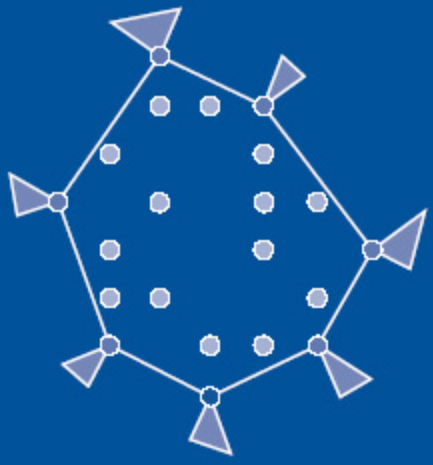
ZURICH LECTURES IN ADVANCED MATHEMATICS

Shmuel Onn

Nonlinear Discrete Optimization

An Algorithmic Theory



European Mathematical Society


S E
M M
E S

Zurich Lectures in Advanced Mathematics

Edited by

Mathematics in Zurich has a long and distinguished tradition, in which the writing of lecture notes volumes and research monographs plays a prominent part. The *Zurich Lectures in Advanced Mathematics* series aims to make some of these publications better known to a wider audience. The series has three main constituents: lecture notes on advanced topics given by internationally renowned experts, graduate text books designed for the joint graduate program in Mathematics of the ETH and the University of Zurich, as well as contributions from researchers in residence at the mathematics research institute, FIM-ETH. Moderately priced, concise and lively in style, the volumes of this series will appeal to researchers and students alike, who seek an informed introduction to important areas of current research.

Previously published in this series:

Yakov B. Pesin, *Lectures on partial hyperbolicity and stable ergodicity*
Sun-Yung Alice Chang, *Non-linear Elliptic Equations in Conformal Geometry*
Sergei B. Kuksin, *Randomly forced nonlinear PDEs and statistical hydrodynamics in 2 space dimensions*
Pavel Etingof, *Calogero-Moser systems and representation theory*
Guus Balkema and Paul Embrechts, *High Risk Scenarios and Extremes – A geometric approach*
Demetrios Christodoulou, *Mathematical Problems of General Relativity I*
Camillo De Lellis, *Rectifiable Sets, Densities and Tangent Measures*
Paul Seidel, *Fukaya Categories and Picard–Lefschetz Theory*
Alexander H.W. *Schmitt, Geometric Invariant Theory and Decorated Principal Bundles*
Michael Farber, *Invitation to Topological Robotics*
Alexander Barvinok, *Integer Points in Polyhedra*
Christian Lubich, *From Quantum to Classical Molecular Dynamics: Reduced Models and Numerical Analysis*

Published with the support of the Huber-Kudlich-Stiftung, Zürich

Shmuel Onn

Nonlinear Discrete Optimization

An Algorithmic Theory

European Mathematical Society

Author:

Prof. Shmuel Onn
Davidson Faculty of IE & M
Technion – Israel Institute of Technology
Technion City, Haifa 32000
Israel

E-mail: onn@ie.technion.ac.il

2010 Mathematics Subject Classification: 05Axx, 05Cxx, 05Dxx, 05Exx, 11Dxx, 11Hxx, 11Pxx, 13Pxx, 14Qxx, 15Axx, 15Bxx, 51Mxx, 52Axx, 52Bxx, 52Cxx, 62Hxx, 62Kxx, 62Qxx, 65Cxx, 68Qxx, 68Rxx, 68Wxx, 90Bxx, 90Cxx

Key words: integer programming, combinatorial optimization, optimization, linear programming, stochastic programming, randomized algorithm, approximation algorithm, polynomial time, transportation problem, multi index transportation problem, transshipment problem, multicommodity flow, congestion game, spanning tree, matroid, submodular function, matching, partitioning, clustering, polytope, zonotope, edge direction, totally unimodular matrix, test set, Graver base, contingency table, statistical table, multiway table, disclosure control, data security, privacy, algebraic statistics, experimental design, Frobenius number, Grobner base, Hilbert scheme, zero-dimensional ideal

ISBN 978-3-03719-093-7

The Swiss National Library lists this publication in The Swiss Book, the Swiss national bibliography, and the detailed bibliographic data are available on the Internet at http://www.helveticat.ch.

Contact address:

European Mathematical Society Publishing House
Seminar for Applied Mathematics
ETH-Zentrum FLI C4
CH-8092 Zürich
Switzerland

Phone: +41 (0)44 632 34 36
Email: info@ems-ph.org
Homepage: www.ems-ph.org

Printing and binding: Druckhaus Thomas Müntzer GmbH, Bad Langensalza, Germany
∞ Printed on acid free paper
9 8 7 6 5 4 3 2 1

To Ruth, Amos and Naomi

Preface

This monograph develops an algorithmic theory of nonlinear discrete optimization. It introduces a simple and useful setup which enables the polynomial time solution of broad fundamental classes of nonlinear combinatorial optimization and integer programming problems in variable dimension. An important part of this theory is enhanced by recent developments in the algebra of Graver bases. The power of the theory is demonstrated by deriving the first polynomial time algorithms in a variety of application areas within operations research and statistics, including vector partitioning, matroid optimization, experimental design, multicommodity flows, multiindex transportation and privacy in statistical databases.

The monograph is based on twelve lectures which I gave within the *Nachdiplom Lectures* series at ETH Zürich in Spring 2009, broadly extending and updating five lectures on convex discrete optimization which I gave within the *Séminaire de Mathématiques Supérieures* series at Université de Montréal in June 2006. I thank the support of the Israel Science Foundation and the support and hospitality of ETH Zürich, Université de Montréal, and Mathematisches Forschungsinstitut Oberwolfach, where parts of the research underlying the monograph took place.

I am indebted to Jesus De Loera, Raymond Hemmecke, Jon Lee, Uriel G. Rothblum and Robert Weismantel for their collaboration in developing the theory described herein. I am also indebted to Komei Fukuda for his interest and very helpful suggestions, and to many other colleagues including my co-authors on articles cited herein and my audience at the Nachdiplom Lectures at ETH.

Finally, I am especially grateful to my wife Ruth, our son Amos, and our daughter Naomi, for their support throughout the writing of this monograph.

Haifa, May 2010 Shmuel Onn

Contents

1 Introduction

Discrete optimization problems are fundamental to every area of science and engineering. The mathematical modeling of a real-life problem as a discrete optimization problem consists of representing the feasible scenarios by integer points in some high dimensional space and the cost or utility associated with these scenarios by a real-valued function on this space. The problem is then to efficiently compute an optimal point which enables to identify an optimal scenario. The general mathematical nonlinear discrete optimization problem can be setup as follows.

Nonlinear discrete optimization problem. Given a set $S \subseteq \mathbb{Z}^n$ of integer points, an integer $d \times n$ matrix W, and a real-valued function $f\colon \mathbb{Z}^d \to \mathbb{R}$, find $x \in S$ which minimizes or maximizes the objective function $f(Wx)$, that is, solve the following:

$$\min\big\{f(Wx) : x \in S\big\} \quad \text{or} \quad \max\big\{f(Wx) : x \in S\big\}.$$

Several explanatory notes are in order here. The problem is *discrete* since the feasible points are integer. The dimension n is always a *variable* part of the input. The composite form $f(Wx)$ of the objective function results in no loss of generality: taking $d := n$ and $W := I_n$ the identity matrix, the objective becomes an arbitrary function $f(x)$ on S. While discrete optimization problems are typically computationally very hard and often intractable, this composite form enables a finer classification of efficiently solvable problems. We determine broad classes of triples S, W, f for which the problem can be solved in polynomial time (usually deterministically but sometimes randomly or approximately). The composite form is also natural and useful in modeling: the problem can be interpreted as *multicriterion* or *multiplayer* optimization, where row W_i of the matrix gives a linear function $W_i x$ representing the value of feasible point $x \in S$ under criterion i or its utility to player i, and the objective value $f(Wx) = f(W_1 x, \ldots, W_d x)$ is the "centralized" or "social" balancing of the d criteria or d player utilities.

The function f is sometimes the restriction to $\mathbb{Z}^d \subset \mathbb{R}^d$ of a function $f\colon \mathbb{R}^d \to \mathbb{R}$ defined on the entire space. We assume most of the time that f is presented by a *comparison oracle* that, queried on $x, y \in \mathbb{Z}^d$, asserts whether or not $f(x) \leq f(y)$. Therefore, we do not have a problem with the fact that the actual values of f may be real (possibly nonrational) numbers. This is a very broad presentation that reveals little information on the function and makes the problem harder to solve, but is very expressive and allows for flexibility in using the algorithms that we develop. Often, we assume that the function possesses some additional structure such as being convex or separable on the variables.

The *weight matrix* W is typically assumed to have a fixed number d of rows. In particular, with $d := 1$, $W := w \in \mathbb{Z}^n$ and f the identity on $\mathbb{R}$, the objective function $f(Wx) = wx = \sum_{i=1}^{n} w_i x_i$ is linear, which is the case considered in most literature on discrete optimization and is already hard and often intractable. The matrix W is given explicitly, and the computational complexity of the problem depends on the encoding of its entries (binary versus unary).

The computational complexity of the nonlinear discrete optimization problem most heavily depends on the presentation of the set S of feasible points. Accordingly, the algorithmic theory splits into two major branches as follows.

The first branch of the theory is ***nonlinear integer programming***, where the feasible set S consists of the integer points which satisfy a system of linear inequalities given explicitly by an integer matrix A and a right-hand side vector b:

$$S := \{x \in \mathbb{Z}^n : Ax \leq b\}.$$

The second branch of the theory is ***nonlinear combinatorial optimization***, where the feasible set $S \subseteq \{0,1\}^n$ consists of $\{0,1\}$-valued vectors and is often interpreted as the set $S = \{\mathbf{1}_F : F \in \mathcal{F}\}$ of indicators of a family $\mathcal{F} \subseteq 2^N$ of subsets of a ground set $N := \{1,\ldots,n\}$ with $\mathbf{1}_F := \sum_{j\in F} \mathbf{1}_j$, where $\mathbf{1}_j$ is the jth standard unit vector in $\mathbb{R}^n$. The set S is presented implicitly through some given compact structure or by a suitable oracle. A typical compact structure is a graph, where S is defined to be the set of indicators of subsets of edges that satisfy a given combinatorial property, such as being a matching or a forest. Typical oracles include a *membership oracle*, that queried on $x \in \{0,1\}^n$, asserts whether or not $x \in S$, and a *linear-optimization oracle*, that queried on $w \in \mathbb{Z}^n$ solves the linear optimization problem $\max\{wx : x \in S\}$ over the feasible set S.

We are interested, throughout, in the situation where the feasible set S is *finite*. This holds by definition in combinatorial optimization, where $S \subseteq \{0,1\}^n$. It also holds in most natural integer programming applications; moreover, typically the feasible set can be made finite by more careful modeling. As demonstrated in Section 1.3.3, nonlinear discrete optimization over infinite sets is quite hopeless even in one variable. Nonetheless, we do allow the input set to be infinite, and our algorithms are required to identify this situation in polynomial time as well.

Therefore, throughout this monograph, and in all formal statements, an algorithm is said to *solve a nonlinear discrete optimization problem* if, for any given S, it either finds an optimal solution $x \in S$ or asserts that S is infinite or empty.

There is a massive body of knowledge and literature on linear discrete optimization including linear combinatorial optimization and linear integer programming. But lately, there has been tremendous progress on nonlinear discrete optimization as well. The purpose of this monograph is to provide a comprehensive, unified treatment of nonlinear discrete optimization that incorporates these new developments. Our goal is twofold: first, to enable users of discrete optimization to benefit from these new developments and the recently attained polynomial time solvability of broad fundamental classes of nonlinear discrete optimization problems; second, to stimulate further research on these fascinating important classes of problems, their mathematical structure, computational complexity, and numerous applications.

1.1 Outline of the monograph

The main body of the monograph can be divided into three parts: Chapter 2 on *convex discrete maximization*, Chapters 3–5 on *nonlinear integer programming*, and Chapter 6 on *nonlinear combinatorial optimization*. The three parts, and in fact the individual chapters

as well, can be quite easily read independently of each other, just browsing now and then for relevant definitions and results as needed.

The monograph can also be divided into *theory* versus *applications*. The applications are discussed in Sections 1.2, 2.5, 4.3, 5.2, and 6.3, which can be read independently of the theoretical development. All other sections of Chapters 2–6 develop the theory and can be read independently of the applications sections.

The next introductory, Section 1.2, describes two prototypical examples of classes of combinatorial optimization and integer programming problems. These and other applications motivate the theory developed herein and are discussed in more detail and solved under various assumptions in the later applications sections. We conclude the introduction in Section 1.3 with some preliminary technical issues.

In Chapter 2, we consider *convex discrete maximization*, that is, the problem $\max\{f(Wx) : x \in S\}$ with $f\colon \mathbb{Z}^d \to \mathbb{R}$ convex. The methods used in this chapter are mostly geometric. We provide several polynomial time algorithms for convex maximization in various situations. These results apply to both combinatorial optimization and integer programming branches of our theory. The main result of this chapter is Theorem 2.16 which enables convex maximization in polynomial time using the *edge directions* of the polytope $\mathrm{conv}(S)$. We also discuss various direct applications including matroids and vector partitioning problems.

In Chapters 3–5, we study *nonlinear integer programming*, that is, optimizing a (non)linear function over a set S given by inequalities, mostly of the form:

$$S := \left\{x \in \mathbb{Z}^n : Ax = b,\ l \le x \le u\right\} \tag{1.1}$$

for some integer matrix A, right-hand side b, and $l, u \in \mathbb{Z}^n_\infty$ with $\mathbb{Z}_\infty := \mathbb{Z} \uplus \{\pm\infty\}$. The methods used here are mostly algebraic. These chapters proceed as follows.

In Chapter 3, we introduce the *Graver basis* of an integer matrix and show that it can be used to optimize in polynomial time linear and various nonlinear objective functions over sets of the form (1.1). The main result of this chapter is Theorem 3.12 which enables the polynomial time minimization of separable convex functions over sets of form (1.1). This in particular implies that the Graver basis enables linear integer programming in variable dimension in polynomial time. Combining this with the results of Chapter 2, we further show that the Graver basis also enables convex maximization over sets of form (1.1) in polynomial time.

In Chapter 4, we introduce the theory of *n-fold integer programming*. This theory, which incorporates the results of Chapter 3, enables the first polynomial time solution of very broad fundamental classes of linear and nonlinear integer programming problems in variable dimension. In particular, Theorems 4.10 and 4.12 enable, respectively, maximization and minimization of broad classes of convex functions over n-fold programs in polynomial time. In fact, as shown in Chapter 5, *every* integer program *is* an n-fold integer program. We discuss some of the numerous applications of this powerful theory including linear and nonlinear multicommodity transportation and transshipment problems. Discussion of further applications to multiway tables is postponed to Chapter 5. We also show that similar methods enable the first polynomial time solution, in Theorem 4.19, of the important and extensively studied *stochastic integer programming problem*.

In Chapter 6, we discuss *multiway tables*. Such tables occur naturally in any context involving multiply-indexed variables and are used extensively in operations research and statistics. We prove the universality Theorem 5.1 which shows that every integer program is a program over $l \times m \times 3$ tables and conclude the universality Theorem 5.12 of n-fold integer programming. These results provide powerful tools for establishing the presumable limits of polynomial time solvability of table problems. We discuss applications of the n-fold integer programming theory of Chapter 4 and the universality theorems to multiindex transportation problems and privacy in statistical databases. We also introduce and discuss the *Graver complexity* of graphs and digraphs, new important and fascinating invariants.

Finally, in Chapter 6, we discuss *nonlinear combinatorial optimization*, that is, the problem $\min\{f(Wx) : x \in S\}$ with f arbitrary and $S \subseteq \{0,1\}^n$ presented compactly or by an oracle. We solve the problem in polynomial time for several combinatorial structures S using various methods. In particular, we provide, in Theorems 6.8, 6.12, and 6.23, respectively, a deterministic algorithm for matroids, a randomized algorithm for two matroid intersections, and an approximative algorithm for independence systems. This approximation is of an unusual flavor and the quality of the approximative solution is bounded in terms of certain Frobenius numbers derived from the entries of the weight matrix W. We also establish an exponential lower bound on the running time needed to solve the problem to optimality. We conclude with some concrete applications including experimental design in statistics and universal Gröbner bases in computational algebra.

1.2 Two prototypical classes of examples

We now describe one prototypical class of examples of combinatorial optimization problems and one prototypical class of examples of integer programming problems, discussed in Sections 1.2.1 and 1.2.2, respectively. The special cases of these problems with linear objective functions are classical and had been studied extensively in the literature. The nonlinear optimization extensions are solved under various assumptions later in the monograph as applications of the theory which we develop.

1.2.1 Nonlinear matroid problems

Matroids and spanning trees

A *matroid* is a pair $M = (N, \mathcal{B})$, where N is a finite *ground set*, typically taken to be $N := \{1, \ldots, n\}$, and $\mathcal{B}$ is a nonempty family of subsets of N, called *bases* of the matroid, such that for every $B, B' \in \mathcal{B}$, and $j \in B \setminus B'$, there is a $j' \in B'$ such that $B \setminus \{j\} \cup \{j'\} \in \mathcal{B}$. All bases turn out to have the same cardinality, called the *rank* of M. A subset $I \subseteq N$ is called *independent* in the matroid if $I \subseteq B$ for some $B \in \mathcal{B}$. The family of independent sets of M is denoted by $\mathcal{I}$ and determines M.

A basic model is the *graphic matroid* of a graph $G = (V, N)$: its ground set is the set N of edges of G; its independent sets are subsets of edges forming forests; its bases are inclusion-maximal forests. In particular, if G is connected then its bases are the spanning trees. A broader model is the *vectorial matroid* of a matrix A over a field $\mathbb{F}$: its ground

set is the set N of indices of columns of A; its independent sets are subsets of indices of columns of A which are linearly independent over $\mathbb{F}$; its bases are the subsets of indices of columns of A forming bases of the column space of A. Graphic matroids are very special vectorial matroids over $\mathbb{R}$: given a graph $G = (V, N)$ with set of edges N, orient its edges arbitrarily and let D be the $V \times N$ incidence matrix of the resulting digraph, which is defined by $D_{v,e} := -1$ if edge $e \in N$ leaves vertex $v \in V$, $D_{v,e} := 1$ if e enters v, and $D_{v,e} := 0$ otherwise. Then the graphic matroid of G is precisely the vectorial matroid of D.

A matroid can be presented either through a compact given structure, such as graph or matrix for graphic or vectorial matroid, or by a suitable oracle. Two natural oracles are a *basis oracle* that, queried on $B \subseteq N$, asserts whether or not $B \in \mathcal{B}$, and an *independence oracle*, that queried on $I \subseteq N$, asserts whether or not $I \in \mathcal{I}$. Both oracles are easily realizable from a graph or matrix presentation.

The classical linear optimization problem over a matroid is the following: given matroid $M = (N, \mathcal{B})$ and weight vector $w \in \mathbb{Z}^n$, find a basis $B \in \mathcal{B}$ of maximum weight $w(B) := \sum_{j \in B} w_j$. Letting $S := \{\mathbf{1}_B : B \in \mathcal{B}\} \subseteq \{0,1\}^n$ be the set of indicators of bases, the problem can be written in the form $\max\{wx : x \in S\}$.

This classical problem can be easily solved even when the matroid is presented by an independence oracle, by the following well-known *greedy algorithm* that goes back to [32]: initialize $I := \emptyset$; while possible, pick an element $j \in N \setminus I$ of largest weight w_j such that $I := I \uplus \{j\} \in \mathcal{I}$, set $I := I \uplus \{j\}$, and repeat; output $B := I$. Further details on classical matroid theory can be found in [98].

This is a good point to illustrate the sensitivity of the complexity of a problem to the presentation of the feasible set. A basis oracle presentation of a matroid *does not* admit a polynomial time solution even with linear objective $w := 0$. Indeed, for each $B \subseteq N$ let $M_B := (N, \{B\})$ be the matroid with single basis B. Any algorithm that makes less than $2^n - 1$ oracle queries leaves at least two subsets $B, B' \subset N$ unqueried, in which case, if the oracle presents either M_B or $M_{B'}$ then it replies "no" to all queries, and the algorithm cannot tell whether the oracle presents M_B or $M_{B'}$ and hence cannot tell whether the optimal basis is B or B'.

We proceed to define the general, nonlinear, optimization problem over a matroid. The data for the problem consist of a matroid $M = (N, \mathcal{B})$, an integer $d \times n$ weight matrix W, and a function $f\colon \mathbb{Z}^d \to \mathbb{R}$. Each column W^j of W can be interpreted as vectorial utility of element $j \in N$ in the ground set, and each row W_i can be interpreted as a linear form representing the values of the ground set elements under criterion i. So $W_{i,j}$ is the value of element j under criterion i. The objective value of independent set or basis $F \subseteq N$ is the balancing $f(W(F)) := f(W\mathbf{1}_F)$ by f of the utility of F under the d given criteria. So the problem is as follows.

Nonlinear matroid optimization. Given a matroid $M = (N, \mathcal{B})$ on ground set $N := \{1, \ldots, n\}$, an integer $d \times n$ matrix W, and a function $f\colon \mathbb{Z}^d \to \mathbb{R}$, solve

$$\max\{f(Wx) : x \in S\}$$

with $S \subseteq \{0,1\}^n$ the set of (indicators of) bases or independent sets of M:

$$S := \{\mathbf{1}_B : B \in \mathcal{B}\} \quad \text{or} \quad S := \{\mathbf{1}_I : I \in \mathcal{I}\}.$$

Here is a concrete example of a nonlinear matroid optimization application.

Example 1.1 (maximum norm spanning tree, see Figure 1.1). Let d be a positive integer and $1 \leq p \leq \infty$. Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be the l_p norm $f := \|\cdot\|_p$ on $\mathbb{R}^d$ given by $\|y\|_p^p = \sum_{i=1}^d |y_i|^p$ for $1 \leq p < \infty$ and $\|y\|_\infty = \max_{i=1}^d |y_i|$. Let G be a connected graph with set of edges $N := \{1, \ldots, n\}$. Let W be an integer $d \times n$ weight matrix with $W_{i,j}$ the value of edge j under criterion i. The problem is to find a spanning tree T of G with utility vector of maximum l_p norm $\|\sum_{j \in T} W^j\|_p$ and is the nonlinear matroid optimization problem over the graphic matroid of G.

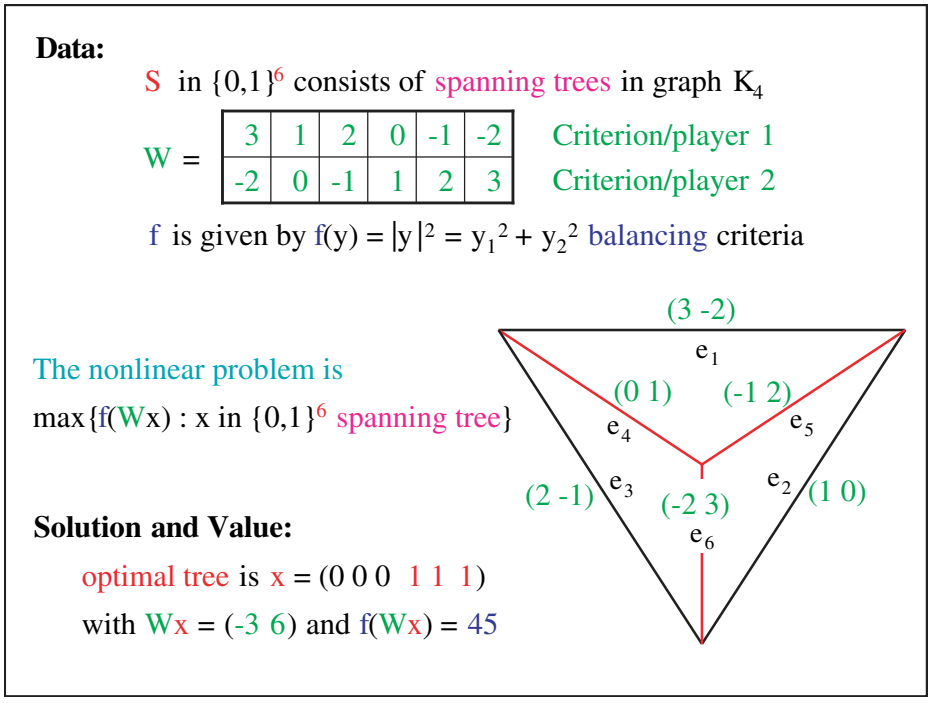


Figure 1.1: Maximum norm spanning tree example

The nonlinear matroid optimization problem is solved in Section 2.5.2 for convex f and under suitable assumptions in Section 6.1.2 for arbitrary f. This in particular applies to nonlinear spanning tree problems as in Example 1.1. One concrete application area is in model fitting in experimental design [10] and is discussed in Section 6.3.2. Another useful application is a polynomial time algorithm for computing the *universal Gröbner basis* of any system of polynomials with a finite set of common zeros in fixed number of variables [6], [85] and is discussed in Section 6.3.3.

Matroid intersections and independence systems

We proceed to introduce two broad extensions of nonlinear matroid optimization.

Nonlinear matroid intersection. Given k matroids $M_i = (N, \mathcal{B}_i)$ on common n element ground set N, integer $d \times n$ matrix W, and function $f\colon \mathbb{Z}^d \to \mathbb{R}$, solve

$$\max\big\{f(Wx) : x \in S\big\}$$

with $S \subseteq \{0,1\}^n$ the set of common bases or common independent sets:

$$S := \big\{\mathbf{1}_B : B \in \mathcal{B}_1 \cap \cdots \cap \mathcal{B}_k\big\} \quad \text{or} \quad S := \big\{\mathbf{1}_I : I \in \mathcal{I}_1 \cap \cdots \cap \mathcal{I}_k\big\}.$$

For $k \geq 3$, even the linear problem is hard: the NP-hard *traveling salesman problem* is reducible to linear three-matroid intersection, see Section 2.5.2.

For $k = 2$, the nonlinear (two) matroid intersection problem is solved under suitable assumptions in Section 2.5.2 for convex f and in Section 6.1.3 for arbitrary f.

The set of common independent sets of several matroids is a special case of the following generic monotonically closed down structure. An *independence system* (sometimes termed *simplicial complex*) is a nonempty set $S \subseteq \{0,1\}^n$ such that $z \in \{0,1\}^n$ and $z \leq x \in S$ imply $z \in S$. We also consider the following problem.

Nonlinear independence system optimization. Given independence system $S \subseteq \{0,1\}^n$, integer $d \times n$ matrix W, and $f : \mathbb{Z}^d \to \mathbb{R}$, solve $\max\{f(Wx) : x \in S\}$.

This is a very broad problem – any reasonable set of $\{0,1\}$-vectors can be closed down to become an independence system – and so is very hard to solve. In Section 6.2, we provide, under suitable restrictions, an approximative solution to this problem whose quality is bounded by certain Frobenius numbers derived from the entries of W and show that finding a true optimal solution requires exponential time.

1.2.2 Nonlinear multicommodity flows

Multiindex transportation problems

The classical *transportation problem* concerns the minimum cost routing of a discrete commodity subject to supply, demand, and channel capacity constraints. The data for the problem is as follows. There are m suppliers and n consumers. Supplier i supplies s_i units, and consumer j consumes c_j units. For each supplier i and consumer j, there is a capacity (upper bound) $u_{i,j}$ on the number of units that can be routed from i to j and a cost $w_{i,j}$ per unit routed from i to j. A *transportation* is a nonnegative integer $m \times n$ matrix x, with $x_{i,j}$ the number of units to be routed from i to j, that satisfies the capacity constraints $x_{i,j} \leq u_{i,j}$ and the supply and consumption constraints $\sum_{j=1}^n x_{i,j} = s_i$, $\sum_{i=1}^m x_{i,j} = c_j$ for all i, j. So the set of feasible transportations is the set of nonnegative integer matrices with row sums s_i, column sums c_j, and entry upper bounds $u_{i,j}$, given by

$$S := \big\{x \in \mathbb{Z}_+^{m\times n} : \textstyle\sum_{j=1}^n x_{i,j} = s_i,\ \sum_{i=1}^m x_{i,j} = c_j,\ x_{i,j} \leq u_{i,j}\big\}. \tag{1.2}$$

The transportation problem is to find a transportation x of minimum total cost $wx := \sum_{i=1}^m \sum_{j=1}^n w_{i,j}x_{i,j}$, that is, the linear integer programming problem:

$$\min\big\{wx : x \in \mathbb{Z}_+^{m\times n},\ \textstyle\sum_j x_{i,j} = s_i,\ \sum_i x_{i,j} = c_j,\ x_{i,j} \leq u_{i,j}\big\}. \tag{1.3}$$

It is well known [54] that the matrix defining the system of inequalities in (1.3) is *totally unimodular*, implying that the underlying polytope is *integer*, that is,

$$\begin{aligned}\operatorname{conv}(S) &= \operatorname{conv}\left\{x \in \mathbb{Z}_+^{m\times n} : \textstyle\sum_j x_{i,j} = s_i,\ \sum_i x_{i,j} = c_j,\ x_{i,j} \le u_{i,j}\right\}\\ &= \left\{x \in \mathbb{R}_+^{m\times n} : \textstyle\sum_j x_{i,j} = s_i,\ \sum_i x_{i,j} = c_j,\ x_{i,j} \le u_{i,j}\right\}.\end{aligned} \tag{1.4}$$

Since the minimum of a linear function over a polytope is attained at a vertex, (1.4) implies that problem (1.3) can be solved in polynomial time by *linear programming* [59], [87] (see Section 2.3.4 for a more detailed discussion of totally unimodular systems).

We proceed to discuss a fundamental and much more difficult extension of the problem. The *multiindex* transportation problem, introduced by Motzkin in [75], is the problem of finding a minimum cost *multiindexed* nonnegative integer array $x = (x_{i_1,\dots,i_d})$ with specified sums over some of its lower dimensional subarrays (termed *margins* in statistics). For simplicity, we discuss now only the case of triple-index problems with line-sum constraints and postpone discussion of the general case to Section 5.2.1. The data for the triple-index, line-sum problem of format $l \times m \times n$ consists of $mn + ln + lm$ line sums, that is, nonnegative integer numbers:

$$v_{*,j,k}, \quad v_{i,*,k}, \quad v_{i,j,*}, \quad 1 \le i \le l,\ 1 \le j \le m,\ 1 \le k \le n,$$

replacing the supplies and consumptions of the classical problem, and an integer $l \times m \times n$ cost array w. The problem is the linear integer programming problem:

$$\min\left\{wx : x \in \mathbb{Z}_+^{l\times m\times n},\ \textstyle\sum_i x_{i,j,k} = v_{*,j,k},\ \sum_j x_{i,j,k} = v_{i,*,k},\ \sum_k x_{i,j,k} = v_{i,j,*}\right\}.$$

The matrix which defines the system of inequalities of the triple-index transportation problem is *not* totally unimodular. Therefore, the underlying polytope is typically not integer, and, as the next example shows, we have strict containment:

$$\begin{aligned}&\operatorname{conv}\left\{x \in \mathbb{Z}_+^{l\times m\times n},\ \textstyle\sum_i x_{i,j,k} = v_{*,j,k},\ \sum_j x_{i,j,k} = v_{i,*,k},\ \sum_k x_{i,j,k} = v_{i,j,*}\right\}\\ &\subsetneq \left\{x \in \mathbb{R}_+^{l\times m\times n},\ \textstyle\sum_i x_{i,j,k} = v_{*,j,k},\ \sum_j x_{i,j,k} = v_{i,*,k},\ \sum_k x_{i,j,k} = v_{i,j,*}\right\}.\end{aligned}$$

Example 1.2 (real-feasible integer-infeasible tri-index transportation). Consider the $6 \times 4 \times 3$ transportation problem with the following line sums:

$$\left(v_{i,j,*}\right) = \begin{pmatrix}1&0&0&1\\0&1&1&0\\1&1&0&0\\0&0&1&1\\1&0&1&0\\0&1&0&1\end{pmatrix}, \quad \left(v_{i,*,k}\right) = \begin{pmatrix}1&1&0\\1&1&0\\1&0&1\\1&0&1\\0&1&1\\0&1&1\end{pmatrix}, \quad \left(v_{*,j,k}\right) = \begin{pmatrix}1&1&1\\1&1&1\\1&1&1\\1&1&1\end{pmatrix}.$$

It can be shown that the following fractional point is the unique feasible one:

$$(x_{i,j,1}) = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad (x_{i,j,2}) = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix},$$

$$(x_{i,j,3}) = \frac{1}{2}\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

As suggested by Example 1.2 and the preceding discussion, the linear multiindex problem is NP-hard and hence presumably computationally intractable. In fact, in Section 5.1 we show that the problem already over $l \times m \times 3$ arrays is *universal*: every integer programming problem *is* an $l \times m \times 3$ transportation problem.

More generally, we consider the nonlinear multiindex problem stated as follows (with the precise definition of a list of hierarchical margins postponed to Section 5.2.3).

Nonlinear multiindex transportation problem. Given list v of hierarchical integer margins for $m_1 \times \cdots \times m_d$ arrays and a function $f\colon \mathbb{Z}^{m_1 \times \cdots \times m_d} \to \mathbb{R}$, solve

$$\min\left\{f(x) : x \in \mathbb{Z}_+^{m_1 \times \cdots \times m_d},\ x \text{ has the given margins } v\right\}.$$

In spite of the hardness of even the linear problem indicated above, we solve the (non)linear multiindex problem under suitable assumptions in Sections 5.2.1 and 5.2.3.

Multicommodity transshipment problems

Another broad extension of the transportation problem is the *multicommodity transshipment problem*. This is a very general flow problem which seeks minimum cost routing of several discrete commodities over a digraph subject to vertex demand and edge capacity constraints. The problem data is as follows (see Figure 1.2 for a trivial example). There is a digraph G with s vertices and t edges. There are l types of commodities. Each commodity has a demand vector $d^k \in \mathbb{Z}^s$ with d_v^k, the demand for commodity k at vertex v (interpreted as supply when positive and consumption when negative). Each edge e has a capacity u_e (upper bound on the combined flow of all commodities on it). A *multicommodity transshipment* is a vector $x = (x^1, \ldots, x^l)$ with $x^k \in \mathbb{Z}_+^t$ for all k and x_e^k the flow of commodity k on edge e, satisfying the capacity constraint $\sum_{k=1}^{l} x_e^k \le u_e$ for each edge e and demand constraint $\sum_{e\in\delta^+(v)} x_e^k - \sum_{e\in\delta^-(v)} x_e^k = d_v^k$ for each vertex v and commodity k (with $\delta^+(v), \delta^-(v)$ the sets of edges entering and leaving vertex v).

The cost of transshipment x is defined as follows. There are cost functions $f_e, g_e^k : \mathbb{Z} \to \mathbb{Z}$ for each edge and each edge-commodity pair. The transshipment cost on edge e is

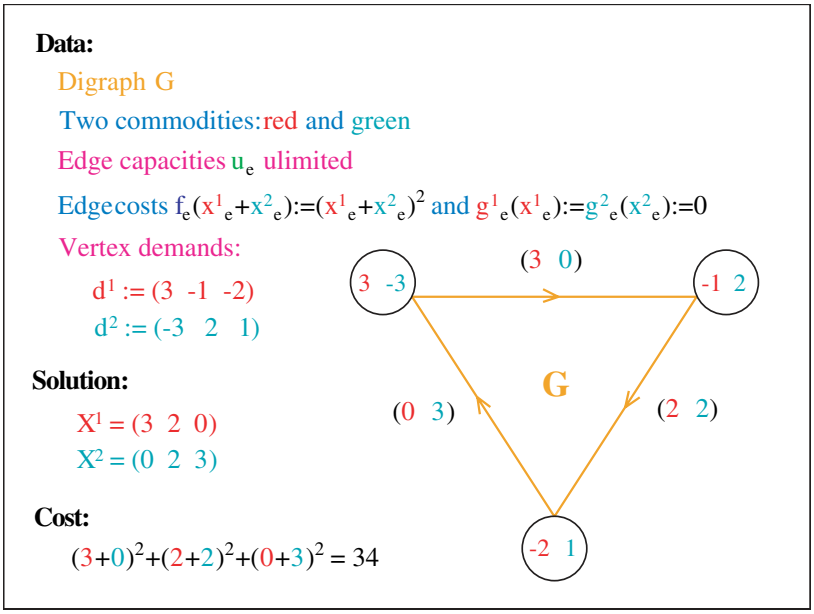


Figure 1.2: Multicommodity transshipment example

$f_e(\sum_{k=1}^{l} x_e^k) + \sum_{k=1}^{l} g_e^k(x_e^k)$ with the first term being the value of f_e on the combined flow of all commodities on e, and the second term being the sum of costs that depends on both the edge and the commodity. The total cost is

$$\sum_{e=1}^{t} \Big(f_e\Big(\sum_{k=1}^{l} x_e^k \Big) + \sum_{k=1}^{l} g_e^k\big(x_e^k\big) \Big).$$

The cost can in particular be convex such as $\alpha_e |\sum_{k=1}^{l} x_e^k|^{\beta_e} + \sum_{k=1}^{l} \gamma_e^k |x_e^k|^{\delta_e^k}$ for some nonnegative integers $\alpha_e, \beta_e, \gamma_e^k, \delta_e^k$, which takes into account the increase in cost due to channel congestion when subject to heavy traffic or communication load [88] (with the standard linear special case obtained by $\beta_e = \delta_e^k = 1$).

So we have the following very general nonlinear multicommodity flow problem.

Nonlinear multicommodity transshipment problem. Given a digraph G with s vertices and t edges, l commodity types, demand $d_v^k \in \mathbb{Z}$ for each commodity k and vertex v, edge capacities $u_e \in \mathbb{Z}_+$, and cost functions $f_e, g_e^k \colon \mathbb{Z} \to \mathbb{Z}$, solve

$$\min \sum_{e} \Big(f_e\Big(\sum_{k=1}^{l} x_e^k \Big) + \sum_{k=1}^{l} g_e^k\big(x_e^k\big) \Big)$$

$$\text{subject to} \quad x_e^k \in \mathbb{Z}_+, \quad \sum_{e \in \delta^+(v)} x_e^k - \sum_{e \in \delta^-(v)} x_e^k = d_v^k, \quad \sum_{k=1}^{l} x_e^k \le u_e.$$

This problem, already with linear costs, is very difficult. It is NP-hard for two commodities over the bipartite digraphs $K_{m,n}$ (oriented from one side to the other) and for variable number l of commodities over $K_{3,n}$. Nonetheless, we do solve the (non)linear problem in polynomial time in two broad situations in Sections 4.3.1 and 4.3.2. In particular, our theory provides the first solution for the linear problem with two commodities over $K_{3,n}$ and with l commodities over the tiny graph $K_{3,3}$.

1.3 Notation, complexity, and finiteness

We conclude our introduction with some notation and preliminary technical issues.

1.3.1 Notation

We use $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{Z}$, $\mathbb{Z}_+$, for the reals, nonnegative reals, integers, and nonnegative integers, respectively. We use $\mathbb{R}_\infty := \mathbb{R} \uplus \{\pm\infty\}$ and $\mathbb{Z}_\infty := \mathbb{Z} \uplus \{\pm\infty\}$ for the extended reals and integers. The absolute value of a real number r is denoted by $|r|$ and its sign by $\mathrm{sign}(r) \in \{-1, 0, 1\}$. The ith standard unit vector in $\mathbb{R}^n$ is denoted by $\mathbf{1}_i$. We use $\mathbf{1} := \sum_{i=1}^n \mathbf{1}_i$ for the vector with all entries equal to 1. The *support* of $x \in \mathbb{R}^n$ is the index set $\mathrm{supp}(x) := \{i : x_i \neq 0\}$ of nonzero entries of x. The *indicator* of subset $I \subseteq N := \{1, \ldots, n\}$ is the vector $\mathbf{1}_I := \sum_{i \in I} \mathbf{1}_i$, so $\mathrm{supp}(\mathbf{1}_I) = I$. Vectors are typically regarded as columns unless they are rows of a matrix or otherwise specified. When vectors in a list are indexed by subscripts $w_i \in \mathbb{R}^n$, their entries are indicated by pairs of subscripts, as $w_i = (w_{i,1}, \ldots, w_{i,n})$. When vectors in a list are indexed by superscripts $x^j \in \mathbb{R}^n$, their entries are indicated by subscripts, as $x^j = (x^j_1, \ldots, x^j_n)$. 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 functions on $\mathbb{R}^n$ via the inner product wx. Thus, we refer to elements of $\mathbb{R}^n$ as points, vectors, or linear functions, as is appropriate from the context. The l_p norm on $\mathbb{R}^n$ is defined by $\|x\|_p^p := \sum_{i=1}^n |x_i|^p$ for $1 \leq p < \infty$ and $\|x\|_\infty := \max_{i=1}^n |x_i|$. The rows of a matrix W are denoted by W_i, the columns by W^j, and the entries by $W_{i,j}$. The inner product of matrices lying in the same matrix space is $W \cdot X := \sum_i \sum_j W_{i,j} X_{i,j}$. For matrix W, we use $\|W\|_\infty := \max_{i,j} |W_{i,j}|$. Additional, more specific notation is introduced wherever needed, recalled in later occurrences and appropriately indexed.

1.3.2 Complexity

Explicit numerical data processed by our algorithms is assumed to be rational, and hence algorithmic time complexity is as in the standard Turing machine model, see, for example [55], [64]. When numerical data is used implicitly, such as in the case of a function f presented by a comparison oracle, whose precise values are irrelevant, we can and are being sloppy about whether these values are rationals or reals.

The input to our algorithms typically consists of integer numbers, vectors, matrices, and finite sets of such objects. The *binary length* of an integer $z \in \mathbb{Z}$ is the number of bits in its binary encoding, which is $\langle z \rangle := 1 + \lceil \log_2(|z| + 1) \rceil$ with the extra bit for the sign. The length of a rational number presented as a fraction $r = \frac{p}{q}$ 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 its entries, that is, $\langle A\rangle \geq mn$. Thus, $\langle A\rangle$ already accounts for mn and hence we usually do not account for m, n separately. Yet, sometimes, especially in results related to n-fold integer programming, we do emphasize n as part of the input. Similarly, the length of a finite set E of numbers, vectors or matrices, is the sum of lengths of its elements, and hence, since $\langle E\rangle \geq |E|$, accounts for its cardinality as well.

Sometimes we assume part of the input is encoded in unary. The *unary length* of an integer $z \in \mathbb{Z}$ is the number $|z| + 1$ of bits in its unary encoding, again, with an extra bit for the sign. The unary length of rational number, vector, matrix, or finite sets of such objects is defined again as the sums of lengths of their numerical constituents and is again no smaller than the number of such constituents.

Both binary and unary lengths of any $\pm\infty$ entry of any lower or upper bound vector l, u over the set $\mathbb{Z}_\infty = \mathbb{Z} \uplus \{\pm\infty\}$ of extended integers are constant.

An algorithm is *polynomial time* if its running time is polynomial in the length of the input. In every formal algorithmic statement, we indicate the length of the input by explicitly listing every input object and indicating if it affects the running time through its unary length or binary length. For example, saying that "an algorithm runs in time polynomial in W and $\langle A, b\rangle$", where W is a weight matrix and A, b define the feasible set S through an inequality system, means that the time is polynomial in the unary length of W and the binary length of A, b.

Often, as in [44], [72], parts of the input, such as the feasible set S or the objective function f, are presented by oracles. The running time then counts also the number of oracle queries. An oracle algorithm is *polynomial time* if its running time, including the number of oracle queries and the length of manipulated numbers including answers to oracle queries, is polynomial in the input length.

1.3.3 Finiteness

We typically assume that the objective function f in a nonlinear discrete optimization problem is presented by a mere comparison oracle. Under such broad presentation, if the feasible set S is infinite then the problem is quite hopeless even in dimension $n = 1$. To see this, consider the following family of simple univariate nonlinear integer programs with convex functions f_u parameterized by $0 \leq u \leq \infty$ as follows:

$$\max\big\{f_u(x) : x \in \mathbb{Z}_+\big\}, \quad f_u(x) := \begin{cases} -x & \text{if } x < u, \\ x - 2u & \text{if } x \geq u. \end{cases}$$

Consider any algorithm attempting to solve the problem and let u be the maximum value of x in all queries made by the algorithm to the oracle of f. Then the algorithm cannot distinguish between the problem with f_u having unbounded objective values and the problem with f_∞ having optimal objective value 0.

So as already noted, we are interested in the situation where the set S is finite. We define the *radius* $\rho(S)$ of a set $S \subseteq \mathbb{Z}^n$ to be its l_∞ radius, which is given by

$$\rho(S) := \sup\big\{\|x\|_\infty : x \in S\big\} \quad \text{with} \quad \|x\|_\infty := \max\big\{\big|x_i\big| : i = 1, \ldots, n\big\}.$$

So $\rho(S)$ is the smallest $\rho \in \mathbb{Z}_\infty$ for which the box $[-\rho, \rho]^n$ contains S. When dealing with arbitrary, oracle presented, sets $S \subseteq \mathbb{Z}^n$, mostly in Chapter 2, the radius may affect the running time of some algorithms, but we *do not* require that it is an explicit part of the input, and get along without knowing it in advance.

In combinatorial optimization, with $S \subseteq \{0,1\}^n$, we always have $\rho(S) \leq 1$. In integer programming, with $S = \{x \in \mathbb{Z}^n : Ax \leq b\}$ given by inequalities, mostly in standard form $S = \{x \in \mathbb{Z}^n : Ax = b,\ l \leq x \leq u\}$, the binary length of $\rho(S)$ is polynomially bounded in the binary length $\langle A, b, l, u\rangle$ of the data by Cramer's rule, see, for example [90]. Therefore, in these contexts, the radius is already polynomial in the data and does not play a significant role in Chapters 3–5 on integer programming and in Chapter 6 on combinatorial optimization.

Finally, we note again that, throughout, and in all formal statements, an algorithm is said to *solve a nonlinear discrete optimization problem* if, for any given S, it either finds an optimal solution $x \in S$, or asserts that S is infinite or empty.

Notes

Background on the classical theory of linear integer programming can be found in the book [90] by Schrijver. More recent sources on integer programming containing also material on nonlinear optimization and on *mixed integer* programming, where some of the variables are integer and some are real, are the book [14] by Bertsimas and Weismantel and survey [49] by Hemmecke, Köppe, Lee, and Weismantel. Development of an algorithmic theory of integer programming in *fixed dimension* using generating functions can be found in the book [9] by Barvinok. The algorithmic theory of integer programming in variable dimension developed here has some of its origins in the work of Sturmfels described in his book [95]. Among the numerous sources on cutting methods for integer programming, let us mention the classical paper [18] by Chvátal on Gomory cuts, the paper [73] by Lovász and Schrijver and survey [65] by Laurent and Rendl on more recent semidefinite cutting methods, and the survey [21] by Cornuéjols on cutting methods for mixed integer programming. Background on the classical theory of linear combinatorial optimization can be found in the trilogy [91] by Schrijver. Geometric development of the algorithmic equivalence of separation and optimization via the ellipsoid method and its many applications in combinatorial optimization can be found in the books [44] by Grötschel, Lovász and Schrijver, and [72] by Lovász.

Let us note that many of the polynomial time algorithms that result from the theory developed in this monograph have running times which are polynomials of very large degree. Therefore, an important role of our theory is to enable to identify that a (non)linear discrete optimization problem can be at all solved in polynomial time. Then there is hope that a more efficient, ad-hoc algorithm can be designed for such a problem. In particular, there is much room for improvements in the polynomial running times for some of the many applications discussed herein.

2 Convex Discrete Maximization

In this chapter, we study the *convex discrete maximization problem*, namely,

$$\max\left\{f(Wx) : x \in S\right\} \tag{2.1}$$

with *convex* function $f : \mathbb{Z}^d \to \mathbb{R}$, integer $d \times n$ matrix W, and set $S \subseteq \mathbb{Z}^n$.

While the dimension n is variable as always, the number d of rows of W, which may represent d criteria or player utilities, is fixed. Note that even for fixed $d = 1$ the problem includes linear integer programming which is generally NP-hard.

In most of this chapter, we assume that we can do linear optimization over S to begin with. So we assume that $S \subseteq \mathbb{Z}^n$ is presented by a *linear-optimization oracle* that, queried on $w \in \mathbb{Z}^n$, solves the linear optimization problem $\max\{wx : x \in S\}$ over S, that is, either finds an optimal solution or asserts that S is infinite or empty.

The geometry of the set S in $\mathbb{R}^n$ and its *image* $WS := \{Wx : x \in S\}$ under the projection by W in $\mathbb{R}^d$ plays a central role here. In Section 2.1, we discuss this image and the *fibers* of points under the projection by W, outline a general strategy for convex maximization, and demonstrate the difficulties involved. In Section 2.2, we consider the simpler situation, where the radius $\rho(S)$ of the feasible set and the weight matrix W are unary encoded, and solve the problem using the ellipsoid method. In Section 2.3, we consider the more involved situation, where $\rho(S)$ and W are binary encoded. In Section 2.3.1, we discuss necessary preliminaries on *edge directions* and *zonotopes*. We proceed in Section 2.3.2 to prove Theorem 2.16 which is the main result of this chapter and uses edge directions of $\mathrm{conv}(S)$ to solve the convex maximization problem over S in polynomial time. In Section 2.3.3, we reconsider the case of unary-encoded data and provide an alternative solution, using the algorithm of Theorem 2.16 based on zonotopes, which avoids the ellipsoid method. In Section 2.3.4, we apply the algorithm of Theorem 2.16 for efficient convex maximization over totally unimodular systems using the *circuits* of the underlying matrix. In Section 2.4, we focus on combinatorial optimization and use Theorem 2.16 to show that convex maximization over a set $S \subseteq \{0,1\}^n$ can be done in polynomial time even when S is presented by a mere membership oracle. We conclude in Section 2.5 with some direct applications to vector partitioning and nonlinear matroid and matroid intersection problems. The results of this chapter are incorporated in Chapters 3–5 and enable efficient convex integer programming with many more applications discussed in Sections 4.3 and 5.2.

The following table enables quick navigation among some of the theorems in this chapter that provide polynomial time solution of the convex maximization problem (2.1). Additional results are in the applications, Section 2.5. The first row indicates assumptions on the data ($S \subseteq \mathbb{Z}^n$ or $S \subseteq \{0,1\}^n$, linear-optimization oracle or membership oracle, edge-directions given or not), and the second row indicates the dependency of the running time on the radius $\rho(S)$ and matrix W.

$S \subseteq \mathbb{Z}^n$ Linear-optimization oracle	$S \subseteq \mathbb{Z}^n$ Linear-optimization oracle Edge directions given	$S \subseteq \{0,1\}^n$ Membership oracle Edge directions given
Theorem 2.10 Polynomial in $\rho(S), W$	Theorem 2.16 Polynomial in $\langle \rho(S), W \rangle$	Theorem 2.22 Polynomial in $\langle W \rangle$

2.1 Image and fibers

Consider the general nonlinear discrete optimization problem with $S \subset \mathbb{Z}^n$ finite, W integer $d \times n$ matrix and $f : \mathbb{Z}^d \to \mathbb{Z}$ function presented by comparison oracle:

$$\max \big\{ f(Wx) : x \in S \big\}. \tag{2.2}$$

We can regard W as the linear projection $W : \mathbb{R}^n \to \mathbb{R}^d$ that maps x to $y := Wx$. Define the *image* of S under the projection W to be the finite set as follows:

$$WS := \{Wx : x \in S\} \subset \mathbb{Z}^d.$$

Define the *fiber* of $y \in \mathbb{R}^d$ to be its preimage under the projection W, given by

$$W^{-1}(y) := \big\{ x \in \mathbb{R}^n : Wx = y \big\}.$$

These definitions suggest the following naïve strategy for solving our problem (2.2).

Procedure 2.1 (naïve strategy for nonlinear discrete optimization).

1. Compute the image $WS = \{Wx : x \in S\}$.
2. Determine a point $y \in WS$ maximizing f over WS.
3. Find and output a feasible point $x \in W^{-1}(y) \cap S$ in the fiber of y.

Obviously, any point x obtained that way is an optimal solution to problem (2.2).

Unfortunately, steps 1 and 3 are very hard. In particular, the number of image points may be exponential in the input length, in which case the image cannot be even written down, let alone computed, efficiently. The problems in steps 1 and 3 are closely related to the following problem which is of interest in its own right.

Fiber problem. Given set $S \subseteq \mathbb{Z}^n$, integer $d \times n$ matrix W, and point $y \in \mathbb{Z}^d$, find $x \in W^{-1}(y) \cap S$, that is, $x \in S$ satisfying $Wx = y$, or assert that none exists.

The fiber problem is generally very difficult. Even deciding if $W^{-1}(y) \cap S \neq \emptyset$ is often NP-complete for S presented by inequalities and often requires exponentially many queries when S is presented by an oracle. The fiber problem for arbitrary f is studied further and solved in several situations by various ways in Chapter 6.

We now restrict attention to the case of convex f, which is the subject of the present chapter. Consider the convex hull of S and that of the image WS:

$$P := \text{conv}(S) \subset \mathbb{R}^n, \quad Q := \text{conv}(WS) = WP \subset \mathbb{R}^d.$$

Since S and WS are finite, both P and Q are convex polytopes. Let $\text{vert}(P) \subseteq S$ and $\text{vert}(Q) \subseteq WS$ be their vertex sets. The next lemma shows that problem (2.2) with convex f always has an optimal solution in the fiber of some vertex of Q.

Lemma 2.2. *Suppose v is a vertex of $\text{conv}(WS)$ maximizing a convex function f. Then $F := W^{-1}(v) \cap S \neq \emptyset$ and any $x \in F$ is optimal for $\max\{f(Wx) : x \in S\}$.*

Proof. Let $P := \text{conv}(S)$ and $Q := \text{conv}(WS) = WP$. Since the function $f(y)$ is convex on $\mathbb{R}^d$ and the composite function $f(Wx)$ is convex on $\mathbb{R}^n$, we have

$$\begin{aligned}\max\left\{f(Wx) : x \in S\right\} &= \max\left\{f(Wx) : x \in P\right\}\\ &= \max\left\{f(y) : y \in Q\right\}\\ &= \max\left\{f(v) : v \in \text{vert}(Q)\right\}.\end{aligned}$$

Let $v \in \text{vert}(Q)$ be any vertex attaining the maximum on the right-hand side. Since $v \in WS$, we have $F := W^{-1}(v) \cap S \neq \emptyset$. Moreover, any point $x \in F$ satisfies $f(Wx) = f(v)$ and hence attains the maximum value and is an optimal solution. □

Lemma 2.2 suggests the following strategy for convex discrete maximization.

Procedure 2.3 (strategy for convex discrete maximization).

1. Compute the set $V := \text{vert}(\text{conv}(WS))$ of vertices of the image.
2. Determine a point $v \in V$ maximizing f over V.
3. Find and output a feasible point $x \in W^{-1}(v) \cap S$ in the fiber of v.

In general, steps 1 and 3 of this variant of the naïve strategy remain difficult. In particular, the number of vertices of $\text{conv}(WS)$ may be exponential in the input length even in dimension $d = 2$. So again, writing all vertices down, let alone computing them, cannot usually be done in polynomial time. We remark that for polyhedra it is generally impossible to compute the vertices even in time polynomial in the *output*, that is, in the number of vertices [60]. However, in this chapter we do overcome these difficulties and, following the general outline of this strategy, solve the problem in polynomial time in two different broad situations.

We note that for the nonlinear discrete optimization problem with arbitrary, not necessarily convex, function f, the set of vertices of $\text{conv}(WS)$ is not enough and the entire image WS is needed. While the vertex set, which is easier to compute, does give some information on the image, this information is generally not enough to determine the entire image, since in general we have a strict containment as follows:

$$WS \subsetneq \text{conv}(WS) \cap \mathbb{Z}^d.$$

Indeed, there may be *holes* in $\text{conv}(WS)$, that is, integer points whose fibers do not contain any feasible point. So when solving in later chapters nonlinear optimization problems other than convex maximization, we have to use other methods.

We conclude this section with an example illustrating the notions discussed above.

Example 2.4. Let S be the set of $m \times m$ permutation matrices, which are the feasible points in the classical *assignment problem*. It has the inequality description:

$$S = \{x \in \mathbb{Z}_+^{m\times m} : \sum_{j=1}^m x_{i,j} = 1,\ \sum_{i=1}^m x_{i,j} = 1\}. \tag{2.3}$$

The linear optimization problem over S is a special case of the classical transportation problem (1.2). As explained in Section 1.2.2, it is polynomial time solvable by linear programming and so a linear-optimization oracle for S is readily available.

We note that even for this simple set S whose convex hull is just the set of bistochastic matrices, the fiber problem for $d = 2$ and $\{0, 1\}$-valued W includes the so-called *exact bipartite matching problem* whose complexity is long open [12], [76].

Consider a specific small example (see Figure 2.1) with $m = 3$, $d = 2$, weight

$$W = (W^k_{i,j}) \in \mathbb{Z}^{2\times 9}, \quad \text{where } W^1 := \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix},\ W^2 := \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

and separable convex function $f : \mathbb{R}^2 \to \mathbb{R}$ given by $f(y) := (3y_1 - 4)^2 + (3y_2 - 4)^2$. We write W^1, W^2 and points $x \in S$ as 3×3 matrices but regard them as vectors in $\mathbb{R}^9$. So the image of $x \in S$ is $Wx := (W^1 \cdot x, W^2 \cdot x)$ with $W^k \cdot x := \sum_{i,j} W^k_{i,j} x_{i,j}$. The

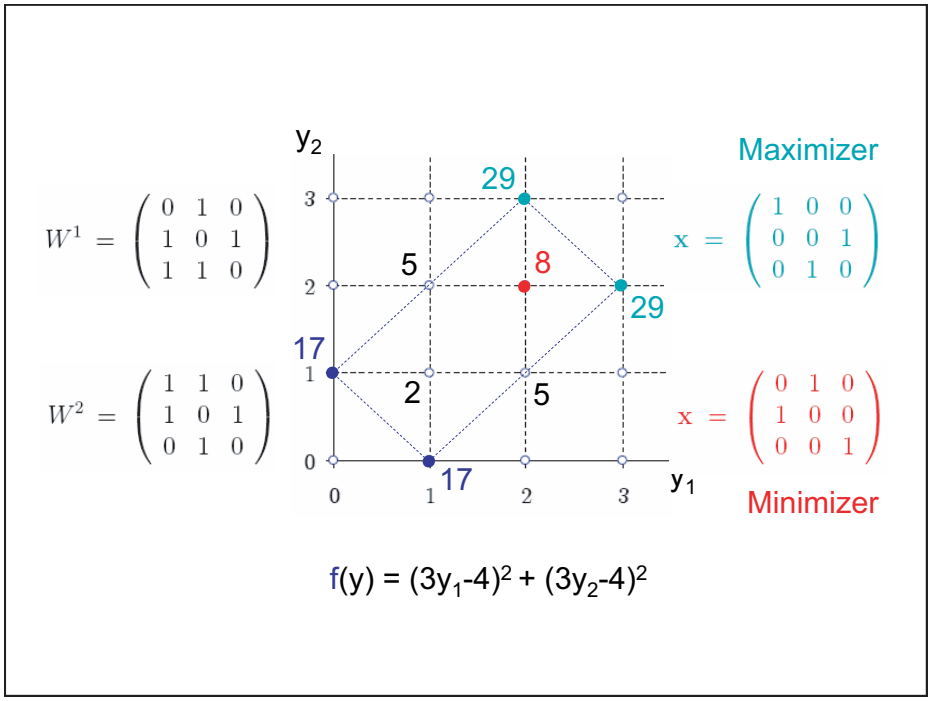


Figure 2.1: Example of the naïve strategy for nonlinear discrete optimization

feasible set S consists of 6 matrices (in general $|S| = m!$ is exponential):

$$S = \left\{ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \right\}.$$

The image consists of 5 points (designated by colored points in Figure 2.1),

$$WS = \left\{ \left(W^1 \cdot x, W^2 \cdot x\right) : x \in S \right\} = \left\{ (1,0), (0,1), (2,2), (3,2), (2,3) \right\}. \tag{2.4}$$

The convex hull $\text{conv}(WS)$ of the image is indicated in Figure 2.1 by a dashed blue line. It contains 8 integer points - the 5 image points and 3 holes $(1,1), (2,1), (1,2)$. The objective function value $f(y) = (3y_1 - 4)^2 + (3y_2 - 4)^2$ is indicated for each of these 8 integer points. The maximum objective value 29 is attained at the two green vertices $(3,2), (2,3)$ of $\text{conv}(WS)$. Therefore, for instance, the point:

$$x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

in the fiber $W^{-1}(2,3)$ of vertex $(2,3)$ is optimal for $\max\{f(Wx) : x \in S\}$. The minimum objective value 2 over the integer points in $\text{conv}(WS)$ is attained at the point $(1,1)$ which lies in the interior of the polytope. However, this point is a hole, whose fiber $W^{-1}(1,1)$ contains no feasible point, as are the points $(2,1), (1,2)$ attaining the next smallest value 5. So the minimum value of a point in the image is only 8 and is attained by the red point $(2,2)$. Therefore, for instance, the point:

$$x = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

which is in the fiber $W^{-1}(2,2)$ an optimal solution for $\min\{f(Wx) : x \in S\}$.

2.2 Small radius and weight

In this section, we consider the simpler situation where the radius $\rho(S)$ of the set $S \subseteq \mathbb{Z}^n$ and the integer $d \times n$ weight matrix W are assumed to be unary encoded.

The main result here is Theorem 2.10: for every fixed d, we can maximize any convex composite function $f(Wx)$ over S in time polynomial in $\rho(S)$ and W.

Here, we solve the problem using the ellipsoid method. In Section 2.3.3, we give an alternative solution which uses the results of Section 2.3 based on zonotopes instead.

We begin with the following simple useful lemma which shows that a linear-optimization oracle for S allows to compute the radius $\rho(S)$. Therefore, in the algorithms to follow, we do not need $\rho(S)$ to be an explicit part of the input.

Lemma 2.5. *Using $2n$ queries of a linear-optimization oracle presenting $S \subseteq \mathbb{Z}^n$, it is possible to either conclude that S is infinite or empty or find its radius $\rho(S)$.*

Proof. By querying the oracle on $w := \pm\mathbf{1}_i$ for $i = 1, \ldots, n$, solve the $2n$ problems:

$$\min / \max \{x_i : x \in S\}, \quad i = 1, \ldots, n.$$

If the oracle returns an optimal solution for each of these problems then the maximum absolute value among the optimal objective function values of these $2n$ problems is the radius $\rho(S)$ of S. Otherwise, the set S is infinite or empty. □

Let $r := n\rho(S)\|W\|_\infty$ with $\|W\|_\infty = \max_{i,j} |W_{i,j}|$. Define $l, u \in \mathbb{Z}^d$ by $l_i := -r$, $u_i := r$ for all i. Then $l \leq Wx \leq y$ for all $x \in S$ so the image WS is contained in

$$Y := \{y \in \mathbb{Z}^d : l \leq y \leq u\}.$$

With d fixed and $\rho(S)$ and W unary encoded, the cardinality of Y is polynomial:

$$|Y| = (2r+1)^d = \big(2n\rho(S)\|W\|_\infty + 1\big)^d.$$

So computing the image WS reduces to solving polynomially many fiber problems,

$$WS = \{y \in Y : W^{-1}(y) \cap S \neq \emptyset\}.$$

But the fiber problem remains hard: its complexity is open even for $d = 2$, S the set of permutation matrices (so $\rho(S) = 1$), and $\{0,1\}$-valued W; see Example 2.4.

We proceed next to find a way around the integer fiber problem, which makes it possible to efficiently realize our convex maximization Strategy 2.3.

We begin with the *rational* fiber problem, of finding a *rational* point in the intersection $W^{-1}(y) \cap \text{conv}(S)$ of the fiber of a given point y with $\text{conv}(S)$. If S is finite then this intersection is a polytope, and we want to determine a *vertex* of it.

Suppose first that we have an explicit description $\text{conv}(S) = \{x \in \mathbb{R}^n : Ax \leq b\}$ of $\text{conv}(S)$ by inequalities, we can then solve the rational fiber problem as follows.

Lemma 2.6. *There is an algorithm that, given integer $m \times n$ matrix A, integer $d \times n$ matrix W, $b \in \mathbb{Z}^m$, and $y \in \mathbb{Z}^d$, with $P := \{x \in \mathbb{R}^n : Ax \leq b\}$ bounded, either obtains a vertex x of the intersection $W^{-1}(y) \cap P$ of the fiber of y and P or asserts that this intersection is empty, in time which is polynomial in $\langle A, b, W, y\rangle$.*

Proof. The intersection is a polytope having the following inequality description:

$$W^{-1}(y) \cap P = \{x \in \mathbb{R}^n : Ax \leq b,\ Wx = y\}.$$

Linear programming [59], [87] enables to either find a vertex or detect that it is empty. □

We proceed to the more general situation where the set $S \subseteq \mathbb{Z}^n$ is presented by a linear-optimization oracle. To solve the rational fiber problem in this broader situation, we use the algorithmic equivalence of separation and linear-optimization from [44], [72] which extends [59] and builds on the ellipsoid method of [101]. This is the only place in the monograph where we use it; moreover, the main result of this section, Theorem 2.10, is proved in Section 2.3.3 in an alternative, self-contained way, without the ellipsoid method.

So we do not go into details of this theory, which can be found in the above references; see also [79] and references therein. Let us just illustrate the main geometric idea by giving a brief outline of the method for approximative minimization of a convex function f over a convex set P. A (strong) *separation oracle* for closed convex set $P \subset \mathbb{R}^n$ is one that, queried on $z \in \mathbb{R}^n$, either asserts that $z \in P$ or returns a *separator* of z from P, that is, a vector $h \in \mathbb{R}^n$ which satisfies $h(x - z) < 0$ for all $x \in P$. A *subgradient oracle* for convex function $f : \mathbb{R}^n \to \mathbb{R}$ is one that, queried on $z \in \mathbb{R}^n$, returns a *subgradient* of f at z, that is, a vector $f'(z) \in \mathbb{R}^n$ such that $f'(z)(x - z) \leq f(x) - f(z)$ for all $x \in \mathbb{R}^n$; in particular, $f'(z)(x - z) \leq 0$ whenever $f(x) \leq f(z)$. Note that if $f(x) = wx$ is linear then $f'(x) = w$ for all $x \in \mathbb{R}^n$. An *ellipsoid* in $\mathbb{R}^n$ is a set of the form $E = \{x \in \mathbb{R}^n : \|A(x - c)\|_2 \leq 1\}$, where A is a nonsingular $n \times n$ matrix and $c \in \mathbb{R}^n$ is the ellipsoid center. The method proceeds as follows. Given convex compact set $P \subset \mathbb{R}^n$ contained in the ball $E_0 := \{x \in \mathbb{R}^n : \|x\|_2 \leq \rho\}$ centered at $c_0 := 0$, it produces, starting from E_0, a sequence of ellipsoids E_k with centers c_k which contain a minimizer of f over P, as follows. At iteration k, both oracles are queried on center c_k. If $c_k \in P$ and $f'(c_k) = 0$ then c_k is a minimizer of f over P and the method terminates. Otherwise, a new ellipsoid E_{k+1} is generated as the one of smallest volume containing the half ellipsoid $\{x \in E_k : h_k(x - c_k) \leq 0\}$, with h_k either the separator of c_k from P if $c_k \notin P$ or $h_k := f'(c_k)$ the subgradient of f at c_k if $c_k \in P$. It can be shown that the volumes of the ellipsoids in this sequence decrease by a constant factor, so after polynomially many iterations, an ellipsoid of sufficiently small volume containing a minimizer of f over P is obtained.

If P has substantial interior and is presented by an oracle (weakly) separating over its interior then it can be shown that some center in the sequence is feasible and has function value which approximates the minimum value of f over P. If P is a rational polytope and f is linear then it can be shown that the method can be used to provide an exact optimal vertex or detect that P is empty. So a separation oracle can be converted to a linear-optimization oracle in polynomial time. Moreover, the method can be reversed to convert a linear-optimization oracle to a separation oracle for P as well. Further details are in the above references.

Using this equivalence of separation and linear optimization, we can solve the rational fiber problem over a set presented by a linear-optimization oracle as well.

Lemma 2.7. *There is an algorithm that, given finite set $S \subset \mathbb{Z}^n$ presented by a linear-optimization oracle, integer $d \times n$ matrix W, and $y \in \mathbb{Z}^d$, either obtains a vertex x of the intersection $W^{-1}(y) \cap \operatorname{conv}(S)$ of the fiber of y and $\operatorname{conv}(S)$ or asserts that this intersection is empty, in time which is polynomial in $\langle \rho(S), W, y \rangle$.*

Proof. Use the linear-optimization oracle of S to realize, via the ellipsoid method, in time which is polynomial in the binary length $\langle \rho(S) \rangle$ of the radius, a separation oracle for $P := \operatorname{conv}(S)$. Now, using it, realize a separation oracle for

$$F := W^{-1}(y) \cap P = \{x \in \mathbb{R}^n : Wx = y\} \cap P$$

as follows. Let $x \in \mathbb{R}^n$ be any query. First check if $Wx = y$ and either conclude $x \in W^{-1}(y)$ or identify an inequality $W_i x \neq y_i$ in which case $h := W_i$ or $h := -W_i$

separates x from $W^{-1}(y)$ and hence from F. If $x \in W^{-1}(y)$ then check if $x \in P$ using the separation oracle of P and either conclude $x \in P$ and hence $x \in F$ or obtain from the oracle a vector h separating x from P and hence from F.

Using this separation oracle for F, realize, via the ellipsoid method again, in time which is polynomial in $\langle \rho(S), W, y \rangle$, a linear-optimization oracle for F, which enables (see [44], [72]) to either obtain a vertex of F or detect that F is empty. □

We can now realize step 1 of our Strategy 2.3 and compute $\text{vert}(\text{conv}(WS))$.

Lemma 2.8. *For every fixed d there is an algorithm that, given a finite nonempty set $S \subset \mathbb{Z}^n$ presented by a linear-optimization oracle and an integer $d \times n$ matrix W, computes the set* $\text{vert}(\text{conv}(WS))$ *in time which is polynomial in $\rho(S)$ and W.*

Proof. First apply the algorithm of Lemma 2.5 and determine the radius $\rho(S)$. Next define $P := \text{conv}(S)$, $Q := \text{conv}(WS)$, $r := n\rho(S)\|W\|_\infty$, and

$$Y := \big\{ y \in \mathbb{Z}^d : -r \le y_i \le r,\ i = 1, \ldots, d \big\}.$$

Since d is fixed, the number of points $|Y| = (2r+1)^d$ of Y is polynomial in $\rho(S)$ and W. Distill the set $U := Y \cap Q$ out of Y as follows. For each $y \in Y$, apply the algorithm of Lemma 2.7; if $W^{-1}(y) \cap P$ is nonempty then $y \in Q$ and hence $y \in U$, whereas if it is empty, $y \notin U$. Since $WS \subseteq U$ and $U \subseteq Q$, we have

$$Q = \text{conv}(WS) \subseteq \text{conv}(U) \subseteq Q,$$

so $Q = \text{conv}(U)$. Now the vertex set of $\text{conv}(U)$ and hence of Q can be computed in polynomial time, either by computing the convex hull of Z in fixed dimension d or by solving a linear program for each $u \in U$ that checks if $u \notin \text{conv}(U \setminus \{u\})$. □

We proceed with step 3 of our convex maximization Strategy 2.3, that of finding a feasible points in the fiber of an optimal vertex of $\text{conv}(WS)$.

Let $P := \text{conv}(S)$ and $Q := \text{conv}(WS)$. Consider any point $y \in Q \cap \mathbb{Z}^d$ and the intersection $F := W^{-1}(y) \cap P$ of the fiber of y with P. The vertices of the polytope F need not all lie in S and need not even all be integer. Even worse, y may be a hole, so that $W^{-1}(y) \cap S = \emptyset$, or F may have only fractional vertices.

Fortunately, fibers of vertices of $\text{conv}(WS)$ are better behaved. We now show that we can efficiently find a feasible point in the fiber of any vertex of the image.

Lemma 2.9. *Let $S \subset \mathbb{Z}^n$ be a finite nonempty set presented by a linear-optimization oracle and let W be an integer $d \times n$ matrix. Then $W^{-1}(v) \cap \text{conv}(S)$ is a nonempty integer polytope whose vertices lie in S for every vertex v of* $\text{conv}(WS)$. *Hence, the algorithm of Lemma 2.7, applied to S, W, and v, returns a feasible $x \in W^{-1}(v) \cap S$.*

Proof. It is well known that if Q is the image of a polytope P under an affine map ω then the preimage $\omega^{-1}(G) \cap P = \{x \in P : \omega(x) \in G\}$ of any face G of Q is a face of P. In particular, if v is a vertex of $Q := WP$, where $P := \text{conv}(S)$, then its preimage $F := W^{-1}(v) \cap P$ under the projection W is a face of P, and hence $\text{vert}(F) = F \cap \text{vert}(P) \subseteq S$. Therefore, any vertex of F, and in particular that vertex x returned by the algorithm of Lemma 2.7, is a feasible point $x \in S$. □

We are now in position to describe our first convex maximization algorithm, which provides an efficient realization of our convex maximization Strategy 2.3. The following result is from [11], extending an earlier result from [12].

Theorem 2.10. *For every fixed d there is an algorithm that, given set $S \subseteq \mathbb{Z}^n$ presented by a linear-optimization oracle, integer $d \times n$ matrix W, and convex function $f : \mathbb{Z}^d \to \mathbb{R}$ presented by a comparison oracle, solves in time which is polynomial in the radius $\rho(S)$ and the weight W, the convex maximization problem:*

$$\max\{f(Wx) : x \in S\}.$$

Proof. First, apply the algorithm of Lemma 2.5 and either conclude that S is infinite or empty and stop or conclude that it is finite and nonempty and continue. Now, use the algorithm of Lemma 2.8 to compute the set $V := \text{vert}(\text{conv}(WS))$. Next, use the comparison oracle of f to find a vertex $v \in V$ maximizing f over V. Finally, use the algorithm of Lemma 2.9 to find a feasible point $x \in W^{-1}(v) \cap S$ in the fiber of the vertex v. By Lemma 2.2, this point x is an optimal solution. □

The algorithm of Theorem 2.10 incorporates the computation of the vertices of the image by the algorithm of Lemma 2.8. This in turn requires repeated use of the algorithm of Lemma 2.7 which converts linear optimization to separation using the ellipsoid method, for each point y in the box Y. This is a very time-consuming process. We now describe a simple faster variant of this procedure which typically inspects only some of the points of Y. It assumes that S is *integer convex*, that is, $S = \text{conv}(S) \cap \mathbb{Z}^n$. This always holds in combinatorial optimization, with $S \subseteq \{0,1\}^n$, and in integer programming, with $S = \{x \in \mathbb{Z}^n : Ax \leq b\}$.

Procedure 2.11 (convex maximization over integer-convex sets).

1. Check by the algorithm of Lemma 2.5 if S is infinite or empty. Suppose not.
2. Let $r := n\rho(S)\|W\|_\infty$ and $Y := \{y \in \mathbb{Z}^d : -r \leq y_i \leq r, i = 1, \ldots, d\}$.
3. Use the oracle of f to label $Y = \{y^1, \ldots, y^k\}$ with $f(y^i) \geq f(y^{i+1})$ for all i.
4. Test the fibers of the y^i in order using the algorithm of Lemma 2.7. Output the first *integer* vertex x^j of $W^{-1}(y^j) \cap \text{conv}(S)$ returned by the algorithm.

To see that this procedure works as well, let y^k be a vertex of $\text{conv}(WS)$ which maximizes f. By Lemma 2.9, the vertex x^k of $W^{-1}(y^k) \cap \text{conv}(S)$ returned by the algorithm of Lemma 2.7 when testing y^k lies in S and is therefore integer. Furthermore, x^k is an optimal solution of the convex maximization problem by Lemma 2.2. Since x^j is the first integer vertex returned, we have $j \leq k$ and, hence, since the y^i are ordered by nonincreasing value of f, we have $f(y^j) \geq (y^k)$. Since $y^j = Wx^j$ and $y^k = Wx^k$, we obtain $f(Wx^j) \geq f(Wx^k)$. Since S is integer convex, we have $x^j \in \text{conv}(S) \cap \mathbb{Z}^n = S$ and hence x^j is feasible. Therefore, x^j is optimal.

2.3 Convex maximization with edge directions

We proceed to consider the more involved situation where the radius $\rho(S)$ and matrix W are binary encoded. In this situation, the number of points in a minimal box $Y = \{y \in \mathbb{Z}^d : l \leq y \leq u\}$ containing the image WS may be exponential in the input length. So methods more sophisticated than those of Section 2.2 are needed.

The main result here is Theorem 2.16: for every fixed d, we can maximize any convex composite function $f(Wx)$ over S endowed with a set E of edge directions of $\text{conv}(S)$, in time polynomial in the binary encoding $\langle \rho(S), W, E \rangle$ of the data.

This section is organized as follows. In Section 2.3.1, we provide the necessary preliminaries on edge directions and zonotopes. These are used in Section 2.3.2 to establish Theorem 2.16 providing our main polynomial time convex maximization algorithm. This algorithm is also used in Section 2.3.3 to provide an alternative procedure, avoiding the ellipsoid method, for solving the problem in the easier situation of Section 2.2.

2.3.1 Edge directions and zonotopes

A *direction* of an edge (1-dimensional face) e of a polyhedron P is any nonzero scalar multiple of $u-v$, where u, v are any two distinct points in e. A *set of all edge directions of* P is a set which contains some direction of each edge of P, see Figure 2.2. We later show how to exploit a set of all edge directions of the convex hull of a set of integer points S for efficient convex maximization over S. The *normal cone* of a polyhedron $P \subseteq \mathbb{R}^n$ at its face F is the relatively open cone C_P^F of those linear functions $h \in \mathbb{R}^n$ 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 $\mathbb{R}^d$ as follows:

$$Z := \text{zone}(E) := \text{conv}\left\{ \textstyle\sum_{i=1}^m \lambda_i e_i : \lambda_i = \pm 1 \right\} \subset \mathbb{R}^d.$$

The following fact, illustrated in Figure 2.3, goes back to Minkowski, see [45], [103].

Lemma 2.12. *Let P be a polytope and let E be a finite set of 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 $h \in C_Z^u$ and let v be a vertex of P at which 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 $0 \leq h(v - w) = h\alpha_e e$. This implies that α_e and λ_e have the same sign and hence $h\alpha_e e > 0$. Therefore, every $h \in C_Z^u$ satisfies $h(v - w) > 0$ for every edge of P containing v. So h is maximized over P uniquely at v and hence is in the cone C_P^v of P at v. This shows that $C_Z^u \subseteq C_P^v$. 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. □

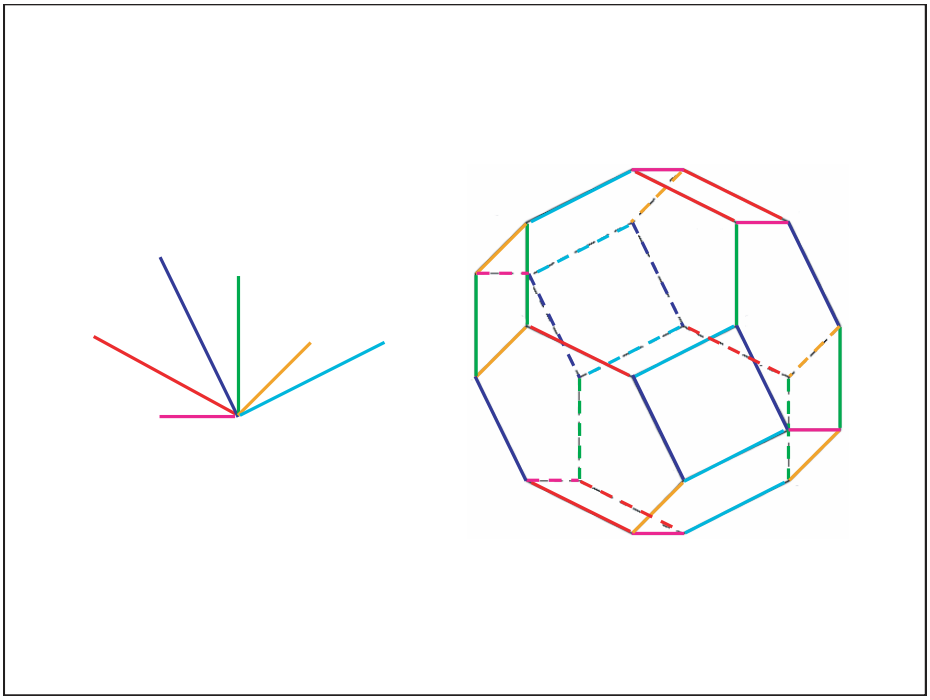


Figure 2.2: Edge directions of a convex polytope

The next lemma provides a bound on the number of vertices of any zonotope and on the complexity of constructing its vertices, each vertex along with a linear function 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 [47], stated in the dual form of 2 partitions. Extensions to p-partitions for any p are in [2], [57]. Further extensions and information on zonotopes and Minkowski sums can be found in [38], [42], [102]. The algorithmic complexity of the problem is settled in [30], [31]. We state the precise bounds on the number of vertices, but outline only a proof that, for every fixed d, the bounds are polynomial, which is all we need in the sequel. Complete details are in the above references.

Lemma 2.13. *The number of vertices of any zonotope* $Z := \mathrm{zone}(E)$ *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 an algorithm that, given* $E \subset \mathbb{Z}^d$, *outputs every vertex* v *of* $Z := \mathrm{zone}(E)$ *along with some* $h_v \in \mathbb{Z}^d$ *maximized over* Z *uniquely at* v, *in time polynomial in* $\langle E \rangle$.

Proof. We only outline a proof that, for every fixed d, the number of vertices is $O(m^{d-1})$ and hence polynomially bounded, and the vertices can be constructed in polynomial time. 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,

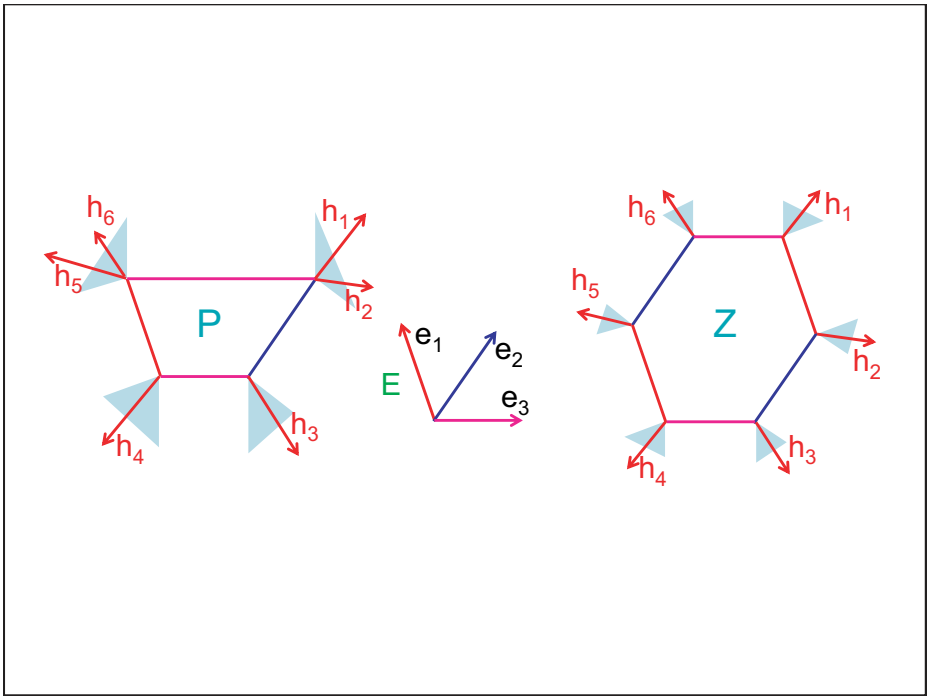


Figure 2.3: A zonotope refining a polytope

$0 \notin E$. The same bound for arbitrary E then follows by using a perturbation argument. More details can be found in [57].

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 $E = H^- \biguplus H^0 \biguplus H^+$ with $H^- := \{e \in E : he < 0\}$, $E^0 := E \cap H$, and $H^+ := \{e \in E : he > 0\}$. The vertices of $Z = \text{zone}(E)$ are in bijection with ordered 2 partitions of E induced by such hyperplanes that avoid E. Indeed, if $E = H^- \biguplus H^+$ then the linear function $h_v := h$ 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 cardinality $d-1$. Since E is generic, M is linearly independent and spans a unique linear hyperplane $\text{lin}(M)$. Let $\hat{H} = \{x \in \mathbb{R}^d : \hat{h}x = 0\}$ be one of the two orientations of the hyperplane $\text{lin}(M)$. Note that $\hat{H}^0 = M$. 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 $H^- = \hat{H}^- \biguplus L$ and $H^+ = \hat{H}^+ \biguplus (M \setminus L)$. The corresponding vertex of the zonotope Z is $v := \sum\{e : e \in H^+\} - \sum\{e : e \in H^-\}$ and the corresponding linear function 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 such subset M, some orientation $\hat{H}$ of $\mathrm{lin}(M)$, and some subset L. Indeed, consider any oriented linear hyperplane $\tilde{H}$ avoiding E. It can be perturbed to a suitable oriented $\hat{H}$ that touches precisely $d-1$ points of E. Put $M := \hat{H}^0$ so that $\hat{H}$ coincides with one of the two orientations of the hyperplane $\mathrm{lin}(M)$ spanned by M, and put $L := \tilde{H}^- \cap M$. Let H be an oriented hyperplane obtained from M, $\hat{H}$ and L by the above procedure. Then the ordered 2-partition $E = H^- \uplus H^+$ induced by H coincides with the ordered 2-partition $E = \tilde{H}^- \uplus \tilde{H}^+$ induced by $\tilde{H}$.

Since there are $\binom{m}{d-1}$ many $(d-1)$-subsets $M \subseteq E$, two orientations $\hat{H}$ of $\mathrm{lin}(M)$, and 2^{d-1} subsets $L \subseteq M$, and d is fixed, the total number of 2 partitions and hence of vertices of Z is bounded by $2^d\binom{m}{d-1} = O(m^{d-1})$. Furthermore, for each choice of M, $\hat{H}$, and L, the linear function $\hat{h}$ defining $\hat{H}$, 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 in time polynomial in $\langle E \rangle$. □

We also need the following simple fact about edge directions of projections.

Lemma 2.14. *If E is a set of all edge directions of polytope P and $\omega : \mathbb{R}^n \to \mathbb{R}^d$ is a linear map then $\omega(E)$ is a set of all the edge directions of the polytope $Q := \omega(P)$.*

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 $e := \frac{1}{\alpha}(v - u)$, a direction of the edge $[u, v]$ of P, we find that $f = \frac{1}{\alpha}(\omega(v) - \omega(u)) = \omega(e) \in \omega(E)$. □

2.3.2 Efficient convex maximization

We proceed to solve the convex maximization problem when W is binary encoded. As in Section 2.2, we follow the general outline of our convex maximization Strategy 2.3. However, the hard steps 1 and 3 are now done simultaneously, since feasible points in fibers of vertices are constructed on the fly as part of the construction of the image polytope, avoiding the use of the time-consuming ellipsoid method. More precisely, we compute a subset $T \subseteq S$ of the feasible set, whose image contains the vertex set of the image of S, that is, $\mathrm{vert}(\mathrm{conv}(WS)) \subseteq WT = \{Wx : x \in T\}$.

Lemma 2.15. *For every fixed d there is an algorithm that, given a finite nonempty set $S \subset \mathbb{Z}^n$ presented by a linear-optimization oracle, integer $d \times n$ weight matrix W, and set $E \subset \mathbb{Z}^n$ of all edge directions of* $\mathrm{conv}(S)$*, computes a subset $T \subseteq S$ such that* $\mathrm{vert}(\mathrm{conv}(WS)) \subseteq WT$*, in time which is polynomial in $\langle \rho(S), W, E \rangle$.*

Proof. Let $P := \mathrm{conv}(S) \subset \mathbb{R}^n$ and $Q := \mathrm{conv}(WS) = WP \subset \mathbb{R}^d$. Since Q is a projection of P, by Lemma 2.14 the projection $D := WE = \{We : e \in E\}$ of E is a set of all edge directions of Q. Let $Z := \mathrm{zone}(D) \subseteq \mathbb{R}^d$ be the zonotope generated by D. Since d is fixed, by Lemma 2.13 we can produce in polynomial time all vertices of Z, every vertex u along with $h_u \in \mathbb{Z}^d$ maximized over Z uniquely at u. For each of these

polynomially many h_u, define $g_u \in \mathbb{Z}^n$ by $g_u := W^T h_u$, query the linear-optimization oracle of S on g_u, and let $x_u \in S$ be the optimal solution obtained from the oracle. Let $z_u := Wx_u \in Q$ be the image of x_u. Since $P = \text{conv}(S)$, we have that x_u is also a maximizer of g_u over P. Since for every $x \in P$ and its image $z := Wx \in Q$ we have $h_u z = g_u x$, we find that z_u is a maximizer of h_u over Q. Now we claim that each vertex v of Q equals some z_u. Indeed, since Z is a refinement of Q by Lemma 2.12, there is some vertex u of Z such that h_u is maximized over Q uniquely at v and hence $v = z_u$. So the set $T := \{x_u : u \in \text{vert}(Z)\} \subseteq S$ of points obtained that way is the desired set. □

We are now in position to prove the main result of this chapter. It extends and unifies earlier results of [83] and [23]. An illustration of the algorithm of Theorem 2.16 below incorporating the algorithm of Lemma 2.15 is provided in Figure 2.4.

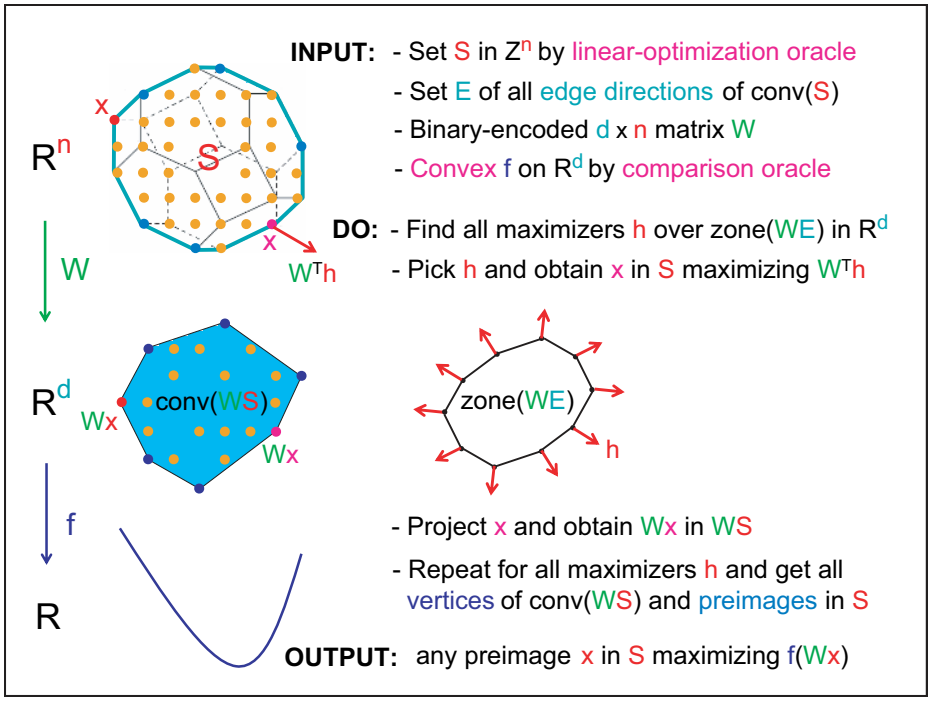


Figure 2.4: Convex discrete maximization algorithm

Theorem 2.16. *For every fixed d there is an algorithm that, given set $S \subseteq \mathbb{Z}^n$ presented by a linear-optimization oracle, integer $d \times n$ matrix W, set $E \subset \mathbb{Z}^n$ of all edge directions of* conv(S)*, and convex function $f : \mathbb{Z}^d \to \mathbb{R}$ presented by a comparison oracle, solves in time polynomial in $\langle \rho(S), W, E \rangle$, the convex problem:*

$$\max \{ f(Wx) : x \in S \}.$$

Proof. First, apply the algorithm of Lemma 2.5 and either conclude that S is infinite or empty and stop, or conclude that it is finite and nonempty and continue. Now, use the algorithm of Lemma 2.15 to compute a subset $T \subseteq S$ such that $\text{vert}(\text{conv}(WS)) \subseteq WT$. Now, inspecting T and using the comparison oracle of f, find point $x^* \in T$ whose image Wx^* maximizes f over WT. Then, by Lemma 2.2,

$$\begin{aligned} f\big(Wx^*\big) &= \max\big\{f(y) : y \in WT\big\} \\ &= \max\big\{f(y) : y \in \text{vert}\,\big(\text{conv}(WS)\big)\big\} \\ &= \max\big\{f(Wx) : x \in S\big\}. \end{aligned}$$

Therefore, x^* is an optimal solution of the given convex maximization problem. □

2.3.3 Small radius and weight revisited

Returning to the situation of Section 2.2, we observe that for unary-encoded data, we can always produce a set of edge directions of polynomial size. This gives a convex maximization algorithm which is different from that given in Section 2.2 and uses the algorithm of Theorem 2.16 instead of the ellipsoid method. It is not clear offhand which of the two is more efficient – this may depend on the specific application of interest. We now describe this variant, providing a second proof of Theorem 2.10.

Theorem 2.10 (revisited). *For every fixed d there is an algorithm that, given set $S \subseteq \mathbb{Z}^n$ presented by a linear-optimization oracle, integer weight $d \times n$ matrix W, and convex function $f : \mathbb{Z}^d \to \mathbb{R}$ presented by a comparison oracle, solves in time polynomial in the radius $\rho(S)$ and weight W, the convex maximization problem:*

$$\max\big\{f(Wx) : x \in S\big\}.$$

Proof. First, apply the algorithm of Lemma 2.5 and either conclude that S is infinite or empty and stop, or conclude that it is finite and nonempty and obtain its radius $\rho(S)$. Next, define $P := \text{conv}(S)$, $Q := \text{conv}(WS)$, $r := n\rho(S)\|W\|_\infty$ as follows:

$$Y := \big\{y \in \mathbb{Z}^d : -r \le y_i \le r,\ i = 1, \dots, d\big\},$$

and

$$D := \{u - v : u, v \in Y\} = \big\{z \in \mathbb{Z}^d : -2r \le z_i \le 2r\big\}. \tag{2.5}$$

Then D is the set of differences of pairs of point of Y and hence a set of all edge directions of Q since $\text{vert}(Q) \subseteq WS \subseteq Y$. Moreover, with d fixed and $\rho(S)$ and W unary encoded, $|D| = (4r + 1)^d$ is polynomially bounded in the input.

Now, invoke the algorithm of Theorem 2.16 incorporating that of Lemma 2.15, using the set D of all edge directions of the image Q, which here is computed directly via (2.5), without going through a set E of all edge directions of P. □

2.3.4 Totally unimodular systems

A matrix A is *totally unimodular* if it is integer and the determinant of every square submatrix of A is -1, 0, or 1. Important examples of totally unimodular matrices are vertex-edge incidence matrices of digraphs and bipartite graphs. The (non)linear optimization problem over a set S given by inequalities of the following form, with totally unimodular matrix A, right-hand side b, and bounds $l, u \in \mathbb{Z}^n_\infty$:

$$S := \{x \in \mathbb{Z}^n : Ax = b,\ l \le x \le u\} \tag{2.6}$$

is quite useful and includes the classical transportation problem (with A the incidence matrix of a bipartite graph) and the classical single-commodity transshipment problem (with A the incidence matrix of a digraph) discussed in Section 1.2.2. Note, however, that the *multiindex* transportation problems and the *multicommodity* transshipment problems discussed in Section 1.2.2 are *not* totally unimodular and therefore much harder and treated by more sophisticated methods in Chapter 4.

It is possible to minimize over totally unimodular systems of the form (2.6) linear functions [54] and separable convex functions [53] in polynomial time. The algorithms exploit the fundamental result of [54] that any polyhedron defined by a totally unimodular matrix A is *integer*, that is, it satisfies the following equality:

$$\text{conv}(S) = \text{conv}\{x \in \mathbb{Z}^n : Ax = b,\ l \le x \le u\} = \{x \in \mathbb{R}^n : Ax = b,\ l \le x \le u\}.$$

We proceed to describe a certain situation where convex maximization over totally unimodular systems of the form (2.6) can also be done in polynomial time.

We need some terminology, and preparatory lemmas also used in later chapters. The *lattice* of an integer $m \times n$ matrix A is the set $\mathcal{L}(A) := \{x \in \mathbb{Z}^n : Ax = 0\}$ of integer vectors in its kernel. We denote the set of nonzero elements in $\mathcal{L}(A)$ by $\mathcal{L}^*(A) := \{x \in \mathbb{Z}^n : Ax = 0,\ x \neq 0\}$. We use a partial order $\sqsubseteq$ on $\mathbb{R}^n$ which extends the coordinate wise partial order $\le$ on the nonnegative orthant $\mathbb{R}^n_+$ and is defined as follows. For $x, y \in \mathbb{R}^n$, we write $x \sqsubseteq y$ and say that x is *conformal* to y if $x_i y_i \ge 0$ and $|x_i| \le |y_i|$ for $i = 1, \ldots, n$, that is, x and y lie in the same orthant of $\mathbb{R}^n$, and each component of x is bounded by the corresponding component of y in absolute value. We use $x \sqsubset y$ to indicate that x is strictly smaller than y under $\sqsubset$, that is, $x \sqsubseteq y$ and $x \neq y$. A finite sum $u := \sum_i v_i$ of vectors in $\mathbb{R}^n$ is called *conformal* if $v_i \sqsubseteq u$ for all i and hence all summands lie in the same orthant.

A *circuit* of A is an element $c \in \mathcal{L}^*(A)$ whose support $\text{supp}(c)$ is minimal under inclusion and whose entries are relatively prime. We denote the set of circuits of A by $\mathcal{C}(A)$. The set of circuits is centrally symmetric, that is, $c \in \mathcal{C}(A)$ if and only if $-c \in \mathcal{C}(A)$. For instance, the set of circuits of the 1×3 matrix $A := (1\ 2\ 1)$ is

$$\mathcal{C}(A) = \pm\{(2, -1, 0), (0, -1, 2), (1, 0, -1)\}.$$

The following property of circuits is well known in one form or another.

Lemma 2.17. *Let A be an $m \times n$ integer matrix of rank r. Any nonzero rational $x \in \mathbb{R}^n$ with $Ax = 0$ is a conformal sum $x = \sum_{i=1}^t \lambda_i c_i$ involving $t \le n - r$ linearly independent circuits $c_i \in \mathcal{C}(A)$ with $\lambda_i \ge 0$ and $\text{supp}(c_i) \not\subseteq \bigcup_{j>i} \text{supp}(c_j)$ for all i.*

Proof. First, we show that for any such x, there are $c \in \mathcal{C}(A)$ and nonnegative λ with $\lambda c \sqsubseteq x$. Suppose indirectly that this is false and let x be a counterexample with minimal support. Then there is an $h \in \mathcal{C}(A)$ with $\mathrm{supp}(h) \subsetneq \mathrm{supp}(x)$ and hence there exists a μ with $y := x - \mu h \sqsubset x$ and $\mathrm{supp}(y) \subsetneq \mathrm{supp}(x)$. Since $y \neq 0$ and $Ay = 0$, there are $c \in \mathcal{C}(A)$ and nonnegative λ with $\lambda c \sqsubseteq y \sqsubseteq x$, a contradiction.

We proceed to prove that every such x is a conformal sum involving circuits with $\mathrm{supp}(c_i) \not\subseteq \bigcup_{j>i} \mathrm{supp}(c_j)$ for each i. This in particular implies that the c_i are linearly independent, and, since all circuits lie in the orthogonal complement of the row space of A, that at most $n - r$ circuits are involved. Suppose indirectly that this is false and let x be a counterexample with minimal support. By what we just proved, there are $c_0 \in \mathcal{C}(A)$ and $\lambda_0 \in \mathbb{R}_+$ with $\lambda_0 c_0 \sqsubseteq x$ and hence $y := x - \lambda_0 c_0 \sqsubseteq x$. Clearly, we can choose λ_0 so that y satisfies $\mathrm{supp}(y) \subsetneq \mathrm{supp}(x)$. Then y is a conformal sum $y = \sum_{i=1}^{s} \lambda_i c_i$ involving circuits c_i with $\mathrm{supp}(c_i) \not\subseteq \bigcup_{j>i} \mathrm{supp}(c_j)$ for each i. But then $x = \lambda_0 c_0 + \sum_{i=1}^{s} \lambda_i c_i$ is a conformal sum involving circuits with $\mathrm{supp}(c_i) \not\subseteq \bigcup_{j>i} \mathrm{supp}(c_j)$ for each i, a contradiction. This completes the proof. □

The circuits of a matrix provide edge directions of the polyhedron it defines.

Lemma 2.18. *For every integer $m \times n$ matrix A, $l, u \in \mathbb{Z}^n_\infty$, and $b \in \mathbb{Z}^m$, the set of circuits $\mathcal{C}(A)$ is a set of all edge-directions of $P = \{x \in \mathbb{R}^n : Ax = b,\ l \leq x \leq u\}$.*

Proof. Consider any edge e of P. Pick two distinct rational points $x, y \in e$ and set $g := y - x$. Then a suitable multiple of g is in $\mathcal{L}^*(A)$ and hence it follows from Lemma 2.17 that $g = \sum_i \lambda_i c_i$ is a conformal sum for suitable circuits $c_i \in \mathcal{C}(A)$ and $\lambda_i \in \mathbb{R}_+$. We claim that $x + \lambda_i c_i \in P$ for all i. Indeed, c_i being a circuit implies $A(x + \lambda_i c_i) = Ax = b$, and $l \leq x, x + g \leq u$ and $\lambda_i c_i \sqsubseteq g$ imply $l \leq x + \lambda_i c_i \leq u$.

Now ,let $w \in \mathbb{R}^n$ be uniquely maximized over P at the edge e. Then $w\lambda_i c_i = w(x + \lambda_i c_i) - wx \leq 0$ for all i. But $\sum w\lambda_i c_i = wg = wy - wx = 0$, implying that in fact $w\lambda_i c_i = 0$ and hence $x + \lambda_i c_i \in e$ 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). □

We now show that Theorem 2.16 enables to maximize convex functions over totally unimodular systems of the form (2.6) in polynomial time when the set of circuits of the matrix defining the system is available. The proof given here incorporates linear programming [59]. In Section 3.3.5, we give an alternative proof which uses results of Chapter 3 on Graver bases and avoids the use of linear programming.

Theorem 2.19. *For every fixed d there is an algorithm that, given totally unimodular $m \times n$ matrix A, its set of circuits $\mathcal{C}(A)$, $l, u \in \mathbb{Z}^n_\infty$, $b \in \mathbb{Z}^m$, integer $d \times n$ matrix W, and convex $f : \mathbb{Z}^d \to \mathbb{R}$ presented by comparison oracle, solves in time polynomial in $\langle A, W, \mathcal{C}(A), l, u, b\rangle$ the convex integer maximization problem:*

$$\max\left\{f(Wx) : x \in \mathbb{Z}^n,\ Ax = b,\ l \leq x \leq u\right\}.$$

Proof. Let $S := \{x \in \mathbb{Z}^n : Ax = b,\ l \leq x \leq u\}$. Since A is totally unimodular, we have $\mathrm{conv}(S) = \{x \in \mathbb{R}^n : Ax = b,\ l \leq x \leq u\}$. Therefore, linear programming enables

to realize in polynomial time a linear-optimization oracle for S. In particular, it allows to either conclude that S is infinite and stop or that it is finite, in which case the binary length $\langle \rho(S) \rangle$ of its radius is polynomial in $\langle A, l, u, b \rangle$, and continue. By Lemma 2.18, the given set $\mathcal{C}(A)$ is a set of all edge directions of $\text{conv}(S)$. Hence, the algorithm of Theorem 2.16 can be applied and provides the polynomial time solution of the convex integer maximization problem. □

We note that an $m \times n$ matrix A has at most $2\sum_{k=0}^{m} \binom{n}{k+1}$ circuits, a bound depending only on m and n and independent of the entries of A. So, if m grows slowly in terms of n, say $m = O(\log n)$, then $\mathcal{C}(A)$ can be computed in subexponential time. In particular, if m is fixed then $\mathcal{C}(A)$ can be computed in polynomial time by checking all $2\sum_{k=0}^{m} \binom{n}{k+1} = O(n^{m+1})$ potential supports of circuits. This has applications to vector partitioning and clustering discussed in Section 2.5.3.

2.4 Convex combinatorial maximization

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 $x \in S$. A membership oracle for S is available in all reasonable applications, but reveals little information on S, making it very hard to use. Yet, we now show that *convex combinatorial maximization*, that is, convex maximization over $S \subseteq \{0,1\}^n$, can be done in polynomial time using but a membership oracle.

The main result here is Theorem 2.22: for every fixed d, we can maximize any convex composite function $f(Wx)$ over $S \subseteq \{0,1\}^n$ presented by a membership oracle, given a set of edge directions of $\text{conv}(S)$, in time polynomial in $\langle W \rangle$.

We need to make use of an additional oracle presentation of S, defined as follows. An *augmentation oracle* for a set $S \subseteq \mathbb{Z}^n$ is one that, queried on $x \in S$ and $w \in \mathbb{Z}^n$, either asserts that x is optimal for the linear optimization problem $\max\{wx : x \in S\}$ or returns a better point $\hat{x} \in S$, that is, one satisfying $w\hat{x} > wx$.

An augmentation oracle for S allows to solve the linear optimization problem $\max\{wx : x \in S\}$ by starting from any feasible $x \in S$ and repeatedly augmenting it until an optimal solution $x^* \in S$ is reached. The following lemma, which is an adaptation of a result of [43], [92] making use of bit-scaling ideas going back to [34], shows that this can be done in time which is polynomial in the binary length of w and the initial point x and in the unary length of the radius $\rho(S)$ of the set S. We need it only for $S \subseteq \{0,1\}^n$, but provide the proof for any finite $S \subset \mathbb{Z}^n$.

Lemma 2.20. *There is an algorithm that, given a finite set $S \subset \mathbb{Z}^n$ presented by an augmentation oracle, $x \in S$, and $w \in \mathbb{Z}^n$, finds an optimal solution $x^* \in S$ to the optimization problem* $\max\{wz : z \in S\}$, *in time polynomial in $\rho(S)$ and $\langle x, w \rangle$.*

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 vector $u_i = (u_{i,1}, \ldots, u_{i,n}) \in \mathbb{Z}^n$ by $u_{i,j} := \text{sign}(w_j) \lfloor 2^{i-k} |w_j| \rfloor$ 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 that 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, namely, $u_i(\hat{y} - \tilde{y}) \geq 1$, since u_i, $\tilde{y}$, $\hat{y}$ are integers. So the number of augmentations from y_{i-1} to y_i is at most the total improvement, which we claim satisfies the following inequality, where $\rho := \rho(S)$:

$$u_i(y_i - y_{i-1}) = (u_i - 2u_{i-1})(y_i - y_{i-1}) + 2u_{i-1}(y_i - y_{i-1}) \leq 2n\rho + 0 = 2n\rho.$$

Indeed, y_{i-1} optimal for u_{i-1} gives $u_{i-1}(y_i - y_{i-1}) \leq 0$; $u_i - 2u_{i-1} \in \{-1, 0, 1\}^n$ and $y_i, y_{i-1} \in S \subseteq [-\rho, \rho]^n$ imply $(u_i - 2u_{i-1})(y_i - y_{i-1}) \leq 2n\rho$.

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 optimization problem. Clearly, the number $2n\rho k$ of calls to the oracle, the number of arithmetic operations, and the binary length of the numbers occurring during the algorithm are polynomial in $\rho(S), \langle x, w \rangle$. □

In combinatorial optimization, with $S \subseteq \{0, 1\}^n$, each edge of $\text{conv}(S)$ is the difference of two $\{0, 1\}$-vectors, and hence each edge direction of $\text{conv}(S)$ is, up to a scalar multiple, a $\{-1, 0, 1\}$-vector. Moreover, any set $E \subset \mathbb{Z}^n$ of all edge directions of $\text{conv}(S)$ with $S \subseteq \{0, 1\}^n$ can be easily converted, by scaling some elements and dropping redundant ones, to one consisting of $\{-1, 0, 1\}$-vectors only. So, henceforth, when assuming that a set $S \subseteq \{0, 1\}^n$ is endowed with a set E of all edge directions of $\text{conv}(S)$, we may and do assume that $E \subseteq \{-1, 0, 1\}^n$.

We proceed to show that an augmentation oracle can be efficiently realized for a set $S \subseteq \{0, 1\}^n$ presented by a membership oracle and endowed with a set of all edge directions of $\text{conv}(S)$. We also need one initial feasible point $x \in S$ to start with. Without such a point, exponential time cannot be generally avoided, as demonstrated in Section 1.2.1 for the set S of bases of an almost trivial matroid.

Lemma 2.21. *There is an algorithm that, given set $S \subseteq \{0, 1\}^n$ presented by membership oracle, point $x \in S$, $w \in \mathbb{Z}^n$, and set $E \subseteq \{-1, 0, 1\}^n$ of all edge directions of* $\text{conv}(S)$*, either returns a better point $\hat{x} \in S$, that is, one satisfying $w\hat{x} > wx$ or asserts that none exists, in time which is polynomial in $\langle w \rangle$ and $|E|$.*

Proof. Setting $e := -e$ if necessary, we may assume that $we \geq 0$ 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 the better point $\hat{x} := x + e$, whereas if there is no such e then terminate asserting that no better point exists.

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 $S \subseteq \{0, 1\}^n$, x is a vertex of $\text{conv}(S)$. Since x is not a maximizer of w, there is an edge $[x, \hat{x}]$ of $\text{conv}(S)$ with $\hat{x}$ a vertex satisfying $w\hat{x} > wx$. Then $e := \hat{x} - x$ is a $\{-1, 0, 1\}$ edge direction of $[x, \hat{x}]$ with $we > 0$ and hence $e \in E$. So the algorithm will find a better point $\hat{x} = x + e$. □

We can now prove a result of [83] that a convex function of the form $f(Wx)$ can be maximized over a set $S \subseteq \{0,1\}^n$ presented by a mere membership oracle in polynomial time when a set of all edge directions of $\text{conv}(S)$ is available.

Theorem 2.22. *For every fixed d there is an algorithm that, given $S \subseteq \{0,1\}^n$ presented by membership oracle, $x \in S$, integer $d \times n$ matrix W, set $E \subseteq \{-1,0,1\}^n$ of all edge directions of* $\text{conv}(S)$*, and convex function $f : \mathbb{Z}^d \to \mathbb{R}$ presented by comparison oracle, solves in time polynomial in $\langle W \rangle$ and $|E|$, the convex problem:*

$$\max \{ f(Wz) : z \in S \}.$$

Proof. First, using the membership oracle of S, an augmentation oracle for S can be realized by the algorithm of Lemma 2.21. Next, using the realized augmentation oracle of S and noting that $\rho(S) \leq 1$, a linear-optimization oracle for S can be realized by the algorithm of Lemma 2.20. Finally, using the realized linear-optimization oracle of S, we can apply the algorithm of Theorem 2.16 and solve the given convex maximization problem in polynomial time as claimed. □

2.5 Some applications

We now describe some direct applications which use Theorems 2.10, 2.19, and 2.22. The algorithm of Theorem 2.16 is incorporated in Chapters 3–5 and enables convex integer programming with many more applications discussed in Sections 4.3 and 5.2.

2.5.1 Quadratic binary programming

Here, we discuss the following simple immediate application. The *quadratic binary programming problem* asks for $x \in \{0,1\}^n$ maximizing the quadratic form $x^T M x$ with M a given $n \times n$ matrix. Consider the case of positive semidefinite $M = W^T W$ of rank d, with W a given integer $d \times n$ matrix. If d is variable then already this restricted version of the problem is NP-hard [46]. Theorem 2.22 implies the following result of [1] that for fixed d the problem is polynomial time solvable.

Corollary 2.23. *For every fixed d there is an algorithm that, given integer $d \times n$ matrix W, solves in time polynomial in $\langle W \rangle$ the quadratic binary program:*

$$\max \{ \|Wx\|_2^2 : x \in \{0,1\}^n \}. \tag{2.7}$$

Proof. Let $S := \{0,1\}^n$. Then the set $E := \{\mathbf{1}_1, \ldots, \mathbf{1}_n\}$, of unit vectors in $\mathbb{R}^n$ is a set of all edge directions of $\text{conv}(S)$ which is simply the n-cube $[0,1]^n$. The set E is easily constructible in time polynomial in n, a membership oracle for S is trivially and efficiently realizable, and $0 \in S$ is a trivial initial feasible point.

Let $f : \mathbb{R}^d \to \mathbb{R}$ be the l_2 norm squared given by $f(y) = \|y\|_2^2 = \sum_{i=1}^d y_i^2$. Then the comparison between $f(y)$ and $f(z)$ can be done for $y, z \in \mathbb{Z}^d$ in time polynomial in $\langle y, z \rangle$, providing an efficient realization of a comparison oracle for f.

This models (2.7) as a convex maximization problem $\max\{f(Wx) : x \in S\}$ which can be solved in polynomial time by the algorithm of Theorem 2.22. □

2.5.2 Matroids and matroid intersections

We now revisit the matroid and matroid intersection problems from Section 1.2.1. We consider convex maximization over bases or independent sets of matroids and two matroid intersections. The matroids can be presented either by independence oracles or by basis oracles. We show the following results on maximizing $f(Wx)$ for convex f, with the dependency of the running time on W indicated. These results are extended in Section 6.1 to matroid optimization with arbitrary nonlinear functions.

Single matroid	Two matroid intersections
Corollary 2.25	Corollary 2.26
Polynomial in $\langle W \rangle$	Polynomial in W

As demonstrated in Section 1.2.1, under a basis oracle presentation, exponentially many queries may be needed even for linear optimization. So we assume that a basis oracle presentation is always endowed with some initial basis to start with. We will use the following simple lemma which establishes the polynomial time equivalence of independence oracle and basis oracle endowed with initial basis.

Lemma 2.24. *A basis oracle can be realized and some basis obtained for a matroid on n elements presented by an independence oracle in time polynomial in n. Conversely, an independence oracle can be realized for a matroid on n elements presented by a basis oracle and endowed with some basis in time polynomial in n.*

Proof. Let $M = (N, \mathcal{B})$ be a matroid and let $\mathcal{I}$ be its family of independent sets.

Suppose M is presented by an independence oracle. A basis oracle is realized as follows: $B \subseteq N$ is a basis if and only if $B \in \mathcal{I}$ and $B \uplus \{i\} \notin \mathcal{I}$ for all $i \in N \setminus B$ which can be checked using the independence oracle. Some basis B can be obtained by the greedy algorithm: initialize $I := \emptyset$; while possible, pick an element $i \in N \setminus I$ such that $I := I \uplus \{i\} \in \mathcal{I}$, set $I := I \uplus \{i\}$, and repeat; output $B := I$.

Suppose M is presented by a basis oracle and endowed with basis B. An independence oracle is realized as follows: given nonempty $I = \{i_1, \ldots, i_r\} \subseteq N$, form a sequence of bases $B = B_0, B_1, \ldots$ as follows: having produced B_{j-1}, check if there is $k \in B_{j-1} \setminus I$ such that $B_{j-1} \cup \{i_j\} \setminus \{k\} \in \mathcal{B}$; if there is, set $B_j := B_{j-1} \cup \{i_j\} \setminus \{k\}$ and continue; else stop. It is not hard to show that I is independent in M if and only if the sequence has been constructed successfully all the way up to B_r. □

Matroids and forests

Theorem 2.22 implies the following result of [48], [81] that maximizing $f(Wx)$ over a matroid with f convex and W binary encoded can be done in polynomial time.

Corollary 2.25. *For every fixed d there is an algorithm that, given n element matroid M presented by either independence oracle or basis oracle endowed with some basis, integer $d \times n$ matrix W, and convex function $f : \mathbb{Z}^d \to \mathbb{R}$ presented by a comparison oracle, solves in time polynomial in $\langle W \rangle$, the convex matroid problem:*

$$\max \big\{ f(Wx) : x \in S \big\}, \tag{2.8}$$

over the bases $S := \{\mathbf{1}_B : B \in \mathcal{B}\}$ or independent sets $S := \{\mathbf{1}_I : I \in \mathcal{I}\}$ of M.

Proof. Consider first problem (2.8) with $S := \{\mathbf{1}_B : B \in \mathcal{B}\}$ the set of matroid bases. We claim that $E := \{\mathbf{1}_i - \mathbf{1}_j : 1 \leq i < j \leq n\}$ is a set of all edge directions of the matroid polytope $P := \text{conv}(S)$. Consider any edge $e = [y, y']$ of P, with $y, y' \in S$, and let $B := \text{supp}(y)$ and $B' := \text{supp}(y')$ be the corresponding bases. 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 this is not the case. Let $h \in \mathbb{R}^n$ be uniquely maximized over P at e, 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 $hy'' = hy - h_i + h_j \geq hy$. 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, which is a contradiction.

Clearly, E can be constructed in time polynomial in n. By Lemma 2.24, we may assume that M is presented by a basis oracle and endowed with some basis B. So we have a membership oracle for S and initial point $\mathbf{1}_B \in S$. Therefore, the algorithm of Theorem 2.22 can be used to solve the matroid problem in polynomial time.

Consider next problem (2.8) with $S := \{\mathbf{1}_I : I \in \mathcal{I}\}$ the set of independent sets. Similar arguments show that a set of all edge directions of $\text{conv}(S)$ is provided by the following:

$$E := \big\{\mathbf{1}_i - \mathbf{1}_j : 1 \leq i < j \leq n\big\} \uplus \big\{\mathbf{1}_i : 1 \leq i \leq n\big\}.$$

Clearly, E can be constructed in time polynomial in n. By Lemma 2.24, we may assume that M is presented by an independence oracle. So we have a membership oracle for S and initial point $0 \in S$. Therefore, the algorithm of Theorem 2.22 can be used to solve the matroid problem in polynomial time in this case as well. □

A concrete application of Corollary 2.25 is for maximum norm forest problems.

Example 1.1 (revisited; maximum norm forest, see Figure 1.1). Let G be graph with n edges and M its graphic matroid, W integer $d \times n$ matrix, and f the l_p norm on $\mathbb{R}^d$. The problem is $\max\{f(Wx) : x \in S\}$ with S the set of either all forests (independent sets of M) or inclusion-maximal forests (bases of M). An independence oracle for M is easily efficiently realizable. The norm f is convex and the comparison of $f(y), f(z)$ for $y, z \in WS$ can be done in time polynomial in $\langle y, z, p\rangle$, by comparing $\|y\|_\infty, \|z\|_\infty$ for $p = \infty$ and $\|y\|_p^p$ and $\|z\|_p^p$ for positive integer p. By Corollary 2.25, this problem is solvable in time polynomial in $\langle W, p\rangle$.

Matroid intersections and bipartite matchings

A classical result in combinatorial optimization is that linear optimization over two matroid intersections can be done in polynomial time [33]. Theorem 2.10 implies the following result of [11] that maximizing $f(Wx)$ over two matroid intersections with f convex and W unary encoded can be done in polynomial time as well.

Corollary 2.26. *For every fixed d there is an algorithm that, given two n element matroids, each presented by either independence oracle or basis oracle endowed with some*

basis, integer $d \times n$ matrix W, and convex $f : \mathbb{Z}^d \to \mathbb{R}$ presented by comparison oracle, solves in time polynomial in W the following convex matroid intersection problem over the set S of common bases or common independent sets:

$$\max\{f(Wx) : x \in S\}, \quad S := \{\mathbf{1}_B : B \in \mathcal{B}_1 \cap \mathcal{B}_2\} \quad \textit{or} \quad S := \{\mathbf{1}_I : I \in \mathcal{I}_1 \cap \mathcal{I}_2\}.$$

Proof. By Lemma 2.24, we may assume that both matroids are presented by independence oracles. As is well known (see [33]), such presentation enables linear optimization over the set S of either common bases or common independent sets of two matroids in polynomial time, providing a realization of a linear-optimization oracle for S. The result now follows by applying the algorithm of Theorem 2.10. □

A concrete application of Corollary 2.26 is for convex assignment problems [12].

Example 2.27 (convex assignment). Consider again the *assignment problem* of Example 2.4, where the feasible set S consists of the $m \times m$ permutation matrices (also interpreted as the perfect matchings in the complete bipartite graph $K_{m,m}$). Let S_1 be the set of $\{0,1\}$-valued $m \times m$ matrices with one 1 per row and let S_2 be the set of $\{0,1\}$-valued $m \times m$ matrices with one 1 per column. Then S_1, S_2 are the sets of indicators of bases of matroids M_1, M_2 on $N := \{(i,j) : 1 \le i,j \le m\}$ and $S = S_1 \cap S_2$ is the set of common bases. Independence oracles for M_1, M_2 are easily efficiently realizable. Corollary 2.26 then implies that the *convex assignment problem* $\max\{f(Wx) : x \in S\}$ can be solved for any given integer $d \times n$ matrix W and convex $f : \mathbb{Z}^d \to \mathbb{R}$ presented by comparison oracle in time polynomial in W.

Under the hypothesis P $\neq$ NP, Corollary 2.26 cannot be extended to $k \ge 3$ matroids, since already the linear matroid intersection problem for $k = 3$ includes the *traveling salesman problem* as a special case. To see this, consider the related NP-complete problem of deciding if a given digraph $G = (V, N)$ with two vertices s, t has a Hamiltonian dipath from s to t. Define three matroids on N as follows: let M_1 be the graphic matroid of the undirected graph underlying G; let M_2 be the matroid with $B \subseteq N$ a basis if, in the subdigraph (V, B), the indegree of s is 0 and the indegree of any $v \neq s$ is 1; let M_3 be the matroid with $B \subseteq N$ a basis if, in (V, B), the outdegree of t is 0 and the outdegree of any $v \neq t$ is 1. Then G has a Hamiltonian path if and only if the three matroids have a common basis.

2.5.3 Vector partitioning and clustering

The vector partitioning problem seeks to partition n items among p players so as to maximize social utility (see Figure 2.5 for a small example in the special case of identical players). Each player i has an integer $q \times n$ utility matrix ${}^iU$ whose kth column ${}^iU^k$ is the utility of item k to player i under q criteria, with entry ${}^iU_{j,k}$ the utility of item k to player i under criterion j. The utility matrix of an ordered partition $\pi = (\pi_1, \dots, \pi_p)$ of the set $\{1, \dots, n\}$ of items is the $q \times p$ matrix:

$$U^\pi := \Big(\sum_{k \in \pi_1}^{1} U^k, \dots, \sum_{k \in \pi_p}^{p} U^k \Big),$$

whose ith column is the sum $\sum_{k\in\pi_i}^{i} U^k$ of utility vectors of items assigned to player i under π, which is the total utility to player i under π. The social utility of π is the balancing of the player utilities by a convex function $f : \mathbb{Z}^{q\times p} \cong \mathbb{Z}^{pq} \to \mathbb{R}$:

$$f(U^\pi) := f\Big(\sum_{k\in\pi_1}^{1} U_{1,k}, \ldots, \sum_{k\in\pi_1}^{1} U_{q,k}, \ldots, \sum_{k\in\pi_p}^{p} U_{1,k}, \ldots, \sum_{k\in\pi_p}^{p} U_{q,k}\Big).$$

In the constrained version, the partition must be of a given *shape*, that is, the number $|\pi_i|$ of items that player i gets has to be a given positive integer λ_i (with $\sum \lambda_i = n$). In the unconstrained version, the number of items per player is unrestricted. If the number p of

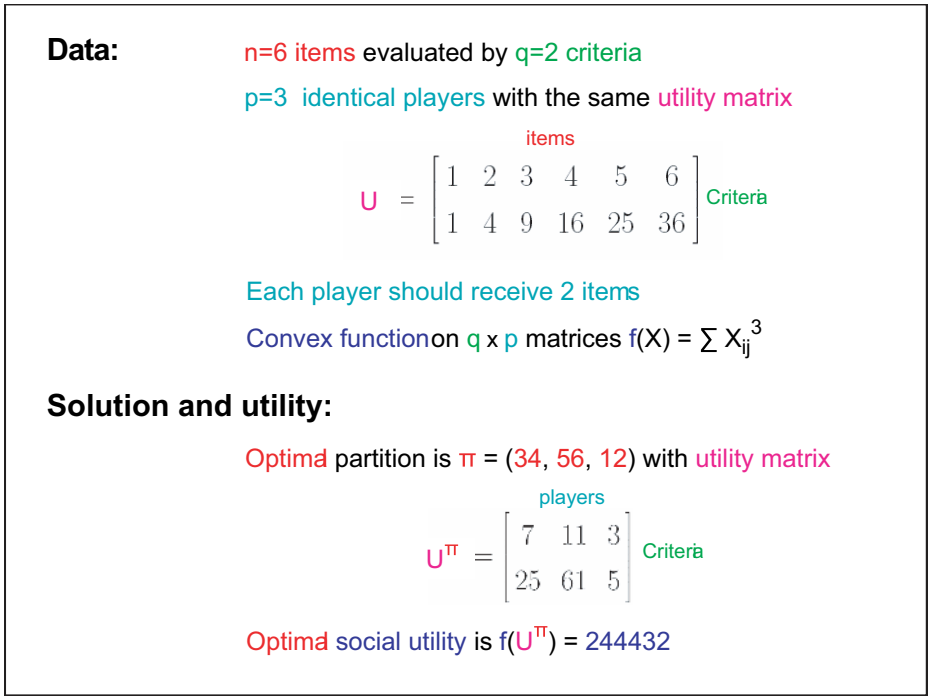


Figure 2.5: Vector partitioning example

players or the number q of criteria is variable then the problem is NP-hard [57]. We now use Theorem 2.19 to show that when both p and q are fixed, both constrained and unconstrained versions of the partitioning problem are polynomial time solvable. The following corollary extends results of [39], [57], [83], [84] on identical players to nonidentical players as well.

Corollary 2.28. *For every fixed numbers p of players and q of criteria, there is an algorithm that, given utility matrices ${}^1U, \ldots, {}^pU \in \mathbb{Z}^{q\times n}$, $1 \le \lambda_1, \ldots, \lambda_p \le n$, and convex function $f : \mathbb{Z}^{pq} \to \mathbb{R}$ presented by a comparison oracle, solves the constrained or unconstrained partitioning problem in time polynomial in $\langle {}^1U, \ldots, {}^pU\rangle$.*

Proof. We demonstrate only the constrained problem, the unconstrained version being similar and simpler. There is an obvious one-to-one correspondence between partitions and matrices $x \in \{0,1\}^{n\times p}$ satisfying $\sum_{i=1}^{p} x_{k,i} = 1$ for all k, where partition π corresponds to matrix x with $x_{k,i} = 1$ if $k \in \pi_i$ and $x_{k,i} = 0$ otherwise. Then the feasible partitions become the integer points of a transportation problem,

$$S := \left\{x \in \mathbb{Z}_+^{n\times p} : \textstyle\sum_{k=1}^{n} x_{k,i} = \lambda_i,\ \sum_{i=1}^{p} x_{k,i} = 1\right\}. \tag{2.9}$$

Identify $\mathbb{Z}^{n\times p} \cong \mathbb{Z}^{np}$ by the coordinate order $x = (x_{1,1}, \ldots, x_{1,p}, \ldots, x_{n,1}, \ldots, x_{n,p})$. Let A denote the $(p+n) \times np$ matrix defining the system of equations in (2.9). Then A is the adjacency matrix of the complete bipartite graph $K_{p,n}$ and is totally unimodular. Moreover, $c \in \mathbb{Z}^{np}$ is a circuit of A if and only if it is supported on the set of edges of a circuit of $K_{p,n}$ with values ± 1 alternating along the circuit. Since p is fixed, the number of circuits of $K_{p,n}$ satisfies $\sum_{i=2}^{p} \binom{p}{i}\binom{n}{i} i!(i-1)! = O(n^p)$, and therefore the set $\mathcal{C}(A)$ of circuits of A can be constructed in polynomial time.

Define an integer $pq \times np$ matrix W by setting $W_{(i,j),(k,i)} :=^i U_{j,k}$ for all i, j, k, and setting all other entries to zero. Then for any partition π and its corresponding vector $x \in \{0,1\}^{np}$, we have $\sum_{k\in\pi_i}^{i} U_{j,k} = (Wx)_{(i,j)}$ for all i, j. This models the problems as a convex program, with W having a fixed number $d := pq$ of rows:

$$\max\left\{f(Wx) : x \in S\right\}.$$

Now, apply the algorithm of Theorem 2.19 to data consisting of the matrix A, its set of circuits $\mathcal{C}(A)$ computed as explained above, $b := (\lambda_1, \ldots, \lambda_p, 1, \ldots, 1)$, $l := 0$, and $u := \mathbf{1}$, and solve the vector partitioning problem in polynomial time. □

A concrete important application of vector partitioning is for clustering.

Example 2.29 (minimal variance clustering). This problem has numerous applications in the analysis of statistical data: group n observed points $u^1, \ldots, u^n$ in $\mathbb{R}^q$ into p clusters $\pi_1, \ldots, \pi_p$ so as to minimize the sum of cluster variances as follows:

$$\sum_{i=1}^{p} \frac{1}{|\pi_i|} \sum_{k\in\pi_i} \left\| u^k - \left(\frac{1}{|\pi_i|} \sum_{k\in\pi_i} u^k\right)\right\|^2.$$

Consider instances where there are $n = pc$ points and the desired clustering is balanced, that is, the clusters should have equal size c. Assume that the observation points are rational, and then suitably scale them to become integer. Suitable manipulation of the sum of variances expression above shows that the problem is equivalent to a constrained vector partitioning problem, with p identical players (the clusters) having a common $q \times n$ utility matrix $U := (u^1, \ldots, u^n)$ with $\lambda_i = c$ for all i and with $f : \mathbb{Z}^{pq} \to \mathbb{R}$ (to be maximized) being the l_2 norm squared as follows:

$$f(z) := \|z\|^2 := \sum_{j=1}^{q} \sum_{i=1}^{p} |z_{j,i}|^2.$$

By Corollary 2.28, we can cluster optimally in polynomial time for all fixed p, q.

Notes

The ellipsoid method of Yudin and Nemirovskii [101] is a powerful tool which, as is well known, led to the first polynomial time algorithm for linear programming by Khachiyan [59] and to the oracle equivalence of separation and linear optimization of Grötschel, Lovász, and Schrijver on their many applications in combinatorial optimization in [44], [72]. As recently shown in [79], without significant increase of computational effort, the ellipsoid method can also be augmented with the computation of proximity certificates for a variety of problems with convex structure including convex minimization, variational inequalities with monotone operators, and computation of Nash equilibria in convex games. But in this monograph, we use it only in deriving Theorem 2.10 in Section 2.2, which has an alternative derivation without the ellipsoid method given in Section 2.3.3. The main result of this chapter, which is Theorem 2.16 on convex maximization using edge directions, is from [83]. It extends earlier results of [81], [84] and is further extended in [23]. It exploits efficient vertex enumeration using the nonstandard information coming from the edge directions. The problem of vertex enumeration of a polyhedron presented by linear inequalities, which is of different flavor, has been studied extensively in the literature. An output-sensitive algorithm for vertex enumeration is given in [5] by Avis and Fukuda, and the hardness of the problem for possibly unbounded polyhedra is shown in [60]. Theorem 2.19 on convex integer maximization over a system defined by a totally unimodular matrix given the circuits of the matrix can be extended to other classes of matrices which define integer polyhedra. Let us mention the important class of *balanced matrices* which strictly contains totally unimodular matrices, studied extensively by Conforti and Cornuéjols, see [19] and the references therein. The results on efficient convex maximization over matroids and matroid intersections are from [48], [81], and [11], respectively. The fruitful interplay between convexity and matroids extends more generally to submodular functions, polymatroids, and beyond; see the books [37] by Fujishige, and [77] by Murota for further details on this line of research. The applications for vector partitioning and clustering come from and extend the series of the papers [39], [57], [83], [84].

3 Nonlinear Integer Programming

In this chapter, we consider nonlinear integer programming, that is, nonlinear optimization over a set of integer points given by linear inequalities, of the form:

$$S := \{x \in \mathbb{Z}^n : Ax = b,\ l \le x \le u\}, \tag{3.1}$$

where A is an integer $m \times n$ matrix, $b \in \mathbb{Z}^m$, and $l, u \in \mathbb{Z}^n_\infty$ with $\mathbb{Z}_\infty = \mathbb{Z} \uplus \{\pm\infty\}$.

A fundamental object in the theory of integer programming is the *Graver basis* introduced by Graver already back in 1975 [41]. However, only very recently, in the series of papers [23], [24], [50], [51], it was established that the Graver basis can be used to solve linear and nonlinear integer programming problems in polynomial time. In this chapter, we describe these important new developments. In Section 3.1, we define the Graver basis and describe some of its basic properties. In Section 3.2, we prove the main result of this chapter, Theorem 3.12, showing that the Graver basis enables to minimize separable convex functions over sets of the form (3.1) in polynomial time. In Section 3.3, we discuss several specializations and extensions of this result to other types of objective functions. We conclude with a short description in Section 3.4 of a simple finite procedure for computing the Graver basis.

The following table enables quick navigation among some of the theorems in this chapter providing polynomial time optimization over sets of the form (3.1).

$\min wx$ (linear objective)	$\min f(x)$ f separable convex	$\max f(Wx)$ f convex	$\min f(Wx)+g(x)$ f,g separable convex
Theorem 3.13	Theorem 3.12	Theorem 3.16	Theorem 3.17

3.1 Graver bases

We begin with the definition of the Graver basis and some of its basic properties. Throughout this section A is an integer $m \times n$ matrix. Recall the following terminology from Section 2.3.4. The *lattice* of A is $\mathcal{L}(A) := \{x \in \mathbb{Z}^n : Ax = 0\}$. The set of nonzero elements in $\mathcal{L}(A)$ is denoted by $\mathcal{L}^*(A) := \{x \in \mathbb{Z}^n : Ax = 0,\ x \neq 0\}$. We use the partial order $\sqsubseteq$ on $\mathbb{R}^n$ defined as follows: for $x, y \in \mathbb{R}^n$, we write $x \sqsubseteq y$ and say that x is *conformal* to y if $x_i y_i \ge 0$ (that is, x and y lie in the same orthant) and $|x_i| \le |y_i|$ for $i = 1, \ldots, n$. A finite sum $u := \sum_i v_i$ of vectors in $\mathbb{R}^n$ is *conformal* if $v_i \sqsubseteq u$ for all i and hence all summands lie in the same orthant.

A simple extension of a classical lemma of Gordan [40] implies that every subset of $\mathbb{Z}^n$ has finitely many $\sqsubseteq$-minimal elements. The following definition is from [41].

Definition 3.1. The *Graver basis* of an integer matrix A is defined to be the finite set $\mathcal{G}(A) \subset \mathbb{Z}^n$ of $\sqsubseteq$-minimal elements in $\mathcal{L}^*(A) = \{x \in \mathbb{Z}^n : Ax = 0,\ x \neq 0\}$.

Recall also from Section 2.3.4 that a *circuit* of A is an element $c \in \mathcal{L}^*(A)$ whose support $\mathrm{supp}(c)$ is minimal under inclusion and whose entries are relatively prime. Like the set $\mathcal{C}(A)$ of circuits, the Graver basis $\mathcal{G}(A)$ is centrally symmetric, that is, $g \in \mathcal{G}(A)$ if and only if $-g \in \mathcal{G}(A)$. It follows directly from the definitions that for every matrix A, the set of circuits is contained in the Graver basis, that is, $\mathcal{C}(A) \subseteq \mathcal{G}(A)$. The converse is typically false. For instance, already for the tiny 1×3 matrix $A := (1\ 2\ 1)$, the Graver basis strictly contains the set of circuits:

$$\mathcal{G}(A) = \pm\{(2,-1,0),(0,-1,2),(1,0,-1),(1,-1,1)\} = \mathcal{C}(A) \uplus \pm\{(1,-1,1)\}.$$

We have the following fundamental property of the Graver basis.

Lemma 3.2. *Every vector $x \in \mathcal{L}^*(A)$ is a conformal sum $x = \sum_i g_i$ of Graver basis elements $g_i \in \mathcal{G}(A)$, with some elements possibly appearing with repetitions.*

Proof. By induction on the well partial order $\sqsubseteq$. Consider any $x \in \mathcal{L}^*(A)$. If it is $\sqsubseteq$-minimal in $\mathcal{L}^*(A)$ then $x \in \mathcal{G}(A)$ by definition of the Graver basis and we are done. Otherwise, there is an element $g \in \mathcal{G}(A)$ such that $g \sqsubset x$. Set $y := x - g$. Then $y \in \mathcal{L}^*(A)$ and $y \sqsubset x$, so by induction there is a conformal sum $y = \sum_i g_i$ with $g_i \in \mathcal{G}(A)$ for all i. Now, $x = g + \sum_i g_i$ is a conformal sum of x. □

Recall from Section 2.4 that an *augmentation oracle* for $S \subseteq \mathbb{Z}^n$ is one that, queried on $x \in S$ and $w \in \mathbb{Z}^n$, either asserts that x is optimal for the optimization problem $\max\{wx : x \in S\}$ or returns $\hat{x} \in S$ satisfying $w\hat{x} > wx$. Definition 3.1, made by Graver already back in 1975 [41], is motivated by the following lemma that shows that $\mathcal{G}(A)$ enables to realize an augmentation oracle for a set S of the form (3.1).

Lemma 3.3. *There is an algorithm that, given the Graver basis $\mathcal{G}(A)$ of an integer $m \times n$ matrix A, $l, u \in \mathbb{Z}^n_\infty$, and $x, w \in \mathbb{Z}^n$ with $l \le x \le u$, in time polynomial in $\langle \mathcal{G}(A), l, u, x, w \rangle$, either asserts that x is optimal for the linear optimization problem $\max\{wz : z \in \mathbb{Z}^n,\ Az = b,\ l \le z \le u\}$ with $b := Ax$ or returns a better point $\hat{x}$.*

Proof. Suppose that x is not optimal. Let x^* be any better feasible point and put $h := x^* - x$. Then $Ah = b - b = 0$ so $h \in \mathcal{L}^*(A)$ and hence, by Lemma 3.2, there is a conformal sum $h = \sum_i g_i$ with $g_i \in \mathcal{G}(A)$ for all i. Now, $l \le x, x + h = x^* \le u$ and $g_i \sqsubseteq h$ imply that $l \le x + g_i \le u$ for all i. Also, $A(x + g_i) = Ax = b$ for all i. Therefore, $x + g_i$ is feasible for all i. Now, $0 < wx^* - wx = wh = \sum_i wg_i$ implies that $wg_i > 0$ for some g_i in this sum. Therefore, $x + g_i$ is feasible and better.

So the algorithm is very simple: if $wg > 0$ for some $g \in G$ with $l \le x + g \le u$ then $\hat{x} := x + g$ is an easily computable better point; otherwise, x is optimal. □

Combining Lemmas 3.3 and 2.20, it is possible to use the Graver basis to solve the linear optimization problem over $S = \{x \in \mathbb{Z}^n : Ax = b,\ l \le x \le u\}$. However, the running time of this procedure is polynomial only in the *unary length* of the radius $\rho(S)$ of S. In integer programming, this is *unsatisfactory* since $\rho(S)$ is typically exponential in the natural binary length of the data $\langle A, l, u, b \rangle$ defining S, and the resulting linear optimization algorithm is *not* naturally polynomial.

However, in Section 3.2 we describe the recent important developments from [23], [24], [50], [51], that establish much stronger results, showing that, in fact, the Graver basis *does* enable to solve linear and nonlinear optimization problems over sets of the from (3.1) in time which is genuinely polynomial, in the natural *binary length* of the data $\langle A, l, u, b\rangle$ which defines the feasible set (along with the binary length of the given Graver basis).

We need the following stronger form of Lemma 3.2 which basically follows from the integer analogs of Carathéodory's theorem established in [20] and [94].

Lemma 3.4. *Every $x \in \mathcal{L}^*(A)$ is a conformal sum $x = \sum_{i=1}^t \lambda_i g_i$ which involves $t \leq 2n-2$ nonnegative integer coefficients λ_i and Graver basis elements $g_i \in \mathcal{G}(A)$.*

Proof. We prove the slightly weaker bound $t \leq 2n-1$ from [20]. A proof of the stronger bound can be found in [94]. Consider any $x \in \mathcal{L}^*(A)$ and let $g_1, \ldots, g_s$ be all elements of $\mathcal{G}(A)$ lying in the same orthant as x. Consider the linear program:

$$\max\Big\{\sum_{i=1}^s \lambda_i : x = \sum_{i=1}^s \lambda_i g_i,\ \lambda_i \in \mathbb{R}_+\Big\}. \tag{3.2}$$

By Lemma 3.2, the point x is a nonnegative linear combination of the g_i and hence the program (3.2) is feasible. Since all g_i are nonzero and in the same orthant as x, program (3.2) is also bounded. As is well known, it then has a *basic* optimal solution, that is, an optimal solution $\lambda_1, \ldots, \lambda_s$ with at most n of the λ_i nonzero. Let

$$y := \sum \big(\lambda_i - \lfloor \lambda_i \rfloor\big) g_i = x - \sum \lfloor \lambda_i \rfloor g_i.$$

If $y = 0$ then $x = \sum \lfloor \lambda_i \rfloor g_i$ is a conformal sum of at most n of the g_i and we are done. Otherwise, $y \in \mathcal{L}^*(A)$ and y lies in the same orthant as x, and hence, by Lemma 3.2 again, $y = \sum_{i=1}^s \mu_i g_i$ with all $\mu_i \in \mathbb{Z}_+$. Then $x = \sum(\mu_i + \lfloor \lambda_i \rfloor) g_i$ and hence, since the λ_i form an optimal solution to (3.2), we have $\sum(\mu_i + \lfloor \lambda_i \rfloor) \leq \sum \lambda_i$. Therefore, $\sum \mu_i \leq \sum(\lambda_i - \lfloor \lambda_i \rfloor) < n$ with the last inequality holding since at most n of the λ_i are nonzero. Since the μ_i are integers, at most $n-1$ of them are nonzero. So $x = \sum(\mu_i + \lfloor \lambda_i \rfloor) g_i$ is a conformal sum of x involving at most $2n-1$ of the g_i. □

We remark that the smallest possible bound $t(n)$ that could possibly replace $2n-2$ in the statement of Lemma 3.4, sometimes called the *integer Carathéodory number*, is yet unknown. While it has been conjectured in [20] that $t(n) = n$, it was eventually shown in [17] that in fact $t(n) \geq \lfloor \frac{7}{6} n \rfloor > n$ for any dimension $n \geq 6$.

The Graver basis also enables to check finiteness of feasible integer programs.

Lemma 3.5. *Let $\mathcal{G}(A)$ be the Graver basis of matrix A and let $l, u \in \mathbb{Z}^n_\infty$. If there is some $g \in \mathcal{G}(A)$ satisfying $g_i \leq 0$ whenever $u_i < \infty$ and $g_i \geq 0$ whenever $l_i > -\infty$ then every set of the form $S := \{x \in \mathbb{Z}^n : Ax = b,\ l \leq x \leq u\}$ is either empty or infinite, whereas if there is no such g, then every set S of this form is finite. Clearly, the existence of such g can be checked in time polynomial in $\langle \mathcal{G}(A), l, u\rangle$.*

Proof. First suppose there exists such g. Consider any such S. Suppose S contains some point x. Then for all $\lambda \in \mathbb{Z}_+$, we have $l \leq x + \lambda g \leq u$ and $A(x + \lambda g) = Ax = b$

and hence $x + \lambda g \in S$, so S is infinite. Next, suppose S is infinite. Then the polyhedron $P := \{x \in \mathbb{R}^n : Ax = b,\ l \le x \le u\}$ is unbounded and hence, as is well known, has a recession vector, that is, a nonzero h, which we may assume to be integer, such that $x + \alpha h \in P$ for all $x \in P$ and $\alpha \ge 0$. This implies that $h \in \mathcal{L}^*(A)$ and that $h_i \le 0$ whenever $u_i < \infty$ and $h_i \ge 0$ whenever $l_i > -\infty$. So h is a conformal sum $h = \sum g_i$ of vectors $g_i \in \mathcal{G}(A)$, each of which also satisfies $g_i \le 0$ whenever $u_i < \infty$ and $g_i \ge 0$ whenever $l_i > -\infty$, providing such g. □

3.2 Efficient separable convex minimization

In this section, we consider the following nonlinear integer minimization problem:

$$\min\{f(x) : x \in \mathbb{Z}^n,\ Ax = b,\ l \le x \le u\} \tag{3.3}$$

with A integer $m \times n$ matrix, $b \in \mathbb{Z}^m$, $l, u \in \mathbb{Z}^n_\infty$, and $f : \mathbb{Z}^n \to \mathbb{Z}$ *separable convex function*, that is, $f(x) = \sum_{j=1}^n f_j(x_j)$ with each $f_j : \mathbb{Z} \to \mathbb{Z}$ univariate convex.

The main result here is Theorem 3.12: given the Graver basis $\mathcal{G}(A)$ of A, we can solve problem (3.3) for any separable convex function in polynomial time.

We prove a sequence of six lemmas and then combine them to prove Theorem 3.12. We start with two simple lemmas about univariate convex functions. The first lemma establishes a certain *supermodularity* property of such functions.

Lemma 3.6. *Let $f : \mathbb{R} \to \mathbb{R}$ be a univariate convex function, let r be a real number, and let $s_1, \ldots, s_m$ be real numbers satisfying $s_i s_j \ge 0$ for all i, j. Then we have the following:*

$$f\Big(r + \sum_{i=1}^m s_i\Big) - f(r) \ge \sum_{i=1}^m \big(f\big(r + s_i\big) - f(r)\big).$$

Proof. Define a new function $g : \mathbb{R} \to \mathbb{R}$ by $g(x) := f(r+x) - f(r)$ for all $x \in \mathbb{R}$. Then g is also convex and $g(0) = 0$. Therefore, for all $x \in \mathbb{R}$ and $0 \le \mu \le 1$:

$$g(\mu x) = g\big((1-\mu)0 + \mu x\big) \le (1-\mu)g(0) + \mu g(x) = \mu g(x).$$

Now assume that not all s_i are zero else the claim of the lemma is trivial. Define

$$\lambda_i := \frac{s_i}{\sum_{j=1}^m s_j}, \quad i = 1, \ldots, m.$$

Then $\sum_{i=1}^m \lambda_i = 1$ and $0 \le \lambda_i \le 1$ for all i since all s_i have the same sign. Therefore,

$$\sum_{i=1}^m g\big(s_i\big) = \sum_{i=1}^m g\Big(\lambda_i \sum_{j=1}^m s_j\Big) \le \sum_{i=1}^m \lambda_i g\Big(\sum_{j=1}^m s_j\Big) = g\Big(\sum_{j=1}^m s_j\Big),$$

and hence, as claimed

$$\sum_{i=1}^m \big(f\big(r + s_i\big) - f(r)\big) = \sum_{i=1}^m g\big(s_i\big) \le g\Big(\sum_{j=1}^m s_j\Big) = f\Big(r + \sum_{j=1}^m s_j\Big) - f(r).$$

□

The second lemma shows that univariate convex functions can be minimized efficiently over an interval of integers using repeated bisections.

Lemma 3.7. *There is an algorithm that, given any two integer numbers $r \le s$ and any univariate convex function $f : \mathbb{Z} \to \mathbb{R}$ given by a comparison oracle, solves in time polynomial in $\langle r, s\rangle$ the following univariate integer minimization problem:*

$$\min\big\{f(\lambda) : \lambda \in \mathbb{Z},\ r \le \lambda \le s\big\}.$$

Proof. If $r = s$ then $\lambda := r$ is optimal. So assume that $r \le s - 1$. Consider the integers:

$$r \le \left\lfloor \frac{r+s}{2} \right\rfloor < \left\lfloor \frac{r+s}{2} \right\rfloor + 1 \le s.$$

Use the oracle of f to compare $f(\lfloor \frac{r+s}{2} \rfloor)$ and $f(\lfloor \frac{r+s}{2} \rfloor + 1)$. By convexity of f:

$$f\Big(\Big\lfloor \frac{r+s}{2} \Big\rfloor\Big) = f\Big(\Big\lfloor \frac{r+s}{2} \Big\rfloor + 1\Big) \Longrightarrow \lambda := \Big\lfloor \frac{r+s}{2} \Big\rfloor \text{is a minimum of } f;$$

$$f\Big(\Big\lfloor \frac{r+s}{2} \Big\rfloor\Big) < f\Big(\Big\lfloor \frac{r+s}{2} \Big\rfloor + 1\Big) \Longrightarrow \text{the minimum of } f \text{ is in the interval } \Big[r, \Big\lfloor \frac{r+s}{2} \Big\rfloor\Big];$$

$$f\Big(\Big\lfloor \frac{r+s}{2} \Big\rfloor\Big) > f\Big(\Big\lfloor \frac{r+s}{2} \Big\rfloor + 1\Big) \Longrightarrow \text{the minimum of } f \text{ is in the interval } \Big[\Big\lfloor \frac{r+s}{2} \Big\rfloor + 1, s\Big].$$

Thus, we either obtain the optimal point or bisect the interval $[r, s]$ and repeat. So in $O(\log(s - r)) = O(\langle r, s\rangle)$ bisections, we find an optimal solution $\lambda \in \mathbb{Z} \cap [r, s]$. □

The next two lemmas extend Lemmas 3.6 and 3.7. The first lemma shows the supermodularity of separable convex functions with respect to conformal sums, see also [78].

Lemma 3.8. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be any separable convex function, $x \in \mathbb{R}^n$ any point, and $\sum g_i$ any conformal sum in $\mathbb{R}^n$. Then we have the following inequality:*

$$f\Big(x + \sum g_i\Big) - f(x) \ge \sum \big(f\big(x + g_i\big) - f(x)\big).$$

Proof. Let f_j be univariate convex functions such that $f(x) = \sum_{j=1}^n f_j(x_j)$. Consider any $1 \le j \le n$. Since $\sum g_i$ is a conformal sum, we have $g_{i,j} g_{k,j} \ge 0$ for all i, k and so, setting $r := x_j$ and $s_i := g_{i,j}$ for all i, Lemma 3.6 applied to f_j implies the following:

$$f_j\Big(x_j + \sum_i g_{i,j}\Big) - f_j\big(x_j\big) \ge \sum_i \big(f_j\big(x_j + g_{i,j}\big) - f_j\big(x_j\big)\big). \tag{3.4}$$

Summing the equations (3.4) for $j = 1, \ldots, n$, we obtain the claimed inequality. □

The second lemma enables to find a best improvement step in a given direction.

Lemma 3.9. *There is an algorithm that, given bounds $l, u \in \mathbb{Z}_\infty^n$, direction $g \in \mathbb{Z}^n$, point $x \in \mathbb{Z}^n$ with $l \le x \le u$, and convex function $f : \mathbb{Z}^n \to \mathbb{R}$ presented by comparison oracle, solves in time polynomial in $\langle l, u, g, x\rangle$, the univariate problem:*

$$\min\big\{f(x + \lambda g) : \lambda \in \mathbb{Z}_+,\ l \le x + \lambda g \le u\big\}. \tag{3.5}$$

Proof. Let $S := \{\lambda \in \mathbb{Z}_+ : l \le x+\lambda g \le u\}$ be the feasible set and let $s := \sup S$, which is easy to determine. If $s = \infty$ then conclude that S is infinite and stop. Otherwise, $S = \{0, 1, \ldots, s\}$ and the problem can be solved by the algorithm of Lemma 3.7 minimizing the univariate convex function $h(\lambda) := h(x + \lambda g)$ over S. □

We can now show that the Graver basis of A allows to solve problem (3.3) in polynomial time, provided we are given an initial feasible point to start with. We later show how to find such an initial point as well. Below, and throughout this chapter, $\hat{f}$ denotes the maximum value of $|f(x)|$ over the feasible set and need not be part of the input. An outline of the algorithm is provided in Figure 3.1.

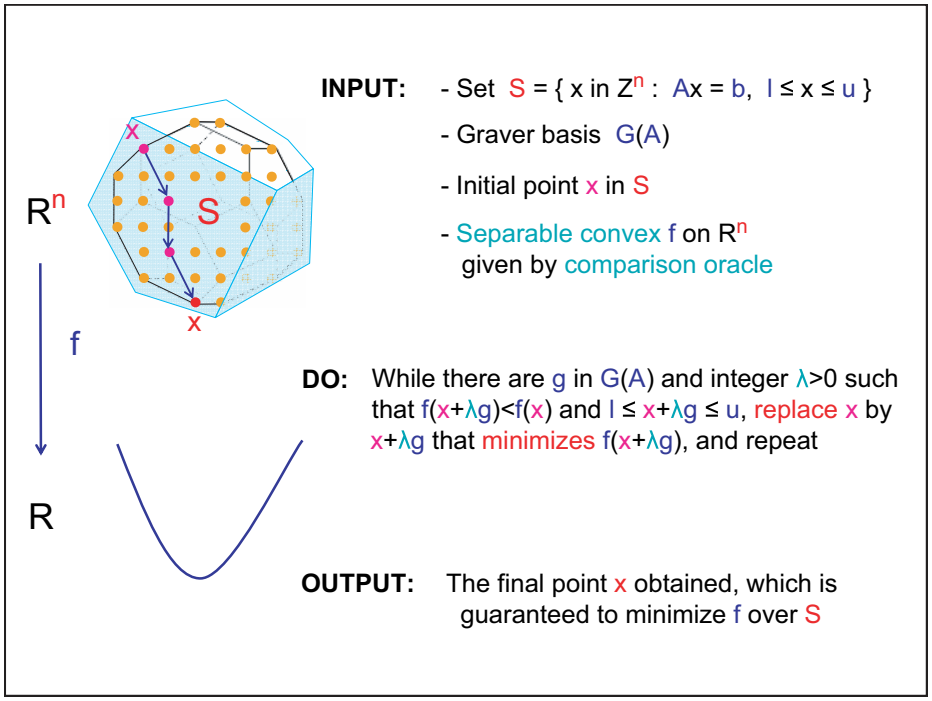


Figure 3.1: Separable convex minimization using Graver bases

Lemma 3.10. *There is an algorithm that, given integer $m \times n$ matrix A, its Graver basis $\mathcal{G}(A)$, vectors $l, u \in \mathbb{Z}^n_\infty$ and $x \in \mathbb{Z}^n$ with $l \le x \le u$, and separable convex function $f : \mathbb{Z}^n \to \mathbb{Z}$ presented by a comparison oracle, solves the integer program:*

$$\min\left\{f(z) : z \in \mathbb{Z}^n,\ Az = b,\ l \le z \le u\right\}, \quad b := Ax, \tag{3.6}$$

in time which is polynomial in the binary length $\langle \mathcal{G}(A), l, u, x, \hat{f}\rangle$ of the data.

Proof. First, apply the algorithm of Lemma 3.5 to $\mathcal{G}(A)$ and l, u and either detect that the feasible set is infinite and stop, or conclude it is finite and continue. Next, produce a

sequence of feasible points $x_0, x_1, \ldots, x_s$ with $x_0 := x$ the given input point, as follows. Having obtained x_k, solve the univariate minimization problem:

$$\min\left\{f\left(x_k+\lambda g\right) : \lambda\in\mathbb{Z}_+,\ g\in\mathcal{G}(A),\ l\le x_k+\lambda g\le u\right\} \tag{3.7}$$

by applying the algorithm of Lemma 3.9 for each $g\in\mathcal{G}(A)$. If the minimal value in (3.7) satisfies $f(x_k+\lambda g)<f(x_k)$ then set $x_{k+1}:=x_k+\lambda g$ and repeat, else stop and output the last point x_s in the sequence. Now, $Ax_{k+1}=A(x_k+\lambda g)=Ax_k=b$ by induction on k, so each x_k is feasible. Since the feasible set is finite and the x_k have decreasing objective values and hence distinct, the algorithm terminates.

We now show that the point x_s output by the algorithm is optimal. Let x^* be any optimal solution to (3.6). Consider any point x_k in the sequence and suppose that it is not optimal. We claim that a new point x_{k+1} will be produced and will satisfy the following:

$$f\left(x_{k+1}\right)-f\left(x^*\right)\le\frac{2n-3}{2n-2}\left(f\left(x_k\right)-f\left(x^*\right)\right). \tag{3.8}$$

By Lemma 3.4, we can write the difference $x^*-x_k=\sum_{i=1}^t\lambda_i g_i$ as conformal sum involving $1\le t\le 2n-2$ elements $g_i\in\mathcal{G}(A)$ with all $\lambda_i\in\mathbb{Z}_+$. By Lemma 3.8:

$$f\left(x^*\right)-f\left(x_k\right)=f\Big(x_k+\sum_{i=1}^t\lambda_i g_i\Big)-f\left(x_k\right)\ge\sum_{i=1}^t\left(f\left(x_k+\lambda_i g_i\right)-f\left(x_k\right)\right).$$

Adding $t(f(x_k)-f(x^*))$ on both sides and rearranging terms, we obtain the following:

$$\sum_{i=1}^t\left(f\left(x_k+\lambda_i g_i\right)-f\left(x^*\right)\right)\le(t-1)\left(f\left(x_k\right)-f\left(x^*\right)\right).$$

Therefore, there is some summand on the left-hand side satisfying the following:

$$f\left(x_k+\lambda_i g_i\right)-f\left(x^*\right)\le\frac{t-1}{t}\left(f\left(x_k\right)-f\left(x^*\right)\right)\le\frac{2n-3}{2n-2}\left(f\left(x_k\right)-f\left(x^*\right)\right).$$

So the point $x_k+\lambda g$ attaining minimum in (3.7) satisfies the following:

$$f\left(x_k+\lambda g\right)-f\left(x^*\right)\le f\left(x_k+\lambda_i g_i\right)-f\left(x^*\right)\le\frac{2n-3}{2n-2}\left(f\left(x_k\right)-f\left(x^*\right)\right),$$

and so indeed $x_{k+1}:=x_k+\lambda g$ will be produced and will satisfy (3.8). This shows that the last point x_s produced and output by the algorithm is indeed optimal.

We proceed to bound the number s of points. Consider any $i<s$ and the intermediate nonoptimal point x_i in the sequence produced by the algorithm. Then $f(x_i)>f(x^*)$ with both values integer, and so repeated use of (3.8) gives the following:

$$\begin{aligned}1\le f\left(x_i\right)-f\left(x^*\right)&=\prod_{k=0}^{i-1}\frac{f\left(x_{k+1}\right)-f\left(x^*\right)}{f\left(x_k\right)-f\left(x^*\right)}\left(f(x)-f\left(x^*\right)\right)\\&\le\left(\frac{2n-3}{2n-2}\right)^i\left(f(x)-f\left(x^*\right)\right),\end{aligned}$$

and therefore

$$i \leq \left(\log \frac{2n-2}{2n-3}\right)^{-1} \log\big(f(x) - f(x^*)\big).$$

Therefore, the number s of points produced by the algorithm is at most one unit larger than this bound, and using a simple bound on the logarithm, we obtain the following:

$$s = O\big(n \log\big(f(x) - f(x^*)\big)\big).$$

Thus, the number of points produced and the total running time are polynomial. □

Next, we show that we can also find an initial feasible point for a given integer program or assert that the given set is empty or infinite, in polynomial time.

Lemma 3.11. *There is an algorithm that, given integer $m \times n$ matrix A, its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}^n_\infty$, and $b \in \mathbb{Z}^m$, in time which is polynomial in $\langle A, \mathcal{G}(A), l, u, b\rangle$, either finds a feasible point $x \in S$ or asserts that S is empty or infinite, where*

$$S := \big\{z \in \mathbb{Z}^n : Az = b,\ l \leq z \leq u\big\}.$$

Proof. Assume that $l \leq u$ and that $l_i < \infty$ and $u_j > -\infty$ for all j, since otherwise there is no feasible point. Also assume that there is no $g \in \mathcal{G}(A)$ satisfying $g_i \leq 0$ whenever $u_i < \infty$ and $g_i \geq 0$ whenever $l_i > -\infty$, since otherwise S is empty or infinite by Lemma 3.5. Now, either detect there is no integer solution to the system of equations $Ax = b$ (without the lower and upper bound constraints) and stop or determine some such solution $\hat{x} \in \mathbb{Z}^n$ and continue; it is well known that this can be done in polynomial time, say, using the Hermite normal form of A, see [90]. Next, define a separable convex function on $\mathbb{Z}^n$ by $f(x) := \sum_{j=1}^n f_j(x_j)$ with

$$f_j(x_j) := \begin{cases} l_j - x_j & \text{if } x_j < l_j, \\ 0 & \text{if } l_j \leq x_j \leq u_j, \\ x_j - u_j & \text{if } x_j > u_j, \end{cases} \quad j = 1, \ldots, n,$$

and extended lower and upper bounds:

$$\hat{l}_j := \min\big\{l_j, \hat{x}_j\big\}, \quad \hat{u}_j := \max\big\{u_j, \hat{x}_j\big\}, \quad j = 1, \ldots, n.$$

Consider the auxiliary separable convex integer program:

$$\min\big\{f(z) : z \in \mathbb{Z}^n,\ Az = b,\ \hat{l} \leq z \leq \hat{u}\big\}. \tag{3.9}$$

First, note that $\hat{x}$ is feasible in (3.9). Next, note that $\hat{l}_j > -\infty$ if and only if $l_j > -\infty$ and $\hat{u}_j < \infty$ if and only if $u_j < \infty$. So there is no $g \in \mathcal{G}(A)$ satisfying $g_i \leq 0$ whenever $\hat{u}_i < \infty$ and $g_i \geq 0$ whenever $\hat{l}_i > -\infty$ and hence the feasible set of (3.9) is finite by Lemma 3.5. Now, apply the algorithm of Lemma 3.10 to (3.9) and obtain an optimal solution x. This can be done in polynomial time since the binary length of $\hat{x}$ and therefore

also of $\hat{l}$, $\hat{u}$ and of the maximum value $\hat{f}$ of $|f(x)|$ over the feasible set of (3.9) are polynomial in the length of the data.

Now, note that every point $z \in S$ is feasible in (3.9), and every point z feasible in (3.9) satisfies $f(z) \geq 0$ with equality if and only if $z \in S$. So, if $f(x) > 0$ then the original set S is empty, whereas if $f(x) = 0$ then $x \in S$ is a feasible point. □

We are finally in position to prove the important result of [50] that the Graver basis allows to solve the nonlinear integer program (3.3) in polynomial time. As before, the bound $\hat{f}$ on $|f(x)|$ over the feasible set need not be part of the input.

Theorem 3.12. *There is an algorithm that, given integer $m \times n$ matrix A, its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}^n_\infty$, $b \in \mathbb{Z}^m$, and separable convex $f : \mathbb{Z}^n \to \mathbb{Z}$ presented by comparison oracle, solves in time polynomial in $\langle A, \mathcal{G}(A), l, u, b, \hat{f}\rangle$ the problem:*

$$\min\left\{f(x) : x \in \mathbb{Z}^n,\ Ax = b,\ l \leq x \leq u\right\}.$$

Proof. First, apply the algorithm of Lemma 3.11 and either conclude that the given integer program is infeasible or the feasible set is infinite and stop or obtain an initial feasible point and continue. Next, apply the algorithm of Lemma 3.10 and either conclude that the feasible set is infinite or obtain an optimal solution. □

3.3 Specializations and extensions

3.3.1 Linear integer programming

Any linear function $wx = \sum_{i=1}^n w_i x_i$ is separable convex. Moreover, an upper bound on $|wx|$ over the feasible set (when finite), which is polynomial in the binary length of the data, readily follows from Cramer's rule. So we have, as an immediate special case of Theorem 3.12, the following important result of [24], that Graver bases enable the polynomial time solution of linear integer programming.

Theorem 3.13. *There is an algorithm that, given an integer $m \times n$ matrix A, its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}^n_\infty$, $b \in \mathbb{Z}^m$, and $w \in \mathbb{Z}^n$, solves in time which is polynomial in $\langle A, \mathcal{G}(A), l, u, b, w\rangle$ the following linear integer programming problem:*

$$\min\left\{wx : x \in \mathbb{Z}^n,\ Ax = b,\ l \leq x \leq u\right\}.$$

3.3.2 Distance minimization

Another useful special case of Theorem 3.12 which is natural in various applications such as image processing, tomography, communication, and error correcting codes is the following result of [50], asserting that Graver bases enable to determine a feasible point which is l_p-closest to a given desired goal point in polynomial time.

Theorem 3.14. *There is an algorithm that, given integer $m \times n$ matrix A, its Graver basis $\mathcal{G}(A)$, positive integer p, vectors $l, u \in \mathbb{Z}^n_\infty$, $b \in \mathbb{Z}^m$, and $\hat{x} \in \mathbb{Z}^n$, solves in time polynomial in p and $\langle A, \mathcal{G}(A), l, u, b, \hat{x}\rangle$, the closest-point problem:*

$$\min\left\{\|x - \hat{x}\|_p : x \in \mathbb{Z}^n,\ Ax = b,\ l \le x \le u\right\}. \tag{3.10}$$

For $p = \infty$, the problem (3.10) can be solved in time polynomial in $\langle A, \mathcal{G}(A), l, u, b, \hat{x}\rangle$.

Proof. For finite p, apply the algorithm of Theorem 3.12 taking f to be the pth power $\|x - \hat{x}\|_p^p$ of the l_p distance. If the feasible set is nonempty and finite (else the algorithm stops) then the maximum value $\hat{f}$ of $|f(x)|$ over it is polynomial in p and $\langle A, l, u, b, \hat{x}\rangle$, and hence an optimal solution can be found in polynomial time.

Consider $p = \infty$. Using Cramer's rule, it is easy to compute an integer ρ with $\langle\rho\rangle$ polynomially bounded in $\langle A, l, u, b\rangle$ that, if the feasible set is finite, provides an upper bound on $\|x\|_\infty$ for any feasible x. Let q be a positive integer satisfying the following:

$$q > \frac{\log n}{\log\left(1 + (2\rho)^{-1}\right)}.$$

Now, apply the algorithm described above for the l_q distance. Assuming that the feasible set is nonempty and finite (else the algorithm stops), let x^* be the feasible point minimizing the l_q distance to $\hat{x}$ obtained by the algorithm. We claim that it also minimizes the l_∞ distance to $\hat{x}$ and hence is the desired optimal solution. Consider any feasible point x. By standard inequalities between the l_∞ and l_q norms:

$$\left\|x^* - \hat{x}\right\|_\infty \le \left\|x^* - \hat{x}\right\|_q \le \|x - \hat{x}\|_q \le n^{\frac{1}{q}}\|x - \hat{x}\|_\infty.$$

Therefore,

$$\left\|x^* - \hat{x}\right\|_\infty - \|x - \hat{x}\|_\infty \le \left(n^{\frac{1}{q}} - 1\right)\|x - \hat{x}\|_\infty \le \left(n^{\frac{1}{q}} - 1\right)2\rho < 1,$$

where the last inequality holds by the choice of q. Since $\|x^* - \hat{x}\|_\infty$ and $\|x - \hat{x}\|_\infty$ are integers, we find that $\|x^* - \hat{x}\|_\infty \le \|x - \hat{x}\|_\infty$. This establishes the claim. □

In particular, for each p the Graver basis enables to solve the integer program:

$$\min\left\{\|x\|_p : x \in \mathbb{Z}^n,\ Ax = b,\ l \le x \le u\right\},$$

which for $p = \infty$ is equivalent to the min-max integer program:

$$\min\left\{\max\left\{\left|x_i\right| : i = 1, \ldots, n\right\} : x \in \mathbb{Z}^n,\ Ax = b,\ l \le x \le u\right\}.$$

3.3.3 Convex integer maximization

We proceed to discuss the *maximization* of a convex function of the composite form $f(Wx)$, with $f : \mathbb{Z}^d \to \mathbb{Z}$ any convex function and W any integer $d \times n$ matrix.

Consider the following pair of polyhedra (with the containment typically strict):

$$\operatorname{conv}\left\{x \in \mathbb{Z}^n : Ax = b, l \le x \le u\right\} \subseteq \left\{x \in \mathbb{R}^n : Ax = b,\ l \le x \le u\right\}.$$

We have shown in Lemma 2.18 that the set of circuits $\mathcal{C}(A)$ is a set of all edge directions of the right-hand side. We now show the analog statement asserting that the Graver basis $\mathcal{G}(A)$ is a set of all edge directions of the left-hand side.

Lemma 3.15. *For any integer $m \times n$ matrix A, $l, u \in \mathbb{Z}^n_\infty$ and $b \in \mathbb{Z}^m$, the Graver basis $\mathcal{G}(A)$ is a set of all edge directions of $P = \text{conv}\{x \in \mathbb{Z}^n : Ax = b,\ l \le x \le u\}$.*

Proof. Consider any edge e of P and pick distinct integer points $x, y \in e$. Then $g := y - x$ is in $\mathcal{L}^*(A)$ and hence Lemma 3.2 implies that $g = \sum_i h_i$ is a conformal sum for suitable $h_i \in \mathcal{G}(A)$. We claim that $x + h_i \in P$ for all i. Indeed, $h_i \in \mathcal{G}(A)$ implies $A(x + h_i) = Ax = b$, and $l \le x, x + g \le u$ and $h_i \sqsubseteq g$ imply $l \le x + h_i \le u$.

Now, let $w \in \mathbb{Z}^n$ be uniquely maximized over P 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. This implies that 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). □

Theorems 3.13 and 2.16 with Lemma 3.15 imply the following result of [23].

Theorem 3.16. *For every fixed d, there is an algorithm that, given integer $m \times n$ matrix A, its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}^n_\infty$, $b \in \mathbb{Z}^m$, integer $d \times n$ matrix W, and convex function $f : \mathbb{Z}^d \to \mathbb{R}$ presented by a comparison oracle, solves in time which is polynomial in $\langle A, W, \mathcal{G}(A), l, u, b\rangle$, the convex integer maximization problem:*

$$\max\left\{f(Wx) : x \in \mathbb{Z}^n,\ Ax = b,\ l \le x \le u\right\}.$$

Proof. Let $S := \{x \in \mathbb{Z}^n : Ax = b,\ l \le x \le u\}$. The algorithm of Theorem 3.13 allows to realize in polynomial time a linear-optimization oracle for S. In particular, it allows to either conclude that S is infinite and stop or that it is finite, in which case the binary length $\langle \rho(S)\rangle$ of its radius is polynomial in $\langle A, l, u, b\rangle$, and continue. By Lemma 3.15, the given Graver basis is a set of all edge directions of $\text{conv}(S)$. Hence, the algorithm of Theorem 2.16 can be applied and provides the polynomial time solution of the given convex integer maximization program. □

3.3.4 Weighted separable convex minimization

We next establish a broad extension of Theorem 3.12, where the objective function is a sum $f(Wx) + g(x)$ involving two separable convex functions $f : \mathbb{Z}^d \to \mathbb{Z}$, $g : \mathbb{Z}^n \to \mathbb{Z}$, which includes a composite term $f(Wx)$. Moreover, the number d of rows of W is allowed to be variable. As usual, it suffices to be given a comparison oracle for the objective function. However, since it is more natural here to have each of f, g given by its own presentation, we assume that each of f, g is given by an *evaluation oracle* that, queried on a vector, returns the value of the function on that vector. As usual, $\hat{f}, \hat{g}$ denote the maximum values of $|f(Wx)|, |g(x)|$ over the feasible set and need not be part of the input. We also allow to incorporate inequalities on Wx in addition to the lower and upper bound inequalities on x.

To solve this problem, we need the Graver basis of an extended, $(m + d) \times (n + d)$ matrix, composed of A, W, the $d \times d$ identity matrix I and the zero $m \times d$ matrix.

Theorem 3.17. *There is an algorithm that, given an integer $m \times n$ matrix A, an integer $d \times n$ matrix W, $l, u \in \mathbb{Z}^n_\infty$, $\hat{l}, \hat{u} \in \mathbb{Z}^d_\infty$, $b \in \mathbb{Z}^m$, the Graver basis $\mathcal{G}(B)$ of*

$$B := \begin{pmatrix} A & 0 \\ W & I \end{pmatrix},$$

and separable convex functions $f : \mathbb{Z}^d \to \mathbb{Z}$, $g : \mathbb{Z}^n \to \mathbb{Z}$ presented by evaluation oracles, solves in time polynomial in $\langle A, W, \mathcal{G}(B), l, u, \hat{l}, \hat{u}, b, \hat{f}, \hat{g} \rangle$, the problem:

$$\min \left\{ f(Wx) + g(x) : x \in \mathbb{Z}^n,\ Ax = b,\ \hat{l} \le Wx \le \hat{u},\ l \le x \le u \right\}. \tag{3.11}$$

Proof. Define $h : \mathbb{Z}^{n+d} \to \mathbb{Z}$ by $h(x, y) := f(-y) + g(x)$ for all $x \in \mathbb{Z}^n$ and $y \in \mathbb{Z}^d$. Clearly, h is separable convex since f, g are. Now, problem (3.11) becomes as follows:

$$\min \left\{ h(x, y) : (x, y) \in \mathbb{Z}^{n+d}, \left(\begin{smallmatrix} A & 0 \\ W & I \end{smallmatrix}\right)\left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right) = \left(\begin{smallmatrix} b \\ 0 \end{smallmatrix}\right),\ l \le x \le u,\ -\hat{u} \le y \le -\hat{l} \right\},$$

and the statement follows at once by applying Theorem 3.12 to this problem. □

3.3.5 Totally unimodular systems revisited

Here, we deduce Theorem 2.19 on convex maximization over totally unimodular systems as a very special case of Theorem 3.16, giving an alternative algorithmic proof incorporating the algorithm of Theorem 3.12 instead of linear programming.

For a matrix A, let $\Delta(A)$ denote the maximum absolute value of a determinant of a square submatrix of A. The following bounds are well known; see, for example, [95].

Lemma 3.18. *For any circuit $c \in \mathcal{C}(A)$ with A integer matrix of rank r, we have the following:*

$$\|c\|_\infty \le \Delta(A) \quad \textit{and} \quad \|c\|_1 \le (r+1)\Delta(A).$$

Proof. Consider any circuit c of A and let $s := |\operatorname{supp}(c)| - 1 \le r$. Let $x \in \mathbb{Z}^{s+1}$ be the restriction of c to its nonzero coordinates. Then there is an $s \times (s+1)$ submatrix B of A of rank s with columns corresponding to the nonzero coordinates of c such that $Bx = 0$. Write $x := (x_0, \ldots, x_s)$ and $B := (B^0, \ldots, B^s)$ and for each j let

$$b_j := \det\left(B^0, \ldots, B^{j-1}, B^{j+1}, \ldots, B^s\right).$$

By Cramer's rule $\sum_{j=0}^{s} (-1)^j b_j B^j = 0$ and hence $|x_j|$ divides $|b_j| \le \Delta(A)$ for all j. Therefore, $\|c\|_\infty = \|x\|_\infty \le \Delta(A)$ and $\|c\|_1 \le (s+1)\|c\|_\infty \le (r+1)\Delta(A)$. □

As noted before, $\mathcal{C}(A) \subseteq \mathcal{G}(A)$ for every matrix A. The next lemma shows that for totally unimodular matrices equality holds, allowing to specialize results on general matrices and Graver bases to totally unimodular matrices and circuits.

Lemma 3.19. *The Graver basis and set of circuits of totally unimodular A satisfy the following:*

$$\mathcal{G}(A) = \mathcal{C}(A).$$

Proof. By Lemma 3.18, any component of any circuit of A satisfies $|c_i| \leq \Delta(A)$. For totally unimodular A, this implies $c_i \in \{-1, 0, 1\}$ for each $c \in \mathcal{C}(A)$. Now, consider any $g \in \mathcal{G}(A)$. Since $g \in \mathcal{L}^*(A)$, Lemma 2.17 implies that $g = \sum_{i=1}^{t} \lambda_i c_i$ is a conformal sum involving circuits $c_i \in \mathcal{C}(A)$ that satisfy $\mathrm{supp}(c_i) \not\subseteq \bigcup_{j>i} \mathrm{supp}(c_j)$ and $\lambda_i \in \mathbb{R}_+$ for each i. Pick any $k \in \mathrm{supp}(c_1) \setminus \bigcup_{j>1} \mathrm{supp}(c_j)$. Then $g_k = \lambda_1 c_{1,k}$ and $c_{1,k} = \pm 1$ and hence $\lambda_1 \geq 1$, which implies that $c_1 \sqsubseteq g$. Since g is a Graver basis element, it must be that $g = c_1$ and hence g is a circuit. So $\mathcal{G}(A) = \mathcal{C}(A)$. □

Theorem 3.16 and Lemma 3.19 imply at once the following theorem from Section 2.3.4.

Theorem 2.19 (revisited). *For every fixed d there is an algorithm that, given totally unimodular $m \times n$ matrix A, its set of circuits $\mathcal{C}(A)$, $l, u \in \mathbb{Z}^n_\infty$, $b \in \mathbb{Z}^m$, integer $d \times n$ matrix W, and convex $f : \mathbb{Z}^d \to \mathbb{R}$ presented by comparison oracle, solves in time polynomial in $\langle A, W, \mathcal{C}(A), l, u, b\rangle$ the convex maximization problem:*

$$\max\left\{f(Wx) : x \in \mathbb{Z}^n,\ Ax = b,\ l \leq x \leq u\right\}.$$

3.4 Bounds on Graver bases

The Graver basis of an $m \times n$ matrix A is typically very large and cannot be written down, let alone computed, in polynomial time. Moreover, unlike the set of circuits $\mathcal{C}(A)$ whose cardinality satisfies the bound $2\sum_{k=0}^{m} \binom{n}{k+1}$ which depends only on m and n, the cardinality of the Graver basis cannot be bounded in terms of m and n, and generally depends on the entries of A as well. Yet, as we next show, the binary length of each element in the Graver basis is polynomially bounded in that of A, implying a simple finite (though exponential) procedure for computing it. Note, though, that in Chapter 4 we introduce a very broad fundamental class of integer programs for which we *can* compute the Graver basis in polynomial time.

Let again $\Delta(A)$ denote the maximum absolute value of the determinant of a square submatrix of matrix A. The Hadamard bound gives $\Delta(A) \leq (\sqrt{m}\|A\|_\infty)^m$ with $\|A\|_\infty = \max_{i,j} |A_{i,j}|$, and therefore $\langle \Delta(A)\rangle = O(m(\log m + \log \|A\|_\infty))$ is polynomially bounded in $\langle A\rangle$. We have the following bounds on Graver basis elements in analogy to the bounds of Lemma 3.18 on circuits, see also [95].

Lemma 3.20. *For any $g \in \mathcal{G}(A)$ with A integer matrix of rank r and n columns:*

$$\|g\|_\infty \leq (n-r)\Delta(A) \quad \textit{and} \quad \|g\|_1 \leq (n-r)(r+1)\Delta(A).$$

Proof. Consider any $g \in \mathcal{G}(A)$. By Lemma 2.17, $g = \sum_{i=1}^{t} \lambda_i c_i$ is a conformal sum involving $t \leq n - r$ circuits $c_i \in \mathcal{C}(A)$ with $\lambda_i \in \mathbb{R}_+$. Now, each $\lambda_i \leq 1$ else $c_i \sqsubset g$,

contradicting $g \in \mathcal{G}(A)$. By the triangle inequality for any $1 \leq p \leq \infty$:

$$\|g\|_p = \Big\| \sum_{i=1}^{t} \lambda_i c_i \Big\|_p \leq \sum_{i=1}^{t} \lambda_i \big\| c_i \big\|_p \leq (n-r) \max \big\{ \|c\|_p : c \in \mathcal{C}(A) \big\}.$$

The bounds now follow from the analog bounds of Lemma 3.18 for circuits. □

Lemma 3.20 implies the following simple finite, though exponential, generic algorithm for computing the Graver basis of any matrix A.

Procedure 3.21 (generic algorithm for computing the Graver basis).

1. Construct the following finite subset of $\mathcal{L}(A)$:

$$L := \big\{ x \in \mathbb{Z}^n : \|x\|_\infty \leq (n-r)\Delta(A),\ \|x\|_1 \leq (n-r)(r+1)\Delta(A),\ Ax = 0 \big\}.$$

2. Distill out the Graver basis $\mathcal{G}(A)$ as the set of $\sqsubseteq$-minimal elements in $L \setminus \{0\}$.

The Graver basis can also be constructed by Gröbner bases methods [95]. However, the computational complexity of these methods is exponential as well.

We now proceed to Chapter 4, where we introduce the class of *n-fold integer programs* for which we can compute the Graver basis in polynomial time.

Notes

Primal methods for integer programming, which iteratively generate a sequence of better and better feasible points, have been studied in the literature for decades, see, for instance, the survey [49] and the references therein. In particular, such methods which make use of the Graver basis introduced in [41] or smaller subsets of the lattice of an integer matrix, which are equivalent to the *Gröbner bases* of the binomial ideal associated with the matrix, are discussed in the book [95] by Sturmfels. However, the polynomial time solvability of integer programming by such methods, in particular in Theorems 3.12, 3.13, and 3.16, was established only recently in the series of papers [23], [24], [50]. Very recent extensions in [66] further show that Graver-based methods also enable the polynomial time solution of integer minimization problems in variable dimension for broad classes of nonconvex quadratic functions and higher degree multivariate polynomial functions.

4 *n*-Fold Integer Programming

In this chapter, we continue our investigation of nonlinear integer programming, that is, nonlinear optimization over sets of integer points given by inequalities:

$$S := \{x \in \mathbb{Z}^n : Ax = b,\ l \le x \le u\}.$$

It is well known that even *linear* optimization over such sets is generally NP-hard. Two fundamental exceptional situations in which linear optimization over such sets can be solved in polynomial time are the following: first, when the dimension n is fixed [69]; second, when the matrix A is totally unimodular [54], [59].

In this chapter, we describe a new important fundamental situation, discovered and developed in the recent series of papers [23], [24], [50], [51], where the optimization of linear and several broad classes of nonlinear objectives can be solved in polynomial time. We now define this class of the so-termed *n-fold integer programs*.

An $(r, s) \times t$ *bimatrix* is a matrix A consisting of two blocks A_1, A_2 with A_1 its $r \times t$ submatrix consisting of the first r rows and A_2 its $s \times t$ submatrix consisting of the last s rows. The *n-fold product* of A is the following $(r + ns) \times nt$ matrix:

$$A^{(n)} := \begin{pmatrix} A_1 & A_1 & \cdots & A_1 \\ A_2 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_2 \end{pmatrix}.$$

The *(nonlinear) n-fold integer programming problem* is the optimization problem over a set presented by inequalities defined by an n-fold product, of the form:

$$S := \{x \in \mathbb{Z}^{nt} : A^{(n)}x = b,\ l \le x \le u\} \tag{4.1}$$

with A an integer $(r, s) \times t$ bimatrix, n positive integer, $b \in \mathbb{Z}^{r+ns}$, and $l, u \in \mathbb{Z}^{nt}_{\infty}$.

Some explanatory notes are in order. First, the dimension of an n-fold integer program is nt and is *variable*. Second, n-fold products $A^{(n)}$ are highly nontotally unimodular: the n-fold product of the simple $(0, 1) \times 1$ bimatrix with A_1 empty and $A_2 := 2$ satisfies $A^{(n)} = 2I_n$ and has exponential determinant 2^n. So this is indeed a class of programs which cannot be solved by methods of fixed dimension or totally unimodular matrices. Third, this class of programs turns out to be very natural and extremely important, with numerous applications, including to integer optimization over multidimensional tables, discussed in this and the next chapter. In fact, it is *universal*: the results of [27], described in Chapter 5, imply that *every* integer program *is* an n-fold integer program over some simple bimatrix A.

In Section 4.1, we study Graver bases of n-fold products of integer bimatrices and show that they can be computed in polynomial time. In Section 4.2, incorporating the results of Chapters 2 and 3, we show that linear optimization, (weighted) separable convex

minimization, distance minimization, and convex maximization over n-fold integer programs can be done in polynomial time. In Section 4.3, we discuss some of the numerous applications of this powerful theory including to (non)linear multicommodity transportation and transshipment problems. Discussion of further applications to multiway tables and proof of the universality of n-fold integer programming are postponed to Chapter 5. In Section 4.4, we discuss *stochastic integer programming*, which turns out to be a certain dual of n-fold integer programming, and use our theory to solve, in Theorem 4.19, stochastic integer programming in polynomial time as well. Finally, in Section 4.5, we discuss a heuristical scheme for approximative optimization over n-fold systems using approximations of the relevant Graver bases.

The following table enables quick navigation among some of the theorems in this chapter providing polynomial time optimization over sets of the form (4.1). Other results include the applications in Section 4.3 and stochastic programming in Section 4.4.

$\min wx$ (Linear objective)	$\min f(x)$ f separable convex	$\max f(Wx)$ f convex	$\min f(Wx)+g(x)$ f,g separable convex
Theorem 4.7	Theorem 4.8	Theorem 4.10	Theorem 4.12

4.1 Graver bases of n-fold products

In this section, we study properties of Graver bases of n-fold products. The main result here is Theorem 4.4: the Graver basis of $A^{(n)}$ is polynomial time computable.

Let A be a fixed integer $(r,s)\times t$ bimatrix with blocks A_1, A_2. For each positive integer n, we index vectors in $\mathbb{Z}^{nt}$ as $x=(x^1,\ldots,x^n)$ with each *brick* x^k in $\mathbb{Z}^t$. The *type* of vector x is the number $\text{type}(x):=|\{k:x^k\neq 0\}|$ of its nonzero bricks.

The following definition from [89] plays an important role in the sequel.

Definition 4.1. The *Graver complexity* of an integer bimatrix A is defined as follows:

$$g(A):=\inf\big\{g\in\mathbb{Z}_+ : \text{type}(x)\leq g \text{ for all } x\in\mathcal{G}(A^{(n)}) \text{ and all } n\big\}.$$

We proceed to establish a result of [89] and its extension in [56] which show that, in fact, the Graver complexity of every integer bimatrix A is finite.

Consider n-fold products $A^{(n)}$ of A. By definition of the n-fold product, $A^{(n)}x=0$ if and only if $A_1\sum_{k=1}^n x^k=0$ and $A_2x^k=0$ for all k. In particular, a necessary condition for x to lie in $\mathcal{L}(A^{(n)})$, and in particular in $\mathcal{G}(A^{(n)})$, is that $x^k\in\mathcal{L}(A_2)$ for all k. Call a vector $x=(x^1,\ldots,x^n)$ *full* if, in fact, $x^k\in\mathcal{L}^*(A_2)$ for all k, in which case $\text{type}(x)=n$, and *pure* if, moreover, $x^k\in\mathcal{G}(A_2)$ for all k. Full vectors, and in particular pure vectors, are natural candidates for lying in the Graver basis $\mathcal{G}(A^{(n)})$ of $A^{(n)}$ and indeed play an important role in its construction.

Consider any full vector $y=(y^1,\ldots,y^m)$. By definition, each brick of y satisfies $y^i\in\mathcal{L}^*(A_2)$ and is therefore a conformal sum $y^i=\sum_{j=1}^{k_i}x^{i,j}$ of some elements $x^{i,j}\in\mathcal{G}(A_2)$ for all i,j. Let $n:=k_1+\cdots+k_m\geq m$ and let x be the pure vector:

$$x=\big(x^1,\ldots,x^n\big):=\big(x^{1,1},\ldots,x^{1,k_1},\ldots,x^{m,1},\ldots,x^{m,k_m}\big).$$

We call the pure vector x an *expansion* of the full vector y, and the full vector y a *compression* of the pure vector x. Note that $A_1 \sum y^i = A_1 \sum x^{i,j}$ and therefore $y \in \mathcal{L}(A^{(m)})$ if and only if $x \in \mathcal{L}(A^{(n)})$. Note also that each full y may have many different expansions and each pure x may have many different compressions.

Lemma 4.2. *Consider any expansion $x = (x^1, \ldots, x^n)$ of any full $y = (y^1, \ldots, y^m)$. If y is in the Graver basis $\mathcal{G}(A^{(m)})$ then x is in the Graver basis $\mathcal{G}(A^{(n)})$.*

Proof. Let $x = (x^{1,1}, \ldots, x^{m,k_m}) = (x^1, \ldots, x^n)$ be any expansion of any full $y = (y^1, \ldots, y^m)$ with $y^i = \sum_{j=1}^{k_i} x^{i,j}$ for each i. Suppose indirectly that $y \in \mathcal{G}(A^{(m)})$ but $x \notin \mathcal{G}(A^{(n)})$. Since $y \in \mathcal{L}^*(A^{(m)})$, we have $x \in \mathcal{L}^*(A^{(n)})$. Since $x \notin \mathcal{G}(A^{(n)})$, there is a $g = (g^{1,1}, \ldots, g^{m,k_m})$ in $\mathcal{G}(A^{(n)})$ satisfying $g \sqsubset x$. Let $h = (h^1, \ldots, h^m)$ be the compression of g defined by $h^i := \sum_{j=1}^{k_i} g^{i,j}$ for each i. Since $g \in \mathcal{L}^*(A^{(n)})$, we have that $h \in \mathcal{L}^*(A^{(m)})$. But $h \sqsubset y$, which contradicts $y \in \mathcal{G}(A^{(m)})$. □

Lemma 4.3. *The Graver complexity $g(A)$ of every integer bimatrix A is finite.*

Proof. We need to bound the type of any element in the Graver basis of the l-fold product of A for any l. Suppose that there is an element z of type m in some $\mathcal{G}(A^{(l)})$. Then its restriction $y = (y^1, \ldots, y^m)$ to its m nonzero bricks is a full vector and is in the Graver basis $\mathcal{G}(A^{(m)})$. Let $x = (x^1, \ldots, x^n)$ be any expansion of y. Then $\text{type}(z) = m \leq n = \text{type}(x)$, and by Lemma 4.2, the pure vector x is in $\mathcal{G}(A^{(n)})$.

Therefore, it suffices to bound the type of any pure element in the Graver basis of the n-fold product of A for any n. Suppose that $x = (x^1, \ldots, x^n)$ is a pure element in $\mathcal{G}(A^{(n)})$ for some n. Let $\mathcal{G}(A_2) = \{g^1, \ldots, g^p\}$ be the Graver basis of A_2 and let G_2 be the $t \times p$ matrix whose columns are the g^i. Let $v \in \mathbb{Z}_+^p$ be the vector with $v_i := |\{k : x^k = g^i\}|$ counting the number of bricks of x which are equal to g^i for each i. Then $\sum_{i=1}^p v_i = \text{type}(x) = n$. Now, $A_1 G_2 v = A_1 \sum_{k=1}^n x^k = 0$, and hence $v \in \mathcal{L}^*(A_1 G_2)$. We claim that, moreover, v is in $\mathcal{G}(A_1 G_2)$. Suppose indirectly that it is not. Then there is a $\hat{v} \in \mathcal{G}(A_1 G_2)$ with $\hat{v} \sqsubset v$, and it is easy to obtain a nonzero $\hat{x} \sqsubset x$ from x by zeroing out some bricks so that $\hat{v}_i = |\{k : \hat{x}^k = g^i\}|$ for all i. Then $A_1 \sum_{k=1}^n \hat{x}^k = A_1 G_2 \hat{v} = 0$ and hence $\hat{x} \in \mathcal{L}^*(A^{(n)})$, contradicting $x \in \mathcal{G}(A^{(n)})$.

So the type of any pure vector, and hence the Graver complexity of A, is at most the largest value $\sum_{i=1}^p v_i$ of any nonnegative vector $v \in \mathcal{G}(A_1 G_2)$. □

We proceed to establish the following result of [24] asserting that Graver bases of n-fold products can be computed in polynomial time. An *n-lifting* of a vector $y = (y^1, \ldots, y^m)$ consisting of m bricks is any vector $z = (z^1, \ldots, z^n)$ consisting of n bricks such that for some $1 \leq k_1 < \cdots < k_m \leq n$, we have $z^{k_i} = y^i$ for $i = 1, \ldots, m$, and all other bricks of z are zero, in particular, $n \geq m$ and $\text{type}(z) = \text{type}(y)$.

Theorem 4.4. *For every fixed integer bimatrix A, there is an algorithm that, given positive integer n, computes the Graver basis $\mathcal{G}(A^{(n)})$ of the n-fold product of A, in time which is polynomial in n. In particular, the cardinality $|\mathcal{G}(A^{(n)})|$ and the binary length $\langle \mathcal{G}(A^{(n)}) \rangle$ of the Graver basis of $A^{(n)}$ are polynomial in n.*

Proof. Let $g := g(A)$ be the Graver complexity of A. Since A is fixed, so is g. Therefore, for every $n \leq g$, the Graver basis $\mathcal{G}(A^{(n)})$, and in particular, the Graver basis $\mathcal{G}(A^{(g)})$ of the g-fold product of A, can be computed in constant time.

Now, consider any $n > g$. We claim that $\mathcal{G}(A^{(n)})$ satisfies the following:

$$\mathcal{G}(A^{(n)}) = \big\{z : z \text{ is an } n\text{-lifting of some } y \in \mathcal{G}\big(A^{(g)}\big)\big\}.$$

Consider any n-lifting z of any $y \in \mathcal{G}(A^{(g)})$. Suppose indirectly that $z \notin \mathcal{G}(A^{(n)})$. Then there is a $z' \in \mathcal{G}(A^{(n)})$ with $z' \sqsubset z$. But then z' is the n-lifting of some $y' \in \mathcal{L}^*(A^{(g)})$ with $y' \sqsubset y$, contradicting $y \in \mathcal{G}(A^{(g)})$. So $z \in \mathcal{G}(A^{(n)})$.

Conversely, consider any $z \in \mathcal{G}(A^{(n)})$. Then $\text{type}(z) \leq g$ and hence z is the n-lifting of some $y \in \mathcal{L}^*(A^{(g)})$. Suppose indirectly that $y \notin \mathcal{G}(A^{(g)})$. Then there exists $y' \in \mathcal{G}(A^{(g)})$ with $y' \sqsubset y$. But then a suitable n-lifting z' of y' satisfies $z' \in \mathcal{L}^*(A^{(n)})$ with $z' \sqsubset z$, contradicting $z \in \mathcal{G}(A^{(n)})$. So $y \in \mathcal{G}(A^{(g)})$.

Now, the number of n-liftings of each $y \in \mathcal{G}(A^{(g)})$ is at most $\binom{n}{g}$, and hence

$$\big|\mathcal{G}(A^{(n)})\big| \leq \binom{n}{g}\big|\mathcal{G}\big(A^{(g)}\big)\big| = O\big(n^g\big).$$

So the set of all n-liftings of vectors in $\mathcal{G}(A^{(g)})$ and hence the Graver basis $\mathcal{G}(A^{(n)})$ of the n-fold product can be computed in time polynomial in n as claimed. □

Here, is an example of computing $\mathcal{G}(A^{(n)})$ by the algorithm of Theorem 4.4.

Example 4.5. Consider the $(2,1) \times 2$ bimatrix A with $A_1 := I_2$ the identity matrix and $A_2 := (1\ 1)$. Then $\mathcal{G}(A_2) = \pm\{(1,-1)\}$ and $g := g(A) = 2$. The g-fold product of A and its Graver basis, which consists of two antipodal pure vectors only, are as follows:

$$A^{(g)} = A^{(2)} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad \mathcal{G}\big(A^{(g)}\big) = \mathcal{G}\big(A^{(2)}\big) = \pm\big\{(1,-1,-1,1)\big\}.$$

We demonstrate how to compute the Graver basis of $A^{(4)}$, which is the 6×8 matrix:

$$A^{(4)} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Applying the algorithm of Theorem 4.4, the Graver basis of $A^{(4)}$ is obtained by taking the $6 = \binom{4}{2}$ many 4 liftings of each of the two elements of $\mathcal{G}(A^{(2)})$ and consists of 12 vectors

given by the rows of the following matrix and their antipodals:

$$\mathcal{G}\big(A^{(4)}\big) = \pm \begin{pmatrix} 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \end{pmatrix}.$$

The algorithm of Theorem 4.4 relies on the mere existence of a finite Graver complexity $g(A)$ for every bimatrix A, since $g(A)$ and the constant Graver bases $\mathcal{G}(A^{(k)})$ for all $k \leq g(A)$ can be built into the algorithm. Nonetheless, the Graver complexity can be computed by a simple finite procedure as we now show. For an integer $r \times p$ matrix B, let $\mathcal{H}(B) := \mathcal{G}(B) \cap \mathbb{Z}^p_+$ denote its so-called *Hilbert basis*, which is often easier to compute than the entire Graver basis (say by a suitable restriction of the simple generic Algorithm 3.21 to the nonnegative orthant).

Procedure 4.6 (computing the Graver complexity of a bimatrix).

1. Compute the Graver basis $\mathcal{G}(A_2)$ (say using Algorithm 3.21).
2. If $\mathcal{G}(A_2) = \emptyset$ then output $g(A) := 0$. Otherwise form the matrix G_2 having as columns the elements of $\mathcal{G}(A_2)$, compute $\mathcal{H}(A_1G_2)$ or $\mathcal{G}(A_1G_2)$, and output the following:

$$g(A) := \max\big\{\mathbf{1}v : v \in \mathcal{H}\big(A_1G_2\big)\big\} = \max\big\{\|v\|_1 : v \in \mathcal{G}\big(A_1G_2\big)\big\}.$$

To justify this procedure, first note that if $\mathcal{G}(A_2)$ is empty, which holds if and only if A_2 has linearly independent columns, then $\mathcal{G}(A^{(n)})$ is also empty for all n and hence indeed $g(A) = 0$. Next, note that if $\mathcal{G}(A_2) = \{g^1, \ldots, g^p\}$ is nonempty then the proof of Lemma 4.3 shows that if $x = (x^1, \ldots, x^n)$ is a pure vector in $\mathcal{G}(A^{(n)})$ then the vector $v \in \mathbb{Z}^p_+$ with $v_i = |\{k : x^k = g^i\}|$ is an element of $\mathcal{H}(A_1G_2) = \mathcal{G}(A_1G_2) \cap \mathbb{Z}^p_+$ with $\mathbf{1}v = \sum_{i=1}^p v_i = n = \text{type}(x)$; but it is also easy to see that, conversely, each vector $v \in \mathcal{H}(A_1G_2)$ with $\mathbf{1}v = n$ gives rise to a pure element $x \in \mathcal{G}(A^{(n)})$ and therefore indeed $g(A) = \max\{\mathbf{1}v : v \in \mathcal{H}(A_1G_2)\}$. Finally, note that a vector g is a column of G_2 if and only if $-g$ is, and so the same property holds for A_1G_2, implying that $g(A) = \max\{\|v\|_1 : v \in \mathcal{G}(A_1G_2)\}$ as well.

4.2 Efficient n-fold integer programming

Combining Theorem 4.4 with the results of Chapter 3, we now obtain five theorems from [23], [24], [50], [51] on (non)linear n-fold integer programming polynomial time.

As in Chapter 3, when functions f, g are present, $\hat{f}, \hat{g}$ denote the maximum values of $|f(Wx)|, |g(x)|$ over the feasible set and need not be part of the input.

Theorem 4.7. *For each fixed integer $(r,s)\times t$ bimatrix A, there is an algorithm that, given positive integer n, $l,u\in\mathbb{Z}^{nt}_{\infty}$, $b\in\mathbb{Z}^{r+ns}$, and $w\in\mathbb{Z}^{nt}$, solves in time which is polynomial in n and $\langle l,u,b,w\rangle$ the following linear n-fold integer program:*

$$\min\left\{wx : x\in\mathbb{Z}^{nt},\ A^{(n)}x=b,\ l\leq x\leq u\right\}.$$

Proof. First compute the Graver basis $\mathcal{G}(A^{(n)})$ by the algorithm of Theorem 4.4. Next, apply the algorithm of Theorem 3.13 and solve the given program. □

Theorem 4.8. *For each fixed integer $(r,s)\times t$ bimatrix A, there is an algorithm that, given n, $l,u\in\mathbb{Z}^{nt}_{\infty}$, $b\in\mathbb{Z}^{r+ns}$, and separable convex $f:\mathbb{Z}^{nt}\to\mathbb{Z}$ presented by a comparison oracle, solves in time polynomial in n and $\langle l,u,b,\hat{f}\rangle$ the following problem:*

$$\min\left\{f(x) : x\in\mathbb{Z}^{nt},\ A^{(n)}x=b,\ l\leq x\leq u\right\}.$$

Proof. First compute the Graver basis $\mathcal{G}(A^{(n)})$ by the algorithm of Theorem 4.4. Next, apply the algorithm of Theorem 3.12 and solve the given program. □

Theorem 4.9. *For each fixed integer $(r,s)\times t$ bimatrix A, there is an algorithm that, given positive integers n and p, $l,u\in\mathbb{Z}^{nt}_{\infty}$, $b\in\mathbb{Z}^{r+ns}$, and $\hat{x}\in\mathbb{Z}^{nt}$, solves in time which is polynomial in n, p, and $\langle l,u,b,\hat{x}\rangle$ the closest-point problem:*

$$\min\left\{\|x-\hat{x}\|_p : x\in\mathbb{Z}^{nt},\ A^{(n)}x=b,\ l\leq x\leq u\right\}. \tag{4.2}$$

For $p=\infty$, the problem (4.2) can be solved in time polynomial in n and $\langle l,u,b,\hat{x}\rangle$.

Proof. First compute the Graver basis $\mathcal{G}(A^{(n)})$ by the algorithm of Theorem 4.4. Next, apply the algorithm of Theorem 3.14 and solve the given problem. □

In particular, for each p we can solve the minimum norm n-fold integer program:

$$\min\left\{\|x\|_p : x\in\mathbb{Z}^{nt},\ A^{(n)}x=b,\ l\leq x\leq u\right\},$$

which for $p=\infty$ is equivalent to the min-max n-fold integer program:

$$\min\left\{\max\left\{\left|x_i\right| : i=1,\ldots,nt\right\} : x\in\mathbb{Z}^{nt},\ A^{(n)}x=b,\ l\leq x\leq u\right\}.$$

Theorem 4.10. *For each fixed d and $(r,s)\times t$ integer bimatrix A, there is an algorithm that, given n, bounds $l,u\in\mathbb{Z}^{nt}_{\infty}$, integer $d\times nt$ matrix W, $b\in\mathbb{Z}^{r+ns}$, and convex function $f:\mathbb{Z}^{d}\to\mathbb{R}$ presented by a comparison oracle, solves in time polynomial in n and $\langle W,l,u,b\rangle$ the convex n-fold integer maximization problem:*

$$\max\left\{f(Wx) : x\in\mathbb{Z}^{nt},\ A^{(n)}x=b,\ l\leq x\leq u\right\}.$$

Proof. First compute the Graver basis $\mathcal{G}(A^{(n)})$ by the algorithm of Theorem 4.4. Next, apply the algorithm of Theorem 3.16 and solve the given program. □

We proceed to establish a broad extension of Theorem 3.12, where the objective function is a sum $f(Wx)+g(x)$ involving two separable convex functions f, g, which includes a composite term $f(Wx)$. Moreover, the number d of rows of W is allowed to be variable. We also allow to incorporate inequalities on Wx in addition to the lower and upper bound inequalities on x. We need the following lemma.

Lemma 4.11. *For every fixed integer $(r,s)\times t$ bimatrix A and $(p,q)\times t$ bimatrix W, there is an algorithm that, given n, computes in time polynomial in n the Graver basis $\mathcal{G}(B)$ of the following $(r+ns+p+nq)\times(nt+p+nq)$ matrix:*

$$B := \begin{pmatrix} A^{(n)} & 0 \\ W^{(n)} & I \end{pmatrix}.$$

Proof. Let D be the $(r+p, s+q)\times(t+p+q)$ bimatrix with blocks defined by the following:

$$D_1 := \begin{pmatrix} A_1 & 0 & 0 \\ W_1 & I_p & 0 \end{pmatrix}, \quad D_2 := \begin{pmatrix} A_2 & 0 & 0 \\ W_2 & 0 & I_q \end{pmatrix}.$$

Apply the algorithm of Theorem 4.4 and compute in polynomial time the Graver basis $\mathcal{G}(D^{(n)})$ of the n-fold product of D, which is the following matrix:

$$D^{(n)} = \left(\begin{array}{ccccccccccc} A_1 & 0 & 0 & A_1 & 0 & 0 & \cdots & A_1 & 0 & 0 \\ W_1 & I_p & 0 & W_1 & I_p & 0 & \cdots & W_1 & I_p & 0 \\ \hline A_2 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ W_2 & 0 & I_q & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & A_2 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & W_2 & 0 & I_q & \cdots & 0 & 0 & 0 \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & \cdots & A_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & W_2 & 0 & I_q \end{array}\right).$$

Suitable row and column permutations applied to $D^{(n)}$ give the following matrix:

$$C := \left(\begin{array}{cccc|cccc|cccc} A_1 & A_1 & \cdots & A_1 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ A_2 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_2 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \hline W_1 & W_1 & \cdots & W_1 & I_p & I_p & \cdots & I_p & 0 & 0 & \cdots & 0 \\ W_2 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & I_q & 0 & \cdots & 0 \\ 0 & W_2 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & I_q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W_2 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & I_q \end{array}\right).$$

Obtain the Graver basis $\mathcal{G}(C)$ in polynomial time from $\mathcal{G}(D^{(n)})$ by permuting the entries of each element of the latter by the permutation of the columns of $\mathcal{G}(D^{(n)})$ that is used to get C (row permutations do not affect the Graver basis).

Now, note that the matrix B can be obtained from C by dropping all but the first p columns in the second block. Consider any element in the Graver basis of C, indexed, according to the block structure, in the following way:

$$\left(x^1, x^2, \ldots, x^n, y^1, y^2, \ldots, y^n, z^1, z^2, \ldots, z^n\right).$$

Clearly, if $y^k = 0$ for $k = 2, \ldots, n$ then the following restriction of this element:

$$\left(x^1, x^2, \ldots, x^n, y^1, z^1, z^2, \ldots, z^n\right)$$

is in the Graver basis of B. On the other hand, if

$$\left(x^1, x^2, \ldots, x^n, y^1, z^1, z^2, \ldots, z^n\right)$$

is any element in $\mathcal{G}(B)$ then its extension:

$$\left(x^1, x^2, \ldots, x^n, y^1, 0, \ldots, 0, z^1, z^2, \ldots, z^n\right)$$

is in $\mathcal{G}(C)$. So the Graver basis of B can be obtained in polynomial time by the following:

$$\mathcal{G}(B) := \left\{ \left(x^1, \ldots, x^n, y^1, z^1, \ldots, z^n\right) : \left(x^1, \ldots, x^n, y^1, 0, \ldots, 0, z^1, \ldots, z^n\right) \in \mathcal{G}(C) \right\}.$$

Therefore, the Graver basis of B can indeed be computed in polynomial time. □

We can now prove our next theorem. As usual, it suffices to be given a comparison oracle for the objective function. However, as with Theorem 3.17, it is more natural here to have each of the functions f, g involved given by its own presentation, and so we assume that each of f, g is given by an evaluation oracle.

Theorem 4.12. *For every fixed integer $(r, s) \times t$ bimatrix A and integer $(p, q) \times t$ bimatrix W, there is an algorithm that, given n, $l, u \in \mathbb{Z}_\infty^{nt}$, $\hat{l}, \hat{u} \in \mathbb{Z}_\infty^{p+nq}$, $b \in \mathbb{Z}^{r+ns}$, and separable convex functions $f : \mathbb{Z}^{p+nq} \to \mathbb{Z}$, $g : \mathbb{Z}^{nt} \to \mathbb{Z}$ presented by evaluation oracles, solves in time which is polynomial in n and $\langle l, u, \hat{l}, \hat{u}, b, \hat{f}, \hat{g} \rangle$ the following problem:*

$$\min \left\{ f\left(W^{(n)} x\right) + g(x) : x \in \mathbb{Z}^{nt},\ A^{(n)} x = b,\ \hat{l} \le W^{(n)} x \le \hat{u},\ l \le x \le u \right\}.$$

Proof. First, use the algorithm of Lemma 4.11 to compute the Graver basis of the following:

$$B := \begin{pmatrix} A^{(n)} & 0 \\ W^{(n)} & I \end{pmatrix}.$$

Next, apply the algorithm of Theorem 3.17 and solve the given problem. □

4.3 Some applications

The theory of n-fold integer programming has numerous applications in a variety of areas. Here, we discuss applications to (non)linear multicommodity flow problems. In Chapter 5, we discuss applications to multiway tables and privacy in databases.

Recall from Section 1.2.2 the following very general nonlinear multicommodity flow problem (also see Figure 1.2). There is a digraph G with s vertices and t edges. There are l commodities. Each commodity has a demand vector $d^k \in \mathbb{Z}^s$ with d^k_v the demand for commodity k at vertex v (interpreted as supply when positive and consumption when negative). Each edge e has a capacity u_e (upper bound on the combined flow of all commodities on it). A *multicommodity transshipment* is a vector $x = (x^1, \ldots, x^l)$ with $x^k \in \mathbb{Z}^t_+$ for all k and x^k_e the flow of commodity k on edge e, satisfying the capacity constraint $\sum_{k=1}^l x^k_e \leq u_e$ for each edge e and demand constraint $\sum_{e\in\delta^+(v)} x^k_e - \sum_{e\in\delta^-(v)} x^k_e = d^k_v$ for each vertex v and commodity k with $\delta^+(v), \delta^-(v)$ the sets of edges entering and leaving vertex v. There are cost functions $f_e, g^k_e : \mathbb{Z} \to \mathbb{Z}$ for each edge and each edge-commodity pair. The cost of transshipment x on edge e is $f_e(\sum_{k=1}^l x^k_e) + \sum_{k=1}^l g^k_e(x^k_e)$, where the first term is the value of f_e on the combined flow of all commodities on e and the second term the sum of costs for each edge-commodity pair. The cost can in particular be convex such as $\alpha_e|\sum_{k=1}^l x^k_e|^{\beta_e} + \sum_{k=1}^l \gamma^k_e|x^k_e|^{\delta^k_e}$ for some nonnegative integers $\alpha_e, \beta_e, \gamma^k_e, \delta^k_e$, accounting for increase in cost due to channel congestion when subject to heavy traffic or communication load [88]. The linear problem is the special case with $\beta_e = \delta^k_e$=1. The total cost is the sum of costs over all edges.

The theory of n-fold integer programming provides the first polynomial time algorithms for the problem in two broad situations discussed in Sections 4.3.1 and 4.3.2.

4.3.1 Nonlinear many-commodity transshipment

Here, we consider the problem with *variable* number l of commodities over a fixed (but arbitrary) digraph – the so-termed *many-commodity* transshipment problem. This problem may seem at first very restricted; however, even deciding if a feasible many-transshipment exists (regardless of its cost) is NP-complete already over the complete bipartite digraphs $K_{3,n}$ (oriented from one side to the other) with only 3 vertices on one side [51]; moreover, even over the single tiny digraph $K_{3,3}$, the only solution available to date is the one given below via n-fold integer programming.

As usual, $\hat{f}, \hat{g}$ denote the maximum values of $|f|, |g|$ over the feasible set and need not be part of the input. It is usually easy to determine an upper bound on these values from the problem data. For instance, in the special case of linear cost functions f, g, bounds which are polynomial in the binary length of the costs α_e, γ^k_e, capacities u, and demands d^k_v, follow easily from Cramer's rule.

We now obtain a result of [51] on (non)linear many-commodity transshipment.

Corollary 4.13. *For every fixed digraph G, there is an algorithm that, given l commodities, demand $d^k_v \in \mathbb{Z}$ for each commodity k and vertex v, edge capacities $u_e \in \mathbb{Z}_+$, and convex costs $f_e, g^k_e : \mathbb{Z} \to \mathbb{Z}$ presented by evaluation oracles, solves in time polynomial*

in l and $\langle d_v^k, u_e, \hat{f}, \hat{g}\rangle$ the many-commodity transshipment problem:

$$\min \sum_e \Big(f_e\Big(\sum_{k=1}^{l} x_e^k\Big) + \sum_{k=1}^{l} g_e^k\big(x_e^k\big)\Big)$$

$$\text{s.t.} \quad x_e^k \in \mathbb{Z}, \qquad \sum_{e\in\delta^+(v)} x_e^k - \sum_{e\in\delta^-(v)} x_e^k = d_v^k, \qquad \sum_{k=1}^{l} x_e^k \le u_e, \qquad x_e^k \ge 0.$$

Proof. Assume that G has s vertices and t edges and let D be its $s \times t$ vertex-edge incidence matrix. Let $f : \mathbb{Z}^t \to \mathbb{Z}$ and $g : \mathbb{Z}^{lt} \to \mathbb{Z}$ be the separable convex functions defined by $f(y) := \sum_{e=1}^{t} f_e(y_e)$ with $y_e := \sum_{k=1}^{l} x_e^k$ and $g(x) := \sum_{e=1}^{t}\sum_{k=1}^{l} g_e^k(x_e^k)$. Let $x = (x^1, \ldots, x^l)$ be the vector of variables with $x^k \in \mathbb{Z}^t$ the flow of commodity k for each k. Then the problem can be rewritten as follows:

$$\min\big\{ f\big(\textstyle\sum_{k=1}^{l} x^k\big) + g(x) : x \in \mathbb{Z}^{lt},\ Dx^k = d^k,\ \sum_{k=1}^{l} x^k \le u,\ x \ge 0\big\}.$$

We can now proceed in two ways.

First way: extend the vector of variables to $x = (x^0, x^1, \ldots, x^l)$ with $x^0 \in \mathbb{Z}^t$ representing an additional slack commodity. Then the capacity constraints become $\sum_{k=0}^{l} x^k = u$ and the cost function becomes $f(u - x_0) + g(x^1, \ldots, x^l)$ which is also separable convex. Now, let A be the $(t, s)\times t$ bimatrix with first block $A_1 := I_t$ the $t \times t$ identity matrix and second block $A_2 := D$. Let $d^0 := Du - \sum_{k=1}^{l} d^k$ and let $b := (u, d^0, d^1, \ldots, d^l)$. Then the problem becomes the $(l + 1)$-fold integer program:

$$\min\big\{ f\big(u - x^0\big) + g\big(x^1, \ldots, x^l\big) : x \in \mathbb{Z}^{(l+1)t},\ A^{(l+1)}x = b,\ x \ge 0\big\}. \tag{4.3}$$

By Theorem 4.8, this program can be solved in polynomial time as claimed.

Second way: let A be the $(0, s) \times t$ bimatrix with first block A_1 empty and second block $A_2 := D$. Let W be the $(t, 0) \times t$ bimatrix with first block $W_1 := I_t$ the $t \times t$ identity matrix and second block W_2 empty. Let $b := (d^1, \ldots, d^l)$. Then the problem is precisely the following l-fold integer program:

$$\min\big\{ f\big(W^{(l)}x\big) + g(x) : x \in \mathbb{Z}^{lt},\ A^{(l)}x = b,\ W^{(l)}x \le u,\ x \ge 0\big\}.$$

By Theorem 4.12, this program can be solved in polynomial time as claimed. □

We also point out the following immediate consequence of Corollary 4.13.

Corollary 4.14. *For fixed s, the (convex) many-commodity transshipment problem with variable l commodities on any s-vertex digraph is polynomial time solvable.*

4.3.2 Nonlinear multicommodity transportation

Here, we consider the problem with fixed (but arbitrary) number l of commodities over any bipartite subdigraph of $K_{m,n}$ (oriented from one side to the other) – the so-called multicommodity *transportation* problem – with fixed number m of suppliers and *variable*

number n of consumers. This is very natural in operations research applications, where few facilities serve many customers. The problem is difficult even for $l = 2$ commodities: deciding if a feasible 2-commodity transportation exists (regardless of its cost) is NP-complete already over $K_{m,n}$ [27]; moreover, even over the bipartite digraphs $K_{3,n}$ with only $m = 3$ suppliers, the only available solution to date is the one given below via n-fold integer programming.

This problem seems harder than the one discussed in the previous subsection (with no seeming analog for non-bipartite digraphs), and the formulation below is more delicate. Therefore, it is convenient to change the labeling of the data a little bit as follows (see Figure 4.1). We denote edges by pairs (i, j), where $1 \le i \le m$ is a supplier and

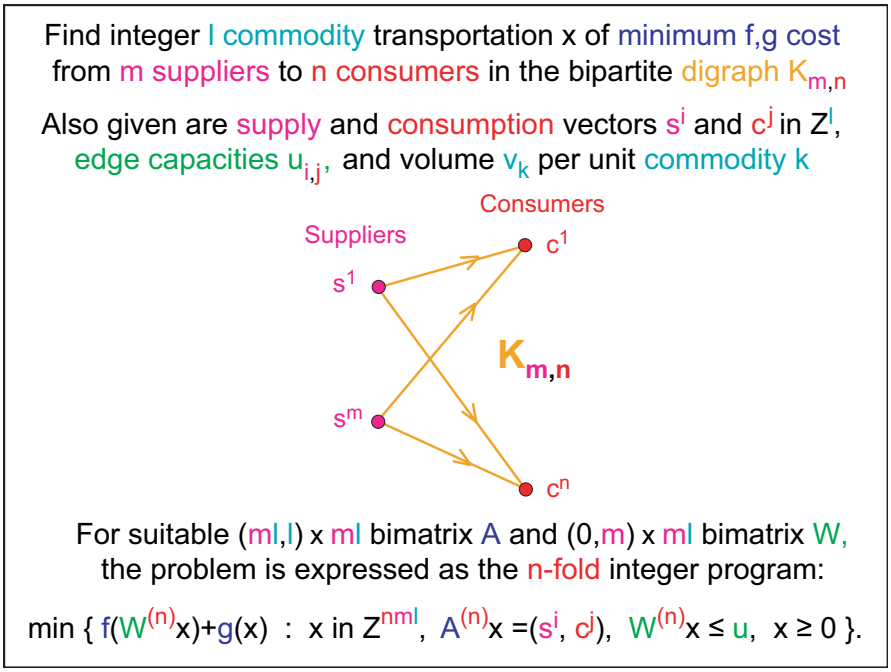


Figure 4.1: Multicommodity transportation problem

$1 \le j \le n$ is a consumer. The demand vectors are replaced by (nonnegative) supply and consumption vectors: each supplier i has a supply vector $s^i \in \mathbb{Z}^l_+$ with s^i_k its supply in commodity k, and each consumer j has a consumption vector $c^j \in \mathbb{Z}^l_+$ with c^j_k its consumption in commodity k. Each commodity k has its own volume $v_k \in \mathbb{Z}_+$ per unit flow. A multicommodity transportation is indexed as follows, with $x^j_{i,k}$ the flow of commodity k from supplier i to consumer j:

$$x = \left(x^1, \ldots, x^n\right), \quad x^j = \left(x^j_{1,1}, \ldots, x^j_{1,l}, \ldots, x^j_{m,1}, \ldots, x^j_{m,l}\right).$$

The capacity constraint on edge (i, j) is $\sum_{k=1}^{l} v_k x^j_{i,k} \le u_{i,j}$ and the cost is $f_{i,j}(\sum_{k=1}^{l} v_k x^j_{i,k}) + \sum_{k=1}^{l} g^j_{i,k}(x^j_{i,k})$ with $f_{i,j}, g^j_{i,k} : \mathbb{Z} \to \mathbb{Z}$ convex.

We assume below that the underlying digraph is $K_{m,n}$ (with edges oriented from suppliers to consumers), since the problem over any subdigraph G of $K_{m,n}$ reduces to that over $K_{m,n}$ by forcing 0 capacity on all edges not present in G.

We now obtain a result of [51] on (non)linear multicommodity transportation.

Corollary 4.15. *For fixed l commodities, m suppliers, and volumes v_k, there is an algorithm that, given n customers, supplies and demands $s^i, c^j \in \mathbb{Z}^l_+$, $u_{i,j} \in \mathbb{Z}_+$, and convex costs $f_{i,j}, g^j_{i,k} : \mathbb{Z} \to \mathbb{Z}$ presented by evaluation oracles, solves in time polynomial in n and $\langle s^i, c^j, u, \hat{f}, \hat{g}\rangle$ the multicommodity transportation problem:*

$$\min \sum_{i,j} \Big(f_{i,j}\Big(\sum_k v_k x^j_{i,k}\Big) + \sum_{k=1}^{l} g^j_{i,k}\big(x^j_{i,k}\big)\Big)$$

$$\text{s.t.} \quad x^j_{i,k} \in \mathbb{Z}, \quad \sum_j x^j_{i,k} = s^i_k, \quad \sum_i x^j_{i,k} = c^j_k, \quad \sum_{k=1}^{l} v_k x^j_{i,k} \le u_{i,j}, \quad x^j_{i,k} \ge 0.$$

Proof. Construct bimatrices A and W as follows. Let D be the $(l,0) \times l$ bimatrix with first block $D_1 := I_l$ and second block D_2 empty. Let V be the $(0,1) \times l$ bimatrix with first block V_1 empty and second block $V_2 := (v_1, \dots, v_l)$. Let A be the $(ml, l) \times ml$ bimatrix with first block $A_1 := I_{ml}$ and second block $A_2 := D^{(m)}$. Let W be the $(0, m) \times ml$ bimatrix with first block W_1 empty and second block $W_2 := V^{(m)}$. Let b be the $(ml+nl)$-vector $b := (s^1, \dots, s^m, c^1, \dots, c^n)$.

Let $f : \mathbb{Z}^{nm} \to \mathbb{Z}$ and $g : \mathbb{Z}^{nml} \to \mathbb{Z}$ be the separable convex functions defined by $f(y) := \sum_{i,j} f_{i,j}(y_{i,j})$ with $y_{i,j} := \sum_{k=1}^{l} v_k x^j_{i,k}$ and $g(x) := \sum_{i,j} \sum_{k=1}^{l} g^j_{i,k}(x^j_{i,k})$.

Now, note that $A^{(n)}x$ is an $(ml + nl)$-vector, whose first ml entries are the flows from each supplier of each commodity to all consumers, and whose last nl entries are the flows to each consumer of each commodity from all suppliers. So the supply and consumption equations are encoded by $A^{(n)}x = b$. Next, note that the nm-vector $y = (y_{1,1}, \dots, y_{m,1}, \dots, y_{1,n}, \dots, y_{m,n})$ satisfies $y = W^{(n)}x$. So the capacity constraints become $W^{(n)}x \le u$ and the cost function becomes $f(W^{(n)}x) + g(x)$. Therefore, the problem is precisely the following n-fold integer program:

$$\min \big\{ f\big(W^{(n)}x\big) + g(x) : x \in \mathbb{Z}^{nml},\ A^{(n)}x = b,\ W^{(n)}x \le u,\ x \ge 0 \big\}.$$

By Theorem 4.12, this program can be solved in polynomial time as claimed. □

4.4 Stochastic integer programming

Stochastic integer programming arises in decision making under uncertainty and is an important and extensively studied area with a variety of applications, see, for instance, [70] and the references therein. We now show that suitable adjustments of the methods of n-fold integer programming enable to solve, for the first time, stochastic integer programming problems in polynomial time as well.

In a stochastic integer program, part of the data is random, and decisions are made in two stages – before and after the realizations of the random data occur. The data for the

program is as follows. There are r first-stage decision variables arranged in a vector $y^0 = (y_1^0, \ldots, y_r^0)$ and s second-stage decision variables arranged in a vector $y = (y_1, \ldots, y_s)$. There are deterministic lower and upper bounds $l^0, u^0 \in \mathbb{Z}_\infty^r$ and cost $w^0 \in \mathbb{Z}^r$ for the first-stage decision vector, and random lower and upper bounds $\mathbf{l}, \mathbf{u} \in \mathbb{Z}_\infty^s$ and cost $\mathbf{w} \in \mathbb{Z}^s$ for the second-stage decision vector. The first-stage and second-stage decision vectors are tied together by a common system of linear equations defined by fixed $r \times t$ integer matrix A_1, fixed $s \times t$ integer matrix A_2, and random right-hand side vector $\mathbf{b} \in \mathbb{Z}^t$. The objective is to minimize the sum of direct costs of first-stage decisions and expected costs of second-stage decisions. So the *stochastic integer programming problem* is as follows:

$$\min\left\{w^0 y^0 + \mathbb{E}\left[\mathbf{c}\left(y^0\right)\right] : y^0 \in \mathbb{Z}^r,\ l^0 \le y^0 \le u^0\right\}, \tag{4.4}$$

where

$$\mathbf{c}\left(y^0\right) := \min\left\{\mathbf{w}y : y \in \mathbb{Z}^s,\ y^0 A_1 + y A_2 = \mathbf{b},\ \mathbf{l} \le y \le \mathbf{u}\right\}.$$

In this section, vectors are interpreted as either rows or columns, as is clear from the context; in particular, in the equation system above, y^0, y, and $\mathbf{b}$ are rows.

It is a common practice to assume that the sample space is finite (possibly the result of discretization of an originally infinite space). So we assume that there are n possible sample points, the so-termed *scenarios*, where, for $k = 1, \ldots, n$, scenario k occurs with probability p_k, with corresponding realizations l^k, u^k, w^k, b^k of the random part of the data. Introducing, for each $k = 1, \ldots, n$, a copy $y^k := (y_1^k, \ldots, y_s^k)$ of the second-stage decision vector for scenario k, the stochastic integer program (4.4) can be replaced by its following equivalent deterministic counterpart:

$$\min\left\{w^0 y^0 + \textstyle\sum_{k=1}^n p_k w^k y^k : y_i^k \in \mathbb{Z},\ y^0 A_1 + y^k A_2 = b^k,\ l^k \le y^k \le u^k\right\}. \tag{4.5}$$

As is common, we refer to program (4.5) simply also as a *stochastic integer program*. Note that the number n of scenarios, which is possibly the result of discretization of an infinite space, is typically very large and hence is assumed to be variable.

Now, arrange the variables in an $(r + ns)$-vector $y := (y^0, y^1, \ldots, y^n)$ consisting of $n + 1$ *bricks*, and likewise, the costs as $w := (w^0, p_1 w^1, \ldots, p_n w^n) \in \mathbb{Z}^{r+ns}$, the lower and upper bounds as $l := (l^0, l^1, \ldots, l^n)$, $u := (u^0, u^1, \ldots, u^n) \in \mathbb{Z}_\infty^{r+ns}$, and the right-hand sides as $b := (b^1, \ldots, b^n) \in \mathbb{Z}^{nt}$. Let A be the $(r, s) \times t$ bimatrix with blocks A_1 and A_2. Then the program (4.5) can be rewritten in the form:

$$\min\left\{wy : y \in \mathbb{Z}^{r+ns},\ yA^{(n)} = b,\ l \le y \le u\right\}. \tag{4.6}$$

The system of equations in this program involves the n-fold product $A^{(n)}$ of A. However, (4.6) is *not* quite an n-fold program but rather a certain dual of it, since the row vector y of variables multiplies $A^{(n)}$ from the *left*. Therefore, in order to apply Graver bases methods to program (4.6), the relevant Graver basis is the Graver basis $\mathcal{G}((A^{(n)})^T)$ of the *transpose* of $A^{(n)}$ rather than of $A^{(n)}$ itself, that is, the set of $\sqsubseteq$-minimal elements among

the nonzero integer solutions of the system:

$$\begin{pmatrix} y^0 & y^1 & y^2 & \cdots & y^n \end{pmatrix} \begin{pmatrix} A_1 & A_1 & \cdots & A_1 \\ A_2 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 & \cdots & 0 \end{pmatrix}.$$

While $\mathcal{G}(A^{(n)})$ can be computed in time polynomial in n by Theorem 4.4, the Graver basis $\mathcal{G}((A^{(n)})^T)$ of the transpose *cannot* be computed in polynomial time, since, as the next example shows, even its cardinality can be exponential in n.

Example 4.16 (exponential Graver basis of stochastic integer program). Let A be the $(1,2) \times 1$ matrix with first block $A_1 = 1$ and second block $A_2 := (1\ 1)^T$. Then $A^{(n)}$ is totally unimodular and $\mathcal{G}(A^{(n)}) = \mathcal{C}(A^{(n)}) = \emptyset$ is empty. But the Graver basis $\mathcal{G}((A^{(n)})^T) = \mathcal{C}((A^{(n)})^T)$ of its totally unimodular transpose:

$$(A^{(n)})^T = \left(\begin{array}{c|c|c|c|c|c|c|c} 1 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ \hline 1 & 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline 1 & 0 & 0 & 0 & 0 & \cdots & 1 & 1 \end{array}\right)$$

consists of $2^{n+1} + 2n$ elements, $2n$ with $y^0 = 0$ and 2^{n+1} with $y^0 = \pm 1$ of the form:

$$\pm\big(1, y^1, \ldots, y^n\big), \quad y^k \in \big\{(-1,0),(0,-1)\big\}, \quad k = 1, \ldots, n.$$

We proceed to derive the recent result of [50] building on [52], showing that, nonetheless, stochastic integer programming *can* be solved in polynomial time. We use the following weaker but useful property of Graver bases of n-fold products which does hold for Graver bases of transposes of n-fold products as well. We give below only the proof for n-fold products. The proof for their transposes can be found in [52] which builds on an extension in [74] of the Gordan lemma [40].

Lemma 4.17. *For every fixed bimatrix A, there are finite sets G and H such that:*

1. *for all n and $g = (g^1, \ldots, g^n) \in \mathcal{G}(A^{(n)})$, we have $g^k \in G$ for all k;*
2. *for all n and $h = (h^0, h^1, \ldots, h^n) \in \mathcal{G}((A^{(n)})^T)$, we have $h^k \in H$ for all k.*

Proof. We provide the proof of part 1 only. The proof of part 2 can be found in [52]. Let $g(A)$ be the Graver complexity of A. By the results of Section 4.1 (see proof of Theorem 4.4), any element $g = (g^1, \ldots, g^n) \in \mathcal{G}(A^{(n)})$ is the n-lifting of some $z = (z^1, \ldots, z^{g(A)}) \in \mathcal{G}(A^{(g(A))})$. So each brick of g is equal to some brick of z. Therefore, G can be taken to be the set of all bricks appearing in some element of the constant basis $\mathcal{G}(A^{(g(A))})$. So G depends only on A and is independent of n. □

We need one more lemma, which replaces Lemmas 3.5 and 3.9 and provides the engine needed for an iterative greedy augmentation process analog to that of Lemma 3.10, without requiring the entire Graver basis to be given explicitly.

Lemma 4.18. *For every fixed integer $(r,s)\times t$ bimatrix A, there is an algorithm that, given n and vectors $l,u\in\mathbb{Z}_\infty^{r+ns}$ and $w,y\in\mathbb{Z}^{r+ns}$ with $l\le y\le u$, and letting*

$$S:=\big\{z\in\mathbb{Z}^{r+ns}:zA^{(n)}=b,\ l\le z\le u\big\},\quad b:=yA^{(n)},$$

either asserts that S is infinite or concludes that it is finite and finds $\hat{y}\in S$ with

$$w\hat{y}\le\min\big\{w(y+\lambda h):\lambda\in\mathbb{Z}_+,\ h\in\mathcal{G}\big((A^{(n)})^T\big),\ l\le y+\lambda h\le u\big\},\tag{4.7}$$

in time which is polynomial in n and $\langle l,u,w,y\rangle$.

Proof. Let H be the fixed set of bricks for A guaranteed to exist by Lemma 4.17. Adding $0\in\mathbb{Z}^r$ and $0\in\mathbb{Z}^s$ to H if necessary, we assume that these bricks are in H. Define the following:

$$L:=\big\{h\in\mathbb{Z}^{r+ns}:h^0\in H\cap\mathbb{Z}^r,\ h^k\in H\cap\mathbb{Z}^s,\ \big(h^0,h^k\big)A=0,\ k=1,\dots,n\big\}.$$

Since H contains every brick of every element of the Graver basis of $(A^{(n)})^T$, and $h\in\mathbb{Z}^{r+ns}$ satisfies $hA^{(n)}=0$ if and only if $(h^0,h^k)A=0$ for $k=1,\dots,n$, we have the following:

$$\mathcal{G}\big((A^{(n)})^T\big)\subset L\subseteq\mathcal{L}\big((A^{(n)})^T\big).\tag{4.8}$$

For each k and each brick $h^k\in H$ (in $\mathbb{Z}^r$ for $k=0$ and in $\mathbb{Z}^s$ for other k), determine

$$\lambda\big(k,h^k\big):=\sup\big\{\lambda\in\mathbb{Z}_+:l^k\le y^k+\lambda h^k\le u^k\big\}\in\mathbb{Z}_+\uplus\{\infty\}.\tag{4.9}$$

Note that there are at most $|H\cap\mathbb{Z}^r|+n|H\cap\mathbb{Z}^s|=O(n)$ such that $\lambda(k,h^k)$ and each can be easily computed by inspection, so all can be computed in polynomial time.

First, we claim that S is infinite if and only if there is a nonzero element $h=(h^0,\dots,h^n)\in L$ such that $\lambda(k,h^k)=\infty$ for $k=0,\dots,n$. Indeed, if such an h exists then $y+\lambda h\in S$ for all $\lambda\in\mathbb{Z}_+$ so S is infinite; if S is infinite then, by Lemma 3.5, there is such an $h\in\mathcal{G}((A^{(n)})^T)\subset L$. To check this infiniteness criterion in polynomial time, for each $h^0\in H\cap\mathbb{Z}^r$ with $\lambda(0,h^0)=\infty$, try to extend h^0 to an element $h=(h^0,h^1,\dots,h^n)\in L$ as follows: for $k=1,\dots,n$ search for a brick $h^k\in H\cap\mathbb{Z}^s$, nonzero if possible, satisfying $\lambda(k,h^k)=\infty$ and $(h^0,h^k)A=0$. Then S is infinite if and only if this process constructs some such nonzero h.

Second, assuming that S is finite, we show how to find $\hat{y}\in S$ satisfying (4.7). By (4.8), we can take $\hat{y}:=y+\lambda h$, where (λ,h) is any optimal pair for the problem:

$$\min\big\{w\lambda h:\lambda\in\mathbb{Z}_+\uplus\{\infty\},\ h\in L,\ l\le y+\lambda h\le u\big\}\tag{4.10}$$

with the convention $\lambda h:=0\in\mathbb{Z}^{r+ns}$ for the pair $\lambda=\infty$ and $h=0$. Since $0\in L$ and S is finite, this pair is the only feasible one with $\lambda=\infty$. Since $|L|$ may be exponential (see Example 4.16), we proceed to solve (4.10) indirectly as follows.

Let $\Lambda\subset\mathbb{Z}_+\uplus\{\infty\}$ be the set of $\lambda(k,h^k)$ obtained in (4.9) for all k and h^k. Produce a set $T\subseteq\Lambda\times L$ of pairs (λ,h) by applying the following process: for each $\lambda\in\Lambda$ and each brick $h^0\in H\cap\mathbb{Z}^r$ satisfying $\lambda(0,h^0)\ge\lambda$, try to extend h^0 to $h=(h^0,h^1,\dots,h^n)\in L$

as follows: for $k = 1, \ldots, n$ search for a brick $h^k \in H \cap \mathbb{Z}^s$ satisfying $\lambda(k, h^k) \geq \lambda$ and $(h^0, h^k)A = 0$, and if there are any, pick one with minimum $w^k h^k$; if h was completed successfully then add the pair (λ, h) to T. Note that $|T| \leq |\Lambda| \cdot |H \cap \mathbb{Z}^r| = O(n)$ and can be constructed in polynomial time.

We now claim that an optimal pair (λ, h) for (4.10) can be taken as any optimal pair for the following problem, which can be solved immediately by inspecting T as follows:

$$\min \{w\lambda h : (\lambda, h) \in T\}. \tag{4.11}$$

First, note that any $(\lambda, h) \in T$ satisfies $l \leq y + \lambda h \leq u$ so is feasible in (4.10). Second, consider any pair $(\bar{\lambda}, \bar{h})$ which is optimal for the problem (4.10). Let

$$\hat{\lambda} := \min \{\lambda(k, \bar{h}^k) : k = 0, \ldots, n\} \in \Lambda.$$

Then, in the process which generates the set T, when considering $\hat{\lambda}$ and $\bar{h}^0$, the vector $\bar{h}$ is a feasible extension; let $\hat{h}$ be the optimal extension chosen in the process. Then $(\hat{\lambda}, \hat{h}) \in T$. Now, $(\bar{\lambda}, \bar{h})$, $(0, 0)$ feasible in (4.10) imply $\hat{\lambda} \geq \bar{\lambda}$, $w\bar{h} \leq 0$, and so

$$w\hat{\lambda}\hat{h} = \hat{\lambda} \sum_k w^k \hat{h}^k \leq \hat{\lambda} \sum_k w^k \bar{h}^k = \hat{\lambda} w\bar{h} \leq \bar{\lambda} w\bar{h} = w\bar{\lambda}\bar{h}.$$

Therefore, $(\hat{\lambda}, \hat{h})$ is also optimal for (4.10) as claimed. This completes the proof. □

We are now in position to prove the following important result of [50] which shows that stochastic integer programming can be solved in polynomial time.

Theorem 4.19. *For each fixed integer $(r, s) \times t$ bimatrix A, there is an algorithm that, given positive integer n, $l, u \in \mathbb{Z}_\infty^{r+ns}$, $b \in \mathbb{Z}^{nt}$, and $w \in \mathbb{Z}^{r+ns}$, solves in time which is polynomial in n and $\langle l, u, b, w \rangle$ the following stochastic integer program:*

$$\min \{wy : y \in \mathbb{Z}^{r+ns},\ yA^{(n)} = b,\ l \leq y \leq u\}. \tag{4.12}$$

Proof. First, we show the following analogs of Lemmas 3.10 and 3.11 for stochastic integer programming (restricted to optimization of linear objective functions).

Claim 1 (analog of Lemma 3.10). Given an initial feasible point y to the stochastic integer program (4.12), we can solve the program in polynomial time.

Proof of Claim 1. First, apply the algorithm of Lemma 4.18 and either detect that the feasible set is infinite and stop, or conclude it is finite and continue. Next, produce a sequence of feasible points $y^0, y^1, \ldots, y^s$ with $y^0 := y$ the given input point, as follows. Having obtained y^k, obtain a new feasible point y^{k+1} satisfying the following:

$$wy^{k+1} \leq \min \{w(y^k + \lambda h) : \lambda \in \mathbb{Z}_+,\ h \in \mathcal{G}((A^{(n)})^T),\ l \leq y^k + \lambda h \leq u\}, \tag{4.13}$$

by applying again the algorithm of Lemma 4.18. If $wy^{k+1} < wy^k$ then repeat, else stop and output the last point y^s. The proof now proceeds like that of Lemma 3.10, noting that a linear function wy is separable convex, and that inequality (4.13) assures that the sequence of feasible points converges to an optimal point at least as fast as in the algorithm of Lemma 3.10 and hence in polynomial time. □

Claim 2 (analog of Lemma 3.11). We can either find a point feasible in (4.12) or detect the program is infeasible or the feasible set is infinite in polynomial time.

Proof of Claim 2. Assume that $l \leq u$ and that $l_j < \infty$ and $u_j > -\infty$ for all j, else there is no feasible point. Now, either detect that there is no integer solution to the system of equations $yA^{(n)} = b$ (without the lower and upper bound constraints) and stop, or determine some such solution $y \in \mathbb{Z}^{r+ns}$ and continue (in polynomial time, using the Hermite normal form of $(A^{(n)})^T$, see [90]). Let

$$I := \{j : l_j \leq y_j \leq u_j\} \subseteq \{1, \ldots, r+ns\}$$

be the set of indices of entries of y which satisfy their lower and upper bounds. While $I \subsetneq \{1, \ldots, r+ns\}$, repeat the following procedure. Pick any index $i \notin I$. Then either $y_i < l_i$ or $y_i > u_i$. We describe the procedure only in the former case, the latter being symmetric. Update the lower and upper bounds by setting the following:

$$\hat{l}_j := \min\{l_j, y_j\}, \quad \hat{u}_j := \max\{u_j, y_j\}, \quad j = 1, \ldots, r+ns.$$

Solve in polynomial time the following linear integer program, in which y is feasible:

$$\max\{z_i : z \in \mathbb{Z}^{r+ns},\ zA^{(n)} = b,\ \hat{l} \leq z \leq \hat{u},\ z_i \leq u_i\}, \tag{4.14}$$

by applying the algorithm of Claim 1 above. Now, for all j, we have that $\hat{l}_j > -\infty$ if and only if $l_j > -\infty$ and $\hat{u}_j < \infty$ if and only if $u_j < \infty$. Therefore, if the algorithm asserts that the feasible set of (4.14) is infinite, then the original program (4.12) either has an infinite feasible set or is infeasible. So assume that the algorithm finds an optimal solution z. If $z_i < l_i$ then (4.12) is infeasible. Otherwise, set $y := z$, $I := \{j : l_j \leq y_j \leq u_j\}$, and repeat. Note that in each iteration, the cardinality of I increases by at least one. Therefore, after at most $r + ns$ iterations, either the algorithm detects that the original program is infeasible or has an infinite feasible set, or $I = \{1, \ldots, r+ns\}$ is obtained, in which case the current point y is feasible. □

To complete the proof of the theorem: first, apply the algorithm of Claim 2 and either conclude the program is infeasible or the feasible set is infinite and stop, or obtain some feasible point and continue. Next, apply the algorithm of Claim 1 and either conclude the feasible set is infinite or obtain an optimal solution. □

As for n-fold integer programming, a natural extension of (4.12) is the following *nonlinear stochastic integer programming problem* with objective $f : \mathbb{Z}^{r+ns} \to \mathbb{Z}$:

$$\min\{f(y) : y \in \mathbb{Z}^{r+ns},\ yA^{(n)} = b,\ l \leq y \leq u\}.$$

In [50], this more general problem is solved for *splittable* convex functions presented by suitable oracles. Moreover, using an extension of the finiteness Lemma 4.17 from [4], the polynomial time solvability of convex stochastic integer programming can be extended to the so-called *multistage* stochastic integer programming problems, where a multistage process of alternating decisions and observations takes place.

4.5 Graver approximation scheme

Consider again the linear or separable convex n-fold integer programming problem:

$$\min\{f(x) : x \in \mathbb{Z}^{nt},\ A^{(n)}x = b,\ l \le x \le u\}. \tag{4.15}$$

The solution of problem (4.15) by the algorithm of Theorem 4.8 involves two major tasks: first, the construction of the Graver basis $\mathcal{G}(A^{(n)})$ by the algorithm of Theorem 4.4; second, the iterative greedy augmentation of an initial point to an optimal one using the Graver basis $\mathcal{G}(A^{(n)})$ by the algorithm of Lemma 3.10.

As shown in the proof of Theorem 4.4, the cardinality of $\mathcal{G}(A^{(n)})$ is $O(n^g)$ with $g := g(A)$ the Graver complexity of A. So the time needed for constructing $\mathcal{G}(A^{(n)})$, as well as for its examination in finding a greedy augmentation in each iteration, involves an $O(n^g)$ term. While this is polynomial since g is fixed, it may be very large even for small A. We therefore proceed to suggest a scheme, parameterized by an integer $k \le g(A)$, which constructs an approximating subset G_k^n of the Graver basis $\mathcal{G}(A^{(n)})$ and uses it for augmentation, resulting in approximative solutions at lower complexity. We define this approximating set by the following:

$$G_k^n := \{x \in \mathcal{G}(A^{(n)}) : \text{type}(x) \le k\}. \tag{4.16}$$

The next lemma bounds the size of G_k^n and suggests an efficient procedure for computing it. It refines Theorem 4.4 which gives the special case of $G_g^n = \mathcal{G}(A^{(n)})$.

Lemma 4.20. *For any integer bimatrix A and pair of integers $1 \le k \le n$, we have the following:*

$$G_k^n := \{x \in \mathcal{G}(A^{(n)}) : \text{type}(x) \le k\} = \{x : x \text{ is an } n\text{-lifting of some } y \in \mathcal{G}(A^{(k)})\}.$$

For fixed A,k, the cardinality of G_k^n and the time needed to compute it are $O(n^k)$.

Proof. Consider any n-lifting x of any $y \in \mathcal{G}(A^{(k)})$. Suppose indirectly that $x \notin G_k^n$. Since $\text{type}(x) = \text{type}(y) \le k$ this implies that $x \notin \mathcal{G}(A^{(n)})$. Therefore, there exists some $x' \in \mathcal{G}(A^{(n)})$ with $x' \sqsubset x$. But then x' is the n-lifting of some $y' \in \mathcal{L}^*(A^{(k)})$ with $y' \sqsubset y$, which contradicts $y \in \mathcal{G}(A^{(k)})$. So $x \in G_k^n$.

Conversely, consider any $x \in G_k^n$. Then x is the n-lifting of some $y \in \mathcal{L}^*(A^{(k)})$. Suppose indirectly that $y \notin \mathcal{G}(A^{(k)})$. Then there exists some $y' \in \mathcal{G}(A^{(k)})$ with $y' \sqsubset y$. But then a suitable n-lifting x' of y' satisfies $x' \in \mathcal{L}^*(A^{(n)})$ with $x' \sqsubset x$, which contradicts $x \in \mathcal{G}(A^{(n)})$. So $y \in \mathcal{G}(A^{(k)})$.

Now, the number of n-liftings of each $y \in \mathcal{G}(A^{(k)})$ is at most $\binom{n}{k}$, and hence

$$|G_k^n| \le \binom{n}{k} |\mathcal{G}(A^{(k)})|.$$

If A and k are fixed then the Graver basis $\mathcal{G}(A^{(k)})$ is constant and hence the cardinality of G_k^n and the time needed to compute it are $O(n^k)$ as claimed. □

We proceed to outline our *Graver approximation scheme* for the n-fold integer programming problem (4.15), parameterized by $1 \leq k \leq g := g(A)$. For simplicity, we assume that the input includes an initial feasible point x_0 and that the feasible set is finite (which can be checked by linear programming). We also assume that $g(A) \geq 1$. The case $g(A) = 0$ occurs if and only if the columns of A_2 are linearly independent, in which case the same holds for $A^{(n)}$ for all n, and the given initial point is the only feasible one and hence already the optimal solution. Our scheme uses the following sequence of increasingly better approximations of $\mathcal{G}(A^{(n)})$:

$$\emptyset = G_0^n \subseteq G_1^n \subseteq \cdots \subseteq G_g^n = \mathcal{G}(A^{(n)}). \tag{4.17}$$

The containment relations in this sequence follow directly from the definition of the G_k^n in (4.16), and the equality $G_g^n = \mathcal{G}(A^{(n)})$ follows from the definition of the Graver complexity. For each parameter value $1 \leq k \leq g$, the scheme obtains recursively an initial feasible point x_{k-1}, computes G_k^n, and uses it for repeated greedy augmentations as long as possible, terminating with a new, typically better, feasible point x_k. Thus, the scheme produces a sequence of points satisfying the following:

$$f(x_0) \geq f(x_1) \geq \cdots \geq f(x_g) = f^*,$$

where the last point x_g, obtained from the scheme with parameter $k = g$ which uses the full Graver basis $G_g^n = \mathcal{G}(A^{(n)})$, is guaranteed by Lemma 3.10 to be an optimal solution attaining the optimal objective function value f^*.

Procedure 4.21 (Graver scheme for n-fold integer programming).

1. Obtain an initial feasible point x_{k-1} by applying the scheme recursively.
2. Compute the approximation G_k^n of $\mathcal{G}(A^{(n)})$ by the algorithm of Lemma 4.20.
3. Use the algorithm of Lemma 3.10 with G_k^n instead of $\mathcal{G}(A^{(n)})$ to iteratively greedily augment x_{k-1} to the best attainable point x_k and output this point.

The time taken by step 2 is $O(n^k)$ by Lemma 4.20 and hence polynomial in n. The running time of the iterative step 3 is harder to estimate and depends on the quality of the approximation of $\mathcal{G}(A^{(n)})$ by G_k^n and on the type of function f.

So the suggested Graver approximation scheme allows to tradeoff precision for complexity, where increased parameter value k typically results in improved approximation x_k with better objective value $f(x_k)$ but increased running time, with the highest parameter value $k = g = g(A)$ resulting in a true optimal solution x_g with $f(x_g) = f^*$ and running time $O(n^g)$, which, nonetheless, is polynomial.

To estimate the quality of an approximating point x_k and decide whether to settle for it or to increment k and apply the scheme for the next parameter value and hence for a longer time, it is possible to approximate in polynomial time the optimal objective value $\tilde{f}$ of the *continuous relaxation* of the problem (4.15):

$$\tilde{f} := \min\left\{f(x) : x \in \mathbb{R}^{nt},\ A^{(n)}x = b,\ l \leq x \leq u\right\} \leq f^*, \tag{4.18}$$

by efficiently minimizing a convex function over a polytope, see, for instance, [80].

The mere existence of a finite Graver complexity $g := g(A)$ for every A suffices to realize Scheme 4.21, since g and the constant Graver bases $\mathcal{G}(A^{(k)})$ for $k \leq g$ can be built into the algorithm. In particular, the value g allows to identify the highest parameter $k = g$, for which the scheme outputs a point $x_k = x_g$ which is a true optimal solution. Note that the relaxation (4.18) *cannot* be used to compute f^* and identify optimality via $f(x_k) = f^*$ since there is typically a gap $\tilde{f} < f^*$.

In practice, preparatory preprocessing is useful, where the Graver complexity $g(A)$ and each of the Graver bases $\mathcal{G}(A^{(k)})$ for $k \leq g$ are actually computed in constant time once and for all (say, by Algorithms 4.6 and 3.21). Then, given any n, Scheme 4.21 with any parameter $k \leq g$ can be quickly applied.

Notes

The core of the theory of n-fold integer programming is developed in the series of papers [23], [24], [50]. An important component in this theory is the notion of Graver complexity introduced in [89] which extends [3] and is extended in [56]. The applications to multicommodity flows are from [51]. Multicommodity flows have been studied extensively in the literature with different emphases. Let us mention the paper [68], which investigates max-flow min-cut properties and their use in approximation algorithms for multicommodity flow problems, and the references therein. Stochastic integer programming is an important and extensively studied area with a variety of applications, see [70] and the references therein. Part 2 of the finiteness Lemma 4.17 has been established in [52]. Theorem 4.19 providing the polynomial time solution of stochastic integer programming is from [50].

5 Multiway Tables and Universality

Multiway tables occur naturally in any context involving multiply-indexed variables. They have been studied extensively for decades, in operations research in the context of high dimensional transportation problems, and in statistics in the context of privacy in statistical databases and disclosure control, see [3], [7], [8], [22], [29], [36], [58], [62], [63], [75], [97], [100], and the references therein. In this chapter, we study multiway tables in depth, completely settle the algorithmic complexity of multiway table problems, and discuss some of their important applications in these areas.

A *d-way table* is an $m_1 \times \cdots \times m_d$ array $x = (x_{i_1,\ldots,i_d})$ of *nonnegative integers*. We say that multiway table x has *format* $m_1 \times \cdots \times m_d$ and call $m_1, \ldots, m_d$ the *sides* of x. Following statistical terminology, a *margin* of table x is the sum of entries in some lower dimensional subarray of x and can be a *line-sum*, *plane-sum*, *hyperplane-sum*, and so on. Table problems involve the set of multiway tables with some of their margins specified. For clarity, through most of this chapter we discuss line-sum constraints, which are the most difficult and already illustrate the main ideas developed herein. But all of our algorithmic results hold much more generally for any *hierarchical margins*, as discussed later in Section 5.2.3.

The set of multiway tables with some of their margins specified is the set of nonnegative integer arrays satisfying a system of linear equations, and therefore is the set of feasible points in a suitable integer program. For instance, the set of 3-way tables of format $l \times m \times n$ with all line-sums specified is as follows:

$$S := \left\{ x \in \mathbb{Z}_+^{l \times m \times n} : \sum_i x_{i,j,k} = v_{*,j,k},\ \sum_j x_{i,j,k} = v_{i,*,k},\ \sum_k x_{i,j,k} = v_{i,j,*} \right\},$$

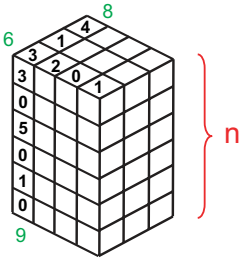
where the specified line-sums are $mn + ln + lm$ given nonnegative integer numbers:

$$v_{*,j,k},\ v_{i,*,k},\ v_{i,j,*}, \quad 1 \le i \le l,\ 1 \le j \le m,\ 1 \le k \le n.$$

The set of margin-constrained multiway tables of format $m_1 \times \cdots \times m_d \times n$ is shown later in this chapter to form an n-fold system (see Figure 5.1). Therefore, the theory of n-fold integer programming developed in Chapter 4 applies and provides the first polynomial time algorithms for treating such tables with $m_1, \ldots, m_d$ fixed and n variable – the so-termed *long tables*. In contrast, we show that margin-constrained multiway tables with two variable sides, in fact, already 3-way tables of format $l \times m \times 3$ – the so-termed *short tables* – are *universal* for integer programming. This universality theorem provides a powerful tool in establishing the presumable limits of polynomial time solvability of multiway table problems and is used in the applications Section 5.2 to contrast the polynomial time solvability over long multiway tables which is attainable by n-fold integer programming.

In Section 5.1, we prove the universality Theorem 5.1 showing that every integer program is one over $l \times m \times 3$ line-sum constrained tables. In Section 5.2, we discuss important applications of the n-fold integer programming theory from Chapter 4 and the universality theorem to multiindex transportation problems and privacy in statistical databases.

Consider $m_1 \times \cdots \times m_d \times n$ tables with given margins such as line-sums:



$$A^{(n)} = \begin{pmatrix} A_1 & A_1 & \cdots & A_1 \\ A_2 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_2 \end{pmatrix}$$

n

Such tables form an n-fold system $\{x : A^{(n)}x = b,\ x \geq 0,\ x \text{ integer}\}$ for suitable bimatrix A determined by $m_1, \ldots, m_d$, where A_1 controls equations of margins which involve summation over layers, whereas A_2 controls equations of margins involving summation within a single layer at a time.

Figure 5.1: Multiway tables

In Section 5.3, we prove that n-fold integer programming is universal in Theorem 5.12 and use it to suggest a heuristical scheme for arbitrary (non)linear integer programming that builds on the Graver approximation scheme of Section 4.5. Finally, in Section 5.4 we discuss the *Graver complexity of (di)graphs*, new fascinating invariants that control the complexity of multiway table and multicommodity flow problems. We establish an exponential lower bound in Theorem 5.19 providing unusual type of evidence supporting the P $\neq$ NP hypothesis.

The table below enables quick navigation among most results of this chapter.

	Polynomial time solvability	Universality and intractability
n-fold integer programming	n-folds of bimatrices (Theorems of Chapter 4)	n-folds of m-folds Theorem 5.12
Multiindex transportation	Long, $m_1 \times \cdots \times m_d \times n$ Corollaries 5.3 and 5.4	Short, $l \times m \times 3$ Theorem 5.1 Corollary 5.2
Privacy in databases	Long, $m_1 \times \cdots \times m_d \times n$ Corollary 5.8	Short, $l \times m \times 3$ Corollary 5.5 Corollary 5.7
Hierarchical margins	Long, $m_1 \times \cdots \times m_d \times n$ Corollaries 5.10 and 5.11	Short, $l \times m \times 3$ Theorem 5.1

5.1 The universality theorem

A *d-way polytope* is the set of $m_1 \times \cdots \times m_d$ nonnegative *real* arrays $x = (x_{i_1,\ldots,i_d})$ satisfying given margin constraints. So the integer points in such a polytope are precisely the d-way tables of the same format which satisfy the same margin constraints. For instance, the 3-way line-sum polytope of format $l \times m \times n$ is as follows:

$$T := \left\{x \in \mathbb{R}_+^{l\times m\times n} : \sum_i x_{i,j,k} = v_{*,j,k},\ \sum_j x_{i,j,k} = v_{i,*,k},\ \sum_k x_{i,j,k} = v_{i,j,*}\right\},$$

and the corresponding set of line-sum constrained 3-way tables is $S = T \cap \mathbb{Z}^{l\times m\times n}$.

We now establish the *universality theorem* of [27] which shows that, quite remarkably, *any* rational polytope is a 3-way $l \times m \times 3$ line-sum polytope. A polytope $P \subset \mathbb{R}^p$ is *representable* as a polytope $Q \subset \mathbb{R}^q$ if there is an injection $\sigma : \{1,\ldots,p\} \to \{1,\ldots,q\}$ such that the coordinate-erasing projection:

$$\pi_\sigma : \mathbb{R}^q \longrightarrow \mathbb{R}^p : x = \big(x_1,\ldots,x_q\big) \longmapsto y = \big(y_1,\ldots,y_p\big) := \big(x_{\sigma(1)},\ldots,x_{\sigma(p)}\big)$$

provides a bijection between Q and P and between $Q \cap \mathbb{Z}^q$ and $P \cap \mathbb{Z}^p$. If P is representable as Q then the two are structurally isomorphic in any reasonable sense. If moreover the injection σ and the corresponding projection π_σ are polynomial time computable, then P and Q are also algorithmically isomorphic in any reasonable sense. The two are linearly equivalent and hence all (non)linear programming problems over the two are polynomial time equivalent; they are combinatorially equivalent and hence they have the same face numbers and facial structure; they are integrally equivalent and hence all (non)linear integer programming and integer counting problems over the two are polynomial time equivalent.

We now provide an outline of the proof of the universality theorem. Complete details and more consequences of this theorem can be found in [26], [27], [28]. In the statement and proof below, we use an abbreviated notation for the line-sums.

Theorem 5.1. *Every rational polytope $P = \{y \in \mathbb{R}_+^d : Ay = b\}$ is in polynomial time computable integer preserving bijection with some $l \times m \times 3$ line-sum polytope:*

$$T := \left\{x \in \mathbb{R}_+^{l\times m\times 3} : \sum_i x_{i,j,k} = z_{j,k},\ \sum_j x_{i,j,k} = v_{i,k},\ \sum_k x_{i,j,k} = u_{i,j}\right\}. \quad (5.1)$$

More precisely, there is an algorithm that, given integer A and b with P bounded, produces l, m and integer line-sums $u = (u_{i,j})$, $v = (v_{i,k})$ and $z = (z_{j,k})$, in time polynomial in $\langle A, b\rangle$, such that the polytope P is representable as the polytope T.

Proof. The proof consists of three polynomial time constructive steps, each representing a polytope of a given format as a polytope of another given format.

First step. 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 enables the rest of the steps to run in polynomial time. For each variable y_j, let k_j be the maximum number of bits in the binary representation of the absolute value of any coefficient of y_j in the system $Ay = b$, that

is, $k_j := \max_i \lfloor \log_2 |a_{i,j}| \rfloor$. Now introduce variables $x_{j,0}, \ldots, x_{j,k_j}$ and relate them by the equations $2x_{j,i} - x_{j,i+1} = 0$ for all i. 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 step. 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 *plane-sum* polytope of the form:

$$\big\{x \in \mathbb{R}_+^{r \times s \times h} : \textstyle\sum_{i,j} x_{i,j,k} = z_k,\ \sum_{i,k} x_{i,j,k} = v_j,\ \sum_{j,k} x_{i,j,k} = u_i\big\}.$$

Note that a face of such a polytope 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". Since Q is a polytope and hence bounded, we can compute by Cramer's rule an integer upper bound U on the value of any coordinate y_j of any $y \in Q$. For each j, let r_j be the maximum between the sum of positive coefficients of y_j and sum of absolute values of negative coefficients of y_j in the system $Ay = b$:

$$r_j := \max\Big(\sum_k \big\{a_{k,j} : a_{k,j} > 0\big\},\ \sum_k \big\{\big|a_{k,j}\big| : a_{k,j} < 0\big\}\Big).$$

Assume that A is a $c \times d$ matrix. Let $r := \sum_{j=1}^{d} r_j$, $R := \{1, \ldots, r\}$, $h := c + 1$ 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, non-forbidden, 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 as follows:

$$\begin{aligned} F := \big\{x \in \mathbb{R}_+^{r \times r \times h} : {} & x_{i,j,k} = 0 \text{ for all } (i,j,k) \notin E, \\ & \text{and } \textstyle\sum_{i,j} x_{i,j,k} = z_k,\ \sum_{i,k} x_{i,j,k} = v_j,\ \sum_{j,k} x_{i,j,k} = u_i\big\}. \end{aligned}$$

We also indicate the injection $\sigma : \{1, \ldots, d\} \to R \times R \times H$ giving the desired embedding of coordinates y_j as coordinates $x_{i,j,k}$ and the representation of the polytope Q as F. Roughly, each equation $k = 1, \ldots, c$ 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, $j = 1, \ldots, d$ is encoded in a "vertical box" $R_j \times R_j \times H$, where $R = \biguplus_{j=1}^{d} R_j$ is the natural partition of R with $|R_j| = r_j$ for all $j = 1, \ldots, d$, that is, with $R_j := \{1 + \sum_{l<j} r_l, \ldots, \sum_{l \leq j} r_l\}$.

Now, all "vertical" plane-sums are defined to have the same value U, that is, $u_j := v_j := U$ for $j = 1, \ldots, r$. All entries not in the union $\biguplus_{j=1}^{d} R_j \times R_j \times H$ of the variable boxes are 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 $r_1 \geq 2$.

In the case $r_1 = 1$, we have $R_1 = \{1\}$; the box, which is just the single line $\{1\} \times \{1\} \times H$, has 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 $\bar{y}_1 := U - y_1$ (and likewise for other variables). The vertical sums u, v then force $\bar{y}_1 = U - y_1 = U - x_{1,1,k^+} = x_{1,1,k^-}$, so the complement of y_1 is also embedded.

Consider next the case $r_1 \geq 2$. For each $s = 1, \ldots, r_1$, the line $\{s\} \times \{s\} \times H$ contains one enabled entry $(s, s, k^+(s))$, and the line $\{s\} \times \{1 + (s \mod r_1)\} \times H$ contains one enabled entry $(s, 1+(s \mod r_1), k^-(s))$. All other entries of $R_1 \times R_1 \times H$ are 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 the following:

$$x_{s,s,k^+(s)} = x_{1,1,k^+(1)} = y_1 \quad \text{and} \quad x_{s,1+(s \bmod r_1),k^-(s)} = U - x_{1,1,k^+(1)} = \bar{y}_1$$

for each $s = 1, \ldots, r_1$. So both y_1 and $\bar{y}_1$ are each embedded in r_1 distinct entries.

To clarify the above description, it is helpful to visualize the $R \times R$ matrix $(x_{i,j,+})$ whose entries are the vertical line-sums $x_{i,j,+} := \sum_{k=1}^{h} x_{i,j,k}$. For instance, if we have three variables with $r_1 = 3$, $r_2 = 1$, $r_3 = 2$ then we get $R_1 = \{1, 2, 3\}$, $R_2 = \{4\}$, $R_3 = \{5, 6\}$, and the line-sums matrix $x = (x_{i,j,+})$ is as follows:

$$\begin{pmatrix} x_{1,1,+} & x_{1,2,+} & 0 & 0 & 0 & 0 \\ 0 & x_{2,2,+} & x_{2,3+} & 0 & 0 & 0 \\ x_{3,1,+} & 0 & x_{3,3,+} & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{4,4,+} & 0 & 0 \\ 0 & 0 & 0 & 0 & x_{5,5,+} & x_{5,6,+} \\ 0 & 0 & 0 & 0 & x_{6,5,+} & x_{6,6,+} \end{pmatrix} = \begin{pmatrix} y_1 & \bar{y}_1 & 0 & 0 & 0 & 0 \\ 0 & y_1 & \bar{y}_1 & 0 & 0 & 0 \\ \bar{y}_1 & 0 & y_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & U & 0 & 0 \\ 0 & 0 & 0 & 0 & y_3 & \bar{y}_3 \\ 0 & 0 & 0 & 0 & \bar{y}_3 & y_3 \end{pmatrix}.$$

We now encode the equations by defining the horizontal plane-sums z and the indices $k^+(s)$, $k^-(s)$ above as follows. For $k = 1, \ldots, c$, consider the kth equation $\sum_j a_{k,j} y_j = b_k$. Define the index sets as follows:

$$J^+ := \big\{j : a_{k,j} > 0\big\}, \quad J^- := \big\{j : a_{k,j} < 0\big\},$$

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 $z_h = z_{c+1} := r \cdot U - \sum_{k=1}^{c} z_k$. Now, with $\bar{y}_j := U - y_j$ the complement of variable y_j as above, the kth equation can be rewritten as follows:

$$\sum_{j \in J^+} a_{k,j} y_j + \sum_{j \in J^-} \big|a_{k,j}\big| \bar{y}_j = \sum_{j=1}^{d} a_{k,j} y_j + U \cdot \sum_{j \in J^-} \big|a_{k,j}\big| = b_k + U \cdot \sum_{j \in J^-} \big|a_{k,j}\big| = z_k.$$

To encode this equation, we simply "pull down" to the corresponding kth horizontal plane as many copies of each variable y_j or $\bar{y}_j$ 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 $k^+(s) := c + 1$ or $k^-(s) := c + 1$.

For instance, if $c = 8$, the first variable y_1 has $r_1 = 3$, its coefficient $a_{4,1} = 3$ in the fourth equation is positive, its coefficient $a_{7,1} = -2$ in the seventh equation is negative, and $a_{k,1} = 0$ for $k \neq 4, 7$, then we set $k^+(1) = k^+(2) = k^+(3) := 4$ (so $\sigma(1) := (1, 1, 4)$ embedding y_1 as $x_{1,1,4}$), $k^-(1) = k^-(2) := 7$, and $k^-(3) := h = 9$. This completes the encoding and provides the desired representation.

Third step. We show that any 3-way plane-sum polytope, with possibly additional upper bound inequalities on the entries, of the form:

$$F := \left\{ y \in \mathbb{R}_+^{r\times s\times h} : \sum_{i,j} y_{i,j,k} = c_k,\ \sum_{i,k} y_{i,j,k} = b_j,\ \sum_{j,k} y_{i,j,k} = a_i,\ y_{i,j,k} \le e_{i,j,k} \right\}$$

can be represented as a 3-way line-sum polytope (with no entry upper bounds):

$$T := \left\{ x \in \mathbb{R}_+^{l\times m\times 3} : \sum_I x_{I,J,K} = z_{J,K},\ \sum_J x_{I,J,K} = v_{I,K},\ \sum_K x_{I,J,K} = u_{I,J} \right\}.$$

In particular, this implies that any face F of a 3-way plane-sum polytope, prescribed by a set E of enabled entries, can be represented as a 3-way line-sum polytope T, by setting an upper-bound $e_{i,j,k} := 0$ on each forbidden entry $(i,j,k) \notin E$ and redundant upper-bound $e_{i,j,k} := U$ on each enabled entry $(i,j,k) \in E$. We describe the presentation, but omit the proof that it is indeed valid; further details on this step can be found in [26], [27]. 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 $l := r \cdot s$ and $m := h + r + s$. The first index I of each entry $x_{I,J,K}$ is a pair $I = (i,j)$ in the l-set:

$$\left\{ (1,1), \ldots, (1,s), (2,1), \ldots, (2,s), \ldots, (r,1), \ldots, (r,s) \right\}.$$

The second index J of each entry $x_{I,J,K}$ is a pair $J = (s,t)$ in the m-set:

$$\left\{ (1,1), \ldots, (1,h), (2,1), \ldots, (2,r), (3,1), \ldots, (3,s) \right\}.$$

The last index K simply ranges in the 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 $\max\{a_1, \ldots, a_r\}$ and $\max\{b_1, \ldots, b_s\}$. The line-sums are set to be as follows:

$$u_{(i,j),(1,q)} = e_{i,j,q}, \quad u_{(i,j),(2,q)} = \begin{cases} U & \text{if } q = i, \\ 0 & \text{otherwise,} \end{cases} \quad u_{(i,j),(3,q)} = \begin{cases} U & \text{if } q = j, \\ 0 & \text{otherwise,} \end{cases}$$

$$v_{(i,j),p} = \begin{cases} U & \text{if } p = 1, \\ \displaystyle\sum_k e_{i,j,k} & \text{if } p = 2, \\ U & \text{if } p = 3, \end{cases} \quad z_{(p,q),1} = \begin{cases} c_q & \text{if } p = 1, \\ s \cdot U - a_q & \text{if } p = 2, \\ 0 & \text{if } p = 3, \end{cases}$$

$$z_{(p,q),2} = \begin{cases} \left(\displaystyle\sum_{i,j} e_{i,j,q} \right) - c_q & \text{if } p = 1, \\ 0 & \text{if } p = 2, \\ b_q & \text{if } p = 3, \end{cases} \quad z_{(p,q),3} = \begin{cases} 0 & \text{if } p = 1, \\ a_q & \text{if } p = 2, \\ r \cdot U - b_q & \text{if } p = 3. \end{cases}$$

Finally, applying the first step to the given polytope P, applying the second step to the resulting Q, and applying the third step to the resulting F, we obtain in polynomial time the desired $l \times m \times 3$ line-sum polytope T representing P. □

5.2 Some applications

We now discuss some of the applications of the n-fold integer programming theory of Chapter 4 and the universality Theorem 5.1 to multiway table problems, including multiindex transportation, privacy in statistical databases, and extensions.

5.2.1 Multiindex transportation problems

Recall from Section 1.2.2 that the multiindex transportation problem of [75] is the problem of finding a minimum cost multiway table with specified margins. So it is the most fundamental linear optimization problem over multiway tables.

Here, we discuss line-sums only. The extension to the much broader setting of hierarchical margin constraints is postponed to Section 5.2.3. So we consider the problem

$$\min\left\{wx : x\in\mathbb{Z}_+^{m_1\times\cdots\times m_d},\ \sum_{i_1} x_{i_1,\ldots,i_d}=v_{*,i_2,\ldots,i_d},\ldots,\ \sum_{i_d} x_{i_1,\ldots,i_d}=v_{i_1,\ldots,i_{d-1},*}\right\},$$

where the specified line-sums are $\sum_{k=1}^{d}\prod_{i\neq k} m_i$ given nonnegative integers:

$$v_{*,i_2,\ldots,i_d},\ldots,v_{i_1,\ldots,i_{d-1},*},\quad 1\leq i_1\leq m_1,\ldots,\ 1\leq i_d\leq m_d.$$

For $d=2$, this program is totally unimodular and can be solved in polynomial time. However, already for $d=3$ it is generally not, and the problem is much harder. Consider the problem over $l\times m\times n$ tables. If l,m,n are all fixed then the problem is solvable in polynomial time (in the natural binary length of the line-sums), but even in this very restricted situation one needs off-hand the algorithm of [69] for integer programming in fixed dimension lmn. If l,m,n are all variable then the problem is NP-hard [58]. The in-between cases are much more delicate and were resolved only recently. If two sides are variable and one is fixed then the problem is still NP-hard, even over short $l\times m\times 3$ tables with fixed $n=3$ [26], [27], as is implied by the universality Theorem 5.1. If two sides are fixed and one is variable, then the problem can be solved in polynomial time by n-fold integer programming. We proceed to discuss in more detail these in-between cases.

We start with the case of short, $l\times m\times 3$ tables, with two variable sides.

Corollary 5.2. *It is NP-complete to decide, given l,m, and binary-encoded integer line-sums $v_{i,j,*},v_{i,*,k},v_{*,j,k}$, if the following set of $l\times m\times 3$ tables is nonempty:*

$$S:=\left\{x\in\mathbb{Z}_+^{l\times m\times 3} : \sum_i x_{i,j,k}=v_{*,j,k},\ \sum_j x_{i,j,k}=v_{i,*,k},\ \sum_k x_{i,j,k}=v_{i,j,*}\right\}.$$

Proof. The problem is obviously in NP. To illustrate the power of the universality theorem we give two reductions. First, we reduce to it directly the *integer programming feasibility problem* of deciding, given integer A,b, if $Q:=\{y\in\mathbb{R}_+^n : Ay=b\}$ has an integer point, which is well known to be NP-complete. It is well known (see for instance [90]) that Q has an integer point if and only if the following polytope:

$$P:=\left\{(y,z)\in\mathbb{R}_+^{2n} : Ay=b,\ y_i+z_i=U,\ i=1,\ldots,n\right\}$$

has an integer point, where U is a suitable positive integer upper bound which can be efficiently computed from A, b by Cramer's rule. We can then apply the polynomial time algorithm of Theorem 5.1 to P and produce l, m and line-sums as above such that P and hence Q have integer points if and only if S is nonempty.

Second, we reduce to it the *subset-sum problem*, well known to be NP-complete, of deciding, given positive integers $a_0, a_1, \ldots, a_n$, if there is an $I \subseteq \{1, \ldots, n\}$ with $a_0 = \sum_{i \in I} a_i$, which holds if and only if there is an integer point in the polytope:

$$P := \left\{(y, z) \in \mathbb{R}^{2n}_+ : \textstyle\sum_{i=1}^n a_i y_i = a_0,\ y_i + z_i = 1\ i = 1 \ldots, n\right\}.$$

Now apply the polynomial time algorithm of Theorem 5.1 to P and produce l, m and line-sums as above such that P has an integer point if only if S is nonempty. □

The above NP-completeness statement indicates that multiway tables, already of format $l \times m \times 3$, are wildly behaved. The universality Theorem 5.1 provides a powerful tool in demonstrating such behavior, as the next example shows.

Example 1.2 (revisited; real-feasible integer-infeasible transportation). Using universality, we automatically recover the smallest known example, first discovered in [97], of a rational-nonempty, integer-empty, triple-index transportation polytope, as follows. We start with the polytope $P = \{y \geq 0 : 2y = 1\}$ in one variable, containing a single rational point but no integer point. The algorithm of Theorem 5.1 then represents it as a $6 \times 4 \times 3$ line-sum polytope T with line-sums given by the following three matrices (the verification of this is left to the reader):

$$\left(v_{i,j,*}\right) = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \quad \left(v_{i,*,k}\right) = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \quad \left(v_{*,j,k}\right) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

By the universality Theorem 5.1, the polytope T is integer-equivalent to P and hence also contains a single rational point (see Example 1.2) but no integer point.

We proceed to show that the important result, first discovered in [24], that the multi-index transportation problem with two sides fixed and one side variable, can be solved in polynomial time by n-fold integer programming. In fact, this result holds more generally for long, $m_1 \times \cdots \times m_d \times n$ multiway tables, of any dimension, with $m_1, \ldots, m_d$ fixed and n variable. Here, we give the proof for line-sum constraints and in Section 5.2.3 extend it to any hierarchical margin constraints. We note that even over $3 \times 3 \times n$ tables with line-sum constraints, the only solution of the problem available to date is the one below using n-fold integer programming.

Corollary 5.3. *For every fixed $d, m_1, \ldots, m_d$, there is an algorithm that, given n, integer $m_1 \times \cdots \times m_d \times n$ cost w and line-sums $v = ((v_{*,i_2,\ldots,i_{d+1}}), \ldots, (v_{i_1,\ldots,i_d,*}))$, solves in*

time polynomial in n and $\langle w, v\rangle$ the multiindex transportation problem:

$$\min wx = \sum_{i_1,\ldots,i_{d+1}} w_{i_1,\ldots,i_{d+1}} x_{i_1,\ldots,i_{d+1}}$$
$$\text{s.t.} \quad x \in \mathbb{Z}_+^{m_1\times\cdots\times m_d\times n},\ \sum_{i_1} x_{i_1,\ldots,i_{d+1}} = v_{*,i_2,\ldots,i_{d+1}},\ \ldots,\ \sum_{i_{d+1}} x_{i_1,\ldots,i_{d+1}} = v_{i_1,\ldots,i_d,*}.$$

Proof. Re-index all arrays as $x = (x^1,\ldots,x^n)$ with each $x^{i_{d+1}} = (x_{i_1,\ldots,i_d,i_{d+1}})$ a suitably indexed $m_1m_2\cdots m_d$ vector representing the i_{d+1}th layer of x. Define

$$t := r := m_1m_2\cdots m_d, \quad s := \sum_{k=1}^{d}\prod_{i\neq k} m_i.$$

Let $b := (b^0, b^1,\ldots,b^n) \in \mathbb{Z}^{r+ns}$, where $b^0 := (v_{i_1,\ldots,i_d,*})$ and for $i_{d+1} = 1,\ldots,n$,

$$b^{i_{d+1}} := \big((v_{*,i_2,\ldots,i_d,i_{d+1}}),\ldots,(v_{i_1,\ldots,i_{d-1},*,i_{d+1}})\big).$$

Let A be the $(t,s)\times t$ bimatrix with first block $A_1 := I_t$ the $t\times t$ identity matrix and second block A_2 a matrix defining the line-sum equations on $m_1\times\cdots\times m_d$ arrays. Then the equations $A_1(\sum_{i_{d+1}} x^{i_{d+1}}) = b^0$ represent the line-sum equations $\sum_{i_{d+1}} x_{i_1,\ldots,i_{d+1}} = v_{i_1,\ldots,i_d,*}$, where summations over layers occur, whereas the equations $A_2x^{i_{d+1}} = b^{i_{d+1}}$ for $i_{d+1} = 1,\ldots,n$ represent all other line-sum equations, where summations are within a single layer at a time (see Figure 5.1). So the multiindex transportation problem is encoded as the n-fold integer program:

$$\min\big\{wx : x\in\mathbb{Z}^{nt},\ A^{(n)}x = b,\ x\geq 0\big\}.$$

Using the algorithm of Theorem 4.7, this n-fold integer program, and hence the given multiindex transportation problem, can be solved in polynomial time. □

Corollary 5.3 extends immediately to nonlinear objective functions of the forms in Theorems 4.8–4.10. So we have the following solution of the nonlinear multiindex transportation problem of Section 1.2.2 for long tables, stated here for line-sums and extended in Section 5.2.3 to hierarchical margins. As usual, the function f is given by a comparison oracle and $\hat{f}$ is the maximum value of $|f(x)|$ over the feasible set.

Corollary 5.4. *For any fixed $d, m_1,\ldots,m_d$, there are algorithms that, given n and integer line-sums v, solve the following over $m_1\times\cdots\times m_d\times n$ tables satisfying v:*

1. $\min f(x)$ *with f given separable convex, in time polynomial in n and $\langle v,\hat{f}\rangle$;*
2. $\min\|x-\hat{x}\|_p$ *with $\hat{x}$ given goal table, in time polynomial in n, p and $\langle v,\hat{x}\rangle$;*
3. $\max f(w^1x,\ldots,w^kx)$ *with fixed number k of given integer $m_1\times\cdots\times m_d\times n$ weights w^i and given convex f, in time polynomial in n and $\langle v,w^1,\ldots,w^k\rangle$.*

5.2.2 Privacy in statistical databases

A common practice in the disclosure of sensitive data contained in a multiway table is to release some of the table margins rather than the entries of the table. 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 all tables having the same margins as those of the source table in the database. In particular, if this set consists of a unique value, that of the source table, then this entry can be exposed and privacy violated. This raises the following fundamental problem.

Entry uniqueness problem. Given a list of margin values and entry index, is the value which can occur in that entry in all tables with these margins unique?

The complexity of this problem turns out to behave in analogy to the complexity of the multiindex transportation problem discussed in Section 5.2.1. Consider the problem for $d = 3$ over $l \times m \times n$ tables. It is polynomial time decidable when l, m, n are all fixed, and coNP-complete when l, m, n are all variable [58]. The in-between cases are much more delicate and were resolved only recently. If two sides are variable and one is fixed then the problem is still coNP-complete, even over short $l \times m \times 3$ tables with fixed $n = 3$ [82]. If two sides are fixed and one is variable, then the problem can be solved in polynomial time by n-fold integer programming. We proceed to discuss in more detail these in-between cases.

We start with the case of short, $l \times m \times 3$ tables with two variable sides. Not only is the problem coNP-complete in this case, but also is much stronger; using Theorem 5.1, we can now obtain a result of [28] that *any set of nonnegative integers* is the set of values of an entry of some $l \times m \times 3$ tables with some specified line-sums.

Corollary 5.5. *For every finite set $S \subset \mathbb{Z}_+$ of nonnegative integers, there exist l, m, and line-sums for $l \times m \times 3$ tables, such that the set of values occurring in a given entry $x_{r,s,t}$ in all $l \times m \times 3$ tables x having these line-sums, is precisely S.*

Proof. Consider any finite set $S = \{s_1, \ldots, s_h\} \subset \mathbb{Z}_+$. Consider the polytope:

$$P := \left\{ y \in \mathbb{R}^{h+1}_+ : y_0 - \textstyle\sum_{j=1}^h s_j y_j = 0,\ \sum_{j=1}^h y_j = 1 \right\}.$$

By Theorem 5.1, there are (polynomial time computable) l, m and line-sums:

$$v_{*,j,k}, \quad v_{i,*,k}, \quad v_{i,j,*}, \quad 1 \le i \le l,\ 1 \le j \le m,\ 1 \le k \le 3,$$

defining a 3-way polytope T which represents P. Permuting the indices of arrays if necessary, we may assume that the coordinate y_0 is embedded in the entry $x_{r,s,t}$ under this presentation. Since P and T are integer-equivalent, the set of values attained by the entry $x_{r,s,t}$ in all tables with these line-sums is, as desired, as follows:

$$\left\{ x_{r,s,t} : x \in T \cap \mathbb{Z}^{l \times m \times 3} \right\} = \left\{ y_0 : y \in P \cap \mathbb{Z}^{h+1} \right\} = S.$$

□

Using the construction of Corollary 5.5 incorporating that of Theorem 5.1, we obtain the following (probably smallest possible) example of line-sums for $6 \times 4 \times 3$ tables, where some specific entry attains the set of values $\{0, 2\}$ which has a *gap*.

Example 5.6 (gap in the set of values of a table entry, see Figure 5.2). Applying our construction to the polytope $P = \{y \geq 0 : y_0 - 2y_1 = 0,\ y_1 + y_2 = 1\}$ in three variables, we obtain line-sums for $16 \times 11 \times 3$ tables such that the set of values of a specific entry in all tables with these line-sums is precisely $\{0, 2\}$ and has a gap. These can be reduced to line-sums for $6 \times 4 \times 3$ tables, given below and in Figure 5.2, where the designated entry also attains the values $0, 2$ only:

$$(v_{i,j,*}) = \begin{pmatrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 2 & 2 \\ 2 & 2 & 0 & 0 \\ 0 & 2 & 0 & 2 \\ 2 & 0 & 2 & 0 \end{pmatrix}, \quad (v_{i,*,k}) = \begin{pmatrix} 2 & 2 & 0 \\ 1 & 1 & 0 \\ 2 & 0 & 2 \\ 3 & 0 & 1 \\ 0 & 2 & 2 \\ 0 & 1 & 3 \end{pmatrix}, \quad (v_{*,j,k}) = \begin{pmatrix} 2 & 1 & 2 \\ 2 & 1 & 2 \\ 2 & 1 & 2 \\ 2 & 3 & 2 \end{pmatrix}.$$

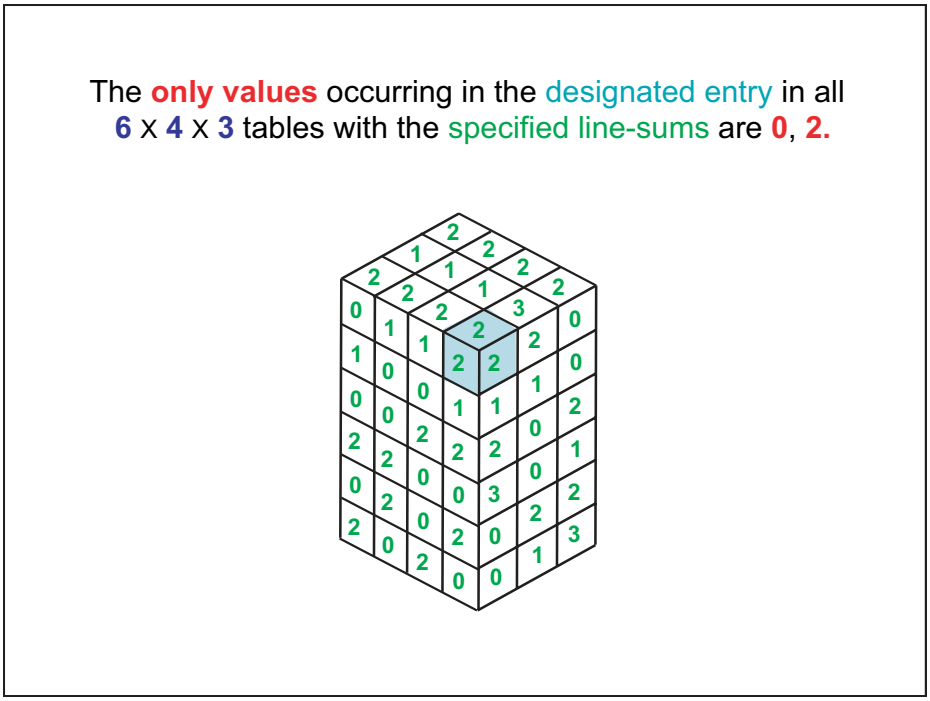


Figure 5.2: Set of entry values with a gap

The universality Theorem 5.1 also implies the following result of [82] on the complexity of entry uniqueness over $l \times m \times 3$ tables with two variable sides.

Corollary 5.7. *It is coNP-complete to decide, given l, m, and binary-encoded integer line-sums $v_{i,j,*}, v_{i,*,k}, v_{*,j,k}$, if $x_{1,1,1}$ attains a unique value in all tables in the following set:*

$$S := \left\{x \in \mathbb{Z}_+^{l\times m\times 3} : \sum_i x_{i,j,k} = v_{*,j,k},\ \sum_j x_{i,j,k} = v_{i,*,k},\ \sum_k x_{i,j,k} = v_{i,j,*}\right\}.$$

Proof. The problem is obviously in coNP. We reduce to it the *complement* of the subset-sum problem of deciding, given positive integers $a_0, a_1, \ldots, a_n$, if there is *no* $I \subseteq \{1, \ldots, n\}$ with $a_0 = \sum_{i\in I} a_i$, which is coNP-complete. Consider the polytope:

$$P := \left\{(y, z) \in \mathbb{R}_+^{2(n+1)} : a_0 y_0 - \sum_{i=1}^n a_i y_i = 0,\ y_i + z_i = 1,\ i = 0, 1, \ldots, n\right\}.$$

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 $I \subseteq \{1, \ldots, n\}$ 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 $i \in \{1, \ldots, n\} \setminus I$, and $z_i = 1 - y_i$ for all i. Now, apply the polynomial time algorithm of Theorem 5.1 and produce l, m and line-sums for $l \times m \times 3$ tables defining a 3-way polytope T which represents P. Permuting the indices of arrays if necessary, we may assume that the coordinate y_0 is embedded in the entry $x_{1,1,1}$ under this presentation. Since P and T are integer-equivalent, 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* subset $I \subseteq \{1, \ldots, n\}$ which satisfies $a_0 = \sum_{i\in I} a_i$. □

We proceed to show the result of [82] that entry uniqueness over tables with two sides fixed and one side variable can be decided in polynomial time by n-fold integer programming. In fact, this result holds more generally for long, $m_1 \times \cdots \times m_d \times n$ tables, of any dimension, with $m_1, \ldots, m_d$ fixed and n variable. Here, we give the proof for line-sums and in Section 5.2.3 extend it to any hierarchical margin constraints. We note that even over $3 \times 3 \times n$ tables with line-sum constraints, the only solution of the problem available to date is the one below using n-fold integer programming.

Corollary 5.8. *For every fixed $d, m_1, \ldots, m_d$, there is an algorithm that, given n, integer line-sums $v = ((v_{*,i_2,\ldots,i_{d+1}}), \ldots, (v_{i_1,\ldots,i_d,*}))$, and index $(k_1, \ldots, k_{d+1})$, solves in time polynomial in n and $\langle v\rangle$, the entry uniqueness problem of deciding if $x_{k_1,\ldots,k_{d+1}}$ attains a unique value in all multiway tables in the following set S:*

$$\left\{x \in \mathbb{Z}_+^{m_1\times\cdots\times m_d\times n} : \sum_{i_1} x_{i_1,\ldots,i_{d+1}} = v_{*,i_2,\ldots,i_{d+1}}, \ldots,\ \sum_{i_{d+1}} x_{i_1,\ldots,i_{d+1}} = v_{i_1,\ldots,i_d,*}\right\}.$$

Proof. By Corollary 5.3, we can solve in polynomial time both n-fold programs as follows:

$$l := \min\left\{x_{k_1,\ldots,k_{d+1}} : x \in S\right\},$$
$$u := \max\left\{x_{k_1,\ldots,k_{d+1}} : x \in S\right\}.$$

Clearly, entry $x_{k_1,\ldots,k_{d+1}}$ attains a unique value in all tables with the given line-sums if and only if $l = u$, which can therefore be tested in polynomial time. □

The algorithm of Corollary 5.8 and its extension to any hierarchical margins which is described in Section 5.2.3 allow statistical agencies to efficiently check possible 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 compromise privacy and hence fewer margins should be released.

We note that long tables, with one side much larger than the others, often arise in practical applications. For instance, in health statistical tables, the long factor may be the age of an individual, whereas other factors may be binary (yes-no) or ternary (subnormal, normal, and supnormal). Moreover, it is always possible to merge categories of factors, with the resulting coarser tables approximating the original ones, making the algorithm of Corollary 5.8 applicable.

Finally, we describe a procedure based on a suitable adaptation of the algorithm of Corollary 5.8, that constructs the entire set of values that can occur in a specified entry, rather than just deciding its uniqueness. Here, S is the set of tables satisfying the given line-sums, and the running time is output-efficient, that is, polynomial in the input length plus the number of elements in the output set.

Procedure 5.9 (constructing the set of values in an entry).

1. Initialize $l := -\infty$, $u := \infty$, and $E := \emptyset$;
2. solve in polynomial time the following linear n-fold integer programs:

$$\hat{l} := \min\left\{x_{k_1,\ldots,k_{d+1}} : l \le x_{k_1,\ldots,k_{d+1}} \le u,\ x \in S\right\},$$
$$\hat{u} := \max\left\{x_{k_1,\ldots,k_{d+1}} : l \le x_{k_1,\ldots,k_{d+1}} \le u,\ x \in S\right\};$$

3. if the problems in step 2 are feasible then update $l := \hat{l} + 1$, $u := \hat{u} - 1$, $E := E \uplus \{\hat{l}, \hat{u}\}$, and repeat step 2, else stop and output the set of values E.

5.2.3 Extensions to hierarchical margins

We now extend the results of Sections 5.2.1 and 5.2.2 to sets of multiway tables satisfying a much broader class of margin constraints. Consider any $m_1 \times \cdots \times m_d$ table x. For any tuple $(i_1, \ldots, i_d)$ with $i_j \in \{1, \ldots, m_j\} \uplus \{*\}$, the corresponding *margin* $x_{i_1,\ldots,i_d}$ of x is defined to be the sum of entries of x over all coordinates j for which $i_j = *$. The *support* of the tuple $(i_1, \ldots, i_d)$ and of the margin $x_{i_1,\ldots,i_d}$ is the set $\mathrm{supp}(i_1, \ldots, i_d) := \{j : i_j \neq *\}$ of nonsummed coordinates. For instance, if x is a $9 \times 5 \times 7 \times 8$ table then it has $63 = 9 \cdot 7$ margins with support $F = \{1, 3\}$ such as the following:

$$x_{6,*,7,*} = \sum_{i_2=1}^{5} \sum_{i_4=1}^{8} x_{6,i_2,7,i_4}.$$

A list of margins is *hierarchical* if, for some family $\mathcal{F}$ of subsets of $\{1, \ldots, d\}$, it consists of all margins with support in $\mathcal{F}$. Given a family $\mathcal{F}$ of subsets of $\{1, \ldots, d\}$ and a list $v = (v_{i_1,\ldots,i_d})$ of integer values for all margins supported on $\mathcal{F}$, the set of multiway tables satisfying the corresponding *hierarchical margin constraints* is as follows:

$$S_{\mathcal{F}} := \left\{x \in \mathbb{Z}_+^{m_1 \times \cdots \times m_d} : x_{i_1,\ldots,i_d} = v_{i_1,\ldots,i_d},\ \mathrm{supp}\left(i_1, \ldots, i_d\right) \in \mathcal{F}\right\}. \tag{5.2}$$

In particular, for any $0 \le k \le d$, the margins supported on the family $\mathcal{F}$ of all k-subsets of $\{1,\dots,d\}$ are precisely all margins which are sums of entries over $(d-k)$-dimensional subtables of x. In particular, for the family $\mathcal{F}$ of all $(d-1)$-subsets of $\{1,\dots,d\}$ (respectively, all $(d-2)$-subsets, or all 1-subsets), the set $S_{\mathcal{F}}$ of multiway tables in (5.2) is precisely the set of $m_1 \times \cdots \times m_d$ tables with all line-sums (respectively, all plan-sums, or all hyperplane-sums) specified.

The following important outcome of n-fold integer programming extends Corollaries 5.3 and 5.4 and provides the solution of the nonlinear multiindex transportation problem of Section 1.2.2 for long tables with any hierarchical margin constraints and broad classes of (non)linear objective functions. The function f is given by a comparison oracle and $\hat{f}$ is the maximum value of $|f(x)|$ over the feasible set.

Corollary 5.10. *For every fixed d, family $\mathcal{F}$ of subsets of $\{1,\dots,d+1\}$, and sides $m_1,\dots,m_d$, there are algorithms that, given n and list $v = (v_{i_1,\dots,i_{d+1}})$ of integer values for all margins supported on $\mathcal{F}$, solve the following (non)linear multiindex transportation problems over the set $S_{\mathcal{F}}$ of $m_1 \times \cdots \times m_d \times n$ tables satisfying v:*

1. $\min wx$ *with given $m_1 \times \cdots \times m_d \times n$ cost w, in time polynomial in n, $\langle v, w\rangle$;*
2. $\min f(x)$ *with f given separable convex, in time polynomial in n and $\langle v, \hat{f}\rangle$;*
3. $\min \|x - \hat{x}\|_p$ *with $\hat{x}$ given goal table, in time polynomial in n, p, and $\langle v, \hat{x}\rangle$;*
4. $\max f(w^1 x,\dots,w^k x)$ *with fixed number k of given integer $m_1 \times \cdots \times m_d \times n$ weights w^i and given convex f, in time polynomial in n and $\langle v, w^1,\dots,w^k\rangle$.*

Proof. Re-index the arrays as $x = (x^1,\dots,x^n)$ with each $x^j = (x_{i_1,\dots,i_d,j})$ a suitably indexed $m_1 m_2 \cdots m_d$ vector representing the jth layer of x. Then each of the above problems can be encoded as one over a suitable n-fold system:

$$S_{\mathcal{F}} = \left\{x \in \mathbb{Z}_+^{nt} : A^{(n)} x = b\right\}$$

with A an $(r,s) \times t$ bimatrix with $t := m_1 m_2 \cdots m_d$ and r, s, A_1 and A_2 determined by $\mathcal{F}$, and with right-hand side $b := (b^0, b^1,\dots,b^n) \in \mathbb{Z}^{r+ns}$ determined by the margins $v = (v_{i_1,\dots,i_{d+1}})$, such that the equations $A_1(\sum_{j=1}^n x^j) = b^0$ represent the constraints of all margins $x_{i_1,\dots,i_d,*}$ (where summation over layers occurs), whereas the equations $A_2 x^j = b^j$ for $j = 1,\dots,n$ represent the constraints of all margins $x_{i_1,\dots,i_d,j}$ with $j \ne *$ (where summations are within a single layer at a time).

The algorithms of Theorems 4.7–4.10 now enable to solve these n-fold programs and hence the given multiindex transportation problems in polynomial time. □

A similar extension of Corollary 5.8 allows statistical agencies to efficiently make learned decisions about secure disclosure of hierarchical lists of margins.

Corollary 5.11. *For every fixed d, family $\mathcal{F}$ of subsets of $\{1,\dots,d+1\}$, and sides $m_1,\dots,m_d$, there is an algorithm that, given n, list $v = (v_{i_1,\dots,i_{d+1}})$ of integer values for all margins supported on $\mathcal{F}$, and index $(k_1,\dots,k_{d+1})$, solves in time polynomial in n and $\langle v\rangle$ the entry uniqueness problem of deciding if $x_{k_1,\dots,k_{d+1}}$ attains a unique value over the set $S_{\mathcal{F}}$ of $m_1 \times \cdots \times m_d \times n$ tables satisfying v.*

Proof. By Corollary 5.10, we can solve in polynomial time both n-fold programs:

$$l := \min\left\{x_{k_1,\ldots,k_{d+1}} : x \in S_{\mathcal{F}}\right\},$$
$$u := \max\left\{x_{k_1,\ldots,k_{d+1}} : x \in S_{\mathcal{F}}\right\}.$$

Clearly, entry $x_{k_1,\ldots,k_{d+1}}$ attains a unique value in all tables with the given margins if and only if $l = u$, which can therefore be tested in polynomial time. □

Finally, as for the special case of line-sums in Section 5.2.2, we have an output-efficient algorithm for constructing the entire set of values that can occur in a specified entry over the set $S_{\mathcal{F}}$ of tables satisfying given hierarchical margin constraints.

Procedure 5.9 (revisited; constructing the set of values in an entry).

1. Initialize $l := -\infty$, $u := \infty$, and $E := \emptyset$;
2. solve in polynomial time the following linear n-fold integer programs:

$$\hat{l} := \min\left\{x_{k_1,\ldots,k_{d+1}} : l \le x_{k_1,\ldots,k_{d+1}} \le u,\ x \in S_{\mathcal{F}}\right\},$$
$$\hat{u} := \max\left\{x_{k_1,\ldots,k_{d+1}} : l \le x_{k_1,\ldots,k_{d+1}} \le u,\ x \in S_{\mathcal{F}}\right\};$$

3. if the problems in step 2 are feasible then update $l := \hat{l} + 1$, $u := \hat{u} - 1$, $E := E \uplus \{\hat{l}, \hat{u}\}$, and repeat step 2, else stop and output the set of values E.

5.3 Universality of *n*-fold integer programming

Let us introduce the following notation. For an integer $s \times t$ matrix D, let $\boxminus D$ be the $(t, s) \times t$ bimatrix with first block the $t \times t$ identity and second block D:

$$\boxminus D := \begin{pmatrix} I_t \\ D \end{pmatrix}.$$

Now, introduce a special form of the n-fold product, defined for a *matrix* D, by the following:

$$D^{[n]} := (\boxminus D)^{(n)} = \begin{pmatrix} I_t & I_t & \cdots & I_t \\ D & 0 & \cdots & 0 \\ 0 & D & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D \end{pmatrix}.$$

We consider such n-fold products of such m-fold products of the tiny fixed 1×3 matrix $(1\,1\,1)$. Note that $(1\,1\,1)^{[m]}$ is precisely the $(3+m)\times 3m$ incidence matrix of the complete bipartite graph $K_{3,m}$. For instance, for $m = 2$, it is the matrix:

$$\begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^{[2]} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

We can now rewrite Theorem 5.1 in the following compact and elegant form.

Theorem 5.12. *Every rational polytope* $\{y \in \mathbb{R}^d_+ : Ay = b\}$ *stands in polynomial time computable integer preserving bijection with some polytope of the special form:*

$$\left\{x \in \mathbb{R}^{3mn}_+ : \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^{[m][n]} x = a\right\}. \tag{5.3}$$

Proof. Identify the space $\mathbb{R}^{n\times m\times 3}$ of arrays with $\mathbb{R}^{3mn}$ by the coordinate ordering:

$$x = \left(x_{1,1,1}, x_{1,1,2}, x_{1,1,3}, \ldots, x_{1,m,1}, x_{1,m,2}, x_{1,m,3}, \ldots, x_{n,1,1}, \ldots, x_{n,m,3}\right).$$

Then the line-sum equations on arrays become the equations $(1\ 1\ 1)^{[m][n]}x = a$ on vectors. Therefore, the polytope (5.3) becomes the 3-way line-sum polytope (5.1) and the statement follows from the universality Theorem 5.1 for 3-way tables. □

Theorem 5.12 shows the universality of n-fold integer programming: every (non)linear integer program can be lifted in polynomial time to an equivalent n-fold program over a simple $\{0,1\}$-valued bimatrix $\boxminus(1\ 1\ 1)^{[m]}$ completely determined by a single parameter m. Indeed, consider any (bounded) integer program:

$$\min\left\{f(y) : y \in \mathbb{Z}^d_+,\ Ay = b\right\}. \tag{5.4}$$

Construct in polynomial time an injection $\sigma : \{1,\ldots,d\} \to \{1,\ldots,3mn\}$ embedding each coordinate y_i as some coordinate $x_{\sigma(i)}$ and representing the polytope $\{y \in \mathbb{R}^d_+ : Ay = b\}$ as a polytope (5.3). Then the problem (5.4) lifts to the following n-fold integer programming problem over the bimatrix $\boxminus(1\ 1\ 1)^{[m]}$:

$$\min\left\{f\left(x_{\sigma(1)},\ldots,x_{\sigma(d)}\right) : x \in \mathbb{Z}^{3mn}_+,\ \left(\boxminus\,(1\ 1\ 1)^{[m]}\right)^{(n)} x = a\right\}. \tag{5.5}$$

Moreover, for every fixed m, the program (5.5) can be solved in polynomial time for linear and a variety of nonlinear functions f by Theorems 4.7–4.10 and 4.12.

Let $g(m) := g(\boxminus(1\ 1\ 1)^{[m]})$ be the Graver complexity of the bimatrix defining the universal program (5.5). When solving this n-fold program for any fixed m, the time needed for constructing the relevant Graver basis $\mathcal{G}((1\ 1\ 1)^{[m][n]})$ and repeatedly using it in the iterative augmentation process involves an $O(n^{g(m)})$ term. While this is polynomial for each fixed m, it may be quite large even for small values of m. So we suggest to apply to (5.5) the approximation Scheme 4.21. This scheme, combined with universality, gives the following universal approximation scheme parameterized by k for arbitrary (non)linear integer programming.

Procedure 5.13 (universal scheme for (non)linear integer programming).

Input: Arbitrary nonlinear integer program (5.4) given by integer matrix A and vector b, function f presented by comparison oracle, and feasible point y_0:

1. check by linear programming if program (5.4) is unbounded. Suppose not;
2. use the algorithm of Theorem 5.12 to lift (5.4) to equivalent program (5.5);

3. obtain an initial feasible point x_{k-1} by applying the scheme recursively, with x_0 the point in program (5.5) corresponding to the point y_0 in program (5.4);
4. compute the Graver basis $\mathcal{G}((1\ 1\ 1)^{[m][k]})$;
5. compute the following approximation of $\mathcal{G}((1\ 1\ 1)^{[m][n]})$:

$$G_k^{m,n} := \{x \in \mathcal{G}((1\ 1\ 1)^{[m][n]}) : \text{type}(x) \le k\}$$
$$= \{x : x \text{ is an } n\text{-lifting of some } z \in \mathcal{G}((1\ 1\ 1)^{[m][k]})\};$$

6. use the algorithm of Lemma 3.10 with $G_k^{m,n}$ instead of $\mathcal{G}((1\ 1\ 1)^{[m][n]})$ to iteratively augment x_{k-1} to the best attainable point x_k and output it.

Several remarks about the complexity of this universal approximation scheme are in order. The time taken by step 2 is polynomial in $\langle A, b\rangle$ by Theorem 5.12. It can be shown that the Graver basis $\mathcal{G}((1\ 1\ 1)^{[m][k]})$ can be obtained by a suitable permutation of coordinates from the Graver basis $\mathcal{G}((1\ 1\ 1)^{[k][m]})$. Therefore, by Theorem 4.4, for fixed k and variable m, the computation of $\mathcal{G}((1\ 1\ 1)^{[m][k]})$ in step 4 and its cardinality are $O(m^{g(k)})$ with $g(k) = g(\boxminus(1\ 1\ 1)^{[k]})$ and hence polynomial in m. In particular, for $k = 2$ they are $O(m^3)$. The time for computing $G_k^{m,n}$ in step 5 and its cardinality for fixed k and variable m and n are as follows:

$$\binom{n}{k}\left|\mathcal{G}((1\ 1\ 1)^{[m][k]})\right| = O\left(n^k m^{g(k)}\right),$$

and so are polynomial in m and n; for $k = 2$ they are $O(n^2m^3)$. The time taken by the iterative step 6 is harder to estimate and depends on the quality of the approximation of $\mathcal{G}((1\ 1\ 1)^{[m][n]})$ provided by $G_k^{m,n}$ and the type of function f.

5.4 Graver complexity of graphs and digraphs

Recall that the *incidence matrix* of a graph or a digraph $G = (V, E)$ is the $V \times E$ matrix D defined as follows: for a graph, $D_{v,e} = 1$ if edge e contains vertex v and $D_{v,e} = 0$ if not; for a digraph, $D_{v,e} = -1$ if edge e leaves vertex v, $D_{v,e} = 1$ if e enters v, and $D_{v,e} = 0$ otherwise. Recall also that for an integer $s \times t$ matrix D, we denote by $\boxminus D$ the $(t, s) \times t$ bimatrix with first block I_t and second block D.

In this section, we study the following new graph and digraph invariant, the significance of which in nonlinear integer programming is explained below.

Definition 5.14. The *Graver complexity* of a graph or a digraph G is the Graver complexity $g(G) := g(\boxminus D)$ of the bimatrix $\boxminus D$ with D the incidence matrix of G.

Unfortunately, our understanding of this invariant is very limited. While it can be computed by the finite Algorithm 4.6, this computation is out of reach even for very small graphs. For instance, already the Graver complexity $g(K_{3,4})$ of the small complete

bipartite graph $K_{3,4}$ on 7 vertices is unknown. Nonetheless, in this section we do establish some bounds on the Graver complexity of the graphs $K_{3,m}$. The main result here is Theorem 5.19: the complexity $g(K_{3,m})$ is exponential.

Here is a small example where we can compute the Graver complexity precisely.

Example 5.15 (the bipartite graph $K_{3,2}$). Let $A := (\boxminus D)$ be the $(6,5) \times 6$ bimatrix with $A_1 = I_6$ and $A_2 = D$ the incidence matrix of the graph $K_{3,2}$:

$$D = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^{[2]} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

We compute the Graver complexity $g(K_{3,2}) = g(\boxminus D) = g(A)$ by Algorithm 4.6. Since $A_2 = D$ is totally unimodular, we have $\mathcal{G}(A_2) = \mathcal{G}(D) = \mathcal{C}(D)$ by Lemma 3.19. Let G_2 be the matrix having as columns the elements of the Graver basis $\mathcal{G}(A_2)$:

$$G_2 := \mathcal{G}(A_2) = \mathcal{G}(D) = \mathcal{C}(D) = \begin{pmatrix} 1 & -1 & 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 & 1 & -1 \end{pmatrix}.$$

The Graver basis of $A_1 G_2 = I_6 G_2 = G_2$ (computed, say, by Algorithm 3.21) has 22 elements which are the columns of the following matrix and their antipodals:

$$\mathcal{G}(A_1 G_2) = \mathcal{G}(A_2) = \pm \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 \\ 0 & 1 & 0 & -1 & -1 & 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 \end{pmatrix}.$$

Using Algorithm 4.6, we obtain the Graver complexity of the bipartite graph $K_{3,2}$:

$$g(K_{3,2}) = \max\{\|v\|_1 : v \in \mathcal{G}(A_1 G_2)\} = 3.$$

The time taken by our algorithms for (non)linear n-fold integer programming over a bimatrix A involves an $O(n^{g(A)})$ term needed for constructing the relevant Graver basis $\mathcal{G}(A^{(n)})$ and repeatedly using it in the iterative augmentation process.

Consider the many-commodity transshipment problem over a digraph G solved in Section 4.3.1. The bimatrix underlying the n-fold integer program (4.3) in the proof of Corollary 4.13 is precisely $\boxminus D$ with D the incidence matrix of G. So the time needed to solve the nonlinear many-commodity transshipment problem by the algorithm of Corollary 4.13 involves an $O(n^{g(G)})$ term whose exponent $g(G)$ is the Graver complexity of G. This explains the significance of this digraph invariant.

Likewise, it can be shown that the *many-commodity b-matching problem* over an undirected graph G, the precise definition of which is omitted, can be solved by an n-fold program similar to (4.3), whose solution time involves an $O(n^{g(G)})$ term with $g(G)$ the Graver complexity of the graph G, explaining its significance.

Finally, consider the universal n-fold integer program (5.5) discussed in Section 5.3 as follows:

$$\min\big\{f(x) : x \in \mathbb{Z}_+^{3mn},\ \big(\boxminus\,(1\ 1\ 1)^{[m]}\big)^{(n)}x = a\big\}.$$

The bimatrix defining this universal program is $\boxminus(1\ 1\ 1)^{[m]}$ with $(1\ 1\ 1)^{[m]}$ the incidence matrix of the complete bipartite graph $K_{3,m}$. Therefore, the time needed to solve this universal program for any fixed m involves an $O(n^{g(m)})$ term, where

$$g(m) := g\big(\boxminus\,(1\ 1\ 1)^{[m]}\big) = g\big(K_{3,m}\big).$$

So the Graver complexity $g(m)$ of the complete bipartite graph $K_{3,m}$ controls the solution time of the universal integer program, explaining its special importance. In particular, since integer programming feasibility is NP-complete, $g(m)$ must grow as a function of m under the hypothesis P $\neq$ NP. We show below that, in fact, it grows *exponentially fast*, as $g(m) = \Omega(2^m)$. This provides a new type of evidence of unusual flavor supporting the P $\neq$ NP hypothesis. An exponential upper bound $g(m) = O(m^4 6^m)$ holds as well and can be easily derived from Cramer's rule and Hadamard's bound. Narrowing the gap between these bounds remains a challenging and important problem. While $g(1) = g(K_{3,1}) = 0$, $g(2) = g(K_{3,2}) = 3$, and $g(3) = g(K_{3,3}) = 9$, the value of $g(m) = g(K_{3,m})$ is unknown for all $m \geq 4$.

We proceed to establish the claimed exponential lower bound on $g(m)$. Throughout, let $A := \boxminus(1\ 1\ 1)^{[m]}$ be the bimatrix with first block $A_1 = I_{3m}$ and second block $A_2 = (1\ 1\ 1)^{[m]}$ the incidence matrix of $K_{3,m}$. So $g(m) = g(K_{3,m}) = g(A)$. Now, A_2 is totally unimodular, so by Lemma 3.19, its Graver basis equals its set of circuits. Let G_2 denote the matrix having as columns the elements of $\mathcal{G}(A_2) = \mathcal{C}(A_2)$.

Our starting point is the following lemma.

Lemma 5.16. *Any circuit $h \in \mathcal{C}(G_2)$ of G_2 gives a lower bound $g(m) \geq \|h\|_1$.*

Proof. First, our data satisfies $A_1 G_2 = I_{3m} G_2 = G_2$. Second, $\mathcal{C}(G_2) \subseteq \mathcal{G}(G_2)$ holds. Therefore, computing the Graver complexity by Algorithm 4.6, we obtain the following:

$$g(m) = g(A) = \max\big\{\|v\|_1 : v \in \mathcal{G}(G_2)\big\} \geq \max\big\{\|h\|_1 : h \in \mathcal{C}(G_2)\big\}. \qquad \square$$

We use the following notation with m any integer clear from the context. Let

$$B := \{a, b, c\},\ U := \big\{u_1, \ldots, u_m\big\},\ V := B \uplus U,\ E := B \times U.$$

Then V and E are, respectively, the set of vertices and set of edges of the complete bipartite graph $K_{3,m}$, and index, respectively, the rows and columns of its incidence matrix $A_2 = (1\ 1\ 1)^{[m]}$. It is convenient to interpret each vector $x \in \mathbb{Z}^E$ also as (1) an integer-valued function on the set of edges $E = B \times U$; (2) a $3 \times m$ matrix with rows and columns

indexed, respectively, by B and U. With these interpretations, x is in $\mathcal{C}(A_2) = \mathcal{G}(A_2)$ if and only if (1) as a function on E, its support is a circuit of $K_{3,m}$, along which it alternates in values ± 1 and can be indicated by the sequence $(v_1, v_2, \ldots, v_l)$ of vertices of the circuit of $K_{3,m}$ on which it is supported, with the convention that its value is $+1$ on the first edge (v_1, v_2) in that sequence; (2) as a matrix, it is nonzero, has $0, \pm 1$ entries, has zero row and column sums, and has inclusion-minimal support with respect to these properties.

We also use the following terminology. Suppose that $x^1, \ldots, x^k \in \mathcal{C}(A_2) = \mathcal{G}(A_2)$ are circuits satisfying a relation $\sum_{i=1}^k h_i x^i = 0$ with all h_i positive. We call it a *primitive-relation* if $(h_1, \ldots, h_k)$ is the restriction of a circuit $h \in \mathcal{C}(G_2) \subseteq \mathcal{G}(G_2)$ to the columns of G_2 corresponding to the x^i, in which case $g(m) \geq \sum_{i=1}^k h_i$.

Here is an example that demonstrates this notation and also plays a role below.

Example 5.17 ($m = 4$: lower bound on the Graver complexity of $K_{3,4}$). Consider the following seven circuits of $(1\ 1\ 1)^{[4]}$ and $K_{3,4}$:

		u_1	u_2	u_3	u_4	
$x^1 := (a, u_4, c, u_2, b, u_3)$	=	0	0	-1	1	a
		0	-1	1	0	b
		0	1	0	-1	c
$x^2 := (a, u_2, c, u_3, b, u_1)$	=	-1	1	0	0	a
		1	0	-1	0	b
		0	-1	1	0	c
$x^3 := (a, u_4, b, u_1, c, u_2)$	=	0	-1	0	1	a
		1	0	0	-1	b
		-1	1	0	0	c
$x^4 := (a, u_4, b, u_2, c, u_1)$	=	-1	0	0	1	a
		0	1	0	-1	b
		1	-1	0	0	c
$x^5 := (a, u_1, b, u_2, c, u_3)$	=	1	0	-1	0	a
		-1	1	0	0	b
		0	-1	1	0	c
$x^6 := (a, u_3, b, u_4, c, u_2)$	=	0	-1	1	0	a
		0	0	-1	1	b
		0	1	0	-1	c
$x^7 := (a, u_2, b, u_3, c, u_4)$	=	0	1	0	-1	a
		0	-1	1	0	b
		0	0	-1	1	c

Then the following linear dependency on $x^1, \ldots, x^7$ is a primitive-relation:

$$\sum_{i=1}^{7} h_i x^i = x^1 + 2x^2 + 3x^3 + 3x^4 + 5x^5 + 6x^6 + 7x^7 = 0,$$

which demonstrates the bound $g(4) \geq \sum_{i=1}^{7} h_i = 1 + 2 + 3 + 3 + 5 + 6 + 7 = 27$.

We proceed with the general lower bound. We have the following lemma.

Lemma 5.18. *Suppose there are k circuits x^i of $(1\ 1\ 1)^{[m]}$ which admit a primitive-relation $\sum_i h_i x^i = 0$ with $x^k = (a, u_{m-2}, b, u_{m-1}, c, u_m)$ and h_k odd. Then there are also $k+2$ circuits $\bar{x}^i$ of $(1\ 1\ 1)^{[m+1]}$ which admit a primitive-relation $\sum_i \bar{h}_i \bar{x}^i = 0$ with $\bar{x}^{k+2} = (a, u_{m-1}, b, u_m, c, u_{m+1})$ and $\bar{h}_{k+2}$ odd, where the $\bar{h}_i$ are given by the following:*

$$\bar{h}_i = 2h_i,\ i = 1, \ldots, k-1, \quad \bar{h}_{k+2} = \bar{h}_{k+1} = \bar{h}_k = h_k. \tag{5.6}$$

Proof. Using the natural embedding of the complete bipartite graph $K_{3,m}$ in $K_{3,m+1}$, we can interpret circuits of the former also as circuits of the latter. Put $y^i := x^i$ for $i = 1, \ldots, k-1$ and define the following circuits:

		u_1	$\cdots$	u_{m-2}	u_{m-1}	u_m	u_{m+1}	
$y^k := (a, u_{m-2}, b, u_{m-1}, c, u_{m+1})$	=	0	$\cdots$	1	0	0	-1	a
		0	$\cdots$	-1	1	0	0	b
		0	$\cdots$	0	-1	0	1	c
$y^{k+1} := (a, u_{m-2}, b, u_{m+1}, c, u_m)$	=	0	$\cdots$	1	0	-1	0	a
		0	$\cdots$	-1	0	0	1	b
		0	$\cdots$	0	0	1	-1	c
$y^{k+2} := (a, u_{m+1}, b, u_{m-1}, c, u_m)$	=	0	$\cdots$	0	0	-1	1	a
		0	$\cdots$	0	1	0	-1	b
		0	$\cdots$	0	-1	1	0	c

Note that these circuits satisfy $y^k + y^{k+1} + y^{k+2} = 2x^k$. Suppose that $\sum_i \bar{h}_i y^i = 0$ is a nontrivial relation on the y^i. Without loss of generality, we may assume that the $\bar{h}_i$ are relatively prime integers, at least one of which is positive. Since the edges (a, u_{m+1}), (b, u_{m+1}), (c, u_{m+1}) of $K_{3,m+1}$ are not in $K_{3,m}$ and hence in no circuit y^i for $i < k$, the restrictions of the relation $\sum_i \bar{h}_i y^i = 0$ to these edges (or to the corresponding matrix entries) force the equalities $\bar{h}_{k+2} = \bar{h}_{k+1} = \bar{h}_k$. Therefore,

$$\begin{aligned} 0 = \sum_{i=1}^{k+2} \bar{h}_i y^i &= \sum_{i=1}^{k-1} \bar{h}_i y^i + \bar{h}_k \big(y^k + y^{k+1} + y^{k+2}\big) \\ &= \sum_{i=1}^{k-1} \bar{h}_i x^i + \bar{h}_k \big(2x^k\big) = \sum_{i=1}^{k-1} \bar{h}_i x^i + 2\bar{h}_k x^k, \end{aligned}$$

which is a nontrivial integer relation on the x^i. So there must exist an integer α so that, for all i, the coefficient of x^i in that relation is α times the coefficient of x^i in the relation $\sum_{i=1}^{k} h_i x^i = 0$:

$$\bar{h}_i = \alpha h_i,\ i = 1, \ldots, k-1, \quad 2\bar{h}_k = \alpha h_k.$$

Since all the h_i and at least one of the $\bar{h}_i$ are positive, these equations imply that α is positive. Therefore, all $\bar{h}_i$ are positive, implying that $\sum_i \bar{h}_i y^i = 0$ is a primitive-relation on the y^i. Since h_k is odd, the equation $2\bar{h}_k = \alpha h_k$ implies that α is even and therefore $\alpha = 2\mu$ for some positive integer μ, implying $\bar{h}_k = \mu h_k$. Then μ divides each of the $\bar{h}_i$, which are relatively prime, and therefore $\mu = 1$ and $\alpha = 2$. Therefore, the $\bar{h}_i$ satisfy (5.6), and in particular, $\bar{h}_{k+2} = h_k$ is odd.

Now, apply to the vertices of $K_{3,m+1}$ a permutation that maps u_{m+1}, u_{m-1}, u_m to u_{m-1}, u_m, u_{m+1} in that order and fixes the rest of the vertices. For each $i = 1, \ldots, k+2$, let $\bar{x}^i$ be the circuit of $K_{3,m+1}$ which is the image of y^i under this permutation. Then the $\bar{x}^i$ also satisfy the primitive-relation $\sum_i \bar{h}_i \bar{x}^i = 0$ with the same coefficients $\bar{h}_i$ and $\bar{x}^{k+2} = (a, u_{m-1}, b, u_m, c, u_{m+1})$, as claimed. □

We are now in position to prove the following exponential bound from [13].

Theorem 5.19. *For every $m \geq 4$, we have $g(m) = g(K_{3,m}) \geq 17 \cdot 2^{m-3} - 7$.*

Proof. We prove by induction on m that, for all $m \geq 4$, there are $2m-1$ circuits x^i of $(1\ 1\ 1)^{[m]}$ with $x^{2m-1} = (a, u_{m-2}, b, u_{m-1}, c, u_m)$, satisfying a primitive-relation $\sum_i h_i x^i = 0$ with $h_{2m-1} = 7$ and $\sum_i h_i = 17 \cdot 2^{m-3} - 7$, which implies the bound.

The induction basis at $m = 4$ is provided by the circuits in Example 5.17.

Suppose that the hypothesis holds for some $m \geq 4$ and let x^i be $2m-1$ circuits with corresponding coefficients h^i verifying the hypothesis. Lemma 5.18 applied to this data with $k = 2m-1$ then guarantees the existence of $k+2 = 2m+1 = 2(m+1)-1$ circuits $\bar{x}^i$ with corresponding coefficients $\bar{h}^i$ which satisfy the following:

$$\bar{x}^{2(m+1)-1} = \big(a, u_{m-1}, b, u_m, c, u_{m+1}\big), \quad \bar{h}_{2(m+1)-1} = \bar{h}_{2m} = \bar{h}_{2m-1} = h_{2m-1} = 7,$$

and, moreover,

$$\begin{aligned}\sum_{i=1}^{2m+1} \bar{h}_i &= \sum_{i=1}^{2m-2} 2h_i + 3h_{2m-1} = 2\sum_{i=1}^{2m-1} h_i + h_{2m-1} \\ &= 2\big(17 \cdot 2^{m-3} - 7\big) + 7 = 17 \cdot 2^{(m+1)-3} - 7.\end{aligned}$$

□

Example 5.20 ($m = 5$: lower bound on the Graver complexity of $K_{3,5}$). Using our construction, we obtain the following nine circuits of $(1\ 1\ 1)^{[5]}$ and $K_{3,5}$:

$$\begin{aligned}
x^1 &= \big(a, u_5, c, u_2, b, u_4\big), \quad x^2 = \big(a, u_2, c, u_4, b, u_1\big), \quad x^3 = \big(a, u_5, b, u_1, c, u_2\big), \\
x^4 &= \big(a, u_5, b, u_2, c, u_1\big), \quad x^5 = \big(a, u_1, b, u_2, c, u_4\big), \quad x^6 = \big(a, u_4, b, u_5, c, u_2\big), \\
x^7 &= \big(a, u_2, b, u_4, c, u_3\big), \quad x^8 = \big(a, u_2, b, u_3, c, u_5\big), \quad x^9 = \big(a, u_3, b, u_4, c, u_5\big),
\end{aligned}$$

which satisfy the primitive-relation:

$$2x^1 + 4x^2 + 6x^3 + 6x^4 + 10x^5 + 12x^6 + 7x^7 + 7x^8 + 7x^9 = 0,$$

providing the lower bound $g(K_{3,5}) = g(5) \geq 61$ on the Graver complexity of $K_{3,5}$.

Notes

Multiindex transportation problems and polytopes have been studied for decades, starting with the classical 1952 paper [75] by Motzkin. Theorem 5.1 shows that, remarkably, *every* rational polytope is an $l \times m \times 3$ line-sum transportation polytope. This enables to reduce the study of any algorithmic or structural property of rational convex polytopes to such triple-index polytopes, whose defining matrix is much more structured and depends on two parameters l and m only. A particularly important question, triggered by a letter of M. Hirsch to G. Dantzig in 1957, and open to date, is whether the diameter of the graph of every polytope is polynomially bounded in its dimension and number of facets; see [61] for an up-to-date survey. The universality theorem reduces this question to $l \times m \times 3$ line-sum polytopes. Motivated by their universality, some bounds on the diameter of such multiindex polytopes were recently established in [25]. Let us also mention a universality theorem for $\{0, 1\}$-polytopes in [15] by Billera and Sarangarajan, which asserts that every polytope all of whose vertices are $\{0, 1\}$-valued, is the face of some traveling salesman polytope. Privacy in databases is an area which is becoming increasingly important in the Internet era and is studied extensively by computer scientists, public data agencies, and statisticians. Let us mention the papers [8], [22], and [36], and the references therein, as informative sources corresponding to these three lines of investigation. Corollaries 5.8 and 5.11 provide the first polynomial time solutions of some of the most fundamental problems that occur in this area. It should be noted, though, that the running time of n-fold integer programming algorithms is typically a polynomial of very large degree. It is therefore of interest to study approximations of the Graver basis as in Sections 4.5 and 5.3. Initial experimentations for $3 \times 3 \times n$ multiway tables show very promising results, but much more study is needed. The Graver complexity of a graph and a digraph was introduced in [13]. The importance of these invariants stems from the fact that they form the degree of the polynomial time complexity of solving multiway table and multicommodity flow problems. Unfortunately, they seem to be very hard to compute, and even the complexity $g(K_{3,4})$ of the 7-vertex bipartite graph $K_{3,4}$ is unknown, coincidentally reminiscent of the hardness of the *Shannon capacity* invariant of a graph which is unknown even for the 7-vertex circuit C_7.

6 Nonlinear Combinatorial Optimization

In this chapter, we discuss the *nonlinear combinatorial optimization problem*, namely,

$$\min\{f(Wx) : x \in S\} \tag{6.1}$$

with $S \subseteq \{0,1\}^n$ presented compactly or by an oracle, integer $d \times n$ matrix W, and arbitrary function $f : \mathbb{Z}^d \to \mathbb{Z}$. We consider the minimization form, but our results hold for any function f and hence apply for the maximization form as well.

We assume as usual that the function f is presented by a mere comparison oracle. But unlike previous chapters, were we have assumed and exploited some structure on f, such as being convex, here we allow the function f to be arbitrary, making the problem much harder. In fact, any algorithm that solves problem (6.1) for arbitrary function f, must, whenever the image $WS = \{Wx : x \in S\}$ of S under W contains at least two points, include every image point in some query to the comparison oracle of f and hence must compute the entire image. Indeed, if some point $\hat{y} \in WS$ is not included in any query then the algorithm cannot tell between the possible situations where either $f(\hat{y})$ is strictly smaller or strictly larger than $f(y)$ for every $y \in WS \setminus \{\hat{y}\}$, and hence cannot tell if the optimal solution is in the fiber of $\hat{y}$ or not and cannot solve the problem correctly.

Note that this is in contrast with the special case of maximizing convex functions (or minimizing concave functions) considered in Chapter 2, where it suffices to compute the typically smaller subset $\mathrm{vert}(\mathrm{conv}(WS))$ of vertices of the image.

So the time taken by any algorithm for solving problem (6.1) is at least the cardinality $|WS|$ of the image. Therefore, when W is binary encoded, the problem is intractable and hopeless even in fixed dimension $d = 1$, for instance, the image in $\mathbb{Z}$ of $S := \{0,1\}^n$ under $w := (1, 2, 4, \ldots, 2^{n-1})$ has $|wS| = 2^n$ points, since

$$wS = \left\{ \textstyle\sum_{j=1}^{n} 2^{j-1} x_j : x \in \{0,1\}^n \right\} = \left\{0, 1, \ldots, 2^n - 1\right\}.$$

Therefore, in this chapter we assume that the weight matrix W is unary encoded. We follow the general outline of the Naïve Strategy 2.1 of Section 2.1, consisting of three steps: computing the image WS, finding a point $y \in WS$ minimizing $f(y)$ over WS, and finding a feasible point $x \in W^{-1}(y) \cap S$ in the fiber of y. The key problem is the first step of the strategy, that of computing the image WS.

We solve (6.1) in polynomial time for several types of combinatorial families S, using different implementations of Strategy 2.1 involving a variety of methods. For matroids, in Section 6.1.2, we compute the image exactly, leading to deterministic algorithm in Theorem 6.8. For matroid intersections, in Section 6.1.3, we compute a random subset of the image, leading to randomized algorithm in Theorem 6.12. Finally, for arbitrary independence systems, in Section 6.2.1, we compute an approximating subset of the image, leading to approximative algorithm in Theorem 6.23. This approximation has unusual flavor, and its quality is bounded by certain Frobenius numbers derived from the entries of W. We also establish an exponential lower bound in Theorem 6.24 on the time needed to solve the problem to optimality.

We conclude with some concrete applications including experimental design in statistics and universal Gröbner bases in computational algebra.

The table below enables quick navigation among theorems in this chapter providing polynomial time solution of problem (6.1). Additional results are in the applications Section 6.3. The first row indicates assumptions on the data (combinatorial structure of S, assumptions on d, assumptions on W), and the second row indicates the type of polynomial time algorithm established herein. Our results for matroids and matroid intersections apply to bases and independent sets. Restricting attention to independent sets, all systems considered in the table are independence systems, of generality and difficulty increasing from left to right.

Single matroid d arbitrary W unary encoded	Two matroid intersections d arbitrary W unary encoded	Any independence system $d = 1$ W has entries in $\{a_1, \ldots, a_p\}$
Theorem 6.8 deterministic	Theorem 6.12 randomized	Theorem 6.23 $r(a_1, \ldots, a_p)$-approximative

6.1 Nonlinear matroid optimization

In this section, we solve the nonlinear combinatorial optimization problem (6.1) over matroids and two matroid intersections for arbitrary function f and unary-encoded W. The following table summarizes the time complexities of all our algorithmic results for matroids and matroid intersections, including the results of this section and the results of Section 2.5.2 on the maximization of convex functions.

	Single matroid	Two matroid intersections
Maximizing convex f	Corollary 2.25 polynomial in $\langle W \rangle$	Corollary 2.26 polynomial in W
Optimizing any f	Theorem 6.8 polynomial in W	Theorem 6.12 randomized polynomial in W

6.1.1 Preparation

As noted, our solution of problem (6.1) follows the outline of Strategy 2.1, consisting of three steps: computing the image WS, finding a point $y \in WS$ minimizing $f(y)$ over WS, and finding a feasible point $x \in W^{-1}(y) \cap S$ in the fiber of y.

We begin with some preparatory lemmas which are used in the sequel. As usual $\|W\|_\infty = \max_{i,j} |W_{i,j}|$. Also, for any $0 \leq r \leq n$, let $\{0,1\}^n_r$ be the set of all vectors $x \in \{0,1\}^n$ with precisely r entries equal to 1, that is, with $|\operatorname{supp}(x)| = r$.

The first lemma enables to restrict attention to nonnegative weight matrices.

Lemma 6.1. *The image of $S \subseteq \{0,1\}^n_r$ under an integer $d \times n$ matrix W is obtainable as the translation $WS = \bar{W}S - v$ of the image of S under the nonnegative integer matrix $\bar{W}$ defined by $\bar{W}_{i,j} := W_{i,j} + \|W\|_\infty$ for all i, j with $v := r\|W\|_\infty \mathbf{1} \in \mathbb{Z}^d$.*

Proof. The proof follows at once since $\bar{W}x = Wx + v$ for every $x \in \{0,1\}^n_r$. □

The *restriction* of $S \subseteq \{0,1\}^n$ to a subset $J \subseteq N := \{1, \ldots, n\}$ of the ground set is the set $S^J := \{x \in S : \mathrm{supp}(x) \subseteq J\} \subseteq \{0,1\}^n$ of all points in S supported on J. Recall the *fiber problem* from Section 2.1: given $S \subseteq \mathbb{Z}^n$, integer $d \times n$ matrix W, and point $y \in \mathbb{Z}^d$, find $x \in W^{-1}(y) \cap S$, that is, $x \in S$ with $Wx = y$, or assert that none exists. The next lemma shows that the fiber problem for $S \subseteq \{0,1\}^n$ can be solved by deciding if $y \in WS^J = \{Wx : x \in S^J\}$ for $n+1$ restrictions of S.

Lemma 6.2. *The fiber problem of finding $x \in W^{-1}(y) \cap S$ or asserting that none exists can be solved by deciding if $y \in WS^J$ for $n+1$ restrictions S^J of $S \subseteq \{0,1\}^n$.*

Proof. If $y \notin WS^N$ then assert that there is no feasible point in the fiber of y. Otherwise, set $J_0 := N$ and for $j = 1, 2, \ldots, n$ repeat the following iteration: set $J := J_{j-1} \setminus \{j\}$; if $y \in WS^J$ then set $J_j := J$, else set $J_j := J_{j-1}$. Output the indicator $x := \mathbf{1}_{J_n}$ of the final set J_n obtained that way. Then $x \in W^{-1}(y) \cap S$. □

We conclude our preparation with some properties of multivariate polynomials. Let $x = (x_1, \ldots, x_d)$ be a vector of d variables. Each $u \in \mathbb{Z}^d_+$ serves as an *exponent* of a corresponding *monomial* $x^u := \prod_{i=1}^d x_i^{u_i}$ in x. A (*multivariate*) *polynomial* in x is a linear combination $g = g(x) := \sum g_u x^u$ of finitely many monomials with coefficients g_u in a suitable field. A polynomial is *integer* (*rational*, *real*, *complex*) if all its coefficients are integer (rational, real, complex). We mostly restrict attention to integer or rational polynomials, whose coefficients can be processed by our algorithms. The *support* of g is the finite set $\mathrm{supp}(g) := \{u \in \mathbb{Z}^d_+ : g_u \neq 0\}$ of exponents of all monomials x^u having nonzero coefficients g_u. We denote by $\|g\|_\infty := \|(g_u : u \in \mathbb{Z}^d_+)\|_\infty$ the maximum absolute value of any coefficient of g. The *degree* of g is $\deg(g) := \max\{\|u\|_1 : u \in \mathrm{supp}(g)\}$, that is, the maximum value $\|u\|_1 = \sum_{i=1}^d u_i$ of an exponent of a monomial x^u having a nonzero coefficient g_u. As usual, an *evaluation oracle* for g is one that, queried on $x \in \mathbb{Z}^d$, returns $g(x)$.

We need two lemmas about polynomials. The first one concerns interpolation.

Lemma 6.3. *For any fixed d, there is an algorithm that, given $s \in \mathbb{Z}_+$ and d-variate integer polynomial g presented by evaluation oracle, with $\mathrm{supp}(g) \subseteq \{0, 1, \ldots, s\}^d$, computes $\mathrm{supp}(g)$ and all g_u, $u \in \mathrm{supp}(g)$, in time polynomial in s and $\langle\|g\|_\infty\rangle$.*

Proof. Let $U := \{0, 1, \ldots, s\}^d$ so that $g(x) = \sum_{u \in U} g_u x^u$. For each of the polynomially many integers $t = 1, 2, \ldots, (s+1)^d$, let $x(t)$ be the vector in $\mathbb{Z}^d$ obtained by substituting $x_i := t^{(s+1)^{i-1}}$ for $i = 1, \ldots, d$, and use the evaluation oracle of g to compute $g(x(t))$. We then obtain a system of $(s+1)^d$ linear equations in $(s+1)^d$ indeterminates g_u, $u \in U$,

where, for $t = 1, 2, \ldots, (s+1)^d$, the tth equation is as follows:

$$\sum_{u \in U} g_u \big(x(t)\big)^u = \sum_{u \in U} g_u \prod_{i=1}^{d} \big(t^{(s+1)^{i-1}}\big)^{u_i} = \sum_{u \in U} t^{(\sum_{i=1}^{d} u_i (s+1)^{i-1})} g_u = g\big(x(t)\big).$$

As $u = (u_1, \ldots, u_d)$ runs through U, the sum $\sum_{i=1}^{d} u_i (s+1)^{i-1}$ attains precisely all $|U| = (s+1)^d$ distinct values $0, 1, \ldots, (s+1)^d - 1$. This implies that, under the total order of the points u in U by increasing value of $\sum_{i=1}^{d} u_i (s+1)^{i-1}$, the vector of coefficients of the g_u in the tth equation is precisely the point $(t^0, t^1, \ldots, t^{(s+1)^d - 1})$ on the so-called *moment curve* in $\mathbb{R}^U \cong \mathbb{R}^{(s+1)^d}$. Therefore, the equations are linearly independent. Moreover, each coefficient and each right-hand side of each equation in this system have a polynomially bounded binary length, since

$$\big(x(t)\big)^u \leq \big((s+1)^d\big)^{(s+1)^d - 1}, \quad \big|g\big(x(t)\big)\big| \leq (s+1)^d \big((s+1)^d\big)^{(s+1)^d - 1} \|g\|_\infty.$$

Therefore, the system of equations can be solved in polynomial time, and the coefficients g_u and the support $\operatorname{supp}(g) = \{u \in U : g_u \neq 0\}$ be obtained as claimed. □

The second lemma, from [93], bounds the number of zeros of a polynomial.

Lemma 6.4. *Let g be a nonzero n-variate complex polynomial with $\deg(g) \leq r$ and q a positive integer. Then $g(x) = 0$ for at most $\frac{r}{q}$ of the vectors $x \in \{1, 2, \ldots, q\}^n$.*

Proof. We show by induction on n that $g(x) = 0$ for at most $\frac{r}{q} q^n = r q^{n-1}$ such vectors. For $n = 1$, this holds since any nonzero univariate polynomial of degree at most r has at most r complex roots. Now, suppose that $n \geq 2$ and g is nonzero. Then for some $0 \leq k \leq r$ and some $(n-1)$-variate polynomials $g_0, \ldots, g_k$ with $g_k \neq 0$:

$$g\big(x_1, \ldots, x_n\big) = \sum_{i=0}^{k} g_i\big(x_1, \ldots, x_{n-1}\big) x_n^i.$$

For each vector $(v_1, \ldots, v_{n-1}) \in \{1, \ldots, q\}^{n-1}$ which is not a zero of g_k, we have that $g(v_1, \ldots, v_{n-1}, x_n)$ is a nonzero univariate polynomial in x_n of degree at most k and hence is zero for at most k values of x_n. So g has at most $k q^{n-1}$ such zeros. For each vector $(v_1, \ldots, v_{n-1}) \in \{1, \ldots, q\}^{n-1}$ which is a zero of g_k it may be that $g(v_1, \ldots, v_{n-1}, x_n)$ is zero for all q values of x_n in $\{1, \ldots, q\}$. Since $\deg(g_k) \leq r - k$, by induction g_k has at most $(r-k) q^{n-2}$ zeros $(v_1, \ldots, v_{n-1}) \in \{1, \ldots, q\}^{n-1}$. So g has at most $q(r-k) q^{n-2}$ such zeros. Therefore, the total number of zeros of g at most $k q^{n-1} + q(r-k) q^{n-2} = r q^{n-1}$ and the induction step follows. □

6.1.2 Matroids

The main result of this subsection is Theorem 6.8: for every fixed d, we can minimize any nonlinear composite function $f(Wx)$ with W presented in unary, over the bases or independent sets of a vectorial matroid, in polynomial time.

Let M be the vectorial matroid of an integer $r \times n$ matrix A of rank r and let $S := \{\mathbf{1}_B : B \in \mathcal{B}\}$ be its set of bases. Let W be a nonnegative integer $d \times n$ weight matrix. For each $x \in \{0,1\}^n_r$ let $A^x := [A^j : j \in \text{supp}(x)]$ be the $r \times r$ submatrix of A consisting of the columns corresponding to nonzero entries of x.

For each $y \in \mathbb{Z}^d_+$ define the following:

$$g_y := \sum \left\{ \det^2\left(A^x\right) : x \in W^{-1}(y) \cap S \right\}.$$

Let $b = (b_1, \ldots, b_d)$ be a vector of variables. Define the following polynomial in b:

$$g = g(b) := \sum_y g_y b^y = \sum_y g_y \prod_{i=1}^{d} b_i^{y_i}. \tag{6.2}$$

The following lemma bounds the support and norm of g and shows that the image WS of bases coincides with the support of g. As usual, $\Delta(A)$ denotes the maximum absolute value of a determinant of a square submatrix of A.

Lemma 6.5. *For every integer $r \times n$ matrix A of rank r and nonnegative integer $d \times n$ matrix W, with S the bases of the matroid of A, the polynomial g satisfies the following:*

$$\text{supp}(g) = WS \subseteq \left\{0, 1, \ldots, r\|W\|_\infty\right\}^d, \quad \|g\|_\infty \leq \binom{n}{r} \Delta^2(A).$$

Proof. We have $\det(A^x) \neq 0$ if and only if $x \in S$. Therefore, $g_y \neq 0$ if and only if $W^{-1}(y) \cap S \neq \emptyset$, that is, if and only if $y \in WS$. The first claim then follows by the following:

$$\text{supp}(g) = \left\{y : g_y \neq 0\right\} = WS \subseteq W\{0,1\}^n_r \subseteq \left\{0, 1, \ldots, r\|W\|_\infty\right\}^d.$$

The second claim, bounding $\|g\|_\infty$, follows by noting that for all $y \in \mathbb{Z}^d_+$ we have the following:

$$\left|g_y\right| \leq |S|\Delta^2(A) \leq \left|\{0,1\}^n_r\right|\Delta^2(A) = \binom{n}{r}\Delta^2(A).$$

□

Let $A(b)$ be the $r \times n$ matrix obtained from A by multiplying, for $j = 1, \ldots, n$, column A^j by monomial $b^{W^j} = \prod_{i=1}^d b_i^{W_{i,j}}$ having as exponent column W^j of W:

$$A^j(b) := b^{W^j} A^j = \prod_{i=1}^{d} b_i^{W_{i,j}} A^j, \quad j = 1, \ldots, n.$$

The next lemma provides an efficient evaluation oracle for the polynomial g.

Lemma 6.6. *For every integer $r \times n$ matrix A of rank r and nonnegative integer $d \times n$ matrix W, the following identity of polynomials in variables b, which enables to evaluate $g(b)$ for any given $b \in \mathbb{Z}^d$ in time polynomial in W and $\langle A, b\rangle$, holds:*

$$g(b) = \det\left(A(b)A^T\right).$$

Proof. The classical Binet–Cauchy identity for two $r \times n$ matrices B and C is as follows:

$$\det\left(BC^T\right) = \sum_{x \in \{0,1\}_r^n} \det\left(B^x\right) \det(C^x). \tag{6.3}$$

Applying the Binet–Cauchy identity to $B := A(b)$ and $C := A$, we obtain the claimed identity, with $S \subseteq \{0,1\}_r^n$ the set of bases of the vectorial matroid of A:

$$\begin{aligned}\det\left(A(b)A^T\right) &= \sum_{x \in \{0,1\}_r^n} \det\left(A^x(b)\right) \det(A^x) = \sum_{x \in \{0,1\}_r^n} \prod_{j=1}^{n} \left(b^{W^j}\right)^{x_j} \det^2\left(A^x\right) \\ &= \sum_{x \in S} b^{(\sum_{j=1}^n x_j W^j)} \det^2\left(A^x\right) = \sum_{x \in S} b^{Wx} \det^2\left(A^x\right) \\ &= \sum_y b^y \sum_{x \in W^{-1}(y) \cap S} \det^2\left(A^x\right) = \sum_y g_y b^y = g(b).\end{aligned}$$

Now, given any $b \in \mathbb{Z}^d$, we can clearly evaluate $g(b) = \det(A(b)A^T)$ in time which polynomial in W, $\langle A, b\rangle$, by computing the determinant of an integer matrix. □

We can now compute the image of the bases or independent sets of a matroid.

Lemma 6.7. *For every fixed d, there is an algorithm that, given vectorial matroid M which is presented by integer $r \times n$ matrix A, and integer $d \times n$ matrix W, in time which is polynomial in $\langle A\rangle$ and W, computes the image WS of the set $S := \{\mathbf{1}_B : B \in \mathcal{B}\}$ of bases or set $S := \{\mathbf{1}_I : I \in \mathcal{I}\}$ of independent sets of M.*

Proof. We begin with the statement for bases. Removing some rows of A if necessary, we may assume that A has rank r and therefore $S = \{\mathbf{1}_B : B \in \mathcal{B}\} \subseteq \{0,1\}_r^n$. By Lemma 6.1, we may then assume that W is nonnegative. Define the polynomial g as in (6.2). By Lemma 6.5, we have $\mathrm{supp}(g) \subseteq \{0, 1 \ldots, s\}^d$ with $s := r\|W\|_\infty$ and $\|g\|_\infty \leq \binom{n}{r}\Delta^2(A)$. By Lemma 6.6, we can realize in polynomial time an evaluation oracle for the polynomial g. So we can apply the algorithm of Lemma 6.3 and compute $\mathrm{supp}(g)$, which by Lemma 6.5 provides the image as $WS := \mathrm{supp}(g)$, in time polynomial in s and $\langle\|g\|_\infty\rangle$ and hence polynomial in W and $\langle A\rangle$.

We continue with the statement for $S = \{\mathbf{1}_I : I \in \mathcal{I}\}$ the independent sets. Let $\bar{W} := [W\ 0]$ be the $d \times (n+r)$ matrix obtained by appending to W the $d \times r$ zero matrix and let $\bar{A} := [A\ I_r]$ be the $r \times (n+r)$ matrix obtained by appending to A the $r \times r$ identity matrix. Let $\bar{S} := \{\mathbf{1}_B : B \in \bar{\mathcal{B}}\}$ be the set of bases of the vectorial matroid $\bar{M}$ of $\bar{A}$. Then it is not hard to verify that $WS = \bar{W}\bar{S}$, which can be computed in polynomial time by the above algorithm for bases of $\bar{M}$. □

We can now obtain a result of [10] about nonlinear matroid optimization.

Theorem 6.8. *For every fixed d, there is an algorithm that, given vectorial matroid M of integer $r \times n$ matrix A, integer $d \times n$ matrix W, and function $f : \mathbb{Z}^d \to \mathbb{R}$ presented by comparison oracle, solves in time polynomial in $\langle A\rangle, W$, the problem:*

$$\min\left\{f(Wx) : x \in S\right\} \tag{6.4}$$

over the bases $S := \{\mathbf{1}_B : B \in \mathcal{B}\}$ or independent sets $S := \{\mathbf{1}_I : I \in \mathcal{I}\}$ of M.

Proof. Use the algorithm of Lemma 6.7 and compute the image WS. Then use the comparison oracle of f and obtain a point $y \in WS$ minimizing f over WS.

We proceed to show how to implement the algorithm of Lemma 6.2 to obtain a point $x \in W^{-1}(y) \cap S$. For this, we need to show how to decide if $y \in WS^J$ for the restriction $S^J = \{x \in S : \text{supp}(x) \subseteq J\}$ of S to any subset $J \subseteq \{1, \ldots, n\}$.

Consider any such restriction. Let $\bar{A}$ be the $r \times n$ matrix obtained from A by replacing each column A^j, $j \notin J$, by the zero vector $0 \in \mathbb{Z}^r$. If S is the set of bases and $\text{rank}(\bar{A}) < \text{rank}(A)$ then $S^J = \emptyset$ so $y \notin WS^J$. Otherwise, namely, if S is the set of bases and $\text{rank}(\bar{A}) = \text{rank}(A)$, or S is the set of independent sets, then S^J is precisely the set of bases or independent sets, respectively, of the matroid $\bar{M}$ of $\bar{A}$. Now, compute WS^J by the algorithm of Lemma 6.7 for $\bar{M}$ and check if $y \in WS^J$.

Using this decision procedure for restrictions, we can indeed use the algorithm of Lemma 6.2 and obtain $x \in W^{-1}(y) \cap S$ which is an optimal solution. □

6.1.3 Matroid intersections

We proceed to solve the nonlinear optimization problem over two matroid intersections. The general outline of the solution method is similar to the one over matroids, but the problem is harder and we need to incorporate randomization.

The main result here is Theorem 6.12: for every fixed d, we can minimize any nonlinear composite function $f(Wx)$ with W in unary, over the common bases or independent sets of two vectorial matroids, in randomized polynomial time.

Let M_1, M_2 be vectorial matroids of integer $r \times n$ matrices A_1, A_2 of rank r and let $S := \{\mathbf{1}_B : B \in \mathcal{B}_1 \cap \mathcal{B}_2\}$ be the set of common bases. Let W be a nonnegative integer $d \times n$ weight matrix. As before, for each $x \in \{0,1\}^n_r$ let $A^x_k := [A^j_k : j \in \text{supp}(x)]$ be the corresponding $r \times r$ submatrix of A_k for $k = 1, 2$.

Let $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_d)$ be vectors of n variables and d variables, respectively. For each $y \in \mathbb{Z}^d_+$, define the following polynomial in a:

$$g_y(a) := \sum \left\{ \det\left(A^x_1\right) \det\left(A^x_2\right) a^x : x \in W^{-1}(y) \cap S \right\}, \quad a^x = \prod_{j=1}^{n} a_j^{x_j}. \tag{6.5}$$

Now, define the following polynomial in both vectors of variables a and b:

$$g = g(a,b) := \sum_y g_y(a) b^y, \quad b^y = \prod_{i=1}^{d} b_i^{y_i}. \tag{6.6}$$

We now consider numerical substitutions of vectors $a \in \mathbb{Z}^n$ into $g = g(a,b)$, turning g into a polynomial in b only. In what follows, we often make random substitutions for a, in which case g becomes a random polynomial in b with random support $\text{supp}(g) = \{y \in \mathbb{Z}^d_+ : g_y(a) \neq 0\}$. The following lemma bounds the norm and support of g under such substitutions and shows that any point in the image WS of common bases appears in the random support of g with high probability.

Lemma 6.9. *Let A_1, A_2 be integer $r \times n$ matrices of rank r, S the set of common bases of their vectorial matroids, and W nonnegative integer $d \times n$ matrix. Let g be the polynomial in b obtained by substituting $a \in \{1, 2, \ldots, q\}^n$ into $g(a, b)$. Then*

$$\operatorname{supp}(g) \subseteq WS \subseteq \big\{0, 1, \ldots, r\|W\|_\infty\big\}^d, \quad \|g\|_\infty \leq \binom{n}{r} q^r \Delta\big(A_1\big)\Delta\big(A_2\big).$$

If a is randomly uniformly sampled from $\{1, 2, \ldots, q\}^n$ then, for every $y \in WS$,

$$\mathbb{P}\big[y \in \operatorname{supp}(g)\big] \geq 1 - \frac{r}{q}.$$

Proof. We begin with the claims on the support and norm. For the support, note that if $y \notin WS$ then $W^{-1}(y) \cap S = \emptyset$ and hence $g_y(a) = 0$ for any a. Therefore,

$$\operatorname{supp}(g) = \big\{y : g_y(a) \neq 0\big\} \subseteq WS \subseteq W\{0, 1\}^n_r \subseteq \big\{0, 1, \ldots, r\|W\|_\infty\big\}^d.$$

For the bound on the norm $\|g\|_\infty$, note that for all $y \in \mathbb{Z}^d_+$, we have the following:

$$\big|g_y(a)\big| \leq |S|\Delta\big(A_1\big)\Delta\big(A_2\big)q^r \leq \big|\{0, 1\}^n_r\big|\Delta\big(A_1\big)\Delta\big(A_2\big)q^r = \binom{n}{r} q^r \Delta\big(A_1\big)\Delta\big(A_2\big).$$

For the claim on random substitutions, consider any point $y \in WS$. Consider

$$g_y(a) = \sum \big\{ \det\big(A^x_1\big) \det\big(A^x_2\big) a^x : x \in W^{-1}(y) \cap S \big\}$$

as a polynomial in a. Since distinct $x \in \{0, 1\}^n$ give distinct monomial a^x, no cancelations occur among terms in the sum defining $g_y(a)$. Since y is in the image, there is some $x \in W^{-1}(y) \cap S$ and hence the monomial a^x appears with coefficient $\det(A^x_1) \det(A^x_2)$ which is nonzero since x is a common basis. Therefore, $g_u(a)$ is a nonzero polynomial in a of degree r. The claim now follows from Lemma 6.4 which bounds the probability of $g_y(a) = 0$ which is equivalent to $y \notin \operatorname{supp}(g)$. □

Let $A_1(a, b)$ be the $r \times n$ matrix obtained from the matrix A_1 by multiplying, for $j = 1, \ldots, n$, the column A^j_1 by the monomial $a_j b^{W^j} = a_j \prod_{i=1}^d b_i^{W_{i,j}}$, that is,

$$A^j_1(a, b) := a_j b^{W^j} A^j_1 = a_j \prod_{i=1}^d b_i^{W_{i,j}} A^j_1, \quad j = 1, \ldots, n.$$

The next lemma provides an efficient evaluation oracle for the polynomial g.

Lemma 6.10. *For every integer $r \times n$ matrices A_1, A_2 of rank r and nonnegative integer $d \times n$ matrix W, the following identity, which enables to evaluate $g(a, b)$ for any given $a \in \mathbb{Z}^n$ and $b \in \mathbb{Z}^d$ in time polynomial in W and $\langle A_1, A_2, a, b\rangle$, holds:*

$$g(a, b) = \det\big(A_1(a, b)A^T_2\big).$$

Proof. The Binet–Cauchy identity (6.3) with $B := A_1(a,b)$ and $C := A_2$ gives the claimed identity, with $S \subseteq \{0,1\}^n_r$ the common bases of the matroids of A_1, A_2:

$$\begin{aligned}\det\left(A_1(a,b)A_2^T\right) &= \sum_{x\in\{0,1\}^n_r} \det\left(A_1^x(a,b)\right)\det\left(A_2^x\right)\\ &= \sum_{x\in\{0,1\}^n_r}\prod_{j=1}^{n}\left(a_j b^{W^j}\right)^{x_j}\det\left(A_1^x\right)\det\left(A_2^x\right)\\ &= \sum_{x\in S} a^x b^{Wx}\det\left(A_1^x\right)\det\left(A_2^x\right)\\ &= \sum_y b^y \sum_{x\in W^{-1}(y)\cap S} a^x \det\left(A_1^x\right)\det\left(A_2^x\right)\\ &= \sum_y g_y(a)b^y = g(a,b).\end{aligned}$$

Given $a \in \mathbb{Z}^n$, $b \in \mathbb{Z}^d$, we can evaluate $g(a,b) = \det(A_1(a,b)A_2^T)$ in time polynomial in W, $\langle A_1, A_2, a, b\rangle$, by computing the determinant of an integer matrix. □

We can now show how to compute a random subset of the set of common bases or common independent sets of two matroids, which contains every point of the image with high probability. A *randomized algorithm* is an algorithm that has access to a random bit generator. Its running time then counts also the number of random bits used. A randomized algorithm is *polynomial time* if its running time, including the number of random bits used, is polynomial in the input length.

Lemma 6.11. *For every fixed d, there is a randomized algorithm that, given two vectorial matroids, with M_k presented by integer $r_k \times n$ matrix A_k, and integer $d \times n$ matrix W, in time polynomial in $\langle A_1, A_2\rangle$, W, computes a random subset $Y \subseteq WS$ of the image of $S := \{\mathbf{1}_B : B \in \mathcal{B}_1 \cap \mathcal{B}_2\}$ common bases or $S := \{\mathbf{1}_I : I \in \mathcal{I}_1 \cap \mathcal{I}_2\}$ common independent sets of M_1 and M_2, such that for every point $y \in WS$:*

$$\mathbb{P}[y \in Y] \geq 1 - \frac{1}{2(n+1)}.$$

Proof. We begin with the statement for common bases. Removing some rows of A_k if necessary, we may assume that the rank of A_k is r_k for $k = 1, 2$. If $r_1 \neq r_2$ then there is no common basis so the image is empty. So assume $r := r_1 = r_2$ and therefore $S = \{\mathbf{1}_B : B \in \mathcal{B}_1 \cap B_2\} \subseteq \{0,1\}^n_r$. By Lemma 6.1, we may then assume that W is nonnegative. Define the polynomial g as in (6.6). Let $q := 2r(n+1)$. Consider any $a \in \{1, 2, \ldots, q\}^n$ and let $g = g(a,b)$ be the polynomial in b obtained by substituting a. By Lemma 6.9, we have $\operatorname{supp}(g) \subseteq \{0, 1, \ldots, s\}^d$ with $s := r\|W\|_\infty$ and $\|g\|_\infty \leq \binom{n}{r} q^r \Delta(A_1)\Delta(A_2)$. By Lemma 6.10, we can realize in polynomial time an evaluation oracle for the polynomial g. So we can apply the algorithm of Lemma 6.3 and compute $\operatorname{supp}(g)$ in time polynomial in s and $\langle\|g\|_\infty\rangle$ and hence polynomial in W and $\langle A_1, A_2\rangle$. Now, let a be randomly uniformly sampled from $\{1, 2, \ldots, q\}^n$, let $g = g(a,b)$ be the random polynomial in b

obtained by substituting a, and let $Y := \mathrm{supp}(g) = \{y \in \mathbb{Z}^d_+ : g_y(a) \neq 0\}$ be the random support of g. We claim that this Y satisfies the claim of the lemma. Indeed, as just shown, Y can be computed in polynomial time for any realization of the random vector a, and by Lemma 6.9, for every point $y \in WS$ of the image:

$$\mathbb{P}[y \in Y] \geq 1 - \frac{r}{q} = 1 - \frac{1}{2(n+1)}.$$

We continue with the statement for $S = \{\mathbf{1}_I : I \in \mathcal{I}_1 \cap \mathcal{I}_2\}$ the common independent sets. Let $r := \max\{r_1, r_2\}$. Let $\bar{W} := [W \ \ 0]$ be the $d \times (n + r^2)$ matrix obtained by appending to W the $d \times r^2$ zero matrix. For $k = 1, 2$, let $\hat{A}_k$ be the $r \times n$ matrix obtained by appending $r - r_k$ zero rows to A_k. Let C_1, C_2 be the $r \times r^2$ matrices consisting of the first r rows and last r rows, respectively, of the $(r + r) \times r^2$ incidence matrix of the complete bipartite graph $K_{r,r}$. For $k = 1, 2$, let $\bar{A}_k := [\hat{A}_k \ \ C_k]$ be the $r \times (n+r^2)$ matrix obtained by appending C_k to $\hat{A}_k$, and let $\bar{M}_k$ be the vectorial matroid of $\bar{A}_k$. Finally, let $\bar{S} := \{\mathbf{1}_B : B \in \bar{\mathcal{B}}_1 \cap \bar{\mathcal{B}}_2\}$ be the set of common bases of $\bar{M}_1$, $\bar{M}_2$. Then it is not hard to verify that $WS = \bar{W}\bar{S}$, and the claim follows from the above statement for common bases of $\bar{M}_1$, $\bar{M}_2$. □

We can now obtain a result of [11], providing a randomized polynomial time algorithm for solving the nonlinear matroid intersection problem. By this, we mean that the algorithm either asserts that the problem is infeasible or outputs a point $x \in \{0,1\}^n$ which is an optimal solution with probability at least a half. By repeatedly applying the algorithm several times and picking the best solution if any, the probability of success can be increased at will. In particular, repeating it n times increases the probability of obtaining a true optimal solution to at least $1 - 2^{-n}$ while increasing the running time by a linear factor only.

Theorem 6.12. *For every fixed d, there is a randomized algorithm that, given two vectorial matroids, with M_k presented by integer $r_k \times n$ matrix A_k, integer $d \times n$ matrix W, and function $f : \mathbb{Z}^d \to \mathbb{R}$ presented by comparison oracle, solves over the set of common bases $S := \{\mathbf{1}_B : B \in \mathcal{B}_1 \cap \mathcal{B}_2\}$ or common independent sets $S := \{\mathbf{1}_I : I \in \mathcal{I}_1 \cap \mathcal{I}_2\}$ of M_1, M_2, in time polynomial in $\langle A_1, A_2\rangle$, W, the following problem:*

$$\min\big\{f(Wx) : x \in S\big\}.$$

Proof. First, either detect that S is empty or obtain some point $\bar{x} \in S$ as follows: for common independent sets take $\bar{x} := 0 \in \mathbb{Z}^n$ which is always feasible; for common bases apply the deterministic matroid intersection algorithm of [33]. So from here on, we assume that S is nonempty and some $\bar{x} \in S$ has been obtained.

We begin by describing the polynomial time randomized algorithm, and then prove that it outputs an optimal solution with probability at least a half.

Use the algorithm of Lemma 6.11 and compute a random subset $Y \subseteq WS$ of the image. Set $\bar{y} := W\bar{x}$ and $Y := Y \cup \{\bar{y}\} \subseteq WS$ making sure Y is nonempty. Use the comparison oracle of f to obtain a point $\hat{y} \in Y$ minimizing f over Y.

We proceed to show how to implement a randomized counterpart of the algorithm of Lemma 6.2 which outputs a random candidate $\hat{x}$ for being in $W^{-1}(\hat{y}) \cap S$. For this, we

need to describe a randomized procedure for deciding if $\hat{y} \in WS^J$ for the restriction $S^J = \{x \in S : \mathrm{supp}(x) \subseteq J\}$ of S to any subset $J \subseteq \{1, \ldots, n\}$.

Consider any such restriction. For $k = 1, 2$, let $\bar{A}_k$ be the $r \times n$ matrix obtained from A_k by replacing each column A^j, $j \notin J$, by $0 \in \mathbb{Z}^{r_k}$. If S is the set of common bases and $\mathrm{rank}(\bar{A}_1) < \mathrm{rank}(A_1)$ or $\mathrm{rank}(\bar{A}_2) < \mathrm{rank}(A_2)$ then $S^J = \emptyset$ so $\hat{y} \notin WS^J$. Otherwise, namely, if S is the set of common bases and $\mathrm{rank}(\bar{A}_k) = \mathrm{rank}(A_k)$ for $k = 1, 2$, or S is the set of common independent sets, then S^J is precisely the set of common bases or common independent sets, respectively, of the vectorial matroids $\bar{M}_1$, $\bar{M}_2$ of $\bar{A}_1$, $\bar{A}_2$. Now, compute a random subset $Y(J) \subseteq WS^J$ by the algorithm of Lemma 6.11 for $\bar{M}_1$, $\bar{M}_2$, check if $\hat{y} \in Y(J)$ and decide accordingly.

Incorporating this randomized decision procedure for restrictions into the algorithm of Lemma 6.2, we obtain a randomized algorithm which outputs the random indicator $\hat{x} := \mathbf{1}_{J_n}$ of the set J_n obtained in the final iteration of that algorithm.

We now show that $\hat{x}$ is an optimal solution with probability at least a half. Consider the following random events:

$\mathbf{E}_0$**:** the minimizer $\hat{y}$ of f over Y further satisfies $f(\hat{y}) = \min\{f(y) : y \in WS\}$.

$\mathbf{E}_j$**:** the algorithm of Lemma 6.2 decides correctly in iteration j whether $\hat{y} \in WS^J$.

If all events $\mathbf{E}_0, \mathbf{E}_1, \ldots, \mathbf{E}_n$ hold then the algorithm clearly outputs a point x which is an optimal solution. We proceed to bound the probabilities of these events.

First, let $y^* \in WS$ be any minimizer of f over WS. If $y^* \in Y$ then $f(\hat{y}) = f(y^*)$ by choice of $\hat{y}$ and hence $\hat{y}$ is also a minimizer of f over WS. So, by Lemma 6.11,

$$\mathbb{P}\big(\mathbf{E}_0\big) \geq \mathbb{P}\big[y^* \in Y\big] \geq 1 - \frac{1}{2(n+1)}.$$

Next, consider the decision whether $\hat{y} \in WS^J$ in iteration j of the algorithm of Lemma 6.2, based on computing a random subset $Y(J) \subseteq WS^J$ by the algorithm of Lemma 6.11 and checking if $\hat{y} \in Y(J)$. Then again by Lemma 6.11, we have the following:

$$\mathbb{P}\big(\mathbf{E}_j \mid \hat{y} \notin WS^J\big) = \mathbb{P}\big(\hat{y} \notin Y(J) \mid \hat{y} \notin WS^J\big) = 1,$$

and

$$\mathbb{P}\big(\mathbf{E}_j \mid \hat{y} \in WS^J\big) = \mathbb{P}\big(\hat{y} \in Y(J) \mid \hat{y} \in WS^J\big) \geq 1 - \frac{1}{2(n+1)}.$$

Therefore, we obtain the following:

$$\begin{aligned}\mathbb{P}\big(\mathbf{E}_j\big) &= \mathbb{P}\big(\mathbf{E}_j \mid \hat{y} \notin WS^J\big)\mathbb{P}\big[\hat{y} \notin WS^J\big] + \mathbb{P}\big(\mathbf{E}_j \mid \hat{y} \in WS^J\big)\mathbb{P}\big[\hat{y} \in WS^J\big] \\ &\geq \Big(1 - \frac{1}{2(n+1)}\Big)\big(\mathbb{P}\big[\hat{y} \notin WS^J\big] + \mathbb{P}\big[\hat{y} \in WS^J\big]\big) = 1 - \frac{1}{2(n+1)}.\end{aligned}$$

So the probability that the algorithm outputs an optimal solution $\hat{x}$ is at least as follows:

$$\mathbb{P}\Big(\bigcap_{j=0}^{n} \mathbf{E}_j\Big) = 1 - \mathbb{P}\Big(\bigcup_{j=0}^{n} \mathbf{E}_j^c\Big) \geq 1 - \sum_{j=0}^{n} \mathbb{P}\big(\mathbf{E}_j^c\big) \geq 1 - \sum_{j=0}^{n} \frac{1}{2(n+1)} = \frac{1}{2}.$$

This completes the proof, solving the nonlinear matroid intersection problem. □

6.2 Optimization over independence systems

In this section, we study the nonlinear combinatorial optimization problem (6.1) over an arbitrary *independence system* introduced in Section 1.2.1, that is, over a nonempty set $S \subseteq \{0,1\}^n$ such that $z \in \{0,1\}^n$ and $z \leq x \in S$ imply $z \in S$. This is a very broad problem and hence very difficult. In particular, the set of common independent sets of any number k of matroids is an independence system, so, as explained in Section 2.5.2, already for $d = 1$ includes the traveling salesman problem and is hard.

Therefore, we consider the problem under the following quite restrictive assumptions. First, as in Chapter 2, we assume that we can do linear optimization over S to begin with, that is, we assume that S is presented by a *linear-optimization oracle* that, queried on $w \in \mathbb{Z}^n$, solves the problem $\max\{wx : x \in S\}$. Second, we assume that $d = 1$ so that the weight matrix W becomes simply a weight vector $w \in \mathbb{Z}^n$ and the function f is univariate. Finally, we restrict the entries of w to take on values from a fixed p-tuple $a = (a_1, \ldots, a_p)$ of positive integers, so that $w \in \{a_1, \ldots, a_p\}^n$. Without loss of generality, we assume that $a = (a_1, \ldots, a_p)$ is *primitive*, that is, the a_i are distinct positive integers having greatest common divisor $\gcd(a) := \gcd(a_1, \ldots, a_p) = 1$. But, as elsewhere in this chapter, we do allow the function $f : \mathbb{Z} \to \mathbb{R}$, presented by comparison oracle, to be arbitrary.

It turns out, as we show later, that even under these restrictions, the solution of the nonlinear problem (6.1) to optimality may require exponential time. Nonetheless, we are able to find in polynomial time a solution which is approximative in the following unusual and interesting sense. For a nonnegative integer r, we say that $x^* \in S$ is an *r-best solution* to the minimization problem over S, if there are at most r better objective function *values* attainable by feasible points, that is,

$$\left|\left\{f(wx) : f(wx) < f\left(wx^*\right),\ x \in S\right\}\right| \leq r.$$

In particular, a 0-best solution is optimal. Recall that the *Frobenius number* of a primitive tuple $a = (a_1, \ldots, a_p)$ is the largest integer $\mathrm{F}(a)$ that is not expressible as a nonnegative integer combination of the a_i (see more details in Section 6.2.1). It turns out that we can obtain an $r(a)$-best solution with $r(a)$ depending on a only and bounded by the Frobenius numbers derived from subtuples of a. In particular, for $p = 2$ and any pair $a = (a_1, a_2)$, we have $r(a) \leq \mathrm{F}(a)$ the Frobenius number. So, for $a = (2,3)$, that is, for weight vectors $w \in \{2,3\}^n$, since $\mathrm{F}(2,3) = 1$, our algorithm obtains a 1-best solution in polynomial time. Quite amazingly, it turns out that finding a 0-best (namely, optimal) solution for $w \in \{2,3\}^n$ requires exponential time. The following table emphasizes the contrast between these results.

$w \in \{a_1, \ldots, a_p\}^n$	$w \in \{2,3\}^n$	$w \in \{2,3\}^n$
$r(a_1, \ldots, a_p)$-best solution	1-best solution	0-best (optimal) solution
Theorem 6.23	Theorem 6.23	Theorem 6.24
time polynomial in n	time polynomial in n	time exponential in n

We now proceed to derive these results. In Section 6.2.1, we provide the approximative solution, and in Section 6.2.2, we prove that exponential time is needed for exact solution.

6.2.1 Approximative nonlinear optimization

The main result in this subsection is Theorem 6.23: for every tuple $a = (a_1, \ldots, a_p)$, there exists a constant $r(a)$, such that we can find an $r(a)$-best solution for the problem of minimizing any nonlinear composite function $f(wx)$ with any weight vector $w \in \{a_1, \ldots, a_p\}^n$ over any independence system S, in polynomial time.

As the derivation of this is quite long, we split this subsection into two parts.

Monoids and the Frobenius numbers

We begin with some properties of monoids and the Frobenius numbers needed in the sequel. The key result here is Lemma 6.16 which is of interest in its own right.

Throughout, $a = (a_1, \ldots, a_p)$ is a primitive tuple, that is, the a_i are distinct positive integers having greatest common divisor $\gcd(a) = \gcd(a_1, \ldots, a_p) = 1$. For $p = 1$, the only primitive $a = (a_1)$ is the one with $a_1 = 1$. The *monoid* of $a = (a_1, \ldots, a_p)$ is the set of nonnegative integer combinations of its entries:

$$M(a) := \{\mu a = \textstyle\sum_{i=1}^{p} \mu_i a_i : \mu \in \mathbb{Z}_+^p\}.$$

The *gap set* of a is the set $G(a) := \mathbb{Z}_+ \setminus M(a)$ and is well known to be finite [16]. If all $a_i \geq 2$, then $G(a)$ is nonempty, and its maximum element is known as the *Frobenius number* of a and will be denoted by $\mathrm{F}(a) := \max G(a)$. If some $a_i = 1$ then $G(a) = \emptyset$, in which case we define $\mathrm{F}(a) := 0$ by convention. We also define $G(a) := \emptyset$ and $\mathrm{F}(a) := 0$ by convention for the empty p-tuple $a = ()$ with $p = 0$.

Example 6.13. The gap set and the Frobenius number of the pair $a = (3, 5)$ are as follows:

$$G(a) = \{1, 2, 4, 7\}, \quad \mathrm{F}(a) = 7.$$

Classical results of Schur and Sylvester assert that for all $p \geq 2$ and all $a = (a_1, \ldots, a_p)$ with each $a_i \geq 2$, the Frobenius number obeys the upper bound:

$$\mathrm{F}(a) + 1 \leq \min\{(a_i - 1)(a_j - 1) : 1 \leq i < j \leq p\} \tag{6.7}$$

with equality $\mathrm{F}(a) + 1 = (a_1 - 1)(a_2 - 1)$ holding for $p = 2$, see [16] for details.

For monoid $M(a)$ and $\lambda \in \mathbb{Z}_+^p$, we frequently use the following subset of $M(a)$:

$$M(a, \lambda) := \{\mu a : \mu \in \mathbb{Z}_+^p,\ \mu \leq \lambda\} \subset M(a).$$

Such subsets satisfy the following simple lemma. Here and later on, for integers $z, s \in \mathbb{Z}$ and set of integers $Z \subseteq \mathbb{Z}$, we let $z + sZ := \{z + sx : x \in Z\}$.

Lemma 6.14. *For every primitive p-tuple a and $\lambda \in \mathbb{Z}_+^p$ the set $M(a, \lambda)$ satisfies the following:*

$$M(a, \lambda) \subseteq \{0, 1, \ldots, a\lambda\} \setminus \big(G(a) \cup (a\lambda - G(a))\big). \tag{6.8}$$

Proof. Note that $M(a,\lambda)$ is symmetric on $\{0,1,\ldots,a\lambda\}$, that is, for any integer g we have that $g \in M(a,\lambda)$ if and only if it $a\lambda - g \in M(a,\lambda)$; indeed, $g = a\mu$ with $0 \le \mu \le \lambda$ if and only if $a\lambda - g = a(\lambda-\mu)$ with $0 \le \lambda-\mu \le \lambda$. Clearly, $M(a,\lambda) \subseteq \{0,1,\ldots,a\lambda\} \setminus G(a)$. The claim now follows from the symmetry. □

The following definition plays an important role in the sequel.

Definition 6.15. Call $\lambda \in \mathbb{Z}_+^p$ *saturated* for primitive p-tuple a if we have equality:

$$M(a,\lambda) = \{0,1,\ldots,a\lambda\} \setminus \big(G(a) \cup \big(a\lambda - G(a)\big)\big).$$

In particular, if some $a_i = 1$ then λ saturated for a implies $M(a,\lambda) = \{0,1,\ldots,a\lambda\}$.

Example 6.13 (continued). Let $a = (3,5)$ and $\lambda = (3,4)$. Then $a\lambda = 29$, and the two values $12 = 4\cdot 3 + 0\cdot 5$ and $17 = 4\cdot 3 + 1\cdot 5$ are not in $M(a,\lambda)$ but are in $\{0,1,\ldots,a\lambda\} \setminus (G(a) \cup (a\lambda - G(a)))$. Therefore, this λ is not saturated for this a.

Let as usual $\|a\|_\infty := \max\{a_1,\ldots,a_p\}$. Call $a = (a_1,\ldots,a_p)$ *divisible* if a_i divides a_{i+1} for $i = 1,\ldots,p-1$. The following key lemma asserts that, for every primitive a, every componentwise sufficiently large $\lambda \in \mathbb{Z}_+^p$ is saturated.

Lemma 6.16. *Let $a = (a_1,\ldots,a_p)$ be any primitive p-tuple. Then we have the following:*

1. *every $\lambda \in \mathbb{Z}_+^p$ with $\lambda_i \ge \|a\|_\infty$ for $1 \le i \le p$ is saturated for a;*
2. *every $\lambda \in \mathbb{Z}_+^p$ with $\lambda_i \ge \frac{a_{i+1}}{a_i} - 1$ for $1 \le i < p$ is saturated for divisible a.*

Proof. We begin with part 1. We prove that λ is saturated if all $\lambda_i \ge 2\|a\|_\infty$. The proof of the stronger statement is quite technical and can be found in [67].

Fix any $\lambda = (\lambda_1,\ldots,\lambda_p)$ satisfying $\lambda_i \ge 2\|a\|_\infty$ for all i. Consider any integer $v \in \{0,1,\ldots,a\lambda\} \setminus (G(a) \cup (a\lambda - G(a)))$. We show that $v \in M(a,\lambda)$ as well.

Suppose first that $v \le \frac{1}{2}\lambda a$. Assume indirectly that $v \notin M(a,\lambda)$ and choose $\mu \in \mathbb{Z}_+^p$ such that $v = \mu a$ and μ has smallest possible violation $\sum\{\mu_i - \lambda_i : \mu_i > \lambda_i\}$. Let j be an index such that $\mu_j > \lambda_j$. Now, $v = \mu a \le \frac{1}{2}\lambda a$ and $\frac{1}{2}\lambda_i \ge \|a\|_\infty$ for all i imply the following:

$$\sum_{i=1}^{p} \big((\lambda_i - \|a\|_\infty) - \mu_i\big)a_i \ge \sum_{i=1}^{p} \Big(\frac{1}{2}\lambda_i - \mu_i\Big)a_i \ge 0.$$

So some index k satisfies $(\lambda_k - \|a\|_\infty) - \mu_k \ge 0$ and therefore $\mu_k \le \lambda_k - \|a\|_\infty$.

Now, define a new tuple ν by the following:

$$\nu_j := \mu_j - a_k, \quad \nu_k := \mu_k + a_j, \quad \text{and} \quad \nu_i := \mu_i \quad \text{for all } i \ne j,k.$$

Then $\nu_j > \lambda_j - a_k > 0$ and $\nu_k \le (\lambda_k - \|a\|_\infty) + a_j \le \lambda_k$. So ν is nonnegative, satisfies $\nu a = \mu a = v$, and has smaller violation than μ, which is a contradiction to the choice of μ. Therefore, indeed $v \in M(a,\lambda)$.

Next, suppose that $v > \frac{1}{2}\lambda a$. Put $u := \lambda a - v$. Then $v \notin (G(a) \cup (a\lambda - G(a)))$ implies $u \notin ((a\lambda - G(a)) \cup G(a))$. Since $u < \frac{1}{2}\lambda a$, by what we just proved applied to u,

we conclude that $u \in M(a,\lambda)$, and hence $u = \mu a$ for some $\mu \le \lambda$. Then $v = (\lambda - \mu)a$ and hence $v \in M(a,\lambda)$ as well. This completes the proof of part 1.

We proceed with part 2. We use induction on p. For $p = 1$, we have $a_1 = 1$ and hence $G(a) = \emptyset$. Therefore, it is easy to verify that every $\lambda = (\lambda_1)$ is saturated.

Next, consider $p > 1$. We now use induction on λ_p. Suppose first that $\lambda_p = 0$. Let $a' := (a_1, \ldots, a_{p-1})$ and $\lambda' := (\lambda_1, \ldots, \lambda_{p-1})$. Consider any $0 \le v \le \lambda a = \lambda' a'$. Since λ' is saturated by induction on p, there exists $\mu' \le \lambda'$ with $v = \mu' a'$. Then $\mu := (\mu', 0) \le \lambda$ and $v = \mu a$. So λ is also saturated. Suppose next that $\lambda_p > 0$. Let $\tau := (\lambda_1, \ldots, \lambda_{p-1}, \lambda_p - 1)$. Consider any value $0 \le v \le \tau a = \lambda a - a_p$. Since τ is saturated by induction on λ_p, there is a $\mu \le \tau < \lambda$ with $v = \mu a$ and so $v \in M(a,\tau) \subseteq M(a,\lambda)$. Moreover, $v + a_p = \hat{\mu} a$ with $\hat{\mu} := (\mu_1, \ldots, \mu_{p-1}, \mu_p + 1) \le \lambda$ so $v + a_p \in M(a,\lambda)$ as well. Therefore,

$$\{0, 1, \ldots, \tau a\} \cup \big\{a_p, a_p + 1, \ldots, a\lambda\big\} \subseteq M(a,\lambda). \tag{6.9}$$

Now,

$$\tau a = \sum_{i=1}^{p} \tau_i a_i \ge \sum_{i=1}^{p-1} \lambda_i a_i \ge \sum_{i=1}^{p-1} \Big(\frac{a_{i+1}}{a_i} - 1\Big) a_i = \sum_{i=1}^{p-1} \big(a_{i+1} - a_i\big) = a_p - 1,$$

implying that the left-hand side of (6.9) is in fact equal to $\{0, 1, \ldots, \lambda a\}$. So λ is indeed saturated. This completes the double induction and the proof of part 2. □

Efficient approximation

We proceed to solve the nonlinear optimization problem $\min\{f(wx) : x \in S\}$ with $S \subseteq \{0,1\}^n$ an independence system presented by a linear-optimization oracle, weight vector $w \in \{a_1, \ldots, a_p\}^n$ taking values in a primitive tuple $a = (a_1, \ldots, a_p)$, and arbitrary univariate function $f : \mathbb{Z} \to \mathbb{R}$ presented by a comparison oracle.

The outline of our approach is the following variant of Strategy 2.1. The key step is that of computing a subset $T \subseteq S$ of the feasible set, whose image $wT \subseteq wS$ contains most image points and hence provides a good approximation of the true image wS. Note that here $wS = \{wx : x \in S\} \subset \mathbb{Z}$ is a set of integer numbers. Then, by inspecting T and using the comparison oracle of f, we determine a point $x \in T \subseteq S$ whose image $y := wx$ minimizes f over $wT \subseteq wS$, and output x, which is in the fiber $W^{-1}(y) \cap T \subseteq W^{-1}(y) \cap S$, as the approximative solution.

For any point $x \in \{0,1\}^n$, define the *independence system generated by* x to be as follows:

$$S(x) := \big\{z \in \{0,1\}^n : z \le x\big\}.$$

For any independence system S and any $x \in S$, we have $S(x) \subseteq S$ and hence $wS(x) \subseteq wS$, so the image of $S(x)$ provides an approximation of the image of S.

The set $S(x)$ has 2^k points, where $k := |\operatorname{supp}(x)|$ and hence its cardinality is typically exponential. Nonetheless, as the next lemma shows, for w taking values in a we can compute in polynomial time a subset $T \subseteq S(x)$ such that $wS(x) = wT$.

Here and later on, for $a = (a_1, \ldots, a_p) \in \mathbb{Z}_+^p$ and $w \in \{a_1, \ldots, a_p\}^n$, we use the following notation. Let $N := \{1, \ldots, n\}$ and $N_i := \{j \in N : w_j = a_i\}$ for $i = 1, \ldots, p$, giving a partition $N = \biguplus_{i=1}^p N_i$ of the ground set according to weights of elements. Also, for $x \in \{0,1\}^n$ and $i = 1, \ldots, p$, let $\sigma_i(x) := |\operatorname{supp}(x) \cap N_i|$ and $\sigma(x) := (\sigma_1(x), \ldots, \sigma_p(x))$, so that the weight of x under w satisfies $wx = a\sigma(x)$.

Lemma 6.17. *For every fixed p, there is an algorithm that, given $x \in \{0,1\}^n$, $a \in \mathbb{Z}_+^p$, and $w \in \{a_1, \ldots, a_p\}^n$, in time polynomial in $\langle w \rangle$, obtains $T \subseteq S(x)$ with the following:*

$$wS(x) = \big\{wz : z \in S(x)\big\} = \{wz : z \in T\} = wT.$$

Proof. Using the notation introduced above, let $\sigma := \sigma(x) = (\sigma_1(x), \ldots, \sigma_p(x))$. For each $\mu \in \mathbb{Z}_+^p$ satisfying $\mu \le \sigma$ determine some $x_\mu \le x$ with $\sigma(x_\mu) = \mu$ by zeroing out suitable entries of x. Then $T := \{x_\mu : \mu \in \mathbb{Z}_+^p,\ \mu \le \sigma\}$ is computable in polynomial time since there are $\prod_{i=1}^p (\sigma_i + 1) \le (n+1)^p$ such $\mu \le \sigma$, and

$$\begin{aligned} wS(x) &= \big\{wz = a\sigma(z) : z \in \{0,1\}^n,\ z \le x\big\} = \big\{a\mu : \mu \in \mathbb{Z}_+^p,\ \mu \le \sigma\big\} \\ &= \big\{a\sigma\big(x_\mu\big) = wx_\mu : \mu \in \mathbb{Z}_+^p,\ \mu \le \sigma\big\} = \{wz : z \in T\} = wT. \end{aligned}$$

□

A particularly appealing approximation $wS(x)$ of the image wS is given by any point $\hat{x} \in S$ maximizing the linear function wx over S, since for any such point:

$$\{0, w\hat{x}\} \subseteq wS(\hat{x}) \subseteq wS \subseteq \{0, 1, \ldots, w\hat{x}\}. \tag{6.10}$$

Moreover, such an approximation $wS(\hat{x})$ can be easily computed as follows: first query the linear-optimization oracle of S about w and obtain $\hat{x} \in S$ attaining $w\hat{x} = \max\{wx : x \in S\}$; next compute $wS(\hat{x})$ by the algorithm of Lemma 6.17.

Unfortunately, as the following example shows, the number of points of wS which are missing from $wS(\hat{x})$ cannot generally be bounded by any constant.

Example 6.18 (unbounded gap). Let $a := (1, 2)$ and $n := 4m$. Define three vectors in $\mathbb{Z}^n$ by $x^1 := \sum_{j=1}^{2m} \mathbf{1}_j$, $x^2 := \sum_{j=2m+1}^{4m} \mathbf{1}_j$, and $w := x^1 + 2x^2$, namely,

$$x^1 := (1, \ldots, 1, 0, \ldots, 0), \quad x^2 := (0, \ldots, 0, 1, \ldots, 1), \quad w = (1, \ldots, 1, 2, \ldots, 2).$$

Let S be the following independence system:

$$S := S\big(x^1\big) \cup S\big(x^2\big) = \big\{x \in \{0,1\}^n : x \le x^1\big\} \cup \big\{x \in \{0,1\}^n : x \le x^2\big\}.$$

Then the unique maximizer of wx over S is $\hat{x} := x^2$, with $w\hat{x} = 4m$, and hence

$$\begin{aligned} wS(\hat{x}) &= \{2i : i = 0, 1, \ldots, 2m\} \\ &\subsetneq \big\{i : i = 0, 1, \ldots, 2m\big\} \cup \{2i : i = 0, 1, \ldots, 2m\} = wS, \end{aligned}$$

and

$$\big|wS \setminus wS(\hat{x})\big| = \big|\{1, 3, \ldots, 2m-1\}\big| = m = \frac{n}{4}.$$

This example shows that approximating the image using a single system which is generated by a point is not satisfactory. We proceed to refine the above approach by partitioning S into many pieces, approximating the image of each individually and combining these approximations to get a good overall approximation.

Let $S \subseteq \{0,1\}^n$ be a set, $a = (a_1, \ldots, a_p) \in \mathbb{Z}_+^p$ a tuple, and $w \in \{a_1, \ldots, a_p\}^n$ a weight vector. Fix any $\lambda \in \mathbb{Z}_+^p$. For any $\mu \in \mathbb{Z}_+^p$ satisfying $\mu \leq \lambda$, define the following:

$$S_\mu^\lambda := \left\{ x \in S : \begin{array}{ll} \sigma_i(x) = \mu_i & \text{if } \mu_i < \lambda_i, \\ \sigma_i(x) \geq \mu_i & \text{if } \mu_i = \lambda_i \end{array} \right\}. \tag{6.11}$$

Then each $x \in S$ is in precisely one S_μ^λ, the one with $\mu_i = \min\{\sigma_i(x), \lambda_i\}$ for all i. Therefore, the S_μ^λ with λ fixed and $\mu \leq \lambda$ form a partition of S, that is, we have the following:

$$S = \biguplus_{\mu \leq \lambda} S_\mu^\lambda.$$

We can now outline our approach for approximating the image. We choose λ saturated for a. For each S_μ^λ, we compute a point $x_\mu^\lambda \in S_\mu^\lambda$ maximizing w over S_μ^λ. We then compute $T_\mu^\lambda \subseteq S(x_\mu^\lambda)$ such that $wT_\mu^\lambda = wS(x_\mu^\lambda)$ approximates wS_μ^λ. Finally, we take $T := \bigcup_{\mu \leq \lambda} T_\mu^\lambda$ and show that wT is a good approximation of wS.

We proceed to develop the details of this outlined approach. We need two lemmas to show that we can maximize the linear function wx over each S_μ^λ. For any set $S \subseteq \{0,1\}^n$ and any two subsets $\emptyset \subseteq I \subseteq J \subseteq N = \{1, \ldots, n\}$ let

$$S_I^J := \{x \in S : I \subseteq \operatorname{supp}(x) \subseteq J\}.$$

The restriction of S to J in Section 6.1.1 is the special case $S^J = S_\emptyset^J$. Our first simple lemma reduces linear optimization over such sets to linear optimization over S.

Lemma 6.19. *There is an algorithm that, given any set $S \subseteq \{0,1\}^n$ presented by linear-optimization oracle, $w \in \mathbb{Z}^n$, and $I \subseteq J \subseteq N = \{1, \ldots, n\}$, solves the linear optimization problem* $\max\{wx : x \in S_I^J\}$ *in time which is polynomial in* $\langle w \rangle$.

Proof. Given such S, w, I and J, let $\alpha := n\|w\|_\infty + 1$ and define $u \in \mathbb{Z}^n$ by the following:

$$u_j := \begin{cases} w_j + \alpha, & j \in I, \\ w_j, & j \in J \setminus I, \\ w_j - \alpha, & j \notin J, \end{cases} \qquad j = 1, \ldots, n.$$

Using the linear-optimization oracle of S, solve the linear optimization problem:

$$\max\{ux : x \in S\}.$$

If the oracle asserts that S is empty then so is S_I^J. Otherwise, let $x^* \in S$ be the optimal solution returned by the oracle. Now, note that for all $x \in S_I^J$ and all $z \in S \setminus S_I^J$, if any, we have $ux = wx + |I|\alpha$ and $uz \le wz + (|I| - 1)\alpha$ and therefore,

$$ux - uz \ge \alpha + w(x - z) \ge \alpha - n\|w\|_\infty = 1.$$

This implies that if $x^* \notin S_I^J$ then S_I^J is empty, whereas if $x^* \in S_I^J$ then for all $x \in S_I^J$ we have $wx^* - wx = ux^* - ux \ge 0$ and therefore x^* maximizes w over S_I^J. □

Lemma 6.20. *For every fixed p and $\lambda \in \mathbb{Z}_+^p$, there is an algorithm that, given set $S \subseteq \{0,1\}^n$ presented by linear-optimization oracle, $a \in \mathbb{Z}_+^p$, $w \in \{a_1, \ldots, a_p\}^n$, and $\mu \in \mathbb{Z}_+^p$ with $\mu \le \lambda$, solves* $\max\{wx : x \in S_\mu^\lambda\}$ *in time polynomial in $\langle w \rangle$.*

Proof. Let $K := \{i : \mu_i < \lambda_i\}$. As usual, let $N_i := \{j : w_j = a_i\}$ for $i = 1, \ldots, p$. It is clear that if $\mu_i > |N_i|$ for some i then S_μ^λ is empty. Otherwise, for every choice of subsets $S_i \subseteq N_i$ satisfying $|S_i| = \mu_i$ for $i = 1, \ldots, p$, do the following: let

$$I := \bigcup_{i=1}^{p} S_i, \quad J := \bigcup_{i \in K} S_i \cup \bigcup_{i \notin K} N_i,$$

and use the algorithm of Lemma 6.19 to either detect that S_I^J is empty or obtain a point $x(S_1, \ldots, S_p) \in S_I^J$ which attains $\max\{wx : x \in S_I^J\}$.

It is clear that S_μ^λ is the union of the S_I^J over all choices $S_1, \ldots, S_p$ as above. So if all S_I^J are empty then so is S_μ^λ. Otherwise, the point x^* among the $x(S_1, \ldots, S_p)$, which attains the maximum value wx, is a point S_μ^λ which maximizes w over S_μ^λ.

Now, when $\mu_i \le |N_i|$ for all i, the number of possible choices $S_1, \ldots, S_p$ is as follows:

$$\prod_{i=1}^{p} \binom{|N_i|}{\mu_i} \le \prod_{i=1}^{p} n^{\mu_i} \le \prod_{i=1}^{p} n^{\lambda_i},$$

which is polynomial since λ is fixed, and so we can indeed either conclude that S_μ^λ is empty or find $x^* \in S_\mu^\lambda$ which maximizes w over S_μ^λ in polynomial time. □

For each primitive p-tuple $a = (a_1, \ldots, a_p)$, define a constant $r(a)$ as follows. Let $\lambda := (\lambda_1, \ldots, \lambda_p)$ with $\lambda_i := \|a\|_\infty$ for all i. For each $\mu \in \mathbb{Z}_+^p$ with $\mu \le \lambda$ let

$$I_\mu := \{i : \mu_i = \lambda_i\}, \quad g_\mu := \gcd(a_i : i \in I_\mu), \quad a_\mu := \frac{1}{g_\mu}(a_i : i \in I_\mu). \tag{6.12}$$

Finally, let $r(a)$ be the sum of cardinalities of gap sets of a_μ for all $\mu \in \mathbb{Z}_+^p$ with $\mu \le \lambda$, which is bounded by the sum of corresponding Frobenius numbers, that is,

$$r(a) := \sum_{\mu \le \lambda} |G(a_\mu)| \le \sum_{\mu \le \lambda} \mathrm{F}(a_\mu). \tag{6.13}$$

We can now approximate the image by a subset missing at most $r(a)$ points.

Lemma 6.21. *For every fixed primitive $a = (a_1, \dots, a_p)$, there is an algorithm that, given independence system $S \subseteq \{0,1\}^n$ presented by linear-optimization oracle and $w \in \{a_1, \dots, a_p\}^n$, computes, in time polynomial in n, a subset $T \subseteq S$ satisfying the following:*

$$|wS \setminus wT| \leq r(a).$$

Proof. We begin by describing the polynomial time algorithm, and then prove that the image wT of its output $T \subseteq S$ misses at most $r(a)$ points from wS.

Define $\lambda := (\lambda_1, \dots, \lambda_p)$ by $\lambda_i := \|a\|_\infty$ for every i. For every $\mu \in \mathbb{Z}_+^p$ with $\mu \leq \lambda$ do the following: use the algorithm of Lemma 6.20 to either detect that S_μ^λ is empty or obtain a point $x_\mu^\lambda \in S_\mu^\lambda$ which attains $\max\{wx : x \in S_\mu^\lambda\}$; if $S_\mu^\lambda = \emptyset$ then define $T_\mu^\lambda := \emptyset$, else compute by the algorithm of Lemma 6.17 a subset $T_\mu^\lambda \subseteq S(x_\mu^\lambda)$ of the system $S(x_\mu^\lambda)$ generated by x_μ^λ, such that $wS(x_\mu^\lambda) = wT_\mu^\lambda$. Finally, let

$$T := \bigcup_{\mu \leq \lambda} T_\mu^\lambda.$$

Note that the number of tuples $\mu \leq \lambda$ and hence of applications of the polynomial time algorithms of Lemma 6.17 and Lemma 6.20 is $\prod_{i=1}^p (\lambda_i + 1) = (1 + \|a\|_\infty)^p$ which is constant since a is fixed. Therefore, the algorithm is polynomial time.

We proceed to bound the number of missing image points. Consider any $\mu \leq \lambda$ with $S_\mu^\lambda \neq \emptyset$ and let x_μ^λ be the maximizer of w over S_μ^λ obtained by the algorithm. Let I_μ, g_μ, and a_μ be as in (6.12), and let $h_\mu := \sum\{a_i\mu_i : i \notin I_\mu\}$. For every $x \in \{0,1\}$ let $\sigma_\mu(x) := (\sigma_i(x) : i \in I_\mu)$ where, as usual, $\sigma_i(x) = |\operatorname{supp}(x) \cap N_i|$. Then for every point $x \in S_\mu^\lambda$, we have the following:

$$wx = \sum_{i \notin I_\mu} a_i\sigma_i(x) + \sum_{i \in I_\mu} a_i\sigma_i(x) = \sum_{i \notin I_\mu} a_i\mu_i + g_\mu \sum_{i \in I_\mu} \frac{1}{g_\mu} a_i\sigma_i(x) = h_\mu + g_\mu a_\mu \sigma_\mu(x),$$

and hence $wx \in h_\mu + g_\mu M(a_\mu)$ and $wx \leq wx_\mu^\lambda = h_\mu + g_\mu a_\mu \sigma_\mu(x_\mu^\lambda)$. Therefore,

$$wS_\mu^\lambda \subseteq h_\mu + g_\mu\big(M(a_\mu) \cap \{0, 1, \dots, a_\mu\sigma_\mu(x_\mu^\lambda)\}\big).$$

Let $S(x_\mu^\lambda) = \{x : x \leq x_\mu^\lambda\}$ be the system generated by x_μ^λ. For any $\nu \leq \sigma_\mu(x_\mu^\lambda)$, there is an $x \in S(x_\mu^\lambda)$ obtained by zeroing out suitable entries of x_μ^λ such that $\sigma_\mu(x) = \nu$ and $\sigma_i(x) = \sigma_i(x_\mu^\lambda) = \mu_i$ for $i \notin I_\mu$, and hence $wx = h_\mu + g_\mu a_\mu \nu$. Therefore,

$$h_\mu + g_\mu M\big(a_\mu, \sigma_\mu(x_\mu^\lambda)\big) \subseteq wS(x_\mu^\lambda).$$

Since x_μ^λ is in S_μ^λ, for each $i \in I_\mu$, we have $\sigma_i(x_\mu^\lambda) \geq \mu_i = \lambda_i = \|a\|_\infty \geq \|a_\mu\|_\infty$. Therefore, by Lemma 6.16, we find that $\sigma_\mu(x_\mu^\lambda)$ is saturated for a_μ, and hence,

$$M\big(a_\mu, \sigma_\mu(x_\mu^\lambda)\big) = \big(M(a_\mu) \cap \{0, 1, \dots, a_\mu\sigma_\mu(x_\mu^\lambda)\}\big) \setminus \big(a_\mu\sigma_\mu(x_\mu^\lambda) - G(a_\mu)\big).$$

This implies that

$$wS_\mu^\lambda \setminus wS\big(x_\mu^\lambda\big) \subseteq h_\mu + g_\mu\big(a_\mu\sigma_\mu\big(x_\mu^\lambda\big) - G\big(a_\mu\big)\big).$$

Since $T_\mu^\lambda \subseteq S(x_\mu^\lambda)$ computed by the algorithm satisfies $wT_\mu^\lambda = wS(x_\mu^\lambda)$, we obtain the following:

$$\big|wS_\mu^\lambda \setminus wT_\mu^\lambda\big| = \big|wS_\mu^\lambda \setminus wS\big(x_\mu^\lambda\big)\big| \le \big|G\big(a_\mu\big)\big|.$$

Since the set $T \subseteq S$ computed by the algorithm satisfies the following:

$$wS \setminus wT = w\Big(\bigcup_{\mu\le\lambda} S_\mu^\lambda\Big) \setminus w\Big(\bigcup_{\mu\le\lambda} T_\mu^\lambda\Big) \subseteq \bigcup_{\mu\le\lambda} \big(wS_\mu^\lambda \setminus wT_\mu^\lambda\big),$$

it therefore finally follows that:

$$|wS \setminus wT| \le \sum_{\mu\le\lambda} \big|wS_\mu^\lambda \setminus wT_\mu^\lambda\big| \le \sum_{\mu\le\lambda} \big|\big(G\big(a_\mu\big)\big| = r(a).$$

□

The next lemma provides some estimates on the constant $r(a)$ controlling the approximation quality. Recall our notation in (6.12) and (6.13) for primitive tuple $a \in \mathbb{Z}_+^p$: define $\lambda \in \mathbb{Z}_+^p$ by $\lambda_i := \|a\|_\infty$ for all i and for each $\mu \in \mathbb{Z}_+^p$ with $\mu \le \lambda$ let

$$I_\mu := \big\{i : \mu_i = \lambda_i\big\}, \quad g_\mu := \gcd\big(a_i : i \in I_\mu\big), \quad a_\mu := \frac{1}{g_\mu}\big(a_i : i \in I_\mu\big),$$

and

$$r(a) := \sum_{\mu\le\lambda} \big|G\big(a_\mu\big)\big| \le \sum_{\mu\le\lambda} \mathrm{F}\,\big(a_\mu\big).$$

Lemma 6.22. *For every primitive p-tuple $a = (a_1, \ldots, a_p)$ the following hold:*

1. *an upper bound on $r(a)$ is given by $r(a) \le 2^p\|a\|_\infty^p$;*
2. *for divisible a we have $r(a) = 0$;*
3. *for $p = 2$, that is, for $a = (a_1, a_2)$, we have $r(a) = |G(a)| \le \mathrm{F}(a)$.*

Proof. Define $\lambda \in \mathbb{Z}_+^p$ by $\lambda_i := \|a\|_\infty$ for all i and I_μ, a_μ, $r(a)$ as above. Note that if I_μ is empty or a singleton then $a_\mu = ()$ or $a_\mu = 1$ so $G(a_\mu) = \emptyset$ and $\mathrm{F}(a_\mu) = 0$.

Part 1. First, as noted, for each tuple $\mu \le \lambda$ with $|I_\mu| \le 1$, we have $\mathrm{F}(a_\mu) = 0$. Second, there are at most $2^p\|a\|_\infty^{p-2}$ tuple $\mu \le \lambda$ with $|I_\mu| \ge 2$, and for each, the bound of (6.7) implies $\mathrm{F}(a_\mu) \le \|a\|_\infty^2$. So we obtain, as claimed the following:

$$r(a) \le \sum_{\mu\le\lambda} \mathrm{F}\,\big(a_\mu\big) \le 2^p\|a\|_\infty^{p-2}\|a\|_\infty^2 \;=\; 2^p\|a\|_\infty^p.$$

Part 2. If a is divisible, then the least entry of every nonempty tuple a_μ is 1 and hence $G(a_\mu) = \emptyset$ for every $\mu \le \lambda$. Therefore, $r(a) = 0$.

Part 3. As noted, $G(a_\mu) = \emptyset$ for each $\mu \leq \lambda$ with $|I_\mu| \leq 1$. For $p = 2$, the only $\mu \leq \lambda$ with $|I_\mu| = 2$ is $\mu = \lambda$. Since $a^\lambda_\lambda = a$, we obtain $r(a) = |G(a)| \leq \mathrm{F}(a)$. □

We can finally obtain a result of [67], providing an approximative polynomial time algorithm for nonlinear optimization over independence systems. Recall that $x^* \in S$ is an r-best solution of the minimization problem (6.1) if there are at most r better objective function values attainable by feasible points, that is,

$$\left|\left\{f(wx) : f(wx) < f\left(wx^*\right),\ x \in S\right\}\right| \leq r.$$

Theorem 6.23. *For every primitive p-tuple $a = (a_1, \ldots, a_p)$, there is a constant $r(a)$ and an algorithm that, given any independence system $S \subseteq \{0,1\}^n$ presented by a linear-optimization oracle, weight vector $w \in \{a_1, \ldots, a_p\}^n$, and function $f : \mathbb{Z} \to \mathbb{R}$ presented by a comparison oracle, determines an $r(a)$-best solution of the nonlinear problem $\min\{f(wx) : x \in S\}$, in time polynomial in n. Furthermore:*

1. *the quality of the approximation obtained satisfies the bound $r(a) \leq 2^p \|a\|^p_\infty$;*
2. *if a_i divides a_{i+1} for $1 \leq i < p$ then the algorithm finds an optimal solution;*
3. *for $p = 2$, that is, for $a = (a_1, a_2)$, the algorithm finds an $\mathrm{F}(a)$-best solution.*

Proof. Define $r(a)$ as in (6.13). Compute $T \subseteq S$ such that $|wS \setminus wT| \leq r(a)$ by the algorithm of Lemma 6.21. Now, inspecting T and using the comparison oracle of f, find a point $\hat{x} \in T$ whose image $\hat{y} := w\hat{x}$ attains $f(\hat{y}) = \min\{f(y) : y \in wT\}$. Then

$$\begin{aligned}\left|\left\{f(wx) : f(wx) < f(w\hat{x}),\ x \in S\right\}\right| &= \left|\left\{f(y) : f(y) < f(\hat{y}),\ y \in wS\right\}\right| \\ &= \left|\left\{f(y) : f(y) < f(\hat{y}),\ y \in wS \setminus wT\right\}\right| \\ &\leq |wS \setminus wT| \leq r(a).\end{aligned}$$

Therefore, $\hat{x}$ is an $r(a)$-best solution of the given problem. The additional claims 1, 2, and 3 now follow at once from the corresponding claims of Lemma 6.22. □

6.2.2 Exponential time to exact optimization

We now demonstrate that Theorem 6.23 is best possible in the following sense. Consider $a := (2, 3)$ which is the simplest primitive tuple with nonzero Frobenius number $\mathrm{F}(2, 3) = 1$. Consider weight vectors $w \in \{2, 3\}^n$, that is, where each ground set element has weight either 2 or 3. Then part 3 of Theorem 6.23 assures that we can produce a 1-best solution, that is, either an optimal or a second best solution, in polynomial time. We now show that, in contrast, a 0-best, that is, an optimal, solution for such weight vectors *cannot* be computed in polynomial time.

Theorem 6.24. *There is no polynomial time algorithm for solving the problem $\min\{f(wx) : x \in S\}$ with $S \subseteq \{0,1\}^n$ independence system presented by linear-optimization oracle, $f : \mathbb{Z} \to \mathbb{R}$ presented by comparison oracle, and $w \in \{2, 3\}^n$.*

In fact, the following stronger statement holds. Let $n := 4m$ with $m \geq 2$. Define $w \in \{2, 3\}^n$ by $w_j := 2$ for $j \leq 2m$ and $w_j := 3$ for $j > 2m$. Define f on $\mathbb{Z}$ by $f(5m - 1) := -1$ and $f(z) := 0$ for all $z \neq 5m - 1$. Then at least 2^m queries of the oracle presenting S are needed to determine the optimal objective function value.

Proof. Let $n := 4m$ with $m \geq 2$, $I := \{1, \ldots, 2m\}$ and $J := \{2m+1, \ldots, 4m\}$. Define weight $w \in \{2,3\}^n$ by $w_j := 2$ for $j \in I$ and $w_j := 3$ for $j \in J$, and function f on $\mathbb{Z}$ by $f(5m-1) := -1$ and $f(z) := 0$ for all $z \neq 5m-1$. For $E \subseteq \{1, \ldots, n\}$ and any nonnegative integer k, let $\binom{E}{k}$ be the set of all k-element subsets of E. For $i = 0, 1, 2$, let

$$T_i := \big\{x = \mathbf{1}_A + \mathbf{1}_B : A \in \tbinom{I}{m+i},\ B \in \tbinom{J}{m-i}\big\} \subset \{0,1\}^n.$$

Let S be the independence system generated by $T_0 \cup T_2$, that is,

$$S := \big\{z \in \{0,1\}^n : z \leq x \text{ for some } x \in T_0 \cup T_2\big\}.$$

Then the image of S under w and the minimum value of $f(wx)$ over S satisfy the following:

$$wS = \{0, \ldots, 5m\} \setminus \{1, 5m-1\}, \quad \min\big\{f(wx) : x \in S\big\} = 0.$$

For each $y \in T_1$ let $S_y := S \cup \{y\}$. Note that S_y is also an independence system. Further, the image of S_y under w and the minimum value of $f(wx)$ over S_y satisfy the following:

$$wS_y = \{0, \ldots, 5m\} \setminus \{1\}, \quad \min\big\{f(wx) : x \in S_y\big\} = -1.$$

Finally, for each vector $c \in \mathbb{Z}^n$, let

$$Y(c) := \big\{y \in T_1 : cy > \max\{cx : x \in S\}\big\}.$$

First, we claim that $|Y(c)| \leq \binom{2m}{m-1}$ for every $c \in \mathbb{Z}^n$. Consider two elements $y, z \in Y(c)$ if any. Then $y = \mathbf{1}_A + \mathbf{1}_B$ and $z = \mathbf{1}_U + \mathbf{1}_V$ for some $A, U \in \binom{I}{m+1}$ and $B, V \in \binom{J}{m-1}$. Suppose, indirectly, that $A \neq U$ and $B \neq V$. Pick $a \in A \setminus U$ and $v \in V \setminus B$. Consider the following vectors:

$$x^0 := y - \mathbf{1}_a + \mathbf{1}_v \in T_0,$$
$$x^2 := z + \mathbf{1}_a - \mathbf{1}_v \in T_2.$$

Now $y, z \in Y(c)$ and $x^0, x^2 \in S$ imply the contradiction:

$$c_a - c_v = cy - cx^0 > 0,$$
$$c_v - c_a = cz - cx^2 > 0.$$

So all vectors in $Y(c)$ are of the form $\mathbf{1}_A + \mathbf{1}_B$, with either $A \in \binom{I}{m+1}$ fixed, giving $|Y(c)| \leq \binom{2m}{m-1}$, or $B \in \binom{J}{m-1}$ fixed, giving $|Y(c)| \leq \binom{2m}{m+1} = \binom{2m}{m-1}$, as claimed.

Now, consider any algorithm, and let $c^1, \ldots, c^p \in \mathbb{Z}^n$ be the sequence of oracle queries made by the algorithm. Suppose that $p < 2^m \leq \binom{2m}{m+1}$. Then

$$\Big|\bigcup_{i=1}^{p} Y\big(c^i\big)\Big| \leq \sum_{i=1}^{p} \big|Y\big(c^i\big)\big| \leq p\binom{2m}{m-1} < \binom{2m}{m+1}\binom{2m}{m-1} = |T_1|.$$

So there is some $y \in T_1$ which is not an element of any $Y(c^i)$, that is, satisfies $c^i y \leq \max\{c^i x : x \in S\}$ for each $i = 1, \ldots, p$. So whether the linear-optimization oracle presents S or S_y, on each query c^i it can reply with some $x^i \in S$ attaining the following:

$$c^i x^i = \max\{c^i x : x \in S\} = \max\{c^i x : x \in S_y\}.$$

Therefore, the algorithm cannot tell whether the oracle is presenting S or S_y and therefore cannot tell whether the optimal objective function value is 0 or -1. □

6.3 Some applications

6.3.1 Nonlinear bipartite matching

We begin with a simple immediate application of Theorem 6.12. Consider once again the classical *assignment problem* discussed in Examples 2.4 and 2.27. The feasible set S consists of the $m \times m$ permutation matrices, which are also interpreted as the perfect matchings in the complete bipartite graph $K_{m,m}$. Nonlinear assignment problems have applications in operations research and scheduling concerning the optimal assignment of jobs to processors; see, for instance, [12] for a concrete application to find the minimum so-called *make-span* scheduling.

In Chapter 2, we have demonstrated, in Example 2.27, that the assignment problem is polynomial time solvable for convex objectives. We now obtain a result of [12], solving the problem for arbitrary objectives in randomized polynomial time.

Corollary 6.25. *For every fixed d, there is a randomized algorithm, that, given m, integer $d \times N$ matrix W with columns indexed by $N := \{(i,j) : 1 \leq i, j \leq m\}$, and arbitrary function $f : \mathbb{Z}^d \to \mathbb{R}$ presented by comparison oracle, and letting $S \subset \{0,1\}^N$ be the set of $m \times m$ permutation matrices, solves in time which is polynomial in m and W, the nonlinear assignment (or bipartite matching) problem:*

$$\min\{f(Wx) : x \in S\}.$$

Proof. We use the obvious identification $\mathbb{R}^{m \times m} \cong \mathbb{R}^N$. As in Example 2.27, let $S_1 \subset \{0,1\}^N$ be the set of $m \times m$ matrices with one 1 per row and let $S_2 \subset \{0,1\}^N$ be the set of $m \times m$ matrices with one 1 per column. Then S_1, S_2 are the sets of indicators of bases of matroids M_1, M_2 on $N := \{(i,j) : 1 \leq i, j \leq m\}$, and $S = S_1 \cap S_2$ is the set of common bases. Moreover, M_1, M_2 are the vectorial matroids of the $m \times N$ matrices A_1, A_2 which consist of the first m rows and last m rows, respectively, of the $(m+m) \times N$ incidence matrix of the complete bipartite graph $K_{m,m}$. The corollary now follows at once from Theorem 6.12. □

As noted in Example 2.4, the fiber problem for S the permutation matrices, $d = 2$, and $\{0,1\}$-valued W, includes the so-called *exact bipartite matching problem*, whose complexity is long open [12], [76]. So, an important, more general open problem is to determine conditions under which Corollary 6.25 and Theorem 6.12 could be strengthened to provide deterministic rather than randomized algorithms for the nonlinear assignment problem and nonlinear matroid intersection problem.

6.3.2 Experimental design

We now discuss an application of nonlinear matroid optimization to the area of experimental design. It concerns the choice of a model that can best describe an unknown system which is learned by experimentation, see [86] for more details. Here, we consider the commonly used class of multivariate polynomial models. So the unknown system is assumed to be a real polynomial function $y = g(x) = \sum_u g_u x^u$ of real variables $x = (x_1, \ldots, x_k)$. (See Section 6.1.1 for terminology on polynomials.)

Each finite subset $U \subset \mathbb{Z}_+^k$ provides a *model* for the system: under such model, the system is assumed to be a polynomial $y = g(x) = \sum_{u \in U} g_u x^u$ whose support $\text{supp}(g) = \{u \in \mathbb{Z}_+^k : g_u \neq 0\}$ is contained in U. Assuming model U for the system, the user conducts experiments in order to learn the system. Each experiment consists of feeding the system with some input point $x \in \mathbb{R}^k$ and measuring the corresponding output $y = g(x)$. We assume that admissible experiments are limited to points which lie in a prescribed finite set $P \subset \mathbb{R}^k$, called the *design*. Indeed, in practice, experiments at arbitrary points may be impossible or costly. Once all experiments are made, that is, each point $p \in P$ is input to the system and the corresponding output $y = g(p)$ is measured, the user tries to fit the model, that is, use the information to try and determine the coefficients g_u, $u \in U$.

A model $U = \{u_1, \ldots, u_r\} \subset \mathbb{Z}_+^k$ is *identifiable* by design $P = \{p_1, \ldots, p_r\} \subset \mathbb{R}^k$ if for all possible experiment outcomes $y_1, \ldots, y_r$ at design points $p_1, \ldots, p_r$, there is a unique polynomial $g(x) = \sum_{u \in U} g_u x^u$ with support in U which is consistent with all experiments, that is, satisfies $g(p_i) = y_i$ for $i = 1, \ldots, r$. It is not hard to verify that U is identifiable by P if and only if the $r \times r$ matrix P^U defined by the following:

$$\left(p^U\right)_{i,j} := p_i^{u_j} = \prod_{t=1}^{k} p_{i,t}^{u_{j,t}}, \quad 1 \leq i, j \leq r$$

is invertible. If U is identifiable by P then fitting the model, that is, finding the vector $g = (g_{u_1}, \ldots, g_{u_r})$ of coefficients of the polynomial model given any vector of experiment outcomes $y := (y_1, \ldots, y_r)$, is done by computing $g := (P^U)^{-1} y$.

The problem we consider here is that of choosing the best identifiable model, defined as follows. Given design $P = \{p_1, \ldots, p_r\} \subset \mathbb{R}^k$, set of potential exponents $V = \{v_1, \ldots, v_n\} \subset \mathbb{Z}_+^k$, and objective function $\mathcal{A}$ on models, the so-termed *aberration* in the statistics literature (see, e.g., [99]), the problem is to determine a model $U \subseteq V$ which is identifiable by design P and has minimum aberration $\mathcal{A}(U)$.

Broadly speaking, preferred polynomial models are those in which variables do not occur with high degrees, and the aberration is chosen accordingly.

The following example illustrates some useful aberrations.

Example 6.26. We describe two classes of aberrations $\mathcal{A}(U)$ on models $U \subset \mathbb{Z}_+^k$. Let $\deg_i(x^u) := u_i$ denote the degree of variable x_i in monomial $x^u = \prod_i x_i^{u_i}$.

1. ***Average degree***

   For each model U, consider the following vector, whose ith component is the average degree of x_i over all monomials of the model polynomial $\sum_{u \in U} g_u x^u$:

$$a(U) := \left(\frac{1}{|U|} \sum_{u \in U} \deg_1 \left(x^u\right), \ldots, \frac{1}{|U|} \sum_{u \in U} \deg_k (x^u) \right).$$

Define the aberration by $\mathcal{A}(U) := f(a(U))$ for some function $f : \mathbb{Z}^k \to \mathbb{R}$. In particular, with f the l_p-norm for some p, this aberration is $\mathcal{A}(U) = \|a(U)\|_p$. For $p = \infty$, this aberration is the maximum over variables of average degree:

$$\mathcal{A}(U) = \max\Big\{\frac{1}{|U|}\sum_{u\in U}\deg_1\left(x^u\right), \ldots, \frac{1}{|U|}\sum_{u\in U}\deg_k\left(x^u\right)\Big\}.$$

2. ***Excess degree***

   Let Δ be a positive integer serving as desired upper bound on variable degree. For each model U, consider the following vector, whose ith component is the number of monomials of $\sum_{u\in U} g_u x^u$ in which x_i has degree exceeding Δ:

$$e(U) := \left(\left|\left\{u \in U : \deg_1(x^u) > \Delta\right\}\right|, \ldots, \left|\left\{u \in U : \deg_k\left(x^u\right) > \Delta\right\}\right|\right).$$

   Define the aberration by $\mathcal{A}(U) := f(e(U))$ for some function $f : \mathbb{Z}^k \to \mathbb{R}$. In particular, with f the l_p-norm for some p, this aberration is $\mathcal{A}(U) = \|e(U)\|_p$. For $p = \infty$, this aberration is the maximum number of excesses over variables:

$$\mathcal{A}(U) = \max\left\{\left|\left\{u \in U : \deg_1\left(x^u\right) > \Delta\right\}\right|, \ldots, \left|\left\{u \in U : \deg_k\left(x^u\right) > \Delta\right\}\right|\right\}$$

   with $\mathcal{A}(U) = 0$ if and only if $\deg_i(x^u) \le \Delta$ holds for all i and all $u \in U$.

We proceed to define a broad class of abberations which includes as special cases the average degree and excess degree functions of Example 6.26 with any f.

Given design $P = \{p_1, \ldots, p_r\} \subset \mathbb{R}^k$ and set $V = \{v_1, \ldots, v_n\} \subset \mathbb{Z}_+^k$, let $S(P, V) \subseteq \{0, 1\}^n$ be the set of (indicators of) models $U \subseteq V$ identifiable by P:

$$S(P, V) := \left\{\mathbf{1}_J : J \subseteq \{1, \ldots, n\},\ U = \left\{v_j : j \in J\right\} \text{ is identifiable by } P\right\}.$$

Now, consider aberrations defined for $z \in S(P, V) \subseteq \{0, 1\}^n$ by $\mathcal{A}(z) := f(Wz)$ where, as usual, W is an integer $d \times n$ weight matrix and $f : \mathbb{Z}^d \to \mathbb{R}$ any function.

Example 6.26 (revisited). We now show that, given design $P = \{p_1, \ldots, p_r\} \subset \mathbb{R}^k$ and set $V = \{v_1, \ldots, v_n\} \subset \mathbb{Z}_+^k$ of potential exponents, the average degree and excess degree aberrations over $S(P, V)$ can be expressed as suitable $\mathcal{A}(z) = f(Wz)$.

1. ***Average degree***

   Let $d := k$ and let $W := \frac{1}{r}[v_1, \ldots, v_n]$ be $\frac{1}{r}$ times the $k \times n$ matrix whose columns are the potential exponents in V. Then it is easy to verify that, for any model $z \in S(P, V)$, the average degree vector is precisely $a(U) = Wz$, and hence the average degree aberration for any f is precisely $\mathcal{A}(z) = f(Wz)$.

2. ***Excess degree***

   Given Δ, let $d := k$ and let W be the $k \times n$ matrix defined by $W_{i,j} := 1$ if $\deg_i(x^{v_j}) > \Delta$ and $W_{i,j} := 0$ otherwise. Then it is easy to verify that, for any model $z \in S(P, V)$, the excess degree vector is precisely $e(U) = Wz$, and hence the excess degree aberration for any f is precisely $\mathcal{A}(z) = f(Wz)$.

As a corollary of Theorem 6.8 and Corollary 2.25, we obtain a result of [10] showing that a minimum aberration model can be efficiently found. We restrict attention to designs of rational points, which are processable on a Turing machine.

Corollary 6.27. *For every fixed d, there is an algorithm that, given rational design $P = \{p_1, \ldots, p_r\} \subset \mathbb{R}^k$, set $V = \{v_1, \ldots, v_n\} \subset \mathbb{Z}_+^k$ of potential exponents, integer $d \times n$ matrix W, and function $f : \mathbb{Z}^d \to \mathbb{R}$ presented by comparison oracle, solves, in time polynomial in V, $\langle P \rangle$, and W, the minimum aberration model problem:*

$$\min \big\{ \mathcal{A}(z) := f(Wz) : z \in S(P, V) \big\}.$$

Moreover, if f is concave then the running time is polynomial in V and $\langle P, W \rangle$.

Proof. Define an $r \times n$ matrix P^V by the following:

$$\big(p^V\big)_{i,j} := p_i^{v_j} = \prod_{t=1}^{k} p_{i,t}^{v_{j,t}}, \quad 1 \le i \le r, \quad 1 \le j \le n.$$

Clearly, P^V can be computed in polynomial time. Now, if the rank of P^V is less than r then the set $S(P, V) \subseteq \{0, 1\}^n$ of models U which are identifiable by P is empty. Otherwise, $S(P, V)$ is precisely the set of bases of the vectorial matroid of P^V. The corollary now follows at once from Theorem 6.8 and Corollary 2.25. □

6.3.3 Universal Gröbner bases

We conclude with an application to computational algebra. Let $\mathbb{C}[x]$ denote the algebra of multivariate complex polynomials in variables $x = (x_1, \ldots, x_d)$. The *ideal generated* by a finite system of polynomials $p_1, \ldots, p_m$ is the set $I(p_1, \ldots, p_m)$ of all linear combinations $\sum_{i=1}^m c_i p_i$ of the p_i with polynomial coefficients $c_i \in \mathbb{C}[x]$. An *ideal* is any subset $I \subseteq \mathbb{C}[x]$ of polynomials generated in such a way by some finite system. Every ideal has a finite number of special generating sets termed *reduced Gröbner bases*. The *universal Gröbner basis* of ideal I is the union $\mathcal{U}(I)$ of all its reduced Gröbner bases, and in a sense is the ultimate finite generating set of I for computational purposes. Below, we show how to compute all reduced Gröbner bases and the universal Gröbner basis efficiently, and define them in a nonstandard way on the fly. For standard definitions and further details see [95].

We restrict attention to systems of polynomials which have a finite set of common zeros, that is, vectors $x \in \mathbb{C}^d$ satisfying $p_i(x) = 0$ for $i = 1, \ldots, m$. For such systems, the universal Gröbner basis contains a univariate polynomial in each of the variables, thereby reducing the problem of computing the set of common zeros to that of finding roots of d univariate polynomials. Here is an example.

Example 6.28. Consider the system of $m = 3$ polynomials in $d = 2$ variables:

$$p_1 := x_1^2 - x_2, \quad p_2 := x_2^2 - 7x_2 + 6x_1, \quad p_3 := x_1 x_2 - 3x_2 + 2x_1.$$

The universal Gröbner basis of the ideal I of this system is a set $\mathcal{U}(I) = \{p_1, \ldots, p_7\}$ consisting of the three given polynomials along with four additional ones given by the following:

$$p_4 := x_1^3 - 3x_1^2 + 2x_1, \quad p_5 := x_2^3 - 5x_2^2 + 4x_2,$$
$$p_6 := x_1 + \frac{1}{6}x_2^2 - \frac{7}{6}x_2, \quad p_7 := x_2 - x_1^2.$$

Consider any common zero $x = (x_1, x_2)$ of the system. Then x_1 must be a root of the univariate polynomial $p_4 = p_4(x_1)$ and hence must satisfy $x_1 \in \{0, 1, 2\}$; likewise, x_2 must be a root of the univariate polynomial $p_5 = p_5(x_2)$ and hence must satisfy $x_2 \in \{0, 1, 4\}$. Checking the $9 = 3 \times 3$ potential zeros, we find that the set of common zeros of $\mathcal{U}(I)$ and hence of the original system p_1, p_2, p_3 is $\{(0,0), (1,1), (2,4)\}$, thereby solving the given system of polynomial equations.

A subset $V \subseteq \mathbb{Z}_+^d$ is *basic* for ideal I if for every polynomial $f \in \mathbb{C}[x]$ there is a unique polynomial $f_V \in \mathbb{C}[x]$ with support satisfying $\mathrm{supp}(f_V) \subseteq V$ such that $f - f_V \in I$. The polynomial f_V is called the *normal form* of f under V. The normal form satisfies $f - h \in I$ if and only if $f_V = h_V$; $f \in I$ if and only if $f_V = 0$; $f = f_V$ if and only if $\mathrm{supp}(f) \subseteq V$; $(f + h)_V = f_V + h_V$; $(fh)_V = (f_V h_V)_V$. All basic sets have the same cardinality which is called the *length* of the ideal I, and is finite if and only if the set of common zeros of I is finite. If I has finite length n then n equals the number of common zeros with suitable multiplicities, and in particular is an upper bound on the number of distinct common zeros of I. The set of ideals of length n in d variables is denoted Hilb_n^d and referred to as the *Hilbert scheme*, admitting an embedding into a suitable Grassmanian, see [6].

Example 6.28 (continued). The ideal $I = I(p_1, p_2, p_3)$ in $d = 2$ variables of Example 6.28 has length $n = 3$ and 3 common zeros. The set $V := \{00, 10, 20\}$ is basic for I. We use here abridged notation such as $00 := (0, 0)$ for vectors in $\mathbb{Z}_+^d$. The following table shows a few monomials f with their normal forms under V.

f	1	x_1	x_1^2	x_1^3	x_2
f_V	1	x_1	x_1^2	$3x_1^2 - 2x_1$	x_1^2

(6.14)

f	x_1x_2	$x_1^2x_2$	x_2^2	$x_1x_2^2$	x_2^3
f_V	$3x_1^2 - 2x_1$	$7x_1^2 - 6x_1$	$7x_1^2 - 6x_1$	$15x_1^2 - 14x_1$	$31x_1^2 - 30x_1$

From here on, we restrict attention to ideals of finite length only. Let

$$V_n^d := \{v \in \mathbb{Z}_+^d : \prod_{i=1}^{d} (v_i + 1) \leq n\}.$$

For any finite subset $V \subset \mathbb{Z}_+^d$, let $\sum V := \sum_{u \in V} u$ be the sum of the vectors in V. The following fundamental definition from [6] plays a central role in the sequel.

Definition 6.29. The *basis polytope* of an ideal I of length n in d variables is as follows:

$$\mathcal{P}(I) := \text{conv}\left\{\sum V : V \subseteq V_n^d,\ V \text{ is basic for } I\right\} \subset \mathbb{R}^d.$$

We now explain the role of the polytope $\mathcal{P}(I)$ in constructing the universal Gröbner basis $\mathcal{U}(I)$. Let $\underline{\text{vert}}(\mathcal{P}(I))$ be the set of vertices on the lower envelope of $\mathcal{P}(I)$, namely, those v for which there exists some $h \in \mathbb{R}_+^d$ with strictly positive entries which is minimized over $\mathcal{P}(I)$ uniquely at v. For $V \subseteq \mathbb{Z}_+^d$, let $\min(\bar{V}) \subset \bar{V} := \mathbb{Z}_+^d \setminus V$ be the set of $\leq$-minimal vectors not in V, namely, those $u \in \mathbb{Z}_+^d \setminus V$ for which there is no other $v \in \mathbb{Z}_+^d \setminus V$ with $v \leq u$. Note that $\min(\bar{V})$ is always finite by the lemma of Gordan [40]. The following facts about any ideal of length n in d variables were established in [6] (extending results of [85] on generic radical ideals):

1. every $v \in \underline{\text{vert}}(\mathcal{P}(I))$ satisfies $v = \sum V$ for a *unique* $V \subseteq V_n^d$ basic for I;
2. the reduced Gröbner bases of I are in bijection with $\underline{\text{vert}}(\mathcal{P}(I))$. Moreover, the reduced Gröbner basis corresponding to $v = \sum V$ is given by the following:

$$G_v = G_V = \left\{x^u - x_V^u : u \in \min(\bar{V})\right\}.$$

We proceed to exploit these facts for constructing the universal Gröbner basis. Let

$$U_n^d := V_n^d \cup \left\{v + \mathbf{1}_i : v \in V_n^d,\ 1 \leq i \leq d\right\} \subseteq V_{2n}^d.$$

This set is sufficiently large so that $\min(\bar{V}) \subseteq U_n^d$ for every n-element $V \subseteq V_n^d$. Yet, it is known that for any fixed d, the cardinality of V_n^d and hence also of U_n^d are $O(n(\log n)^{d-1})$, and so are polynomial in n, see [6], [85] and references therein.

A *basic presentation* of ideal I of length n in d variables consists of $V \subseteq V_n^d$ which is basic for I, along with the set $\{x_V^u : u \in U_n^d\}$. It is known that such a presentation can be computed efficiently from any given generating set of I by computing the so-called degree reverse lexicographic Gröbner basis of I, see [35] and the references therein. So we assume that the ideal is presented in this way. We also assume that the polynomials in $\{x_V^u : u \in U_n^d\}$ have rational coefficients and so are processable by a Turing machine. The *binary length* of such a presentation is the sum of binary lengths $\langle \|x_V^u\|_\infty \rangle$ of the given polynomials over all $u \in U_n^d$ (where, as in Section 6.1.1, the l_∞-norm $\|g\|_\infty$ of a polynomial g is the maximum absolute value of any coefficient of g).

Example 6.28 (continued). Consider again the ideal $I = I(p_1, p_2, p_3)$ of length $n = 3$ in $d = 2$ variables of Example 6.28. Then $V_3^2 = \{00, 10, 20, 01, 02\}$ and

$$U_3^2 = \{00, 10, 20, 30, 01, 11, 21, 02, 12, 03\}.$$

Therefore, Table (6.14) provides precisely all normal forms x_V^u under the basic set $V = \{00, 10, 20\}$ for all $u \in U_3^2$. So V and this table is a basic presentation of I.

Our results on matroids imply the following result of [6] which extends [85].

Corollary 6.30. *For every fixed d, there is an algorithm that, given rational basic presented ideal $I \subseteq \mathbb{C}[x]$ of length n in d variables, computes the universal Gröbner basis $\mathcal{U}(I)$ of I in time polynomial in the binary length of the presentation of I.*

Proof. Given the presentation of I construct a $V \times U_n^d$ matrix B, that is, with rows indexed by the given basic set V and columns indexed by U_n^d, by letting, for $u \in U_n^d$ and $v \in V$, the entry $B_{v,u}$ be the coefficient of x^v in the polynomial x_V^u. So the polynomial x_V^u is encoded by column u as $x_V^u = \sum_{v \in V} B_{v,u} x^v$. Let A be the $V \times V_n^d$ submatrix of B consisting of the columns corresponding to $V_n^d \subseteq U_n^d$.

Now, a subset $U \subseteq U_n^d$ is basic for I if and only if the corresponding $V \times U$ submatrix of B is invertible. Let $N := |V_n^d| = O(n(\log n)^{d-1})$ and choose some total order $V_n^d = \{v_1, \ldots, v_N\}$ of the points of V_n^d. Let $S \subseteq \{0,1\}^N$ be the set of bases of the vectorial matroid of A. Let $W := [v_1, \ldots, v_N]$ be the $d \times N$ matrix whose columns are the vectors in V_n^d. Then the basis polytope of I satisfies the following:

$$\mathcal{P}(I) = \operatorname{conv}\Big\{ \sum U : U \subseteq V_n^d,\ U \text{ is basic for } I \Big\} = \operatorname{conv}(WS).$$

Now, compute $Y := \underline{\operatorname{vert}}(\mathcal{P}(I)) = \underline{\operatorname{vert}}(\operatorname{conv}(WS))$ and $Z \subseteq S$ such that $Y = WZ$, in one of several ways using the results of this chapter or Chapter 2, as follows. One possibility is to compute the entire image $WS \subset \mathbb{Z}_+^d$ by the algorithm of Lemma 6.7 or the set $\operatorname{vert}(\operatorname{conv}(WS))$ by the algorithm of Lemma 2.8 (with the greedy algorithm providing a linear-optimization oracle for S, see Section 1.2.1), identify its subset Y by solving a suitable linear program for each of its points, and then obtain Z by finding a feasible point $z \in W^{-1}(y) \cap S$ in the fiber of each $y \in Y$ using the algorithm of Lemma 2.9. This can all be done in time polynomial in $\langle A \rangle$ and W and hence polynomial in the binary length of the presentation of I. Another, faster, possibility, in time polynomial in $\langle A \rangle$ and $\langle W \rangle$, is as follows. Let

$$E := \big\{ W^i - W^j : 1 \le i < j \le N \big\} = W\big\{ \mathbf{1}_i - \mathbf{1}_j : 1 \le i < j \le N \big\} \subset \mathbb{Z}^d.$$

Then E is a set of all edge directions of $\operatorname{conv}(WS)$ (see proof of Corollary 2.25 and Lemma 2.14). Now, apply the algorithm of Lemma 2.15 and compute a set $T \subseteq S$ with $\operatorname{vert}(\operatorname{conv}(WS)) \subseteq WT$. During the algorithm (see proof of Lemma 2.15), each point $t \in T$ comes with $h \in \mathbb{Z}^d$ maximized over WS at Wt; inspection of these t and h then allows to directly distill $Z \subseteq T$ with $\underline{\operatorname{vert}}(\operatorname{conv}(WS)) = WZ$.

Let $Z \subseteq S$ be the set with $\underline{\operatorname{vert}}(\mathcal{P}(I)) = WZ$ computed as above. For each $z \in Z$, do the following. Let $U := \{v_j : z_j = 1\}$ and $v := \sum U = Wz$ be the basic subset of V_n^d and vertex in $\underline{\operatorname{vert}}(\mathcal{P}(I))$ corresponding to basis z. Now, apply suitable row operations to the matrix B so as to make U the new basis, that is, transform B into a $U \times U_n^d$ matrix B_U whose rows are indexed by U and whose submatrix consisting of columns corresponding to U is the identity. Then, for each $u \in U_n^d$, the normal form x_U^u can be read off directly from the column u of B_U by the following:

$$x_U^u = \sum_{w \in U} \big(B_U\big)_{w,u} x^w.$$

Since $\min(\bar{U}) \subseteq U_n^d$, we can read off from B_U the corresponding reduced Gröbner basis $G_U := \{x^u - x_U^u : u \in \min(\bar{U})\}$. Repeating this for every $z \in Z$, we obtain all reduced Gröbner bases. Their union is the universal Gröbner basis $\mathcal{U}(I)$ of I. □

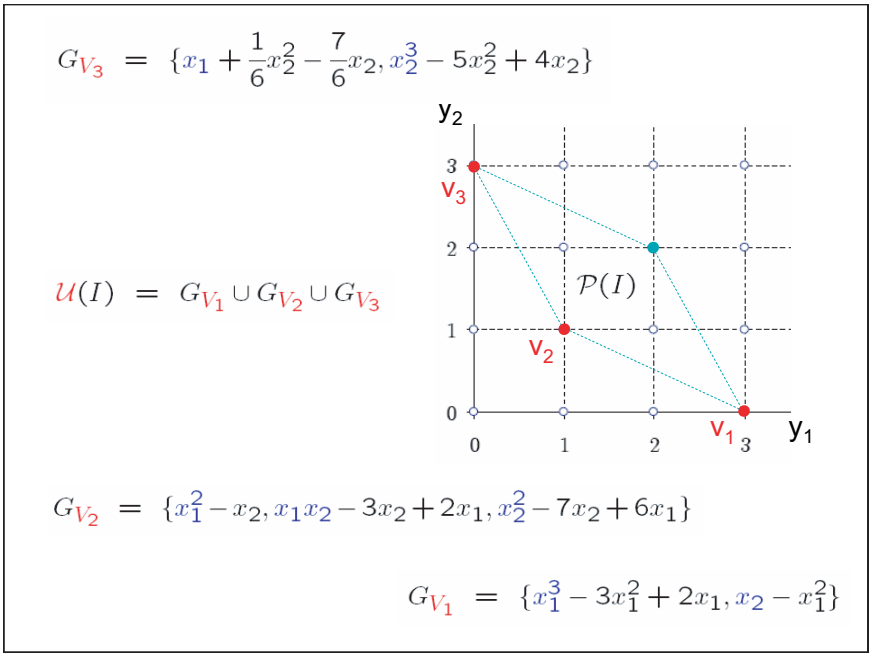


Figure 6.1: Universal Gröbner basis example

Example 6.28 (continued). We now illustrate the algorithm of Corollary 6.30 on the ideal I of length $n = 3$ in $d = 2$ variables in our running Example 6.28 given by the basic presentation with basic set $V_1 := V = \{00, 10, 20\}$ and set $\{x_V^u : u \in U_3^2\}$ of polynomials in table (6.14). Given the presentation, we construct the matrix B_{V_1}:

$$B_{V_1} := \begin{array}{c} \\ 00 \\ 10 \\ 20 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 00 & 10 & 20 & 30 & 01 & 11 & 21 & 02 & 12 & 03 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -2 & 0 & -2 & -6 & -6 & -14 & -30 \\ 0 & 0 & 1 & 3 & 1 & 3 & 7 & 7 & 15 & 31 \end{pmatrix} \end{array}.$$

The matrices A and W having columns indexed by the $N := 5$ elements in V_3^2 are as follows:

$$A := \begin{array}{c} \begin{array}{ccccc} 00 & 10 & 20 & 01 & 02 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -6 \\ 0 & 0 & 1 & 1 & 7 \end{pmatrix} \end{array}, \quad W := \begin{array}{c} \begin{array}{ccccc} 00 & 10 & 20 & 01 & 02 \end{array} \\ \begin{pmatrix} 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 \end{pmatrix} \end{array}.$$

In this small example, we can write explicitly the set S of bases of the vectorial matroid of A, the corresponding subsets of V_3^2 basic for I, and the image WS as follows:

$$S = \{11100, 11010, 11001, 10101, 10011\} \subset \{0,1\}^5$$
$$\{00, 10, 20\},\ \{00, 10, 01\},\ \{00, 10, 02\},\ \{00, 20, 02\},\ \{00, 01, 02\}\ \subset\ V_3^2$$
$$WS = \{30, 11, 12, 22, 03\} \subset \mathbb{Z}_+^2.$$

The polytope $\mathcal{P}(I) = \operatorname{conv}(WS)$ is shown in Figure 6.1. The set $Z \subseteq S$, corresponding subsets of V_3^2 basic for I, and lower vertex set $\underline{\operatorname{vert}}(\mathcal{P}(I)) = WZ$ are as follows:

$$Z = \{11100, 11010, 10011\}$$
$$V_1 = \{00, 10, 20\}, \quad V_2 = \{00, 10, 01\}, \quad V_3 = \{00, 01, 02\}$$
$$\underline{\operatorname{vert}}\big(\mathcal{P}(I)\big) = \underline{\operatorname{vert}}\big(\operatorname{conv}(WS)\big) = WZ = \{30, 11, 03\}.$$

We now compute the three corresponding reduced Gröbner bases. For the given basic set V_1, we have $\min(\bar{V}_1) = \{30, 01\}$ and the reduced Gröbner basis is part of the presentation of I and is read off from columns 30 and 01 of B_{V_1} to be as follows:

$$G_{V_1} = \big\{x_1^3 - 3x_1^2 + 2x_1,\, x_2 - x_1^2\big\} = \big\{p_4, p_7\big\}.$$

For $V_2 = \{00, 10, 01\}$, we make the basis change to obtain the matrix B_{V_2}:

$$B_{V_2} = \begin{array}{c} \\ 00 \\ 10 \\ 01 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 00 & 10 & 20 & 30 & 01 & 11 & 21 & 02 & 12 & 03 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -2 & 0 & -2 & -6 & -6 & -14 & -30 \\ 0 & 0 & 1 & 3 & 1 & 3 & 7 & 7 & 15 & 31 \end{pmatrix} \end{array}.$$

The matrix happens here to remain unchanged expect that its rows are now indexed by V_2. For V_2, we have $\min(\bar{V}_2) = \{20, 11, 02\}$ and the reduced Gröbner basis is read off from columns 20, 11, and 02 of B_{V_2} to be as follows:

$$G_{V_2} = \big\{x_1^2 - x_2,\, x_1x_2 - 3x_2 + 2x_1,\, x_2^2 - 7x_2 + 6x_1\big\} = \big\{p_1, p_2, p_3\big\}.$$

Finally, for $V_3 = \{00, 01, 02\}$, we make the basis change to obtain the matrix B_{V_3}:

$$B_{V_3} = \begin{array}{c} \\ 00 \\ 01 \\ 02 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 00 & 10 & 20 & 30 & 01 & 11 & 21 & 02 & 12 & 03 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7/6 & 1 & 2/3 & 1 & 2/3 & 0 & 0 & -4/3 & -4 \\ 0 & -1/6 & 0 & 1/3 & 0 & 1/3 & 1 & 1 & 7/3 & 5 \end{pmatrix} \end{array}.$$

For V_3, we have $\min(\bar{V}_3) = \{10, 03\}$ and the reduced Gröbner basis is read off from columns 10 and 03 of B_{V_3} to be as follows:

$$G_{V_3} = \big\{x_1 + \frac{1}{6}x_2^2 - \frac{7}{6}x_2,\, x_2^3 - 5x_2^2 + 4x_2\big\} = \big\{p_5, p_6\big\}.$$

The universal Gröbner basis of I is now obtained as the union of all reduced Gröbner bases and is found to be the one provided in the beginning of this section:

$$\mathcal{U}(I) = G_{V_1} \cup G_{V_2} \cup G_{V_3} = \big\{p_1, \ldots, p_7\big\}.$$

Notes

The algorithm for nonlinear matroid optimization and the applications to experimental design in Section 6.3.2 are from [10]; more information on experimental design and on the emerging area of algebraic statistics can be found in [86]. The randomized algorithm for nonlinear matroid intersections in Section 6.1.3 is from [11] extending [12] and has some of its origins in [71]. The understanding of the deterministic time complexity of the fiber problem and the nonlinear optimization problem over various combinatorial families is very limited yet. Of particular interest is the deterministic time complexity of nonlinear matroid intersections, which includes the long-open exact matching problem of [76] as a special case. The approximation providing an r-best solution for nonlinear optimization over independence systems in Section 6.2.1 is from [67] and is of an unusual character. The vast literature on approximation algorithms usually seeks feasible solutions whose objective value is bounded by a constant multiple of the optimal objective function value; more information on this line of study can be found in [96]. It is also unusual and quite remarkable that, as shown in Theorem 6.24, the running time needed to solve the problem to optimality is actually exponential. The results in Section 6.3.3 on universal Gröbner bases of zero dimensional ideals are from [6], extending results of [85] on generic radical ideals. More information on this can be found in these papers, the related paper [35], the book [95] by Sturmfels, and the references therein.

Bibliography

[1] K. Allemand, K. Fukuda, T. M. Liebling, and E. Steiner, A polynomial case of unconstrained zero-one quadratic optimization, *Math. Program.* **91** (2001), 49–52. 33

[2] N. Alon and S. Onn, Separable partitions, *Discrete Appl. Math.* **91** (1999), 39–51. 24

[3] S. Aoki and A. Takemura, Minimal basis for a connected Markov chain over $3 \times 3 \times K$ contingency tables with fixed two-dimensional marginals, *Aust. N. Z. J. Stat.* **45** (2003), 229–249. 73, 74

[4] M. Aschenbrenner and R. Hemmecke, Finiteness theorems in stochastic integer programming, *Found. Comput. Math.* **7** (2007), 183–227. 70

[5] D. Avis and K. Fukuda, A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra, *Discrete Comput. Geom.* **8** (1992), 295–313. 39

[6] E. Babson, S. Onn, and R. Thomas, The Hilbert zonotope and a polynomial time algorithm for universal Gröbner bases, *Adv. in Appl. Math.* **30** (2003), 529–544. 6, 123, 124, 128

[7] M. L. Balinski and F. J. Rispoli, Signature classes of transportation polytopes, *Math. Program.* **60** (1993), 127–144. 74

[8] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar, Privacy, accuracy, and consistency too: a holistic solution to contingency table release, in *Proceedings of the 26th Symposium on Principles of Database Systems* (Beijing, 2007), Association for Computing Machinery, 2007, pp. 273–282. 74, 96

[9] A. Barvinok, *Integer Points in Polyhedra*, Zurich Lectures in Advanced Mathematics, European Mathematical Society, 2008. 13

[10] Y. Berstein, J. Lee, H. Maruri-Aguilar, S. Onn, E. Riccomagno, R. Weismantel, and H. Wynn, Nonlinear matroid optimization and experimental design, *SIAM J. Discrete Math.* **22** (2008), 901–919. 6, 102, 122, 128

[11] Y. Berstein, J. Lee, S. Onn, and R. Weismantel, Parametric nonlinear discrete optimization over well-described sets and matroid intersections, *Math. Program.*, to appear. 22, 35, 39, 106, 128

[12] Y. Berstein and S. Onn, Nonlinear bipartite matching, *Discrete Optim.* **5** (2008), 53–65. 17, 22, 36, 119, 128

[13] ——, The Graver complexity of integer programming, *Ann. Comb.* **13** (2009), 289–296. 95, 96

[14] D. Bertsimas and R. Weismantel, *Optimization over Integers*, Dynamic Ideas, 2005. 13

[15] L. J. Billera and A. Sarangarajan, All 0-1 polytopes are traveling salesman polytopes, *Combinatorica* **16** (1996), 175–188. 96

[16] A. Brauer, On a problem of partitions, *Amer. J. Math.* **64** (1942), 299–312. 109

[17] W. Bruns, J. Gubeladze, M. Henk, A. Martin, and R. Weismantel, A counterexample to an integer analogue of Carathéodory's theorem, *J. Reine Angew. Math.* **510** (1999), 179–185. 42

[18] V. Chvátal, Edmonds polytopes and a hierarchy of combinatorial problems, *Discrete Math.* **4** (1973), 305–337. 13

[19] M. Conforti and G. Cornuéjols, A class of logic problems solvable by linear programming, *J. Assoc. Comput. Mach.* **42** (1995), 1107–1113. 39

[20] W. Cook, J. Fonlupt, and A. Schrijver, An integer analogue of Carathéodory's theorem, *J. Combin. Theory Ser. B* **40** (1986), 63–70. 42

[21] G. Cornuéjols, Valid inequalities for mixed integer linear programs, *Math. Program.* **112** (2008), 3–44. 13

[22] L. H. Cox, On properties of multi-dimensional statistical tables, *J. Statist. Plann. Inference* **117** (2003), 251–273. 74, 96

[23] J. De Loera, R. Hemmecke, S. Onn, U. G. Rothblum, and R. Weismantel, Convex integer maximization via Graver bases, *J. Pure Appl. Algebra* **213** (2009), 1569–1577. 27, 39, 40, 42, 50, 53, 54, 58, 73

[24] J. De Loera, R. Hemmecke, S. Onn, and R. Weismantel, n-fold integer programming, *Discrete Optim.* **5** (2008), 231–241. 40, 42, 48, 53, 54, 56, 58, 73, 81

[25] J. De Loera, E. D. Kim, S. Onn, and F. Santos, Graphs of transportation polytopes, *J. Combin. Theory Ser. A* **116** (2009), 1306–1325. 96

[26] J. De Loera and S. Onn, The complexity of three-way statistical tables, *SIAM J. Comput.* **33** (2004), 819–836. 76, 79, 80

[27] ——, All linear and integer programs are slim 3-way transportation programs, *SIAM J. Optim.* **17** (2006), 806–821. 54, 64, 76, 79, 80

[28] ——, Markov bases of three-way tables are arbitrarily complicated, *J. Symbolic Comput.* **41** (2006), 173–181. 76, 83

[29] A. Dobra, S. E. Fienberg, A. Rinaldo, A. Slavković, and Y. Zhou, Algebraic statistics and contingency table problems: log-linear models, likelihood estimation, and disclosure limitation, in *Emerging Applications of Algebraic Geometry*, IMA Vol. Math. Appl. 149, Springer, 2009, pp. 63–88. 74

[30] H. Edelsbrunner, J. O'Rourke, and R. Seidel, Constructing arrangements of lines and hyperplanes with applications, *SIAM J. Comput.* **15** (1986), 341–363. 24

[31] H. Edelsbrunner, R. Seidel, and M. Sharir, On the zone theorem for hyperplane arrangements, *SIAM J. Comput.* **22** (1993), 418–429. 24

[32] J. Edmonds, Matroids and the greedy algorithm, *Math. Program.* **1** (1971), 127–136. 5

[33] J. Edmonds, Matroid intersection, *Ann. Discrete Math.* **4** (1979), 39–49. 35, 36, 106

[34] J. Edmonds and R. M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, *J. Assoc. Comput. Mach.* **19** (1972), 248–264. 31

[35] J. C. Faugère, P. Gianni, D. Lazard, and T. Mora, Efficient computation of zero-dimensional Gröbner bases by change of ordering, *J. Symbolic Comput.* **16** (1993), 329–344. 124, 128

[36] S. E. Fienberg and A. Rinaldo, Three centuries of categorical data analysis: Log-linear models and maximum likelihood estimation, *J. Statist. Plann. Inference* **137** (2007), 3430–3445. 74, 96

[37] S. Fujishige, *Submodular Functions and Optimization*, Annals of Discrete Mathematics 58, Elsevier, 2nd edition, 2005. 39

[38] K. Fukuda, From the zonotope construction to the Minkowski addition of convex polytopes, *J. Symbolic Comput.* **38** (2004), 1261–1272. 24

[39] K. Fukuda, S. Onn, and V. Rosta, An adaptive algorithm for vector partitioning, *J. Global Optim.* **25** (2003), 305–319. 37, 39

[40] P. Gordan, Über die Auflösung linearer Gleichungen mit reellen Coefficienten, *Math. Ann.* **6** (1873), 23–28. 40, 67, 124

[41] J. E. Graver, On the foundations of linear and integer linear programming. I, *Math. Program.* **9** (1975), 207–226. 40, 41, 53

[42] P. Gritzmann and B. Sturmfels, Minkowski addition of polytopes: computational complexity and applications to Gröbner bases, *SIAM J. Discrete Math.* **6** (1993), 246–269. 24

[43] M. Grötschel and L. Lovász, Combinatorial optimization, in *Handbook of Combinatorics*, Elsevier, 1995, PP. 1541–1597. 31

[44] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, 2nd edition, 1993. 12, 13, 19, 21, 39

[45] B. Grünbaum, *Convex Polytopes*, Springer-Verlag, 2nd edition, 2003. 23

[46] P. L. Hammer, P. Hansen, P. M. Pardalos, and D. J. Rader, Jr., Maximizing the product of two linear functions in 0-1 variables, *Optimization* **51** (2002), 511–537. 33

[47] E. F. Harding, The number of partitions of a set of N points in k dimensions induced by hyperplanes, *Proc. Edinburgh Math. Soc.* **15** (1967), 285–289. 24

[48] R. Hassin and A. Tamir, Maximizing classes of two-parameter objectives over matroids, *Math. Oper. Res.* **14** (1989), 362–375. 34, 39

[49] R. Hemmecke, M. Köppe, J. Lee, and R. Weismantel, Nonlinear integer programming, in *50 Years of Integer Programming 1958–2008: The Early Years and State-of-the-Art Surveys* (M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, eds.), Springer, to appear. 13, 53

[50] R. Hemmecke, S. Onn, and R. Weismantel, A polynomial oracle-time algorithm for convex integer minimization, *Math. Program.*, to appear. 40, 42, 48, 53, 54, 58, 67, 69, 70, 73

[51] ——, n-fold integer programming and nonlinear multi-transshipment, submitted. 40, 42, 54, 58, 62, 65, 73

[52] R. Hemmecke and R. Schultz, Decomposition of test sets in stochastic integer programming, *Math. Program.* **94** (2003), 323–341. 67, 73

[53] D. S. Hochbaum and J. G. Shanthikumar, Convex separable optimization is not much harder than linear optimization, *J. Assoc. Comput. Mach.* **37** (1990), 843–862. 29

[54] A. J. Hoffman and J. B. Kruskal, Integral boundary points of convex polyhedra, in *Linear Inequalities and Related Systems*, Ann. Math. Stud. 38, Princeton University Press, 1956, pp. 223–246. 8, 29, 54

[55] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979. 11

[56] S. Hoşten and S. Sullivant, Finiteness theorems for Markov bases of hierarchical models, *J. Combin. Theory Ser. A* **114** (2007), 311-321. 55, 73

[57] F. K. Hwang, S. Onn, and U. G. Rothblum, A polynomial time algorithm for shaped partition problems, *SIAM J. Optim.* **10** (1999), 70–81. 24, 25, 37, 39

[58] R. Irving and M. R. Jerrum, Three-dimensional statistical data security problems, *SIAM J. Comput.* **23** (1994), 170–184. 74, 80, 83

[59] L. G. Khachiyan, A polynomial algorithm in linear programming, *Sov. Math. Dokl.* **20** (1979), 191–194. 8, 19, 30, 39, 54

[60] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, and V. Gurvich, Generating all vertices of a polyhedron is hard, *Discrete Comput. Geom.* **39** (2008), 174–190. 16, 39

[61] E. D. Kim and F. Santos, An update on the Hirsch conjecture, preprint, arXiv:0907.1186 (2009). 96

[62] V. Klee and C. Witzgall, Facets and vertices of transportation polytopes, in *Mathematics of the Decision Sciences, Part I*, American Mathematical Society, 1968, pp. 257–282. 74

[63] P. Kleinschmidt, C. W. Lee, and H. Schannath, Transportation problems which can be solved by the use of Hirsch-paths for the dual problems, *Math. Program.* **37** (1987), 153–168. 74

[64] D. C. Kozen, *Theory of Computation*, Springer-Verlag, 2006. 11

[65] M. Laurent and F. Rendl, Semidefinite programming and integer programming, in *Handbook on Discrete Optimization* (K. Aardal, G. Nemhauser, and R. Weismantel, eds.), Elsevier, 2005, pp. 393–514. 13

[66] J. Lee, S. Onn, L. Romanchuk, and R. Weismantel, The quadratic Graver cone, quadratic integer minimization, and extensions, submitted. 53

[67] J. Lee, S. Onn, and R. Weismantel, Approximate nonlinear optimization over weighted independence systems, *SIAM J. Discrete Math.* **23** (2009), 1667–1681. 110, 117, 128

[68] T. Leighton and S. Rao, Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms, *J. Assoc. Comput. Mach.* **46** (1999), 787–832. 73

[69] H. W. Lenstra, Jr., Integer programming with a fixed number of variables, *Math. Oper. Res.* **8** (1983), 538–548. 54, 80

[70] F. V. Louveaux and R. Schultz, Stochastic integer programming, in *Stochastic Programming*, Handbooks Oper. Res. Management Sci. 10, Elsevier, 2003, pp. 213-266. 65, 73

[71] L. Lovász, On determinants, matchings, and random algorithms, in *Fundamentals of Computation Theory*, Math. Res. 2, Akademie-Verlag, 1979, pp. 565–574. 128

[72] ——, *An Algorithmic Theory of Numbers, Graphs, and Convexity*, CBMS-NSF Regional Conference Series in Applied Mathematics 50, Society for Industrial and Applied Mathematics (SIAM), 1986. 12, 13, 19, 21, 39

[73] L. Lovász and A. Schrijver, Cones of matrices and set-functions and 0-1 optimization, *SIAM J. Optim.* **1** (1991), 166–190. 13

[74] D. Maclagan, Antichains of monomial ideals are finite, *Proc. Amer. Math. Soc.* **129** (2001), 1609–1615. 67

[75] T. S. Motzkin, The multi-index transportation problem, *Bull. Amer. Math. Soc.* **58** (1952), 494. 8, 74, 80, 96

[76] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani, Matching is as easy as matrix inversion, *Combinatorica* **7** (1987), 105–113. 17, 119, 128

[77] K. Murota, *Discrete Convex Analysis*, SIAM Monographs on Discrete Mathematics and Applications, Society for Industrial and Applied Mathematics (SIAM), 2003. 39

[78] K. Murota, H. Saito, and R. Weismantel, Optimality criterion for a class of nonlinear integer programs, *Oper. Res. Lett.* **32** (2004), 468–472. 44

[79] A. Nemirovskii, S. Onn, and U. G. Rothblum, Accuracy certificates for computational problems with convex structure, *Math. Oper. Res.* **35** (2010), 52–78. 20, 39

[80] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*, SIAM Studies in Applied Mathematics 13, Society for Industrial and Applied Mathematics, 1994. 72

[81] S. Onn, Convex matroid optimization, *SIAM J. Discrete Math.* **17** (2003), 249–253. 34, 39

[82] ——, Entry uniqueness in margined tables, in Proc. PSD 2006 – Symp. on Privacy in Statistical Databses (Rome, Italy), Lec. Not. Comp. Sci. 4302, Springer, 2006, pp. 94–101. 83, 84, 85

[83] S. Onn and U. G. Rothblum, Convex combinatorial optimization, *Discrete Comput. Geom.* **32** (2004), 549–566. 27, 33, 37, 39

[84] S. Onn and L. J. Schulman, The vector partition problem for convex objective functions, *Math. Oper. Res.* **26** (2001), 583–590. 37, 39

[85] S. Onn and B. Sturmfels, Cutting corners, *Adv. in Appl. Math.* **23** (1999), 29–48. 6, 124, 128

[86] G. Pistone, E. M. Riccomagno, and H. P. Wynn, *Algebraic Statistics*, Monographs on Statistics and Applied Probability 89, Chapman & Hall, 2001. 120, 128

[87] J. Renegar, A polynomial-time algorithm, based on Newton's method, for linear programming, *Math. Program.* **40** (1988), 59–93. 8, 19

[88] R. W. Rosenthal, A class of games possessing pure-strategy Nash equilibria, *Internat. J. Game Theory* **2** (1973), 65–67. 10, 62

[89] F. Santos and B. Sturmfels, Higher Lawrence configurations, *J. Combin. Theory Ser. A* **103** (2003), 151–164. 55, 73

[90] A. Schrijver, *Theory of Linear and Integer Programming*, Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons, 1986. 13, 47, 70, 80

[91] ——, *Combinatorial Optimization*, Springer-Verlag, 2003. 13

[92] A. Schulz, R. Weismantel, and G. M. Ziegler, $0/1$-integer programming: optimization and augmentation are equivalent, in *Proceedings of the Third Annual European Symposium on Algorithms*, Lecture Notes in Comput. Sci. 979, Springer, 1995, pp. 473–483. 31

[93] J. T. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, *J. Assoc. Comput. Mach.* **27** (1980), 701–717. 100

[94] A. Sebö, Hilbert bases, Carathéodory's theorem and combinatorial optimization, in *Proceedings of the 1st Integer Programming and Combinatorial Optimization Conference*, University of Waterloo Press, 1990, pp. 431–455. 42

[95] B. Sturmfels, *Gröbner Bases and Convex Polytopes*, University Lecture Series 8, American Mathematical Society, 1996. 13, 51, 52, 53, 122, 128

[96] V. V. Vazirani, *Approximation Algorithms*, Springer-Verlag, 2001. 128

[97] M. Vlach, Conditions for the existence of solutions of the three-dimensional planar transportation problem, *Discrete Appl. Math.* **13** (1986), 61–78. 74, 81

[98] D. J. A. Welsh, *Matroid Theory*, Academic Press, 1976. 5

[99] H. Wu and C. F. J. Wu, Clear two-factor interactions and minimum aberration, *Ann. Statist.* **30** (2002), 1496–1511. 120

[100] V. A. Yemelichev, M. M. Kovalëv, and M. K. Kravtsov, *Polytopes, Graphs and Optimisation*, Cambridge University Press, 1984. 74

[101] D. B. Yudin, and A. S. Nemirovskii, Informational complexity and effective methods for the solution of convex extremal problems, *Matekon* **13** (1977), 25–45. 19, 39

[102] T. Zaslavsky, Facing up to arrangements: face count formulas for partitions of space by hyperplanes, *Mem. Amer. Math. Soc.* **1** (1975). 24

[103] G. M. Ziegler, *Lectures on Polytopes*, Graduate Texts in Mathematics 152, Springer-Verlag, 1995. 23

Index

ZURICH LECTURES IN ADVANCED MATHEMATICS

Shmuel Onn

Nonlinear Discrete Optimization

An Algorithmic Theory

This monograph develops an algorithmic theory of nonlinear discrete optimization. It introduces a simple and useful setup which enables the polynomial time solution of broad fundamental classes of nonlinear combinatorial optimization and integer programming problems in variable dimension. An important part of this theory is enhanced by recent developments in the algebra of Graver bases. The power of the theory is demonstrated by deriving the first polynomial time algorithms in a variety of application areas within operations research and statistics, including vector partitioning, matroid optimization, experimental design, multicommodity flows, multi-index transportation and privacy in statistical databases.

The monograph is intended for graduate students and researchers. It is accessible to anyone with standard undergraduate knowledge and mathematical maturity.

ISBN 978-3-03719-093-7

www.ems-ph.org