$, solves the integer 3-way line-sum transportation problem*

$$\max\Big\{wx : x\in\mathbb{N}^{p\times q\times n}, \sum_i x_{i,j,k}=z_{j,k}, \sum_j x_{i,j,k}=v_{i,k}, \sum_k x_{i,j,k}=u_{i,j}\Big\}.$$

Proof. Re-index $p\times q\times n$ arrays as $x=(x^1,\ldots,x^n)$ with each component indexed as $x^k:=(x^k_{i,j}):=(x_{1,1,k},\ldots,x_{p,q,k})$ suitably indexed as a pq vector representing the kth layer of x. Put $r:=t:=pq$ and $s:=p+q$, and let A be the $(r+s)\times t$ matrix with $A_1:=I_{pq}$ the $pq\times pq$ identity and with A_2 the $(p+q)\times pq$ matrix of equations of the usual 2-way transportation problem for $p\times q$ arrays. Re-arrange the given line-sums in a vector $b:=(b^0,b^1,\ldots,b^n)\in\mathbb{Z}^{r+ns}$ with $b^0:=(u_{i,j})$ and $b^k:=((v_{i,k}),(z_{j,k}))$ for $k=1,\ldots,n$.

This translates the given 3-way transportation problem into an n-fold integer programming problem in standard form,

$$\max\{wx : x \in \mathbb{N}^{nt}, A^{(n)}x = b\},$$

where the equations $A_1(\sum_{k=1}^{n} x^k) = b^0$ represent the constraints $\sum_k x_{i,j,k} = u_{i,j}$ of all line-sums where summation over layers occurs, and the equations $A_2 x^k = b^k$ for $k = 1, \ldots, n$ represent the constraints $\sum_i x_{i,j,k} = z_{j,k}$ and $\sum_j x_{i,j,k} = v_{i,k}$ of all line-sums where summations are within a single layer at a time.

Using the algorithm of Theorem 4.12, this n-fold integer program, and hence the given 3-way transportation problem, can be solved in polynomial time. □

Example 4.14. We demonstrate the encoding of the $p \times q \times n$ transportation problem as an n-fold integer program as in the proof of Corollary 4.13 for $p = q = 3$ (smallest case where the problem is genuinely 3-dimensional). Here we put $r := t := 9$, $s := 6$, write

$$x^k := (x_{1,1,k}, x_{1,2,k}, x_{1,3,k}, x_{2,1,k}, x_{2,2,k}, x_{2,3,k}, x_{3,1,k}, x_{3,2,k}, x_{3,3,k}), \qquad k = 1, \ldots, n,$$

and let the $(9 + 6) \times 9$ matrix A consist of $A_1 = I_9$ the 9×9 identity matrix and

$$A_2 := \begin{pmatrix} 1&1&1&0&0&0&0&0&0 \\ 0&0&0&1&1&1&0&0&0 \\ 0&0&0&0&0&0&1&1&1 \\ 1&0&0&1&0&0&1&0&0 \\ 0&1&0&0&1&0&0&1&0 \\ 0&0&1&0&0&1&0&0&1 \end{pmatrix}.$$

Then the corresponding n-fold integer program encodes the $3 \times 3 \times n$ transportation problem as desired. Already for this case, of $3 \times 3 \times n$ tables, the only known polynomial time algorithm for the transportation problem is the one underlying Corollary 4.13.

Corollary 4.13 has a very broad generalization to multiway transportation problems over long k-way tables of any dimension k; this will be discussed in detail in §6.

4.5.2. Packing Problems and Cutting-Stock

We consider the following rather general class of packing problems which concern maximum utility packing of many items of several types in various bins subject to weight constraints. More precisely, the data is as follows. There are t types of items. Each item of type j has integer weight v_j. There are n_j items of type j to be packed. There are n bins. The weight capacity of bin k is an integer u_k. Finally, there is a utility matrix $w \in \mathbb{Z}^{t \times n}$ where $w_{j,k}$ is the utility of packing one item of type j in bin k. The problem is to find a feasible packing of maximum total utility. By incrementing the number t of types by 1 and suitably augmenting the data, we may assume that the last type t represents "slack items" which occupy the unused capacity in each bin, where the weight of each slack item is 1, the utility of packing any slack item in any bin is 0, and the number of slack items is the total residual weight capacity $n_t := \sum_{k=1}^{n} u_k - \sum_{j=1}^{t-1} n_j v_j$. Let $x \in \mathbb{N}^{t \times n}$ be a variable matrix where $x_{j,k}$ represents the number of items of type j to be packed in bin k. Then the packing problem becomes the following linear integer program,

$$\max\Big\{wx : x \in \mathbb{N}^{t\times n}, \sum_j v_j x_{j,k} = u_k, \sum_k x_{j,k} = n_j\Big\}.$$

We now show that this is in fact an n-fold integer programming problem and therefore, as a consequence of Theorem 4.12, can be solved is polynomial time. While the number t of types and type weights v_j are fixed, which is natural in many bin packing applications, the numbers n_j of items of each type and the bin capacities u_k may be very large.

Corollary 4.15. *For every fixed number t of types and integer type weights $v_1, \ldots, v_t$, there is a polynomial time algorithm that, given n bins, integer item numbers $n_1, \ldots, n_t$, integer bin capacities $u_1, \ldots, u_n$, and $t \times n$ integer utility matrix w, encoded as $[\langle n_1, \ldots, n_t, u_1, \ldots, u_n, w\rangle]$, solves the following integer bin packing problem,*

$$\max\Big\{wx : x \in \mathbb{N}^{t\times n}, \sum_j v_j x_{j,k} = u_k, \sum_k x_{j,k} = n_j\Big\}.$$

Proof. Re-index the variable matrix as $x = (x^1, \ldots, x^n)$ with $x^k := (x^k_1, \ldots, x^k_t)$ where x^k_j represents the number of items of type j to be packed in bin k for all j and k. Let A be the $(t+1)\times t$ matrix with $A_1 := I_t$ the $t\times t$ identity and with $A_2 := (v_1, \ldots, v_t)$ a single row. Re-arrange the given item numbers and bin capacities in a vector $b := (b^0, b^1, \ldots, b^n) \in \mathbb{Z}^{t+n}$ with $b^0 := (n_1, \ldots, n_t)$ and $b^k := u_k$ for all k. This translates the bin packing problem into an n-fold integer programming problem in standard form,

$$\max\{wx : x \in \mathbb{N}^{nt}, A^{(n)}x = b\},$$

where the equations $A_1(\sum_{k=1}^n x^k) = b^0$ represent the constraints $\sum_k x_{j,k} = n_j$ assuring that all items of each type are packed, and the equations $A_2x^k = b^k$ for $k = 1, \ldots, n$ represent the constraints $\sum_j v_j x_{j,k} = u_k$ assuring that the weight capacity of each bin is not exceeded (in fact, the slack items make sure each bin is perfectly packed).

Using the algorithm of Theorem 4.12, this n-fold integer program, and hence the given integer bin packing problem, can be solved in polynomial time. □

Example 4.16 (Cutting-Stock Problem)**.** This is a classical manufacturing problem [18], where the usual setup is as follows: a manufacturer produces rolls of material (such as scotch-tape or band-aid) in one of t different widths $v_1, \ldots, v_t$. The rolls are cut out from standard rolls of common large width u. Given orders by customers for n_j rolls of width v_j, the problem facing the manufacturer is to meet the orders using the smallest possible number of standard rolls. This can be cast as a bin packing problem as follows. Rolls of width v_j become items of type j to be packed. Standard rolls become identical bins, of capacity $u_k := u$ each, where the number of bins is set to be $n := \sum_{j=1}^t \lceil n_j/\lfloor u/v_j\rfloor\rceil$ which is sufficient to accommodate all orders. The utility of each roll of width v_j is set to be its width negated $w_{j,k} := -v_j$ regardless of the standard roll k from which it is cut (paying for the width it takes). Introduce a new roll width $v_0 := 1$, where rolls of that width represent "slack rolls" which occupy the unused width of each standard roll, with utility $w_{0,k} := -1$ regardless of the standard roll k from which it is cut (paying for the unused width it represents), with the number of slack rolls set to be the total residual width $n_0 := nu - \sum_{j=1}^t n_j v_j$. Then the cutting-stock problem becomes a bin packing problem

and therefore, by Corollary 4.15, for every fixed t and fixed roll widths $v_1, \ldots, v_t$, it is solvable in time polynomial in $\sum_{j=1}^{t} \lceil n_j / \lfloor u/v_j \rfloor \rceil$ and $\langle n_1, \ldots, n_t, u \rangle$.

One common approach to the cutting-stock problem uses so-called *cutting patterns*, which are feasible solutions of the knapsack problem $\{y \in \mathbb{N}^t : \sum_{j=1}^{t} v_j y_j \leq u\}$. This is useful when the common width u of the standard rolls is of the same order of magnitude as the demand role widths v_j. However, when u is much larger than the v_j, the number of cutting patterns becomes prohibitively large to handle. But then the values $\lfloor u/v_j \rfloor$ are large and hence $n := \sum_{j=1}^{t} \lceil n_j / \lfloor u/v_j \rfloor \rceil$ is small, in which case the solution through the algorithm of Corollary 4.15 becomes particularly appealing.

5. Convex Integer Programming

In this section we discuss convex integer programming. In particular, we extend the theory of §4 and show that convex n-fold integer programming is polynomial time solvable as well. In §5.1 we discuss convex integer programming over totally unimodular matrices. In §5.2 we show the applicability of Graver bases to convex integer programming. In §5.3 we combine Theorem 2.4, the results of §4, and the preparatory facts from §§5.2, and prove the main result of this section, asserting that convex n-fold integer programming is polynomial time solvable. We conclude with some applications in §5.4.

As in §4, the feasible set S is presented as the set of integer points satisfying an explicitly given system of linear inequalities, given in one of the forms

$$S := \{x \in \mathbb{N}^n : Ax = b\} \quad \text{or} \quad S := \{x \in \mathbb{Z}^n : Ax = b, l \leq x \leq u\},$$

with matrix $A \in \mathbb{Z}^{m \times n}$, right-hand side $b \in \mathbb{Z}^m$, and lower and upper bounds $l, u \in \mathbb{Z}^n_\infty$.

As demonstrated in §1.1, if the polyhedron $P := \{x \in \mathbb{R}^n : Ax = b, l \leq x \leq u\}$ is unbounded then the convex integer programming problem with an oracle presented convex functional is rather hopeless. Therefore, an algorithm that solves the convex integer programming problem should either return an optimal solution, or assert that the program is infeasible, or assert that the underlying polyhedron is unbounded.

Nonetheless, we do allow the lower and upper bounds l, u to lie in $\mathbb{Z}^n_\infty$ rather than $\mathbb{Z}^n$, since often the polyhedron is bounded even though the variables are not bounded explicitly (for instance, if each variable is bounded below only, and appears in some equation all of whose coefficients are positive). This results in broader formulation flexibility. Furthermore, in the next subsections we prove auxiliary lemmas asserting that certain sets cover all edge-directions of relevant polyhedra, which do hold also in the unbounded case. So we now extend the notion of edge-directions, defined in §2.1 for polytopes, to polyhedra. A *direction* of an edge (1-dimensional face) e of a polyhedron P is any nonzero scalar multiple of $y - x$ where x, y are any two distinct points in e. As before, a set *covers all edge-directions of P* if it contains a direction of each edge of P.

5.1. Convex Integer Programming over Totally Unimodular Systems

A matrix A is *totally unimodular* if the determinant of every square submatrix of A lies in $\{-1, 0, 1\}$. Such matrices arise naturally in network flows, ordinary (2-way) transportation problems, and many other situations. A fundamental result in integer programming [65]

asserts that polyhedra defined by totally unimodular matrices are integer. More precisely, if A is an $m \times n$ totally unimodular matrix, $l, u \in \mathbb{Z}^n_\infty$, and $b \in \mathbb{Z}^m$, then

$$P_I := \text{conv}\{x \in \mathbb{Z}^n : Ax = b, l \le x \le u\} = \{x \in \mathbb{R}^n : Ax = b, l \le x \le u\} := P,$$

that is, the underlying polyhedron P coincides with its integer hull P_I. This has two consequences useful in facilitating the solution of the corresponding convex integer programming problem via the algorithm of Theorem 2.4. First, the corresponding linear integer programming problem can be solved by linear programming over P in polynomial time. Second, a set covering all edge-directions of the implicitly given integer hull P_I, which is typically very hard to determine, is obtained here as a set covering all edge-directions of P which is explicitly given and hence easier to determine.

We now describe a well known property of polyhedra of the above form. A *circuit* of a matrix $A \in \mathbb{Z}^{m \times n}$ is a nonzero primitive minimal support element of $\mathcal{L}(A)$. So a circuit is a nonzero $c \in \mathbb{Z}^n$ satisfying $Ac = 0$, whose entries are relatively prime integers, such that no nonzero c' with $Ac' = 0$ has support strictly contained in the support of c.

Lemma 5.1. *For every $A \in \mathbb{Z}^{m \times n}$, $l, u \in \mathbb{Z}^n_\infty$, and $b \in \mathbb{Z}^m$, the set of circuits of A covers all edge-directions of the polyhedron $P := \{x \in \mathbb{R}^n : Ax = b, l \le x \le u\}$.*

Proof. Consider any edge e of P. Pick two distinct points $x, y \in e$ and set $g := y - x$. Then $Ag = 0$ and therefore, as can be easily proved by induction on $|\text{supp}(g)|$, there is a finite decomposition $g = \sum_i \alpha_i c_i$ with α_i positive real number and c_i circuit of A such that $\alpha_i c_i \sqsubseteq g$ for all i, where $\sqsubseteq$ is the natural extension from $\mathbb{Z}^n$ to $\mathbb{R}^n$ of the partial order defined in §4.2. We claim that $x + \alpha_i c_i \in P$ for all i. Indeed, c_i being a circuit implies $A(x + \alpha_i c_i) = Ax = b$; and $l \le x, x + g \le u$ and $\alpha_i c_i \sqsubseteq g$ imply $l \le x + \alpha_i c_i \le u$.

Now let $w \in \mathbb{R}^n$ be a linear functional uniquely maximized over P at the edge e. Then $w\alpha_i c_i = w(x + \alpha_i c_i) - wx \le 0$ for all i. But $\sum(w\alpha_i c_i) = wg = wy - wx = 0$, implying that in fact $w\alpha_i c_i = 0$ and hence $x + \alpha_i c_i \in e$ for all i. This implies that each c_i is a direction of e (in fact, all c_i are the same and g is a multiple of some circuit). □

Combining Theorem 2.4 and Lemma 5.1 we obtain the following statement.

Theorem 5.2. *For every fixed d there is a polynomial time algorithm that, given $m \times n$ totally unimodular matrix A, set $C \subset \mathbb{Z}^n$ containing all circuits of A, vectors $l, u \in \mathbb{Z}^n_\infty$, $b \in \mathbb{Z}^m$, and $w_1, \ldots, w_d \in \mathbb{Z}^n$, and convex $c\colon \mathbb{R}^d \to \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle A, C, l, u, b, w_1, \ldots, w_d\rangle]$, solves the convex integer program*

$$\max\{c(w_1 x, \ldots, w_d x) : x \in \mathbb{Z}^n, Ax = b, l \le x \le u\}.$$

Proof. First, check in polynomial time using linear programming whether the objective function of any of the following $2n$ linear programs is unbounded,

$$\max\{\pm y_i : y \in P\}, \qquad i = 1, \ldots, n, \quad P := \{y \in \mathbb{R}^n : Ay = b, l \le y \le u\}.$$

If any is unbounded then terminate, asserting that P is unbounded. Otherwise, let ρ be the least integer upper bound on the absolute value of all optimal objective values. Then $P \subseteq [-\rho, \rho]^n$ and $S := \{y \in \mathbb{Z}^n : Ay = b, l \le y \le u\} \subset P$ is finite of radius $\rho(S) \le \rho$. In

fact, since A is totally unimodular, $P_I = P = \mathrm{conv}(S)$ and hence $\rho(S) = \rho$. Moreover, by Cramer's rule, $\langle \rho \rangle$ is polynomially bounded in $\langle A, l, u, x \rangle$.

Now, since A is totally unimodular, using linear programming over $P_I = P$ we can simulate in polynomial time a linear discrete optimization oracle for S. By Lemma 5.1, the given set C, which contains all circuits of A, also covers all edge-directions of $\mathrm{conv}(S) = P_I = P$. Therefore we can apply the algorithm of Theorem 2.4 and solve the given convex n-fold integer programming problem in polynomial time. □

While the number of circuits of an $m \times n$ matrix A can be as large as $2\binom{n}{m+1}$ and hence exponential in general, it is nonetheless relatively small in that it is bounded in terms of m and n only and is independent of the matrix A itself. Furthermore, it may happen that the number of circuits is much smaller than the upper bound $2\binom{n}{m+1}$. Also, if in a class of matrices, m grows slowly in terms of n, say $m = O(\log n)$, then this bound is subexponential. In such situations, the above theorem may provide a good strategy for solving convex integer programming over totally unimodular systems.

5.2. Graver Bases and Convex Integer Programming

We now extend the statements of §5.1 about totally unimodular matrices to arbitrary integer matrices. The next lemma shows that the Graver basis of any integer matrix covers all edge-directions of the integer hulls of polyhedra defined by that matrix.

Lemma 5.3. *For every $A \in \mathbb{Z}^{m \times n}$, $l, u \in \mathbb{Z}^n_\infty$, and $b \in \mathbb{Z}^m$, the Graver basis $\mathcal{G}(A)$ of A covers all edge-directions of the polyhedron $P_I := \mathrm{conv}\{x \in \mathbb{Z}^n : Ax = b, l \le x \le u\}$.*

Proof. Consider any edge e of P_I and pick two distinct points $x, y \in e \cap \mathbb{Z}^n$. Then $g := y - x$ is in $\mathcal{L}(A) \setminus \{0\}$. Therefore, by Lemma 4.2, there is a conformal sum $g = \sum_i h_i$ with $h_i \in \mathcal{G}(A)$ for all i. We claim that $x + h_i \in P_I$ for all i. Indeed, first note that $h_i \in \mathcal{G}(A) \subset \mathcal{L}(A)$ implies $Ah_i = 0$ and hence $A(x + h_i) = Ax = b$; and second note that $l \le x, x + g \le u$ and $h_i \sqsubseteq g$ imply that $l \le x + h_i \le u$.

Now let $w \in \mathbb{Z}^n$ be a linear functional uniquely maximized over P_I at the edge e. Then $wh_i = w(x + h_i) - wx \le 0$ for all i. But $\sum(wh_i) = wg = wy - wx = 0$, implying that in fact $wh_i = 0$ and hence $x + h_i \in e$ for all i. Therefore each h_i is a direction of e (in fact, all h_i are the same and g is a multiple of some Graver basis element). □

Combining Theorems 2.4 and 4.4 and Lemma 5.3 we obtain the following statement.

Theorem 5.4. *For every fixed d there is a polynomial time algorithm that, given integer $m \times n$ matrix A, its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}^n_\infty$, $x \in \mathbb{Z}^n$ with $l \le x \le u$, $w_1, \ldots, w_d \in \mathbb{Z}^n$, and convex $c\colon \mathbb{R}^d \to \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle A, \mathcal{G}(A), l, u, x, w_1, \ldots, w_d \rangle]$, solves the convex integer program with $b := Ax$,*

$$\max\{c(w_1 z, \ldots, w_d z) : z \in \mathbb{Z}^n, Az = b, l \le z \le u\}.$$

Proof. First, check in polynomial time using linear programming whether the objective function of any of the following $2n$ linear programs is unbounded,

$$\max\{\pm y_i : y \in P\}, \qquad i = 1, \ldots, n, \quad P := \{y \in \mathbb{R}^n : Ay = b, l \le y \le u\}.$$

If any is unbounded then terminate, asserting that P is unbounded. Otherwise, let ρ be the least integer upper bound on the absolute value of all optimal objective values. Then $P \subseteq [-\rho,\rho]^n$ and $S := \{y \in \mathbb{Z}^n : Ay = b, l \leq y \leq u\} \subset P$ is finite of radius $\rho(S) \leq \rho$. Moreover, by Cramer's rule, $\langle\rho\rangle$ is polynomially bounded in $\langle A, l, u, x\rangle$.

Using the given Graver basis and applying the algorithm of Theorem 4.4 we can simulate in polynomial time a linear discrete optimization oracle for S. Furthermore, by Lemma 5.3, the given Graver basis covers all edge-directions of the integer hull $P_I := \text{conv}\{y \in \mathbb{Z}^n : Ay = b, l \leq y \leq u\} = \text{conv}(S)$. Therefore we can apply the algorithm of Theorem 2.4 and solve the given convex program in polynomial time. □

5.3. Convex n-fold Integer Programming in Polynomial Time

We now extend the result of Theorem 4.11 and show that convex integer programming problems over n-fold systems can be solved in polynomial time as well. As explained in the beginning of this section, the algorithm either returns an optimal solution, or asserts that the program is infeasible, or asserts that the underlying polyhedron is unbounded.

Theorem 5.5. *For every fixed d and fixed $(r+s)\times t$ integer matrix A there is a polynomial time algorithm that, given n, lower and upper bounds $l, u \in \mathbb{Z}^{nt}_{\infty}$, $w_1, \ldots, w_d \in \mathbb{Z}^{nt}$, $b \in \mathbb{Z}^{r+ns}$, and convex functional $c\colon \mathbb{R}^d \to \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle l, u, w_1, \ldots, w_d, b\rangle]$, solves the convex n-fold integer programming problem*

$$\max\{c(w_1 x, \ldots, w_d x) : x \in \mathbb{Z}^{nt}, A^{(n)} x = b, l \leq x \leq u\}.$$

Proof. First, check in polynomial time using linear programming whether the objective function of any of the following $2nt$ linear programs is unbounded,

$$\max\{\pm y_i : y \in P\}, \qquad i = 1, \ldots, nt, \quad P := \{y \in \mathbb{R}^{nt} : A^{(n)} y = b, l \leq y \leq u\}.$$

If any is unbounded then terminate, asserting that P is unbounded. Otherwise, let ρ be the least integer upper bound on the absolute value of all optimal objective values. Then $P \subseteq [-\rho,\rho]^{nt}$ and $S := \{y \in \mathbb{Z}^{nt} : A^{(n)} y = b, l \leq y \leq u\} \subset P$ is finite of radius $\rho(S) \leq \rho$. Moreover, by Cramer's rule, $\langle\rho\rangle$ is polynomially bounded in n and $\langle l, u, b\rangle$.

Using the algorithm of Theorem 4.11 we can simulate in polynomial time a linear discrete optimization oracle for S. Also, using the algorithm of Theorem 4.7 we can compute in polynomial time the Graver basis $\mathcal{G}(A^{(n)})$ which, by Lemma 5.3, covers all edge-directions of $P_I := \text{conv}\{y \in \mathbb{Z}^{nt} : A^{(n)} y = b, l \leq y \leq u\} = \text{conv}(S)$. Therefore we can apply the algorithm of Theorem 2.4 and solve the given convex n-fold integer programming problem in polynomial time. □

Again, as a special case of Theorem 5.5 we recover the following result of [10] concerning convex integer programming in standard form over n-fold matrices.

Theorem 5.6. *For every fixed d and fixed $(r+s)\times t$ integer matrix A there is a polynomial time algorithm that, given n, linear functionals $w_1, \ldots, w_d \in \mathbb{Z}^{nt}$, right-hand side $b \in \mathbb{Z}^{r+ns}$, and convex functional $c\colon \mathbb{R}^d \to \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle w_1, \ldots, w_d, b\rangle]$, solves the convex n-fold integer program in standard form*

$$\max\{c(w_1 x, \ldots, w_d x) : x \in \mathbb{N}^{nt}, A^{(n)} x = b\}.$$

5.4. Some Applications

5.4.1. Transportation Problems and Packing Problems

Theorems 5.5 and 5.6 generalize Theorems 4.11 and 4.12 by broadly extending the class of objective functions that can be maximized in polynomial time over n-fold systems. Therefore all applications discussed in §4.5 automatically extend accordingly.

First, we have the following analog of Corollary 4.13 for the *convex integer transportation problem* over long 3-way tables. This has a very broad further generalization to multiway transportation problems over long k-way tables of any dimension k, see §6.

Corollary 5.7. *For every fixed d, p, q there is a polynomial time algorithm that, given n, arrays $w_1, \ldots, w_d \in \mathbb{Z}^{p\times q\times n}$, line-sums $u \in \mathbb{Z}^{p\times q}$, $v \in \mathbb{Z}^{p\times n}$ and $z \in \mathbb{Z}^{q\times n}$, and convex functional $c\colon \mathbb{R}^d \to \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle w_1, \ldots, w_d, u, v, z\rangle]$, solves the convex integer 3-way line-sum transportation problem*

$$\max\Big\{c(w_1x, \ldots, w_dx) : x \in \mathbb{N}^{p\times q\times n}, \sum_i x_{i,j,k} = z_{j,k}\,, \sum_j x_{i,j,k} = v_{i,k}, \sum_k x_{i,j,k} = u_{i,j}\Big\}.$$

Second, we have the following analog of Corollary 4.15 for convex bin packing.

Corollary 5.8. *For every fixed d, number of types t, and type weights $v_1, \ldots, v_t \in \mathbb{Z}$, there is a polynomial time algorithm that, given n bins, item numbers $n_1, \ldots, n_t \in \mathbb{Z}$, bin capacities $u_1, \ldots, u_n \in \mathbb{Z}$, utility matrices $w_1, \ldots, w_d \in \mathbb{Z}^{t\times n}$, and convex functional $c\colon \mathbb{R}^d \to \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle n_1, \ldots, n_t, u_1, \ldots, u_n, w_1, \ldots, w_d\rangle]$, solves the convex integer bin packing problem,*

$$\max\Big\{c(w_1x, \ldots, w_dx) : x \in \mathbb{N}^{t\times n}, \sum_j v_j x_{j,k} = u_k, \sum_k x_{j,k} = n_j\Big\}.$$

5.4.2. Vector Partitioning and Clustering

The vector partition problem concerns the partitioning of n items among p players to maximize social value subject to constraints on the number of items each player can receive. More precisely, the data is as follows. With each item i is associated a vector $v_i \in \mathbb{Z}^k$ representing its utility under k criteria. The utility of player h under ordered partition $\pi = (\pi_1, \ldots, \pi_p)$ of the set of items $\{1, \ldots, n\}$ is the sum $v_h^\pi := \sum_{i\in\pi_h} v_i$ of utility vectors of items assigned to h under π. The social value of π is the balancing $c(v_{1,1}^\pi, \ldots, v_{1,k}^\pi, \ldots, v_{p,1}^\pi, \ldots, v_{p,k}^\pi)$ of the player utilities, where c is a convex functional on $\mathbb{R}^{pk}$. In the constrained version, the partition must be of a given *shape*, i.e. the number $|\pi_h|$ of items that player h gets is required to be a given number λ_h (with $\sum \lambda_h = n$). In the unconstrained version, there is no restriction on the number of items per player.

Vector partition problems have applications in diverse areas such as load balancing, circuit layout, ranking, cluster analysis, inventory, and reliability, see, e.g., [11, 12, 16, 19, 20] and the references therein. Here is a typical example.

Example 5.9 (Minimal Variance Clustering)**.** This problem has numerous applications in the analysis of statistical data: given n observed points $v_1, \ldots, v_n$ in k-space, group them into p clusters $\pi_1, \ldots, \pi_p$ that minimize the sum of cluster variances given by

$$\sum_{h=1}^{p} \frac{1}{|\pi_h|} \sum_{i \in \pi_h} \left\| v_i - \left(\frac{1}{|\pi_h|} \sum_{i \in \pi_h} v_i \right) \right\|^2 .$$

Consider instances where there are $n = pm$ points and the desired clustering is balanced, that is, the clusters should have equal size m. Suitable manipulation of the sum of variances expression above shows that the problem is equivalent to a constrained vector partition problem, where $\lambda_h = m$ for all h, and where the convex functional $c\colon \mathbb{R}^{pk} \to \mathbb{R}$ (to be maximized) is the Euclidean norm squared, given by

$$c(z) = \|z\|^2 = \sum_{h=1}^{p} \sum_{i=1}^{k} |z_{h,i}|^2 .$$

If either the number of criteria k or the number of players p is variable, the partition problem is intractable since it instantly captures NP-hard problems [12]. When both k, p are fixed, both the constrained and unconstrained versions of the vector partition problem are polynomial time solvable [12,16]. We now show that vector partition problems (either constrained or unconstrained) are in fact convex n-fold integer programming problems and therefore, as a consequence of Theorem 5.6, can be solved is polynomial time.

Corollary 5.10. *For every fixed number p of players and number k of criteria, there is a polynomial time algorithm that, given n, item vectors $v_1, \ldots, v_n \in \mathbb{Z}^k$, $\lambda_1, \ldots, \lambda_p \in \mathbb{N}$, and convex functional $c\colon \mathbb{R}^{pk} \to \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle v_1, \ldots, v_n, \lambda_1, \ldots, \lambda_p \rangle]$, solves the constrained and unconstrained partitioning problems.*

Proof. There is an obvious one-to-one correspondence between partitions and matrices $x \in \{0,1\}^{p \times n}$ with all column-sums equal to one, where partition π corresponds to the matrix x with $x_{h,i} = 1$ if $i \in \pi_h$ and $x_{h,i} = 0$ otherwise. Let $d := pk$ and define d matrices $w_{h,j} \in \mathbb{Z}^{p \times n}$ by setting $(w_{h,j})_{h,i} := v_{i,j}$ for all $h = 1, \ldots, p$, $i = 1, \ldots, n$ and $j = 1, \ldots, k$, and setting all other entries to zero. Then for any partition π and its corresponding matrix x we have $v^{\pi}_{h,j} = w_{h,j} x$ for all $h = 1, \ldots, p$ and $j = 1, \ldots, k$. Therefore, the unconstrained vector partition problem is the convex integer program

$$\max \left\{ c(w_{1,1}x, \ldots, w_{p,k}x) : x \in \mathbb{N}^{p \times n}, \sum_h x_{h,i} = 1 \right\}.$$

Suitably arranging the variables in a vector, this becomes a convex n-fold integer program with a $(0+1) \times p$ defining matrix A, where A_1 is empty and $A_2 := (1, \ldots, 1)$.

Similarly, the constrained vector partition problem is the convex integer program

$$\max \left\{ c(w_{1,1}x, \ldots, w_{p,k}x) : x \in \mathbb{N}^{p \times n}, \sum_h x_{h,i} = 1, \sum_i x_{h,i} = \lambda_h \right\}.$$

This again is a convex n-fold integer program, now with a $(p+1) \times p$ defining matrix A, where now $A_1 := I_p$ is the $p \times p$ identity matrix and $A_2 := (1, \ldots, 1)$ as before.

Using the algorithm of Theorem 5.6, this convex n-fold integer program, and hence the given vector partition problem, can be solved in polynomial time. □

6. Multiway Transportation Problems and Privacy in Statistical Databases

Transportation problems form a very important class of discrete optimization problems. The feasible points in a transportation problem are the multiway tables ("contingency tables" in statistics) such that the sums of entries over some of their lower dimensional sub-tables such as lines or planes ("margins" in statistics) are specified. Transportation problems and their corresponding transportation polytopes have been used and studied extensively in the operations research and mathematical programming literature, as well as in the statistics literature in the context of secure statistical data disclosure and management by public agencies, see [21–30] and references therein.

In this section we completely settle the algorithmic complexity of treating multiway tables and discuss the applications to transportation problems and secure statistical data disclosure, as follows. After introducing some terminology in §6.1, we go on to describe, in §6.2, a universality result that shows that "short" 3-way $r \times c \times 3$ tables, with variable number r of rows and variable number c of columns but fixed small number 3 of layers (hence "short"), are *universal* in a very strong sense. In §6.3 we discuss the general multiway transportation problem. Using the results of §6.2 and the results on linear and convex n-fold integer programming from §4 and §5, we show that the transportation problem is intractable for short 3-way $r \times c \times 3$ tables but polynomial time treatable for "long" $(k+1)$-way $m_1 \times \cdots \times m_k \times n$ tables, with k and the sides $m_1, \ldots, m_k$ fixed (but arbitrary), and the number n of layers variable (hence "long"). In §6.4 we turn to discuss data privacy and security and consider the central problem of detecting entry uniqueness in tables with disclosed margins. We show that as a consequence of the results of §6.2 and §6.3, and in analogy to the complexity of the transportation problem established in §6.3, the entry uniqueness problem is intractable for short 3-way $r \times c \times 3$ tables but polynomial time decidable for long $(k+1)$-way $m_1 \times \cdots \times m_k \times n$ tables.

6.1. Tables and Margins

We start with some terminology on tables, margins and transportation polytopes. A *k-way table* is an $m_1 \times \cdots \times m_k$ array $x = (x_{i_1,\ldots,i_k})$ of nonnegative integers. A *k-way transportation polytope* (or simply *k-way polytope* for brevity) is the set of all $m_1 \times \cdots \times m_k$ nonnegative arrays $x = (x_{i_1,\ldots,i_k})$ such that the sums of the entries over some of their lower dimensional sub-arrays (margins) are specified. More precisely, for any tuple $(i_1, \ldots, i_k)$ with $i_j \in \{1, \ldots, m_j\} \cup \{+\}$, the corresponding *margin* $x_{i_1,\ldots,i_k}$ is the sum of entries of x over all coordinates j with $i_j = +$. The *support* of $(i_1, \ldots, i_k)$ and of $x_{i_1,\ldots,i_k}$ is the set $\mathrm{supp}(i_1, \ldots, i_k) := \{j : i_j \neq +\}$ of nonsummed coordinates. For instance, if x is a $4 \times 5 \times 3 \times 2$ array then it has 12 margins with support $F = \{1, 3\}$ such as $x_{3,+,2,+} = \sum_{i_2=1}^{5} \sum_{i_4=1}^{2} x_{3,i_2,2,i_4}$. A collection of margins is *hierarchical* if, for some family $\mathcal{F}$ of subsets of $\{1, \ldots, k\}$, it consists of all margins $u_{i_1,\ldots,i_k}$ with support in $\mathcal{F}$. In particular, for any $0 \leq h \leq k$, the collection of *all h-margins* of k-tables is the hierarchical collection with $\mathcal{F}$ the family of all h-subsets of $\{1, \ldots, k\}$. Given a hierarchical collection of margins $u_{i_1,\ldots,i_k}$ supported on a family $\mathcal{F}$ of subsets of $\{1, \ldots, k\}$, the corresponding *k-way polytope* is the set of nonnegative arrays with these margins,

$$T_{\mathcal{F}} := \{x \in \mathbb{R}_+^{m_1 \times \cdots \times m_k} : x_{i_1,\ldots,i_k} = u_{i_1,\ldots,i_k}, \mathrm{supp}(i_1, \ldots, i_k) \in \mathcal{F}\}.$$

The integer points in this polytope are precisely the k-way tables with the given margins.

6.2. The Universality Theorem

We now describe the following *universality* result of [6, 8] which shows that, quite remarkably, *any* rational polytope is a short 3-way $r\times c\times 3$ polytope with all line-sums specified. (In the terminology of §6.1 this is the $r \times c \times 3$ polytope $T_{\mathcal{F}}$ of all 2-margins fixed, supported on the family $\mathcal{F} = \{\{1,2\},\{1,3\},\{2,3\}\}$.) By saying that a polytope $P \subset \mathbb{R}^p$ is *representable* as a polytope $Q \subset \mathbb{R}^q$ we mean in the strong sense that there is an injection $\sigma\colon \{1,\ldots,p\} \to \{1,\ldots,q\}$ such that the coordinate-erasing projection

$$\pi\colon \mathbb{R}^q \to \mathbb{R}^p\colon x = (x_1,\ldots,x_q) \mapsto \pi(x) = (x_{\sigma(1)},\ldots,x_{\sigma(p)})$$

provides a bijection between Q and P and between the sets of integer points $Q \cap \mathbb{Z}^q$ and $P \cap \mathbb{Z}^p$. In particular, if P is representable as Q then P and Q are isomorphic in any reasonable sense: they are linearly equivalent and hence all linear programming related problems over the two are polynomial time equivalent; they are combinatorially equivalent and hence they have the same face numbers and facial structure; and they are integer equivalent and therefore all integer programming and integer counting related problems over the two are polynomial time equivalent as well.

We provide only an outline of the proof of the following statement; complete details and more consequences of this theorem can be found in [6, 8].

Theorem 6.1. *There is a polynomial time algorithm that, given $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$, encoded as $[\langle A, b\rangle]$, produces r, c and line-sums $u \in \mathbb{Z}^{r\times c}$, $v \in \mathbb{Z}^{r\times 3}$ and $z \in \mathbb{Z}^{c\times 3}$ such that the polytope $P := \{y \in \mathbb{R}^n_+ : Ay = b\}$ is representable as the 3-way polytope*

$$T := \left\{x \in \mathbb{R}_+^{r\times c\times 3} : \sum_i x_{i,j,k} = z_{j,k}, \sum_j x_{i,j,k} = v_{i,k}, \sum_k x_{i,j,k} = u_{i,j}\right\}.$$

Proof. The construction proving the theorem consists of three polynomial time steps, each representing a polytope of a given format as a polytope of another given format.

First, we show that any $P := \{y \geq 0 : Ay = b\}$ with A, b integer can be represented in polynomial time as $Q := \{x \geq 0 : Cx = d\}$ with C matrix all entries of which are in $\{-1, 0, 1, 2\}$. This reduction of coefficients will enable the rest of the steps to run in polynomial time. For each variable y_j let $k_j := \max\{\lfloor \log_2 |a_{i,j}| \rfloor : i = 1,\ldots m\}$ be the maximum number of bits in the binary representation of the absolute value of any entry $a_{i,j}$ of A. Introduce variables $x_{j,0},\ldots,x_{j,k_j}$, and relate them by the equations $2x_{j,i} - x_{j,i+1} = 0$. The representing injection σ is defined by $\sigma(j) := (j, 0)$, embedding y_j as $x_{j,0}$. Consider any term $a_{i,j}y_j$ of the original system. Using the binary expansion $|a_{i,j}| = \sum_{s=0}^{k_j} t_s 2^s$ with all $t_s \in \{0, 1\}$, we rewrite this term as $\pm \sum_{s=0}^{k_j} t_s x_{j,s}$. It is not hard to verify that this represents P as Q with defining $\{-1, 0, 1, 2\}$-matrix.

Second, we show that any $Q := \{y \geq 0 : Ay = b\}$ with A, b integer can be represented as a face F of a 3-way polytope with all plane-sums fixed, that is, a face of a 3-way polytope $T_{\mathcal{F}}$ of all 1-margins fixed, supported on the family $\mathcal{F} = \{\{1\},\{2\},\{3\}\}$.

Since Q is a polytope and hence bounded, we can compute (using Cramer's rule) an integer upper bound U on the value of any coordinate y_j of any $y \in Q$. Note also that a face of a 3-way polytope $T_{\mathcal{F}}$ is the set of all $x = (x_{i,j,k})$ with some entries forced to zero; these entries are termed "forbidden," and the other entries are termed "enabled."

For each variable y_j, let r_j be the largest between the sum of positive coefficients of y_j and the sum of absolute values of negative coefficients of y_j over all equations,

$$r_j := \max\Big(\sum_k \{a_{k,j} : a_{k,j} > 0\}, \sum_k \{|a_{k,j}| : a_{k,j} < 0\}\Big).$$

Assume that A is of size $m \times n$. Let $r := \sum_{j=1}^n r_j$, $R := \{1, \ldots, r\}$, $h := m + 1$ and $H := \{1, \ldots, h\}$. We now describe how to construct vectors $u, v \in \mathbb{Z}^r, z \in \mathbb{Z}^h$, and a set $E \subset R \times R \times H$ of triples — the enabled, nonforbidden, entries — such that the polytope Q is represented as the face F of the corresponding 3-way polytope of $r \times r \times h$ arrays with plane-sums u, v, z and only entries indexed by E enabled,

$$F := \Big\{x \in \mathbb{R}_+^{r \times r \times h} : x_{i,j,k} = 0 \text{ for all } (i,j,k) \notin E, \text{ and } \sum_{i,j} x_{i,j,k} = z_k, \sum_{i,k} x_{i,j,k} = v_j, \sum_{j,k} x_{i,j,k} = u_i\Big\}.$$

We also indicate the injection $\sigma\colon \{1, \ldots, n\} \to R \times R \times H$ giving the desired embedding of coordinates y_j as coordinates $x_{i,j,k}$ and the representation of Q as F.

Roughly, each equation $k = 1, \ldots, m$ is encoded in a "horizontal plane" $R \times R \times \{k\}$ (the last plane $R \times R \times \{h\}$ is included for consistency with its entries being "slacks"); and each variable y_j, $j = 1, \ldots, n$ is encoded in a "vertical box" $R_j \times R_j \times H$, where $R = \biguplus_{j=1}^n R_j$ is the natural partition of R with $|R_j| = r_j$ for all $j = 1, \ldots, n$, that is, with $R_j := \{1 + \sum_{l<j} r_l, \ldots, \sum_{l \le j} r_l\}$.

Now, all "vertical" plane-sums are set to the same value U, that is, $u_j := v_j := U$ for $j = 1, \ldots, r$. All entries not in the union $\biguplus_{j=1}^n R_j \times R_j \times H$ of the variable boxes will be forbidden. We now describe the enabled entries in the boxes; for simplicity we discuss the box $R_1 \times R_1 \times H$, the others being similar. We distinguish between the two cases $r_1 = 1$ and $r_1 \ge 2$. In the first case, $R_1 = \{1\}$; the box, which is just the single line $\{1\} \times \{1\} \times H$, will have exactly two enabled entries $(1, 1, k^+), (1, 1, k^-)$ for suitable k^+, k^- to be defined later. We set $\sigma(1) := (1, 1, k^+)$, namely embed $y_1 = x_{1,1,k^+}$. We define the *complement* of the variable y_1 to be $\overline{y}_1 := U - y_1$ (and likewise for the other variables). The vertical sums u, v then force $\overline{y}_1 = U - y_1 = U - x_{1,1,k^+} = x_{1,1,k^-}$, so the complement of y_1 is also embedded. Next, consider the case $r_1 \ge 2$. For each $s = 1, \ldots, r_1$, the line $\{s\} \times \{s\} \times H$ (respectively, $\{s\} \times \{1 + (s \bmod r_1)\} \times H$) will contain one enabled entry $(s, s, k^+(s))$ (respectively, $(s, 1 + (s \bmod r_1), k^-(s))$. All other entries of $R_1 \times R_1 \times H$ will be forbidden. Again, we set $\sigma(1) := (1, 1, k^+(1))$, namely embed $y_1 = x_{1,1,k^+(1)}$; it is then not hard to see that, again, the vertical sums u, v force $x_{s,s,k^+(s)} = x_{1,1,k^+(1)} = y_1$ and $x_{s,1+(s \bmod r_1),k^-(s)} = U - x_{1,1,k^+(1)} = \overline{y}_1$ for each $s = 1, \ldots, r_1$. Therefore, both y_1 and $\overline{y}_1$ are each embedded in r_1 distinct entries.

We now encode the equations by defining the horizontal plane-sums z and the indices $k^+(s), k^-(s)$ above as follows. For $k = 1, \ldots, m$, consider the kth equation $\sum_j a_{k,j} y_j = b_k$. Define the index sets $J^+ := \{j : a_{k,j} > 0\}$ and $J^- := \{j : a_{k,j} < 0\}$, and set $z_k := b_k + U \cdot \sum_{j \in J^-} |a_{k,j}|$. The last coordinate of z is set for consistency with u, v to be $z_h = z_{m+1} := r \cdot U - \sum_{k=1}^m z_k$. Now, with $\overline{y}_j := U - y_j$ the complement of variable y_j as above, the kth equation can be rewritten as

$$\sum_{j\in J^+} a_{k,j}y_j + \sum_{j\in J^-} |a_{k,j}|\overline{y}_j = \sum_{j=1}^{n} a_{k,j}y_j + U\cdot\sum_{j\in J^-}|a_{k,j}| = b_k + U\cdot\sum_{j\in J^-}|a_{k,j}| = z_k.$$

To encode this equation, we simply "pull down" to the corresponding kth horizontal plane as many copies of each variable y_j or $\overline{y}_j$ by suitably setting $k^+(s) := k$ or $k^-(s) := k$. By the choice of r_j there are sufficiently many, possibly with a few redundant copies which are absorbed in the last hyperplane by setting $k^+(s) := m+1$ or $k^-(s) := m+1$. This completes the encoding and provides the desired representation.

Third, we show that any 3-way polytope with plane-sums fixed and entry bounds,

$$F := \left\{ y\in\mathbb{R}_+^{l\times m\times n} : \sum_{i,j} y_{i,j,k} = c_k, \sum_{i,k} y_{i,j,k} = b_j, \sum_{j,k} y_{i,j,k} = a_i,\ y_{i,j,k}\le e_{i,j,k} \right\},$$

can be represented as a 3-way polytope with line-sums fixed (and no entry bounds),

$$T := \left\{ x\in\mathbb{R}_+^{r\times c\times 3} : \sum_{I} x_{I,J,K} = z_{J,K}, \sum_{J} x_{I,J,K} = v_{I,K}, \sum_{K} x_{I,J,K} = u_{I,J} \right\}.$$

In particular, this implies that any face F of a 3-way polytope with plane-sums fixed can be represented as a 3-way polytope T with line-sums fixed: forbidden entries are encoded by setting a "forbidding" upper-bound $e_{i,j,k} := 0$ on all forbidden entries $(i,j,k)\notin E$ and an "enabling" upper-bound $e_{i,j,k} := U$ on all enabled entries $(i,j,k)\in E$. We describe the presentation, but omit the proof that it is indeed valid; further details on this step can be found in [5,6,8]. We give explicit formulas for $u_{I,J}, v_{I,K}, z_{J,K}$ in terms of a_i, b_j, c_k and $e_{i,j,k}$ as follows. Put $r := l\cdot m$ and $c := n+l+m$. The first index I of each entry $x_{I,J,K}$ will be a pair $I=(i,j)$ in the r-set

$$\{(1,1),\ldots,(1,m),(2,1),\ldots,(2,m),\ldots,(l,1),\ldots,(l,m)\}.$$

The second index J of each entry $x_{I,J,K}$ will be a pair $J=(s,t)$ in the c-set

$$\{(1,1),\ldots,(1,n),(2,1),\ldots,(2,l),(3,1),\ldots,(3,m)\}.$$

The last index K will simply range in the 3-set $\{1,2,3\}$. We represent F as T via the injection σ given explicitly by $\sigma(i,j,k) := ((i,j),(1,k),1)$, embedding each variable $y_{i,j,k}$ as the entry $x_{(i,j),(1,k),1}$. Let U now denote the minimal between the two values $\max\{a_1,\ldots,a_l\}$ and $\max\{b_1,\ldots,b_m\}$. The line-sums (2-margins) are set to be

$$u_{(i,j),(1,t)} = e_{i,j,t}, \qquad u_{(i,j),(2,t)} = \begin{cases} U & \text{if } t=i,\\ 0 & \text{otherwise,}\end{cases} \qquad u_{(i,j),(3,t)} = \begin{cases} U & \text{if } t=j,\\ 0 & \text{otherwise,}\end{cases}$$

$$v_{(i,j),t} = \begin{cases} U & \text{if } t=1,\\ e_{i,j,+} & \text{if } t=2,\\ U & \text{if } t=3,\end{cases} \qquad z_{(i,j),1} = \begin{cases} c_j & \text{if } i=1,\\ m\cdot U - a_j & \text{if } i=2,\\ 0 & \text{if } i=3,\end{cases}$$

$$z_{(i,j),2} = \begin{cases} e_{+,+,j} - c_j & \text{if } i=1,\\ 0 & \text{if } i=2,\\ b_j & \text{if } i=3,\end{cases} \qquad z_{(i,j),3} = \begin{cases} 0 & \text{if } i=1,\\ a_j & \text{if } i=2,\\ l\cdot U - b_j & \text{if } i=3.\end{cases}$$

Applying the first step to the given rational polytope P, applying the second step to the resulting Q, and applying the third step to the resulting F, we get in polynomial time a 3-way $r \times c \times 3$ polytope T of all line-sums fixed representing P as claimed. □

6.3. The Complexity of the Multiway Transportation Problem

We are now finally in position to settle the complexity of the general multiway transportation problem. The data for the problem consists of: positive integers k (table dimension) and $m_1, \ldots, m_k$ (table sides); family $\mathcal{F}$ of subsets of $\{1, \ldots, k\}$ (supporting the hierarchical collection of margins to be fixed); integer values $u_{i_1,\ldots,i_k}$ for all margins supported on $\mathcal{F}$; and integer "profit" $m_1 \times \cdots \times m_k$ array w. The transportation problem is to find an $m_1 \times \cdots \times m_k$ table having the given margins and attaining maximum profit, or assert than none exists. Equivalently, it is the linear integer programming problem of maximizing the linear functional defined by w over the transportation polytope $T_{\mathcal{F}}$,

$$\max\{wx : x \in \mathbb{N}^{m_1 \times \cdots \times m_k} : x_{i_1,\ldots,i_k} = u_{i_1,\ldots,i_k}, \operatorname{supp}(i_1, \ldots, i_k) \in \mathcal{F}\}.$$

The following result of [5] is an immediate consequence of Theorem 6.1. It asserts that if two sides of the table are variable part of the input then the transportation problem is intractable already for short 3-way tables with $\mathcal{F} = \{\{1,2\},\{1,3\},\{2,3\}\}$ supporting all 2-margins (line-sums). This result can be easily extended to k-way tables of any dimension $k \geq 3$ and $\mathcal{F}$ the collection of all h-subsets of $\{1, \ldots, k\}$ for any $1 < h < k$ as long as two sides of the table are variable; we omit the proof of this extended result.

Corollary 6.2. *It is* NP*-complete to decide, given r, c, and line-sums $u \in \mathbb{Z}^{r\times c}$, $v \in \mathbb{Z}^{r\times 3}$, and $z \in \mathbb{Z}^{c\times 3}$, encoded as $[\langle u, v, z\rangle]$, if the following set of tables is nonempty*,

$$S := \left\{x \in \mathbb{N}^{r\times c\times 3} : \sum_i x_{i,j,k} = z_{j,k}, \sum_j x_{i,j,k} = v_{i,k}, \sum_k x_{i,j,k} = u_{i,j}\right\}.$$

Proof. The integer programming feasibility problem is to decide, given $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$, if $\{y \in \mathbb{N}^n : Ay = b\}$ is nonempty. Given such A and b, the polynomial time algorithm of Theorem 6.1 produces r, c and $u \in \mathbb{Z}^{r\times c}$, $v \in \mathbb{Z}^{r\times 3}$, and $z \in \mathbb{Z}^{c\times 3}$, such that $\{y \in \mathbb{N}^n : Ay = b\}$ is nonempty if and only if the set S above is nonempty. This reduces integer programming feasibility to short 3-way line-sum transportation feasibility. Since the former is NP-complete (see, e.g., [1]), so turns out to be the latter. □

We now show that in contrast, when all sides but one are fixed (but arbitrary), and one side n is variable, then the corresponding long k-way transportation problem for any hierarchical collection of margins is an n-fold integer programming problem and therefore, as a consequence of Theorem 4.12, can be solved is polynomial time. This extends Corollary 4.13 established in §4.5.1 for 3-way line-sum transportation.

Corollary 6.3. *For every fixed k, table sides $m_1, \ldots, m_k$, and family $\mathcal{F}$ of subsets of $\{1, \ldots, k+1\}$, there is a polynomial time algorithm that, given n, integer values $u = (u_{i_1,\ldots,i_{k+1}})$ for all margins supported on $\mathcal{F}$, and integer $m_1 \times \cdots \times m_k \times n$ array w, encoded as $[\langle u, w\rangle]$, solves the linear integer multiway transportation problem*

$$\max\{wx : x \in \mathbb{N}^{m_1\times\cdots\times m_k\times n}, x_{i_1,\ldots,i_{k+1}} = u_{i_1,\ldots,i_{k+1}}, \operatorname{supp}(i_1, \ldots, i_{k+1}) \in \mathcal{F}\}.$$

Proof. Re-index the arrays as $x = (x^1, \ldots, x^n)$ with each $x^j = (x_{i_1,\ldots,i_k,j})$ a suitably indexed $m_1 m_2 \cdots m_k$ vector representing the jth layer of x. Then the transportation problem can be encoded as an n-fold integer programming problem in standard form,

$$\max\{wx : x \in \mathbb{N}^{nt}, A^{(n)}x = b\},$$

with an $(r+s) \times t$ defining matrix A where $t := m_1 m_2 \cdots m_k$ and r, s, A_1 and A_2 are determined from $\mathcal{F}$, and with right-hand side $b := (b^0, b^1, \ldots, b^n) \in \mathbb{Z}^{r+ns}$ determined from the margins $u = (u_{i_1,\ldots,i_{k+1}})$, in such a way that the equations $A_1(\sum_{j=1}^n x^j) = b^0$ represent the constraints of all margins $x_{i_1,\ldots,i_k,+}$ (where summation over layers occurs), whereas the equations $A_2 x^j = b^j$ for $j = 1, \ldots, n$ represent the constraints of all margins $x_{i_1,\ldots,i_k,j}$ with $j \neq +$ (where summations are within a single layer at a time).

Using the algorithm of Theorem 4.12, this n-fold integer program, and hence the given multiway transportation problem, can be solved in polynomial time. □

The proof of Corollary 6.3 shows that the set of feasible points of any long k-way transportation problem, with all sides but one fixed and one side n variable, for any hierarchical collection of margins, is an n-fold integer programming problem. Therefore, as a consequence of Theorem 5.6, we also have the following extension of Corollary 6.3 for the convex integer multiway transportation problem over long k-way tables.

Corollary 6.4. *For every fixed d, k, table sides $m_1, \ldots, m_k$, and family $\mathcal{F}$ of subsets of $\{1, \ldots, k+1\}$, there is a polynomial time algorithm that, given n, integer values $u = (u_{i_1,\ldots,i_{k+1}})$ for all margins supported on $\mathcal{F}$, integer $m_1 \times \cdots \times m_k \times n$ arrays $w_1, \ldots, w_d$, and convex functional $c\colon \mathbb{R}^d \to \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle u, w_1, \ldots, w_d \rangle]$, solves the convex integer multiway transportation problem*

$$\max\{c(w_1 x, \ldots, w_d x) : x \in \mathbb{N}^{m_1 \times \cdots \times m_k \times n}, x_{i_1,\ldots,i_{k+1}} = u_{i_1,\ldots,i_{k+1}}, \mathrm{supp}(i_1, \ldots, i_{k+1}) \in \mathcal{F}\}.$$

6.4. Privacy and Entry-Uniqueness

A common practice in the disclosure of a multiway table containing sensitive data is to release some of the table margins rather than the table itself, see, e.g., [23–25] and the references therein. Once the margins are released, the security of any specific entry of the table is related to the set of possible values that can occur in that entry in any table having the same margins as those of the source table in the data base. In particular, if this set consists of a unique value, that of the source table, then this entry can be exposed and privacy can be violated. This raises the following fundamental *entry-uniqueness problem*: given a consistent disclosed (hierarchical) collection of margin values, and a specific entry index, is the value that can occur in that entry in any table having these margins unique? We now describe the results of [14] that settle the complexity of this problem, and interpret the consequences for secure statistical data disclosure.

First, we show that if two sides of the table are variable part of the input then the entry-uniqueness problem is intractable already for short 3-way tables with all 2-margins (line-sums) disclosed (corresponding to $\mathcal{F} = \{\{1,2\},\{1,3\},\{2,3\}\}$). This can be easily extended to k-way tables of any dimension $k \geq 3$ and $\mathcal{F}$ the collection of all h-subsets of $\{1, \ldots, k\}$ for any $1 < h < k$ as long as two sides of the table are variable; we omit the proof of this extended result. While this result indicates that the disclosing agency

may not be able to check for uniqueness, in this situation, some consolation is in that an adversary will be computationally unable to identify and retrieve a unique entry either.

Corollary 6.5. *It is* coNP*-complete to decide, given* r, c*, and line-sums* $u \in \mathbb{Z}^{r\times c}$, $v \in \mathbb{Z}^{r\times 3}$, $z \in \mathbb{Z}^{c\times 3}$*, encoded as* $[\langle u, v, z\rangle]$*, if the entry* $x_{1,1,1}$ *is the same in all tables in*

$$\left\{x \in \mathbb{N}^{r\times c\times 3} : \sum_i x_{i,j,k} = z_{j,k}, \sum_j x_{i,j,k} = v_{i,k}, \sum_k x_{i,j,k} = u_{i,j}\right\}.$$

Proof. The *subset-sum problem*, well known to be NP-complete, is the following: given positive integers $a_0, a_1, \ldots, a_m$, decide if there is an $I \subseteq \{1, \ldots, m\}$ with $a_0 = \sum_{i\in I} a_i$. We reduce the complement of subset-sum to entry-uniqueness. Given $a_0, a_1, \ldots, a_m$, consider the polytope in $2(m + 1)$ variables $y_0, y_1 \ldots, y_m, z_0, z_1, \ldots, z_m$,

$$P := \left\{(y, z) \in \mathbb{R}_+^{2(m+1)} : a_0y_0 - \sum_{i=1}^{m} a_iy_i = 0, y_i + z_i = 1, i = 0, 1 \ldots, m\right\}.$$

First, note that it always has one integer point with $y_0 = 0$, given by $y_i = 0$ and $z_i = 1$ for all i. Second, note that it has an integer point with $y_0 \neq 0$ if and only if there is an $I \subseteq \{1, \ldots, m\}$ with $a_0 = \sum_{i\in I} a_i$, given by $y_0 = 1$, $y_i = 1$ for $i \in I$, $y_i = 0$ for $i \in \{1, \ldots, m\} \setminus I$, and $z_i = 1 - y_i$ for all i. Lifting P to a suitable $r\times c\times 3$ line-sum polytope T with the coordinate y_0 embedded in the entry $x_{1,1,1}$ using Theorem 6.1, we find that T has a table with $x_{1,1,1} = 0$, and this value is unique among the tables in T if and only if there is *no* solution to the subset-sum problem with $a_0, a_1, \ldots, a_m$. □

Next we show that, in contrast, when all table sides but one are fixed (but arbitrary), and one side n is variable, then, as a consequence of Corollary 6.3, the corresponding long k-way entry-uniqueness problem for any hierarchical collection of margins can be solved is polynomial time. In this situation, the algorithm of Corollary 6.6 below allows disclosing agencies to efficiently check possible collections of margins before disclosure: if an entry value is not unique then disclosure may be assumed secure, whereas if the value is unique then disclosure may be risky and fewer margins should be released. Note that this situation, of long multiway tables, where one category is significantly richer than the others, that is, when each sample point can take many values in one category and only few values in the other categories, occurs often in practical applications, e.g., when one category is the individuals age and the other categories are binary ("yes – no"). In such situations, our polynomial time algorithm below allows disclosing agencies to check entry-uniqueness and make learned decisions on secure disclosure.

Corollary 6.6. *For every fixed* k*, table sides* $m_1, \ldots, m_k$*, and family* $\mathcal{F}$ *of subsets of* $\{1, \ldots, k + 1\}$*, there is a polynomial time algorithm that, given* n*, integer values* $u = (u_{j_1,\ldots,j_{k+1}})$ *for all margins supported on* $\mathcal{F}$*, and entry index* $(i_1, \ldots, i_{k+1})$*, encoded as* $[n, \langle u\rangle]$*, decides if the entry* $x_{i_1,\ldots,i_{k+1}}$ *is the same in all tables in the set*

$$\{x \in \mathbb{N}^{m_1\times\cdots\times m_k\times n} : x_{j_1,\ldots,j_{k+1}} = u_{j_1,\ldots,j_{k+1}}, \mathrm{supp}(j_1, \ldots, j_{k+1}) \in \mathcal{F}\}.$$

Proof. By Theorem 6.3 we can solve in polynomial time both transportation problems

$$l := \min\{x_{i_1,\ldots,i_{k+1}} : x \in \mathbb{N}^{m_1\times\cdots\times m_k\times n}, x \in T_{\mathcal{F}}\},$$
$$u := \max\{x_{i_1,\ldots,i_{k+1}} : x \in \mathbb{N}^{m_1\times\cdots\times m_k\times n}, x \in T_{\mathcal{F}}\},$$

over the corresponding k-way transportation polytope

$$T_{\mathcal{F}} := \{x \in \mathbb{R}_+^{m_1\times\cdots\times m_k\times n} : x_{j_1,\ldots,j_{k+1}} = u_{j_1,\ldots,j_{k+1}}, \mathrm{supp}(j_1,\ldots,j_{k+1}) \in \mathcal{F}\}.$$

Clearly, entry $x_{i_1,\ldots,i_{k+1}}$ has the same value in all tables with the given (disclosed) margins if and only if $l = u$, completing the description of the algorithm and the proof. □

Acknowledgement

This article is reprinted from [66] with kind permission of Springer Science+Business Media.

References

[1] Schrijver, A. (1986) *Theory of Linear and Integer Programming*. Wiley-Interscience Series in Discrete Mathematics, Wiley, Chichester.
[2] Grötschel, M., Lovász, L., and Schrijver, A. (1993) *Geometric Algorithms and Combinatorial Optimization*, vol. 2 of *Algorithms and Combinatorics*. Springer, Berlin, 2nd edn.
[3] Onn, S. (2006), Convex discrete optimization. Lecture Series, Séminaire de mathématiques supérieures, Combinatorial Optimization: Methods and Applications, Université de Montréal, Canada.
[4] Babson, E., Onn, S., and Thomas, R. (2003) The Hilbert zonotope and a polynomial time algorithm for universal Gröbner bases. *Advances in Applied Mathematics*, **30**, 529–544.
[5] De Loera, J. A. and Onn, S. (2004) The complexity of three-way statistical tables. *SIAM Journal on Computing*, **33**, 819–836.
[6] De Loera, J. A. and Onn, S. (2004) All rational polytopes are transportation polytopes and all polytopal integer sets are contingency tables. Bienstock, D. and Nemhauser, G. (eds.), *Integer Programming and Combinatorial Optimization*, vol. 3064 of *Lecture Notes in Computer Science*, pp. 338–351, Springer, Berlin.
[7] De Loera, J. A. and Onn, S. (2006) Markov bases of three-way tables are arbitrarily complicated. *Journal of Symbolic Computation*, **41**, 173–181.
[8] De Loera, J. A. and Onn, S. (2006) All linear and integer programs are slim 3-way transportation programs. *SIAM Journal on Optimization*, **17**, 806–821.
[9] De Loera, J. A., Hemmecke, R., Onn, S., and Weismantel, R. (2008) n-fold integer programming. *Discrete Optimization*, **5**, 231–241.
[10] De Loera, J. A., Hemmecke, R., Onn, S., Rothblum, U. G., and Weismantel, R. (2009) Convex integer maximization via Graver bases. *Journal of Pure and Applied Algebra*, **213**, 1569–1577.
[11] Fukuda, K., Onn, S., and Rosta, V. (2003) An adaptive algorithm for vector partitioning. *Journal of Global Optimization*, **25**, 305–319.
[12] Hwang, F. K., Onn, S., and Rothblum, U. G. (1999) A polynomial time algorithm for shaped partition problems. *SIAM Journal on Optimization*, **10**, 70–81.
[13] Onn, S. (2003) Convex matroid optimization. *SIAM Journal on Discrete Mathematics*, **17**, 249–253.
[14] Onn, S. (2006) Entry uniqueness in margined tables. Domingo-Ferrer, J. and Franconi, L. (eds.), *Privacy in Statistical Databases*, vol. 4302 of *Lecture Notes in Computer Science*, pp. 94–101, Springer, Berlin.
[15] Onn, S. and Rothblum, U. G. (2004) Convex combinatorial optimization. *Discrete & Computational Geometry*, **32**, 549–566.
[16] Onn, S. and Schulman, L. J. (2001) The vector partition problem for convex objective functions. *Mathematics of Operations Research*, **26**, 583–590.
[17] Onn, S. and Sturmfels, B. (1999) Cutting corners. *Advances in Applied Mathematics*, **23**, 29–48.

[18] Gilmore, P. C. and Gomory, R. E. (1961) A linear programming approach to the cutting-stock problem. *Operations Research*, **9**, 849–859.
[19] Barnes, E. R., Hoffman, A. J., and Rothblum, U. G. (1992) Optimal partitions having disjoint convex and conic hulls. *Mathematical Programming, Series A*, **54**, 69–86.
[20] Boros, E. and Hammer, P. L. (1989) On clustering problems with connected optima in Euclidean spaces. *Discrete Mathematics*, **75**, 81–88.
[21] Aoki, S. and Takemura, A. (2003) Minimal basis for a connected Markov chain over $3\times3\times K$ contingency tables with fixed two-dimensional marginals. *Australian & New Zealand Journal of Statistics*, **45**, 229–249.
[22] Balinski, M. L. and Rispoli, F. J. (1993) Signature classes of transportation polytopes. *Mathematical Programming, Series A*, **60**, 127–144.
[23] Cox, L. H. (2003) On properties of multi-dimensional statistical tables. *Journal of Statistical Planning and Inference*, **117**, 251–273.
[24] Domingo-Ferrer, J. and Torra, V. (eds.) (2004) *Privacy in Statistical Databases*, vol. 3050 of *Lecture Notes in Computer Science*. Springer, Berlin.
[25] Doyle, P., Lane, J. I., Theeuwes, J. J. M., and Zayatz, L. V. (eds.) (2001) *Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies*. North-Holland, Amsterdam.
[26] Klee, V. and Witzgall, C. (1968) Facets and vertices of transportation polytopes. Dantzig, G. B. and Veinott, A. F., Jr. (eds.), *Mathematics of the Decision Sciences. Part I*, (Stanford, CA, 1967), vol. 11 of *Lectures in Applied Mathematics*, pp. 257–282, Amer. Math. Soc., Providence, R.I.
[27] Kleinschmidt, P., Lee, C. W., and Schannath, H. (1987) Transportation problems which can be solved by the use of Hirsch-paths for the dual problems. *Mathematical Programming*, **37**, 153–168.
[28] Queyranne, M. and Spieksma, F. C. R. (1997) Approximation algorithms for multi-index transportation problems with decomposable costs. *Discrete Applied Mathematics*, **76**, 239–253.
[29] Vlach, M. (1986) Conditions for the existence of solutions of the three-dimensional planar transportation problem. *Discrete Applied Mathematics*, **13**, 61–78.
[30] Yemelichev, V. A., Kovalev, M. M., and Kravtsov, M. K. (1984) *Polytopes, graphs and optimisation*. Cambridge Univ. Press.
[31] Chvátal, V. (1973) Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, **4**, 305–337.
[32] Lovász, L. (1986) *An Algorithmic Theory of Numbers, Graphs and Convexity*, vol. 50 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, PA.
[33] Yudin, D. B. and Nemirovskii, A. S. (1977) Informational complexity and efficient methods for the solution of convex extremal problems. *Matekon*, **13**, 25–45.
[34] Klee, V. and Kleinschmidt, P. (1987) The d-step conjecture and its relatives. *Mathematics of Operations Research*, **12**, 718–755.
[35] Grünbaum, B. (2003) *Convex Polytopes*, vol. 221 of *Graduate Texts in Mathematics*. Springer, New York, 2nd edn.
[36] Ziegler, G. M. (1995) *Lectures on Polytopes*, vol. 152 of *Graduate Texts in Mathematics*. Springer.
[37] Aho, A. V., Hopcroft, J. E., and Ullman, J. D. (1975) *The Design and Analysis of Computer Algorithms*. Addison-Wesley Series in Computer Science and Information Processing, Addison-Wesley , Reading, MA.
[38] Garey, M. R. and Johnson, D. S. (1979) *Computers and Intractability*. W. H. Freeman, San Francisco, CA.
[39] Tardos, É. (1986) A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, **34**, 250–256.
[40] Harding, E. F. (1966/1967) The number of partitions of a set of N points in k dimensions induced by hyperplanes. *Proceedings of the Edinburgh Mathematical Society. Series II*, **15**, 285–289.
[41] Zaslavsky, T. (1975) Facing up to arrangements: face-count formulas for partitions of space by hyperplanes. *Memoirs of the American Mathematical Society*, **1**.
[42] Alon, N. and Onn, S. (1999) Separable partitions. *Discrete Applied Mathematics*, **91**, 39–51.
[43] Gritzmann, P. and Sturmfels, B. (1993) Minkowski addition of polytopes: computational complexity and applications to Gröbner bases. *SIAM Journal on Discrete Mathematics*, **6**, 246–269.
[44] Edelsbrunner, H., O'Rourke, J., and Seidel, R. (1986) Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal on Computing*, **15**, 341–363.
[45] Edelsbrunner, H., Seidel, R., and Sharir, M. (1991) On the zone theorem for hyperplane arrangements.

New Results and New trends in Computer Science, (Graz, 1991), vol. 555 of *Lecture Notes in Computer Science*, pp. 108–123, Springer, Berlin.

[46] Berstein, Y. and Onn, S. (2008) Nonlinear bipartite matching. *Discrete Optimization*, **5**, 53–65.

[47] Grötschel, M. and Lovász, L. (1995) Combinatorial optimization. *Handbook of Combinatorics, Vol. 1, 2*, pp. 1541–1597, Elsevier, Amsterdam.

[48] Schulz, A. S., Weismantel, R., and Ziegler, G. M. (1995) 0/1-Integer programming: optimization and augmentation are equivalent. Spirakis, P. (ed.), *Algorithms—ESA '95*, vol. 979 of *Lecture Notes in Computer Science*, pp. 473–483, Springer, Berlin.

[49] Edmonds, J. and Karp, R. M. (1972) Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, **19**, 248–264.

[50] Frank, A. and Tardos, É. (1987) An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica*, **7**, 49–65.

[51] Lenstra, A. K., Lenstra, H. W., Jr., and Lovász, L. (1982) Factoring polynomials with rational coefficients. *Mathematische Annalen*, **261**, 515–534.

[52] Khachiyan, L. G. (1979) A polynomial algorithm in linear programming. *Soviet Mathematics. Doklady*, **20**, 191–194.

[53] Allemand, K., Fukuda, K., Liebling, T. M., and Steiner, E. (2001) A polynomial case of unconstrained zero-one quadratic optimization. *Mathematical Programming, Series A*, **91**, 49–52.

[54] Edmonds, J. (1971) Matroids and the greedy algorithm. *Mathematical Programming*, **1**, 127–136.

[55] Hassin, R. and Tamir, A. (1989) Maximizing classes of two-parameter objectives over matroids. *Mathematics of Operations Research*, **14**, 362–375.

[56] Pistone, G., Riccomagno, E., and Wynn, H. P. (2001) *Algebraic Statistics*, vol. 89 of *Monographs on Statistics and Applied Probability*. Chapman & Hall/CRC, Boca Raton, FL.

[57] Schulz, A. S. and Weismantel, R. (2002) The complexity of generic primal algorithms for solving general integral programs. *Mathematics of Operations Research*, **27**, 681–692.

[58] Wallacher, C. and Zimmermann, U. (1992) A combinatorial interior point method for network flow problems. *Mathematical Programming, Series A*, **56**, 321–335.

[59] Graver, J. E. (1975) On the foundations of linear and integer linear programming. I. *Mathematical Programming*, **9**, 207–226.

[60] Sturmfels, B. (1996) *Gröbner Bases and Convex Polytopes*, vol. 8 of *University Lecture Series*. Amer. Math. Soc., Providence, R.I.

[61] Hemmecke, R. (2003) On the positive sum property and the computation of Graver test sets. *Mathematical Programming, Series B*, **96**, 247–269.

[62] Hemmecke, R., Hemmecke, R., and Malkin, P. (2005), 4ti2 version 1.2 – Computation of Hilbert bases, Graver bases, toric Gröbner bases, and more. Available at http://www.4ti2.de/.

[63] Santos, F. and Sturmfels, B. (2003) Higher Lawrence configurations. *Journal of Combinatorial Theory. Series A*, **103**, 151–164.

[64] Hoşten, S. and Sullivant, S. (2007) A finiteness theorem for Markov bases of hierarchical models. *Journal of Combinatorial Theory. Series A*, **114**, 311–321.

[65] Hoffman, A. J. and Kruskal, J. B. (1956) Integral boundary points of convex polyhedra. Kuhn, H. W. and Tucker, A. W. (eds.), *Linear Inequalities and Related Systems*, vol. 38 of *Annals of Mathematics Studies*, pp. 223–246, Princeton Univ. Press, Princeton, NJ.

[66] Onn, S. (2009) Convex discrete optimization. Floudas, C. A. and Pardalos, P. M. (eds.), *Encyclopedia of Optimization*, pp. 513–550, Springer, New York, 2nd edn.