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.

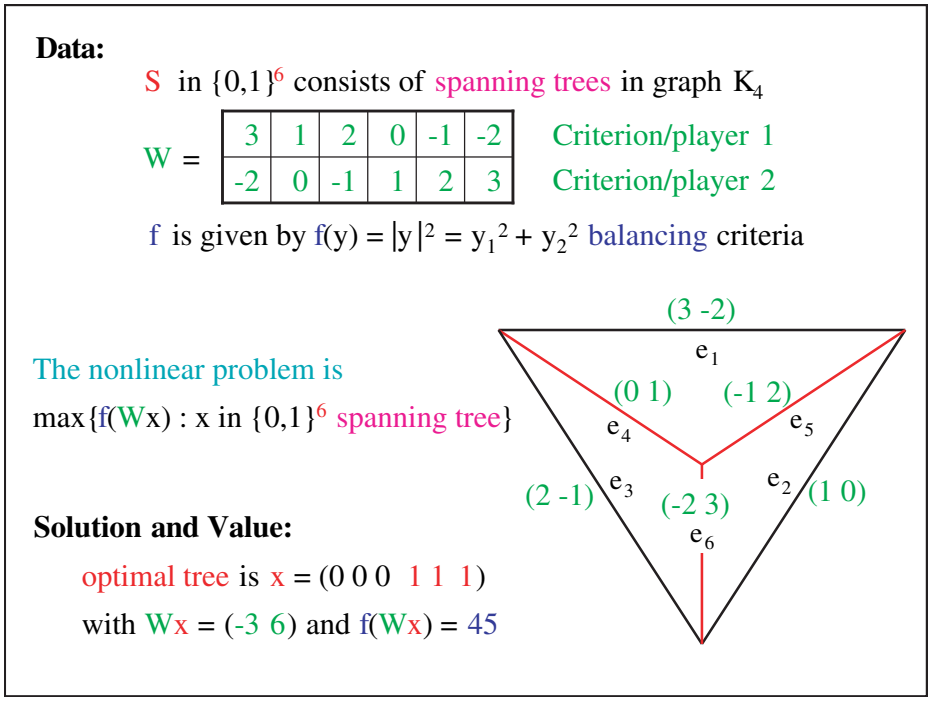

Figure 1.1: Maximum norm spanning tree example

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 $x = (x_{i_1,\ldots,i_d})$ 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

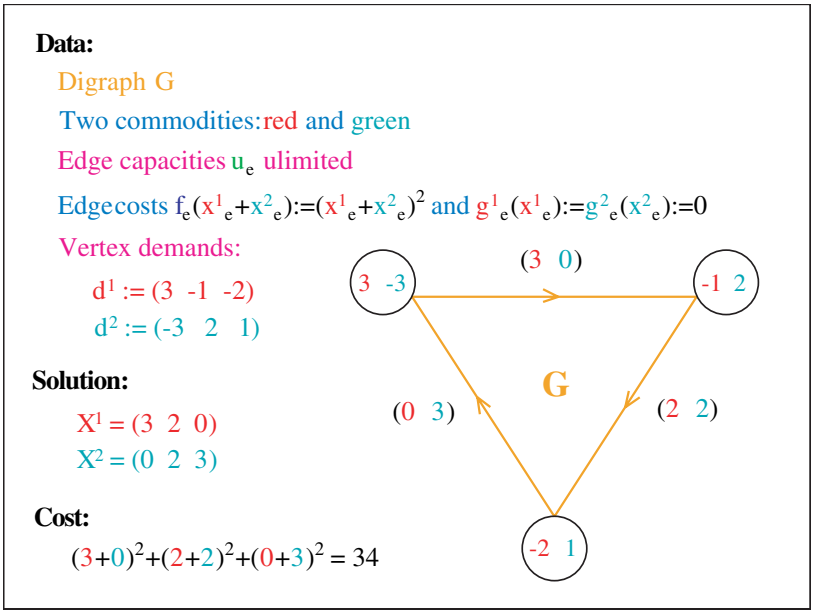

Figure 1.2: Multicommodity transshipment example

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