

Theory and Applications of N-Fold Integer Programming

Shmuel Onn ^{*}

Abstract

We overview our recently introduced theory of n -fold integer programming which enables the polynomial time solution of fundamental linear and nonlinear integer programming problems in variable dimension. We demonstrate its power by obtaining the first polynomial time algorithms in several application areas including multicommodity flows and privacy in statistical databases.

1 Introduction

Linear integer programming is the following fundamental optimization problem,

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

where A is an integer $m \times n$ matrix, $b \in \mathbb{Z}^m$, and $l, u \in \mathbb{Z}_{\infty}^n$ with $\mathbb{Z}_{\infty} := \mathbb{Z} \uplus \{\pm\infty\}$. It is generally NP-hard, but polynomial time solvable in two fundamental situations: the dimension is fixed [18]; the underlying matrix is totally unimodular [15].

Recently, in [4], a new fundamental polynomial time solvable situation was discovered. We proceed to describe this class of 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 following result of [4] asserts that n -fold integer programs are efficiently solvable.

^{*}Supported in part by a grant from ISF - the Israel Science Foundation

Theorem 1.1 [4] *For each fixed integer $(r, s) \times t$ bimatrix A , there is an algorithm that, given positive integer n , bounds $l, u \in \mathbb{Z}_{\infty}^{nt}$, $b \in \mathbb{Z}^{r+ns}$, and $w \in \mathbb{Z}^{nt}$, solves in time which is polynomial in n and in the binary-encoding length $\langle l, u, b, w \rangle$ of the rest of the data, the following so-termed linear n -fold integer programming problem,*

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

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 non totally 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 has numerous applications, the most generic being to integer optimization over multidimensional tables (see §2). In fact it is *universal*: the results of [7] imply that *every* integer program is an n -fold program over some simple bimatrix A (see §4).

The above theorem extends to n -fold integer programming with nonlinear objective functions as well. The following results, from [12], [5] and [13], assert that the minimization and maximization of broad classes of convex functions over n -fold integer programs can also be done in polynomial time. The function f is presented either by a *comparison oracle* that for any two vectors x, y can answer whether or not $f(x) \leq f(y)$, or by an *evaluation oracle* that for any vector x can return $f(x)$.

In the next theorem, f is *separable convex*, namely $f(x) = \sum_i f_i(x_i)$ with each f_i univariate convex. Like linear forms, such functions can be minimized over totally unimodular programs [14]. We show that they can also be efficiently minimized over n -fold programs. The running time depends also on $\log \hat{f}$ with \hat{f} the maximum value of $|f(x)|$ over the feasible set (which need not be part of the input).

Theorem 1.2 [12] *For each fixed integer $(r, s) \times t$ bimatrix A , there is an algorithm that, given $n, l, u \in \mathbb{Z}_{\infty}^{nt}$, $b \in \mathbb{Z}^{r+ns}$, and separable convex $f : \mathbb{Z}^{nt} \rightarrow \mathbb{Z}$ presented by a comparison oracle, solves in time polynomial in n and $\langle l, u, b, \hat{f} \rangle$, the program*

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

An important natural special case of Theorem 1.2 is the following result that concerns finding a feasible point which is l_p -closest to a given desired goal point.

Theorem 1.3 [12] *For each fixed integer $(r, s) \times t$ bimatrix A , there is an algorithm that, given positive integers n and p , $l, u \in \mathbb{Z}_{\infty}^{nt}$, $b \in \mathbb{Z}^{r+ns}$, and $\hat{x} \in \mathbb{Z}^{nt}$, solves in time polynomial in n, p , and $\langle l, u, b, \hat{x} \rangle$, the following distance minimization program,*

$$\min \{\|x - \hat{x}\|_p : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, l \leq x \leq u\}. \quad (1)$$

For $p = \infty$ the problem (1) can be solved in time polynomial in n and $\langle l, u, b, \hat{x} \rangle$.

The next result concerns the *maximization* of a convex function of the composite form $f(Wx)$, with $f : \mathbb{Z}^d \rightarrow \mathbb{Z}$ convex and W an integer matrix with d rows.

Theorem 1.4 [5] *For each fixed d and $(r, s) \times t$ integer bimatrix A , there is an algorithm that, given n , bounds $l, u \in \mathbb{Z}_{\infty}^{nt}$, integer $d \times nt$ matrix W , $b \in \mathbb{Z}^{r+ns}$, and convex function $f : \mathbb{Z}^d \rightarrow \mathbb{R}$ presented by a comparison oracle, solves in time polynomial in n and $\langle W, l, u, b \rangle$, the convex n -fold integer maximization program*

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

Finally, we have the following broad extension of Theorem 1.2 where the objective can include a composite term $f(Wx)$, with $f : \mathbb{Z}^d \rightarrow \mathbb{Z}$ separable convex and W an integer matrix with d rows, and where also inequalities on Wx can be included. As before, \hat{f}, \hat{g} denote the maximum values of $|f(Wx)|, |g(x)|$ over the feasible set.

Theorem 1.5 [13] *For each fixed integer $(r, s) \times t$ bimatrix A and integer $(p, q) \times t$ bimatrix W , there is an algorithm that, given $n, l, u \in \mathbb{Z}_{\infty}^{nt}, \hat{l}, \hat{u} \in \mathbb{Z}_{\infty}^{p+nt}, b \in \mathbb{Z}^{r+ns}$, and separable convex functions $f : \mathbb{Z}^{p+nt} \rightarrow \mathbb{Z}, g : \mathbb{Z}^{nt} \rightarrow \mathbb{Z}$ presented by evaluation oracles, solves in time polynomial in n and $\langle l, u, \hat{l}, \hat{u}, b, f, g \rangle$, the generalized program*

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

The article is organized as follows. In Section 2 we discuss some of the many applications of this theory and use Theorems 1.1–1.5 to obtain the first polynomial time algorithms for these applications. In Section 3 we provide a concise development of the theory of n -fold integer programming and prove our Theorems 1.1–1.5. Sections 2 and 3 can be read in any order. We conclude in Section 4 with a discussion of the universality of n -fold integer programming and of a new (di)-graph invariant, about which very little is known, that is important in understanding the complexity of our algorithms. Further discussion of n -fold integer programming within the broader context of *nonlinear discrete optimization* can be found in [21] and [22].

2 Applications

2.1 Multiway Tables

Multiway tables occur naturally in any context involving multiply-indexed variables. They have been studied extensively in mathematical programming in the context of high dimensional transportation problems (see [27, 28] and the references therein) and in statistics in the context of disclosure control and privacy in statistical

databases (see [3, 9] and the references therein). The theory of n -fold integer programming provides the first polynomial time algorithms for multiway table problems in these two contexts, which are discussed in §2.1.1 and §2.1.2 respectively.

We start with some terminology and background that will be used in the sequel. A *d-way table* is an $m_1 \times \cdots \times m_d$ array $x = (x_{i_1, \dots, i_d})$ of nonnegative integers. A *d-way transportation polytope* (*d-way polytope* for brevity) is the set of $m_1 \times \cdots \times m_d$ nonnegative arrays $x = (x_{i_1, \dots, i_d})$ with specified sums of entries over some of their lower dimensional subarrays (*margins* in statistics). The *d*-way tables with specified margins are the integer points in the *d*-way polytope. For example (see Figure 1), the 3-way polytope of $l \times m \times n$ arrays with specified line-sums (2-margins) is

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

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

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

Our results hold for k -margins for any $0 \leq k \leq d$, and much more generally for any so-called *hierarchical family* of margins. For simplicity of the exposition, however, we restrict attention here to line-sums, that is, $(d-1)$ -margins, only.

We conclude this preparation with the *universality theorem* for multiway tables and polytopes. It provides a powerful tool in establishing the presumable limits of polynomial time solvability of table problems, and will be used in §2.1.1 and §2.1.2 to contrast the polynomial time solvability attainable by n -fold integer programming.

Theorem 2.1 [7] *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*

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

2.1.1 Multi-index transportation problems

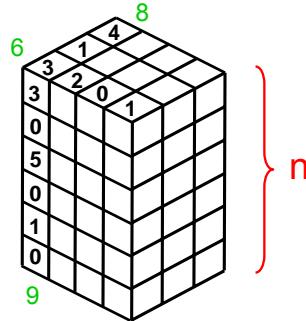
The *multi-index transportation problem* of Motzkin [19] is the integer programming problem over multiway tables with specified margins. For line-sums it is the program

$$\min \left\{ wx : x \in \mathbb{Z}_+^{m_1 \times \cdots \times m_d} : \sum_{i_1} x_{i_1, \dots, i_d} = v_{*,i_2, \dots, i_d}, \dots, \sum_{i_d} x_{i_1, \dots, i_d} = v_{i_1, \dots, i_{d-1}, *} \right\}.$$

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.

Multiway Tables

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



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

Such tables form an n -fold program $\{x : A^{(n)}x = b, x \geq 0, x \text{ integer}\}$ for suitable bimatrix A determined by m_1, \dots, 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 1: Multiway Tables

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-encoding length of the line-sums), but even in this very restricted situation one needs off-hand the algorithm of integer programming in fixed dimension lmn . If l, m, n are all variable then the problem is NP-hard [17]. 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 [6]; moreover, Theorem 2.1 implies that it is NP-hard even over $l \times m \times 3$ tables with fixed $n = 3$. Finally, if two sides are fixed and one is variable, then the problem can be solved in polynomial time by n -fold integer programming. Note that even over $3 \times 3 \times n$ tables, the only solution of the problem available to-date is the one given below using n -fold integer programming.

The polynomial time solvability of the multi-index transportation problem when one side is variable and the others are fixed extends to any dimension d . We have

the following important result on the multi-index transportation problem.

Theorem 2.2 [4] *For every fixed d, m_1, \dots, m_d , there is an algorithm that, given n , integer $m_1 \times \dots \times m_d \times n$ cost w , and integer line-sums $v = ((v_{*,i_2,\dots,i_{d+1}}), \dots, (v_{i_1,\dots,i_d,*}))$, solves in time polynomial in n and $\langle w, v \rangle$, the $(d + 1)$ -index transportation problem*

$$\min \left\{ wx : x \in \mathbb{Z}_+^{m_1 \times \dots \times m_d \times n} : \sum_{i_1} x_{i_1,\dots,i_{d+1}} = v_{*,i_2,\dots,i_{d+1}}, \dots, \sum_{i_{d+1}} x_{i_1,\dots,i_{d+1}} = v_{i_1,\dots,i_d,*} \right\}.$$

Proof. Re-index arrays as $x = (x^1, \dots, x^n)$ with each $x^{i_{d+1}} = (x_{i_1,\dots,i_d,i_{d+1}})$ a suitably indexed $m_1 m_2 \cdots m_d$ vector representing the i_{d+1} -th layer of x . Similarly re-index the array w . Let $t := r := m_1 m_2 \cdots m_d$ and $s := n(m_2 \cdots m_d + \cdots + m_1 \cdots m_{d-1})$. Let $b := (b^0, b^1, \dots, b^n) \in \mathbb{Z}^{r+ns}$, where $b^0 := (v_{i_1,\dots,i_d,*})$ and for $i_{d+1} = 1, \dots, n$,

$$b^{i_{d+1}} := ((v_{*,i_2,\dots,i_d,i_{d+1}}), \dots, (v_{i_1,\dots,i_{d-1},*,i_{d+1}})) .$$

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 $m_1 \times \dots \times m_d$ arrays. Then the equations $A_1(\sum_{i_{d+1}} x^{i_{d+1}}) = b^0$ represent the line-sum equations $\sum_{i_{d+1}} x_{i_1,\dots,i_{d+1}} = v_{i_1,\dots,i_d,*}$ where summations over layers occur, whereas the equations $A_2 x^{i_{d+1}} = b^{i_{d+1}}$ for $i_{d+1} = 1, \dots, n$ represent all other line-sum equations, where summations are within a single layer at a time. Therefore the multi-index transportation problem is encoded as the n -fold integer programming problem

$$\min \{wx : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, x \geq 0\} .$$

Using the algorithm of Theorem 1.1, this n -fold integer program, and hence the given multi-index transportation problem, can be solved in polynomial time. \square

This proof extends immediately to multi-index transportation problems with nonlinear objective functions of the forms in Theorems 1.2–1.5. Moreover, as mentioned before, a similar proof shows that multi-index transportation problems with k -margin constraints, and more generally, hierarchical margin constraints, can be encoded as n -fold integer programming problems as well. We state this as a corollary.

Corollary 2.3 [5] *For every fixed d and m_1, \dots, m_d , the nonlinear multi-index transportation problem, with any hierarchical margin constraints, over $(d + 1)$ -way tables of format $m_1 \times \dots \times m_d \times n$ with variable n layers, are polynomial time solvable.*

2.1.2 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 can be violated. This raises the following fundamental problem.

Entry uniqueness problem: Given hierarchical margin family 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 multi-index transportation problem discussed in §2.1.1. 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 [17]. We discuss next in more detail the in-between cases which are more delicate and were settled only recently.

If two sides are variable and one is fixed then the problem is still coNP-complete, even over $l \times m \times 3$ tables with fixed $n = 3$ [20]. Moreover, Theorem 2.1 implies 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. Figure 2 gives an example of line-sums for $6 \times 4 \times 3$ tables where one entry attains the set of values $\{0, 2\}$ which has a *gap*.

Theorem 2.4 [8] *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 that occur in some fixed entry in all $l \times m \times 3$ tables that have these line-sums, is precisely S .*

Proof. Consider any finite set $S = \{s_1, \dots, s_h\} \subset \mathbb{Z}_+$. Consider the polytope

$$P := \left\{ y \in \mathbb{R}_+^{h+1} : y_0 - \sum_{j=1}^h s_j y_j = 0, \sum_{j=1}^h y_j = 1 \right\} .$$

By Theorem 2.1, there are l, m , and $l \times m \times 3$ polytope T with line-sums

$$v_{*,j,k}, \quad v_{i,*,k}, \quad v_{i,j,*}, \quad 1 \leq i \leq l, \quad 1 \leq j \leq m, \quad 1 \leq k \leq 3 ,$$

such that the integer points in T , which are precisely the $l \times m \times 3$ tables with these line-sums, are in bijection with the integer points in P . Moreover (see [7]), this bijection is obtained by a simple projection from $\mathbb{R}^{l \times m \times 3}$ to \mathbb{R}^{h+1} that erases all but some $h + 1$ coordinates. Let $x_{i,j,k}$ be the coordinate that is mapped to y_0 . Then the set of values that this entry attains in all tables with these line-sums is, as desired,

$$\{x_{i,j,k} : x \in T \cap \mathbb{Z}^{l \times m \times 3}\} = \{y_0 : y \in P \cap \mathbb{Z}^{h+1}\} = S . \quad \square$$

Set of Entry Values With a Gap

The **only values** occurring in the **designated entry** in all **$6 \times 4 \times 3$** tables with the **specified line-sums** are **0, 2**

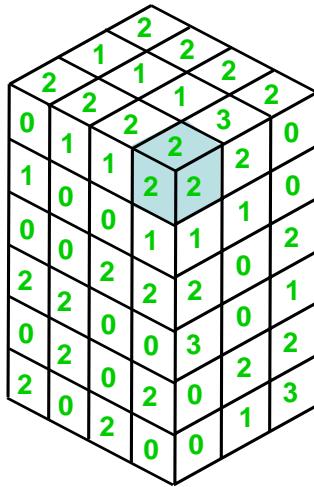


Figure 2: Set of Entry Values With a Gap

Finally, if two sides are fixed and one is variable, then entry uniqueness can be decided in polynomial time by n -fold integer programming. Note that even over $3 \times 3 \times n$ tables, the only solution of the problem available to-date is the one below.

The polynomial time decidability of the problem when one side is variable and the others are fixed extends to any dimension d . It also extends to any hierarchical family of margins, but for simplicity we state it only for line-sums, as follows.

Theorem 2.5 [20] *For every fixed d, m_1, \dots, m_d , there is an algorithm that, given n , integer line-sums $v = ((v_{*,i_2,\dots,i_{d+1}}), \dots, (v_{i_1,\dots,i_d,*}))$, and entry index (k_1, \dots, k_{d+1}) , solves in time which is polynomial in n and $\langle v \rangle$, the corresponding entry uniqueness*

problem, of deciding if the entry $x_{k_1, \dots, k_{d+1}}$ is the same in all $(d+1)$ -tables in the set

$$S := \left\{ x \in \mathbb{Z}_+^{m_1 \times \dots \times m_d \times n} : \sum_{i_1} x_{i_1, \dots, i_{d+1}} = v_{*, i_2, \dots, i_{d+1}}, \dots, \sum_{i_{d+1}} x_{i_1, \dots, i_{d+1}} = v_{i_1, \dots, i_d, *} \right\}.$$

Proof. By Theorem 2.2 we can solve in polynomial time both n -fold programs

$$l := \min \{x_{k_1, \dots, k_{d+1}} : x \in S\},$$

$$u := \max \{x_{k_1, \dots, k_{d+1}} : x \in S\}.$$

Clearly, entry $x_{k_1, \dots, k_{d+1}}$ has the same value in all tables with the given line-sums if and only if $l = u$, which can therefore be tested in polynomial time. \square

The algorithm of Theorem 2.5 and its extension to any family of hierarchical margins 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 be risky and 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 Theorem 2.5 applicable.

Finally, we describe a procedure based on a suitable adaptation of the algorithm of Theorem 2.5, that constructs the entire set of values that can occur in a specified entry, rather than just decides its uniqueness. Here S is the set of tables satisfying the given (hierarchical) margins, and the running time is output-sensitive, that is, polynomial in the input encoding plus the number of elements in the output set.

Procedure for 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:

$$\hat{l} := \min \{x_{k_1, \dots, k_{d+1}} : l \leq x_{k_1, \dots, k_{d+1}} \leq u, x \in S\},$$

$$\hat{u} := \max \{x_{k_1, \dots, k_{d+1}} : l \leq x_{k_1, \dots, k_{d+1}} \leq u, x \in S\}.$$

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 .

2.2 Multicommodity Flows

The multicommodity transshipment problem 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 data for the problem is as follows (see Figure 3 for a small example). There is a digraph G with s vertices

Multicommodity Transshipment Example

Data:

```

digraph G
two commodities: red and green
edge capacities  $u_e$  unlimited
edge costs  $f_e(x^1_e + x^2_e) := (x^1_e + x^2_e)^2$  and  $g^1_e(x^1_e) := g^2_e(x^2_e) := 0$ 
vertex demands:
 $d^1 := (3 \ -1 \ -2)$ 
 $d^2 := (-3 \ 2 \ 1)$ 

```

Solution:

$$X^1 = (3 \ 2 \ 0)$$

$$X^2 = (0 \ 2 \ 3)$$

Cost:

$$(3+0)^2 + (2+2)^2 + (0+3)^2 = 34$$

Figure 3: Multicommodity Transshipment Example

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 $x = (x^1, \dots, x^l)$ 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 $\sum_{k=1}^l x_e^k \leq u_e$ 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 $f_e, g_e^k : \mathbb{Z} \rightarrow \mathbb{Z}$ for each edge and each edge-commodity pair. The transshipment cost on edge e is $f_e(\sum_{k=1}^l x_e^k) + \sum_{k=1}^l g_e^k(x_e^k)$ 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 depend on both the edge and the commodity. The total cost is

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

Our results apply to cost functions which can be standard linear or 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 take into account the increase in cost due to channel congestion when subject to heavy traffic or communication load (with the linear case obtained by $\beta_e = \delta_e^k = 1$).

The theory of n -fold integer programming provides the first polynomial time algorithms for the problem in two broad situations discussed in §2.2.1 and §2.2.2.

2.2.1 The many-commodity transshipment problem

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 [13]; 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} and \hat{g} denote the maximum absolute values of the objective functions f and g over the feasible set. 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-encoding length of the costs α_e, γ_e^k , capacities u_e , and demands d_v^k , are easily obtained by Cramer's rule.

We have the following theorem on (nonlinear) many-commodity transshipment.

Theorem 2.6 [13] *For every fixed digraph G there is an algorithm that, given 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 convex costs $f_e, g_e^k : \mathbb{Z} \rightarrow \mathbb{Z}$ 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,*

$$\begin{aligned} \min \quad & \sum_e \left(f_e \left(\sum_{k=1}^l x_e^k \right) + \sum_{k=1}^l g_e^k(x_e^k) \right) \\ \text{s.t.} \quad & x_e^k \in \mathbb{Z}, \quad \sum_{e \in \delta^+(v)} x_e^k - \sum_{e \in \delta^-(v)} x_e^k = d_v^k, \quad \sum_{k=1}^l x_e^k \leq u_e, \quad x_e^k \geq 0 . \end{aligned}$$

Proof. Assume G has s vertices and t edges and let D be its $s \times t$ vertex-edge incidence matrix. Let $f : \mathbb{Z}^t \rightarrow \mathbb{Z}$ and $g : \mathbb{Z}^{lt} \rightarrow \mathbb{Z}$ 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 $x = (x^1, \dots, x^l)$ 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 in vector form as

$$\min \left\{ f \left(\sum_{k=1}^l x^k \right) + g(x) : x \in \mathbb{Z}^{lt}, Dx^k = d^k, \sum_{k=1}^l x^k \leq u, x \geq 0 \right\} .$$

We can now proceed in two ways.

First way: extend the vector of variables to $x = (x^0, x^1, \dots, x^l)$ 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 $f(u - x_0) + g(x^1, \dots, x^l)$ 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 $b := (u, d^0, d^1, \dots, d^l)$. Then the problem becomes the $(l+1)$ -fold integer program

$$\min \left\{ f(u - x^0) + g(x^1, \dots, x^l) : x \in \mathbb{Z}^{(l+1)t}, A^{(l)}x = b, x \geq 0 \right\} . \quad (2)$$

By Theorem 1.2 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 $b := (d^1, \dots, d^l)$. Then the problem is precisely the following l -fold integer program,

$$\min \left\{ f(W^{(l)}x) + g(x) : x \in \mathbb{Z}^{lt}, A^{(l)}x = b, W^{(l)}x \leq u, x \geq 0 \right\} .$$

By Theorem 1.5 this program can be solved in polynomial time as claimed. \square

We also point out the following immediate corollary of Theorem 2.6.

Corollary 2.7 *For any fixed s , the (convex) many-commodity transshipment problem with variable l commodities on any s -vertex digraph is polynomial time solvable.*

2.2.2 The multicommodity transportation problem

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 the complete bipartite digraphs $K_{m,n}$ [7]; moreover, even over the 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). We now denote edges by pairs (i,j) where

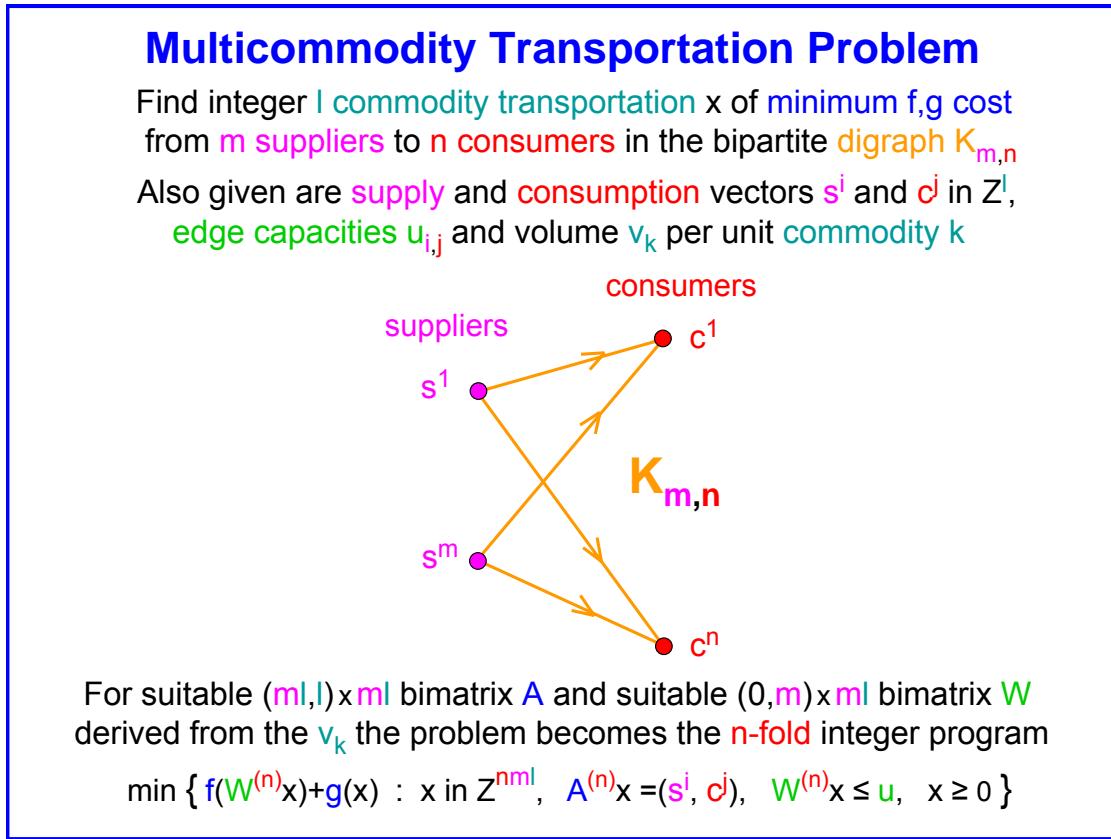


Figure 4: Multicommodity Transportation Problem

$1 \leq i \leq m$ is a supplier and $1 \leq j \leq n$ is a consumer. The demand vectors are now replaced by (nonnegative) supply and consumption vectors: each supplier i has a supply vector $s^i \in \mathbb{Z}_+^l$ with s_k^i its supply in commodity k , and each consumer j has a consumption vector $c^j \in \mathbb{Z}_+^l$ with c_k^j its consumption in commodity k . In addition, here each commodity k may have its own volume $v_k \in \mathbb{Z}_+$ per unit flow. A multicommodity transportation is now indexed as $x = (x^1, \dots, x^n)$ with

$x^j = (x_{1,1}^j, \dots, x_{1,l}^j, \dots, x_{m,1}^j, \dots, x_{m,l}^j)$, where $x_{i,k}^j$ is the flow of commodity k from supplier i to consumer j . The capacity constraint on edge (i, j) is $\sum_{k=1}^l v_k x_{i,k}^j \leq u_{i,j}$ and the cost is $f_{i,j} \left(\sum_{k=1}^l v_k x_{i,k}^j \right) + \sum_{k=1}^l g_{i,k}^j (x_{i,k}^j)$ with $f_{i,j}, g_{i,k}^j : \mathbb{Z} \rightarrow \mathbb{Z}$ convex.

As before, \hat{f}, \hat{g} denote the maximum absolute values of f, g over the feasible set.

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 simply forcing 0 capacity on all edges not present in G .

We have the following theorem on (nonlinear) multicommodity transportation.

Theorem 2.8 [13] *For any fixed l commodities, m suppliers, and volumes v_k , there is an algorithm that, given n , supplies and demands $s^i, c^j \in \mathbb{Z}_+^l$, capacities $u_{i,j} \in \mathbb{Z}_+$, and convex costs $f_{i,j}, g_{i,k}^j : \mathbb{Z} \rightarrow \mathbb{Z}$ 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,*

$$\begin{aligned} \min \quad & \sum_{i,j} \left(f_{i,j} \left(\sum_k v_k x_{i,k}^j \right) + \sum_{k=1}^l g_{i,k}^j (x_{i,k}^j) \right) \\ \text{s.t.} \quad & x_{i,k}^j \in \mathbb{Z}, \quad \sum_j x_{i,k}^j = s_k^i, \quad \sum_i x_{i,k}^j = c_k^j, \quad \sum_{k=1}^l v_k x_{i,k}^j \leq u_{i,j}, \quad x_{i,k}^j \geq 0 . \end{aligned}$$

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

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. Therefore 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 $W^{(n)}x \leq u$ and the cost function becomes $f(W^{(n)}x) + g(x)$. Therefore, the problem is precisely the following n -fold integer program,

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

By Theorem 1.5 this program can be solved in polynomial time as claimed. \square

3 Theory

In §3.1 we define Graver bases of integer matrices and show that they can be used to solve linear and nonlinear integer programs in polynomial time. In §3.2 we show that Graver bases of n -fold products can be computed in polynomial time and, incorporating the results of §3.1, prove our Theorems 1.1–1.5 that establish the polynomial time solvability of linear and nonlinear n -fold integer programming.

To simplify the presentation, and since the feasible set in most applications is finite or can be made finite by more careful modeling, whenever an algorithm detects that the feasible set is infinite, it simply stops. So, throughout our discussion, an algorithm is said to *solve a (nonlinear) integer programming problem* if it either finds an optimal solution x or concludes that the feasible set is infinite or empty.

As noted in the introduction, any nonlinear function f involved is presented either by a mere comparison oracle that for any two vectors x, y can answer whether or not $f(x) \leq f(y)$, or by an evaluation oracle that for any vector x can return $f(x)$.

3.1 Graver Bases and Nonlinear Integer Programming

The *Graver basis* is a fundamental object in the theory of integer programming which was introduced by J. Graver already back in 1975 [11]. However, only very recently, in the series of papers [4, 5, 12], it was established that the Graver basis can be used to solve linear (as well as nonlinear) integer programming problems in polynomial time. In this subsection we describe these important new developments.

3.1.1 Graver bases

We begin with the definition of the Graver basis and some of its basic properties. Throughout this subsection let A be an integer $m \times n$ matrix. The *lattice* of A is the set $\mathcal{L}(A) := \{x \in \mathbb{Z}^n : Ax = 0\}$ of integer vectors in its kernel. We use $\mathcal{L}^*(A) := \{x \in \mathbb{Z}^n : Ax = 0, x \neq 0\}$ to denote the set of nonzero elements in $\mathcal{L}(A)$. We use a partial order \sqsubseteq on \mathbb{R}^n which extends the usual coordinate-wise partial order \leq on the nonnegative orthant \mathbb{R}_+^n and is defined as follows. For two vectors $x, y \in \mathbb{R}^n$ we write $x \sqsubseteq y$ and say that x is *conformal* to y if $x_i y_i \geq 0$ and $|x_i| \leq |y_i|$ for $i = 1, \dots, n$, 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. A suitable extension of the classical lemma of Gordan [10] implies that every subset of \mathbb{Z}^n has finitely many \sqsubseteq -minimal elements. We have the following fundamental definition.

Definition 3.1 [11] 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 $\mathcal{L}^*(A) = \{x \in \mathbb{Z}^n : Ax = 0, x \neq 0\}$.

Note that $\mathcal{G}(A)$ is centrally symmetric, that is, $g \in \mathcal{G}(A)$ if and only if $-g \in \mathcal{G}(A)$. For instance, the Graver basis of the 1×3 matrix $A := [1 \ 2 \ 1]$ consists of 8 elements,

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

Note also that the Graver basis may contain elements, such as $(1, -1, 1)$ in the above small example, whose support involves linearly dependent columns of A . So the cardinality of the Graver basis cannot be bounded in terms of m and n only and depends on the entries of A as well. Indeed, the Graver basis is typically exponential and cannot be written down, let alone computed, in polynomial time. But, as we will show in the next section, for n -fold products it can be computed efficiently.

A finite sum $u := \sum_i v_i$ of vectors in \mathbb{R}^n is called *conformal* if all summands lie in the same orthant and hence $v_i \sqsubseteq u$ for all i . We start with a simple lemma.

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

Proof. We use 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 . \square

We now provide a stronger form of Lemma 3.2 which basically follows from the integer analogs of Carathéodory's theorem established in [2] and [26].

Lemma 3.3 *Any $x \in \mathcal{L}^*(A)$ is a conformal sum $x = \sum_{i=1}^t \lambda_i g_i$ involving $t \leq 2n - 2$ Graver basis elements $g_i \in \mathcal{G}(A)$ with nonnegative integer coefficients $\lambda_i \in \mathbb{Z}_+$.*

Proof. We prove the slightly weaker bound $t \leq 2n - 1$ from [2]. A proof of the stronger bound can be found in [26]. Consider any $x \in \mathcal{L}^*(A)$ and let g_1, \dots, g_s be all elements of $\mathcal{G}(A)$ lying in the same orthant as x . Consider the linear program

$$\max \left\{ \sum_{i=1}^s \lambda_i : x = \sum_{i=1}^s \lambda_i g_i, \quad \lambda_i \in \mathbb{R}_+ \right\} . \quad (3)$$

By Lemma 3.2 the point x is a nonnegative linear combination of the g_i and hence the program (3) is feasible. Since all g_i are nonzero and in the same orthant as x , program (3) is also bounded. As is well known, it then has a *basic* optimal solution, that is, an optimal solution $\lambda_1, \dots, \lambda_s$ with at most n of the λ_i nonzero. Let

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

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), 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 integer, 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 . \square

The Graver basis also enables to check the finiteness of a feasible integer program.

Lemma 3.4 *Let $\mathcal{G}(A)$ be the Graver basis of matrix A and let $l, u \in \mathbb{Z}_\infty^n$. 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 $P := \{x \in \mathbb{R}^n : Ax = b, l \leq x \leq u\}$ 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 $\alpha \geq 0$. This implies that $h \in \mathcal{L}^*(A)$ and that $h_i \leq 0$ whenever $u_i < \infty$ and $h_i \geq 0$ 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 $g_i \leq 0$ whenever $u_i < \infty$ and $g_i \geq 0$ whenever $l_i > -\infty$, providing such g . \square

3.1.2 Separable convex integer minimization

In this subsection we consider the following nonlinear integer minimization problem

$$\min\{f(x) : x \in \mathbb{Z}^n, Ax = b, l \leq x \leq u\}, \quad (4)$$

where A is an integer $m \times n$ matrix, $b \in \mathbb{Z}^m$, $l, u \in \mathbb{Z}_\infty^n$, and $f : \mathbb{Z}^n \rightarrow \mathbb{Z}$ is a separable convex function, that is, $f(x) = \sum_{j=1}^n f_j(x_j)$ with $f_j : \mathbb{Z} \rightarrow \mathbb{Z}$ a univariate convex function for all j . We prove a sequence of lemmas and then combine them to show that the Graver basis of A enables to solve this problem in polynomial time.

We start with two simple lemmas about univariate convex functions. The first lemma establishes a certain *supermodularity* property of such functions.

Lemma 3.5 Let $f : \mathbb{R} \rightarrow \mathbb{R}$ be a univariate convex function, let r be a real number, and let s_1, \dots, s_m be real numbers satisfying $s_i s_j \geq 0$ for all i, j . Then we have

$$f\left(r + \sum_{i=1}^m s_i\right) - f(r) \geq \sum_{i=1}^m (f(r + s_i) - f(r)) .$$

Proof. We use induction on m . The claim holding trivially for $m = 1$, consider $m > 1$. Since all nonzero s_i have the same sign, $s_m = \lambda \sum_{i=1}^m s_i$ for some $0 \leq \lambda \leq 1$. Then

$$r + s_m = (1 - \lambda)r + \lambda \left(r + \sum_{i=1}^{m-1} s_i\right), \quad r + \sum_{i=1}^{m-1} s_i = \lambda r + (1 - \lambda) \left(r + \sum_{i=1}^m s_i\right) ,$$

and so the convexity of f implies

$$\begin{aligned} f(r + s_m) &+ f\left(r + \sum_{i=1}^{m-1} s_i\right) \\ &\leq (1 - \lambda)f(r) + \lambda f\left(r + \sum_{i=1}^m s_i\right) + \lambda f(r) + (1 - \lambda)f\left(r + \sum_{i=1}^m s_i\right) \\ &= f(r) + f\left(r + \sum_{i=1}^m s_i\right). \end{aligned}$$

Subtracting $2f(r)$ from both sides and applying induction, we obtain, as claimed,

$$\begin{aligned} f\left(r + \sum_{i=1}^m s_i\right) - f(r) &\geq f(r + s_m) - f(r) + f\left(r + \sum_{i=1}^{m-1} s_i\right) - f(r) \\ &\geq \sum_{i=1}^m (f(r + s_i) - f(r)) . \quad \square \end{aligned}$$

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

Lemma 3.6 There is an algorithm that, given any two integer numbers $r \leq s$ and any univariate convex function $f : \mathbb{Z} \rightarrow \mathbb{R}$ given by a comparison oracle, solves in time polynomial in $\langle r, s \rangle$ the following univariate integer minimization problem,

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

Proof. If $r = s$ then $\lambda := r$ is optimal. Assume then $r \leq s - 1$. Consider the integers

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

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

$$\begin{aligned} f(\lfloor \frac{r+s}{2} \rfloor) = f(\lfloor \frac{r+s}{2} \rfloor + 1) &\Rightarrow \lambda := \lfloor \frac{r+s}{2} \rfloor \text{ is a minimum of } f; \\ f(\lfloor \frac{r+s}{2} \rfloor) < f(\lfloor \frac{r+s}{2} \rfloor + 1) &\Rightarrow \text{the minimum of } f \text{ is in the interval } [r, \lfloor \frac{r+s}{2} \rfloor]; \\ f(\lfloor \frac{r+s}{2} \rfloor) > f(\lfloor \frac{r+s}{2} \rfloor + 1) &\Rightarrow \text{the minimum of } f \text{ is in the interval } [\lfloor \frac{r+s}{2} \rfloor + 1, s]. \end{aligned}$$

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 $\lambda \in \mathbb{Z} \cap [r, s]$. \square

The next two lemmas extend Lemmas 3.5 and 3.6. The first lemma shows the supermodularity of separable convex functions with respect to conformal sums.

Lemma 3.7 *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be any separable convex function, let $x \in \mathbb{R}^n$ be any point, and let $\sum g_i$ be any conformal sum in \mathbb{R}^n . Then the following inequality holds,*

$$f\left(x + \sum g_i\right) - f(x) \geq \sum (f(x + g_i) - f(x)) .$$

Proof. Let $f_j : \mathbb{R} \rightarrow \mathbb{R}$ be univariate convex functions such that $f(x) = \sum_{j=1}^n f_j(x_j)$. Consider any $1 \leq j \leq n$. Since $\sum g_i$ is a conformal sum, we have $g_{i,j} g_{k,j} \geq 0$ for all i, k and so, setting $r := x_j$ and $s_i := g_{i,j}$ for all i , Lemma 3.5 applied to f_j implies

$$f_j\left(x_j + \sum_i g_{i,j}\right) - f_j(x_j) \geq \sum_i (f_j(x_j + g_{i,j}) - f_j(x_j)) . \quad (5)$$

Summing the equations (5) for $j = 1, \dots, n$, we obtain the claimed inequality. \square

The second lemma concerns finding a best improvement step in a given direction.

Lemma 3.8 *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 $l \leq x \leq u$, and convex function $f : \mathbb{Z}^n \rightarrow \mathbb{R}$ presented by comparison oracle, solves in time polynomial in $\langle l, u, g, x \rangle$, the univariate problem,*

$$\min\{f(x + \lambda g) : \lambda \in \mathbb{Z}_+, l \leq x + \lambda g \leq u\} \quad (6)$$

Proof. Let $S := \{\lambda \in \mathbb{Z}_+ : l \leq x + \lambda g \leq u\}$ 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, $S = \{0, 1, \dots, s\}$ and the problem can be solved by the algorithm of

Lemma 3.6 minimizing the univariate convex function $h(\lambda) := h(x + \lambda g)$ over S . \square

We can now show that the Graver basis of A allows to solve problem (4) in polynomial time, provided we are given an initial feasible point to start with. We will later show how to find such an initial point as well. As noted in the introduction, \hat{f} below denotes the maximum value of $|f(x)|$ over the feasible set (which need not be part of the input). An outline of the algorithm is provided in Figure 5.

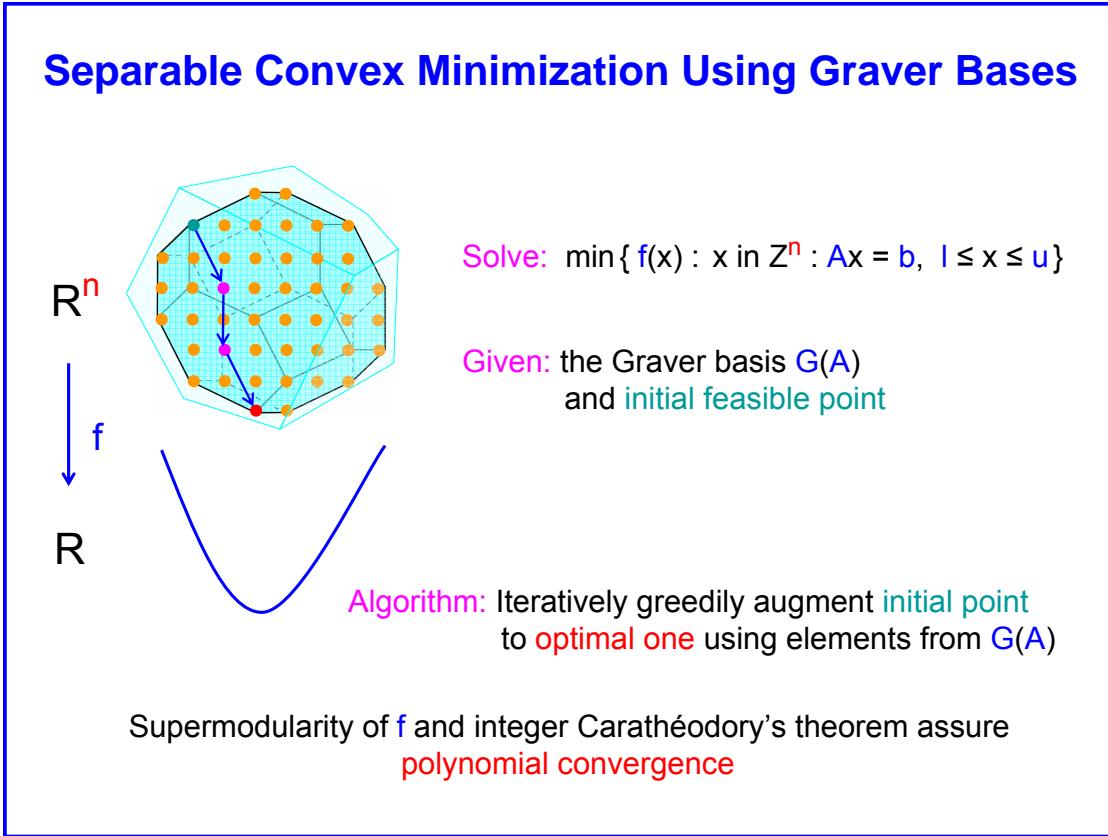


Figure 5: Separable Convex Minimization Using Graver Bases

Lemma 3.9 *There is an algorithm that, given an integer $m \times n$ matrix A , its Graver basis $G(A)$, vectors $l, u \in \mathbb{Z}_\infty^n$ and $x \in \mathbb{Z}^n$ with $l \leq x \leq u$, and separable convex function $f : \mathbb{Z}^n \rightarrow \mathbb{Z}$ presented by a comparison oracle, solves the integer program*

$$\min\{f(z) : z \in \mathbb{Z}^n, Az = b, l \leq z \leq u\}, \quad b := Ax, \quad (7)$$

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

Proof. First, apply the algorithm of Lemma 3.4 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, \dots, x_s with $x_0 := x$ the given input point, as follows. Having obtained x_k , solve the univariate minimization problem

$$\min\{f(x_k + \lambda g) : \lambda \in \mathbb{Z}_+, g \in \mathcal{G}(A), l \leq x_k + \lambda g \leq u\} \quad (8)$$

by applying the algorithm of Lemma 3.8 for each $g \in \mathcal{G}(A)$. If the minimal value in (8) 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 (7). Consider any point x_k in the sequence and suppose it is not optimal. We claim that a new point x_{k+1} will be produced and will satisfy

$$f(x_{k+1}) - f(x^*) \leq \frac{2n-3}{2n-2} (f(x_k) - f(x^*)) \quad (9)$$

By Lemma 3.3, we can write the difference $x^* - x_k = \sum_{i=1}^t \lambda_i g_i$ as conformal sum involving $1 \leq t \leq 2n-2$ elements $g_i \in \mathcal{G}(A)$ with all $\lambda_i \in \mathbb{Z}_+$. By Lemma 3.7,

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

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

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

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

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

So the point $x_k + \lambda g$ attaining minimum in (8) satisfies

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

and so indeed $x_{k+1} := x_k + \lambda g$ will be produced and will satisfy (9). 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 non optimal 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 (9) gives

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

and therefore

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

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

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

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

Next we show that Lemma 3.9 can also be used to find an initial feasible point for the given integer program or assert that none exists in polynomial time.

Lemma 3.10 *There is an algorithm that, given integer $m \times n$ matrix A , its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}_\infty^n$, and $b \in \mathbb{Z}^m$, either finds an $x \in \mathbb{Z}^n$ satisfying $l \leq x \leq u$ and $Ax = b$ or asserts that none exists, in time which is polynomial in $\langle A, \mathcal{G}(A), l, u, b \rangle$.*

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 3.4. 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 [25]. Next define a separable convex function on \mathbb{Z}^n by $f(x) := \sum_{j=1}^n f_j(x_j)$ with

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

and extended lower and upper bounds

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

Consider the auxiliary separable convex integer program

$$\min\{f(z) : z \in \mathbb{Z}^n, Az = b, \hat{l} \leq z \leq \hat{u}\} \quad (10)$$

First note that $\hat{l}_j > -\infty$ if and only if $l_j > -\infty$ and $\hat{u}_j < \infty$ if and only if $u_j < \infty$. Therefore 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 (10) is finite by Lemma 3.4. Next note that \hat{x} is feasible in (10). Now apply the algorithm of Lemma 3.9 to (10) and obtain an optimal solution x . Note that 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 (10) are polynomial in the length of the data.

Now note that every point $z \in S$ is feasible in (10), and every point z feasible in (10) 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. \square

We are finally in position, using Lemmas 3.9 and 3.10, to show that the Graver basis allows to solve the nonlinear integer program (4) in polynomial time. As usual, \hat{f} is the maximum of $|f(x)|$ over the feasible set and need not be part of the input.

Theorem 3.11 [12] *There is an algorithm that, given integer $m \times n$ matrix A , its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}_{\infty}^n$, $b \in \mathbb{Z}^m$, and separable convex $f : \mathbb{Z}^n \rightarrow \mathbb{Z}$ presented by comparison oracle, solves in time polynomial in $\langle A, \mathcal{G}(A), l, u, b, \hat{f} \rangle$ the problem*

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

Proof. First, apply the polynomial time algorithm of Lemma 3.10 and either conclude that the feasible set is infinite or empty and stop, or obtain an initial feasible point and continue. Next, apply the polynomial time algorithm of Lemma 3.9 and either conclude that the feasible set is infinite or obtain an optimal solution. \square

3.1.3 Specializations and extensions

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-encoding length of the data, readily follows from Cramer's rule. Therefore we obtain, as an immediate special case of Theorem 3.11, the following important result, asserting that Graver bases enable the polynomial time solution of linear integer programming.

Theorem 3.12 [4] *There is an algorithm that, given an integer $m \times n$ matrix A , its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}_{\infty}^n$, $b \in \mathbb{Z}^m$, 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,*

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

DISTANCE MINIMIZATION

Another useful special case of Theorem 3.11 which is natural in various applications such as image processing, tomography, communication, and error correcting codes, is the following result, which asserts that the Graver basis enables to determine a feasible point which is l_p -closest to a given desired goal point in polynomial time.

Theorem 3.13 [12] *There is an algorithm that, given integer $m \times n$ matrix A , its Graver basis $\mathcal{G}(A)$, positive integer p , vectors $l, u \in \mathbb{Z}_{\infty}^n$, $b \in \mathbb{Z}^m$, 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 distance minimization problem*

$$\min\{\|x - \hat{x}\|_p : x \in \mathbb{Z}^n, Ax = b, l \leq x \leq u\}. \quad (11)$$

For $p = \infty$ the problem (11) 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 3.11 taking f to be the p -th 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

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

Now apply the algorithm of the first paragraph above for the l_q distance. Assuming the feasible set is nonempty and finite (else the algorithm stops) let x^* be the feasible point which minimizes 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,

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

Therefore

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

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 $\|x^* - \hat{x}\|_\infty \leq \|x - \hat{x}\|_\infty$. This establishes the claim. \square

In particular, for all positive $p \in \mathbb{Z}_\infty$, using the Graver basis we can solve

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

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

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

CONVEX INTEGER MAXIMIZATION

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

We need a result of [23]. A *linear-optimization oracle* for a set $S \subset \mathbb{Z}^n$ is one that, given $w \in \mathbb{Z}^n$, solves the linear optimization problem $\max\{wx : x \in S\}$. 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 one that contains some direction of each edge of P , see Figure 6.

Theorem 3.14 [23] *For all fixed d there is an algorithm that, given a finite set $S \subset \mathbb{Z}^n$ presented by linear-optimization oracle, integer $d \times n$ matrix W , set $E \subset \mathbb{Z}^n$ of all edge-directions of $\text{conv}(S)$, and convex $f : \mathbb{Z}^d \rightarrow \mathbb{R}$ presented by comparison oracle, solves in time polynomial in $\langle \max\{\|x\|_\infty : x \in S\}, W, E \rangle$, the convex problem*

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

We now show that, fortunately enough, the Graver basis of a matrix A is a set of all edge-directions of the integer hull related to the integer program defined by A .

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

Proof. Consider any edge e of P_I and pick two 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 $x + h_i \in P_I$ for all i . Indeed, $h_i \in \mathcal{G}(A)$ implies $A(x + h_i) = Ax = b$, and $l \leq x, x + g \leq u$ and $h_i \sqsubseteq g$ imply $l \leq x + h_i \leq u$.

Now let $w \in \mathbb{Z}^n$ be uniquely maximized over P_I at the edge e . Then $wh_i = w(x + h_i) - wx \leq 0$ 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). \square

Using Theorems 3.12 and 3.14 and Lemma 3.15 we obtain the following theorem.

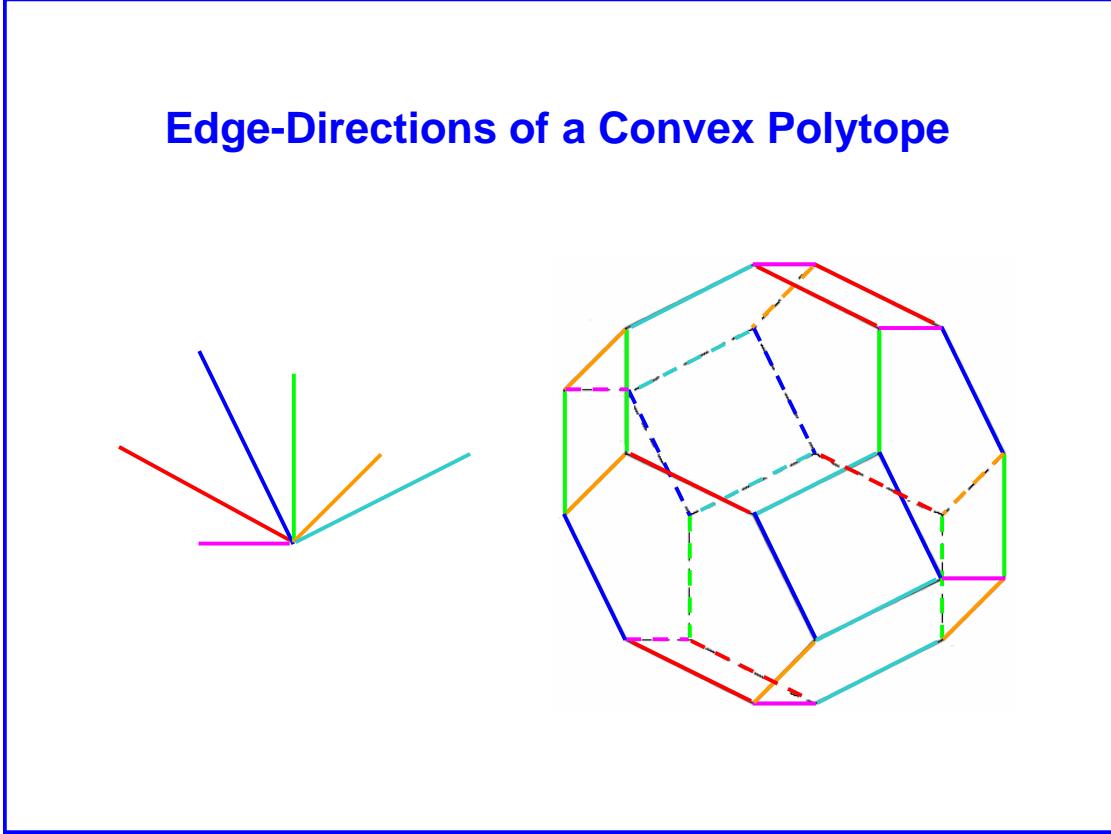


Figure 6: Edge-Directions of a Convex Polytope

Theorem 3.16 [5] For every fixed d there is an algorithm that, given integer $m \times n$ matrix A , its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}_{\infty}^n$, $b \in \mathbb{Z}^m$, integer $d \times n$ matrix W , and convex function $f : \mathbb{Z}^d \rightarrow \mathbb{R}$ 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

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

Proof. Let $S := \{x \in \mathbb{Z}^n : Ax = b, l \leq x \leq u\}$. The algorithm of Theorem 3.12 allows to simulate in polynomial time a linear-optimization oracle for S . In particular, it allows to either conclude that S is infinite and stop or conclude that it is finite, in which case $\langle \max\{\|x\|_{\infty} : x \in S\} \rangle$ 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 $\text{conv}(S) = P_I$. Hence the algorithm of Theorem 3.14 can be applied, and provides the polynomial

time solution of the convex integer maximization program. \square

3.2 N-Fold Integer Programming

In this subsection we focus our attention on (nonlinear) n -fold integer programming. In §3.2.1 we study Graver bases of n -fold products of integer bimatrices and show that they can be computed in polynomial time. In §3.2.2 we combine the results of §3.1 and §3.2.1, and prove our Theorems 1.1–1.5, which establish the polynomial time solvability of linear and nonlinear n -fold integer programming.

3.2.1 Graver bases of n-fold products

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 $x = (x^1, \dots, x^n)$ with each *brick* x^k lying in \mathbb{Z}^t . The *type* of vector x is the number $\text{type}(x) := |\{k : x^k \neq 0\}|$ of nonzero bricks of x .

The following definition plays an important role in the sequel.

Definition 3.17 [24] The *Graver complexity* of an integer bimatrix A is defined as

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

We proceed to establish a result of [24] and its extension in [16] 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 $A_2 x^k = 0$ 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 $x = (x^1, \dots, x^n)$ *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 will indeed play an important role in its construction.

Consider any full vector $y = (y^1, \dots, y^m)$. 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 $n := k_1 + \dots + k_m \geq m$ and let x be the pure vector

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

We call the pure vector x an *expansion* of the full vector y , and we call 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 3.18 Consider any full $y = (y^1, \dots, y^m)$ and any expansion $x = (x^1, \dots, x^n)$ of y . If y is in the Graver basis $\mathcal{G}(A^{(m)})$ then x is in the Graver basis $\mathcal{G}(A^{(n)})$.

Proof. Let $x = (x^{1,1}, \dots, x^{m,k_m}) = (x^1, \dots, x^n)$ be an expansion of $y = (y^1, \dots, y^m)$ with $y^i = \sum_{j=1}^{k_i} x^{i,j}$ for each i . Suppose indirectly $y \in \mathcal{G}(A^{(m)})$ 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 exists an element $g = (g^{1,1}, \dots, g^{m,k_m})$ in $\mathcal{G}(A^{(n)})$ satisfying $g \sqsubset x$. Let $h = (h^1, \dots, h^m)$ be the compression of g defined by $h^i := \sum_{j=1}^{k_i} g^{i,j}$. Since $g \in \mathcal{L}^*(A^{(n)})$ we have $h \in \mathcal{L}^*(A^{(m)})$. But $h \sqsubset y$, contradicting $y \in \mathcal{G}(A^{(m)})$. This completes the proof. \square

Lemma 3.19 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 there is an element z of type m in some $\mathcal{G}(A^{(l)})$. Then its restriction $y = (y^1, \dots, y^m)$ to its m nonzero bricks is a full vector and is in the Graver basis $\mathcal{G}(A^{(m)})$. Let $x = (x^1, \dots, x^n)$ be any expansion of y . Then $\text{type}(z) = m \leq n = \text{type}(x)$, and by Lemma 3.18, 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 $x = (x^1, \dots, x^n)$ is a pure element in $\mathcal{G}(A^{(n)})$ for some n . Let $\mathcal{G}(A_2) = \{g^1, \dots, g^p\}$ 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, note that $A_1 G_2 v = A_1 \sum_{k=1}^n x^k = 0$ 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 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 the Graver basis $\mathcal{G}(A_1 G_2)$. \square

We proceed to establish the following theorem from [4] which asserts that Graver bases of n -fold products can be computed in polynomial time. An n -lifting of a vector $y = (y^1, \dots, y^m)$ consisting of m bricks is any vector $z = (z^1, \dots, z^n)$ consisting of n bricks such that for some $1 \leq k_1 < \dots < k_m \leq n$ we have $z^{k_i} = y^i$ for $i = 1, \dots, m$, and all other bricks of z are zero; in particular, $n \geq m$ and $\text{type}(z) = \text{type}(y)$.

Theorem 3.20 [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-encoding 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

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

Consider any n -lifting z of any $y \in \mathcal{G}(A^{(g)})$. Suppose indirectly $z \notin \mathcal{G}(A^{(n)})$. Then there exists $z' \in \mathcal{G}(A^{(n)})$ 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 $y \notin \mathcal{G}(A^{(g)})$. Then there exists $y' \in \mathcal{G}(A^{(g)})$ with $y' \sqsubset y$. But then the 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

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

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. \square

3.2.2 N-fold integer programming in polynomial time

Combining Theorem 3.20 and the results of §3.1 we now obtain Theorems 1.1–1.4.

Theorem 1.1 [4] *For each fixed integer $(r, s) \times t$ bimatrix A , there is an algorithm that, given positive integer n , $l, u \in \mathbb{Z}_\infty^{nt}$, $b \in \mathbb{Z}^{r+ns}$, 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,*

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

Proof. Compute the Graver basis $\mathcal{G}(A^{(n)})$ using the algorithm of Theorem 3.20. Now apply the algorithm of Theorem 3.12 with this Graver basis and solve the problem. \square

Theorem 1.2 [12] *For each fixed integer $(r, s) \times t$ bimatrix A , there is an algorithm that, given $n, l, u \in \mathbb{Z}_\infty^{nt}$, $b \in \mathbb{Z}^{r+ns}$, and separable convex $f : \mathbb{Z}^{nt} \rightarrow \mathbb{Z}$ presented by a comparison oracle, solves in time polynomial in n and $\langle l, u, b, \hat{f} \rangle$, the program*

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

Proof. Compute the Graver basis $\mathcal{G}(A^{(n)})$ using the algorithm of Theorem 3.20. Now apply the algorithm of Theorem 3.11 with this Graver basis and solve the problem. \square

Theorem 1.3 [12] *For each fixed integer $(r, s) \times t$ bimatrix A , there is an algorithm that, given positive integers n and p , $l, u \in \mathbb{Z}_{\infty}^{nt}$, $b \in \mathbb{Z}^{r+ns}$, and $\hat{x} \in \mathbb{Z}^{nt}$, solves in time polynomial in n , p , and $\langle l, u, b, \hat{x} \rangle$, the following distance minimization program,*

$$\min \{ \|x - \hat{x}\|_p : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, l \leq x \leq u \}. \quad (12)$$

For $p = \infty$ the problem (12) can be solved in time polynomial in n and $\langle l, u, b, \hat{x} \rangle$.

Proof. Compute the Graver basis $\mathcal{G}(A^{(n)})$ using the algorithm of Theorem 3.20. Now apply the algorithm of Theorem 3.13 with this Graver basis and solve the problem. \square

Theorem 1.4 [5] *For each fixed d and $(r, s) \times t$ integer bimatrix A , there is an algorithm that, given n , bounds $l, u \in \mathbb{Z}_{\infty}^{nt}$, integer $d \times nt$ matrix W , $b \in \mathbb{Z}^{r+ns}$, and convex function $f : \mathbb{Z}^d \rightarrow \mathbb{R}$ presented by a comparison oracle, solves in time polynomial in n and $\langle W, l, u, b \rangle$, the convex n -fold integer maximization program*

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

Proof. Compute the Graver basis $\mathcal{G}(A^{(n)})$ using the algorithm of Theorem 3.20. Now apply the algorithm of Theorem 3.16 with this Graver basis and solve the problem. \square

3.2.3 Weighted separable convex integer minimization

We proceed to establish Theorem 1.5 which is a broad extension of Theorem 1.2 that allows the objective function to include a composite term of the form $f(Wx)$, where $f : \mathbb{Z}^d \rightarrow \mathbb{Z}$ is a separable convex function and W is an integer matrix with d rows, and to incorporate inequalities on Wx . We begin with two lemmas. As before, \hat{f}, \hat{g} denote the maximum values of $|f(Wx)|, |g(x)|$ over the feasible set.

Lemma 3.21 *There is an algorithm that, given an integer $m \times n$ matrix A , an integer $d \times n$ matrix W , $l, u \in \mathbb{Z}_{\infty}^n$, $\hat{l}, \hat{u} \in \mathbb{Z}_{\infty}^d$, $b \in \mathbb{Z}^m$, the Graver basis $\mathcal{G}(B)$ of*

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

and separable convex functions $f : \mathbb{Z}^d \rightarrow \mathbb{Z}$, $g : \mathbb{Z}^n \rightarrow \mathbb{Z}$ 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

$$\min \{ f(Wx) + g(x) : x \in \mathbb{Z}^n, Ax = b, \hat{l} \leq Wx \leq \hat{u}, l \leq x \leq u \}. \quad (13)$$

Proof. Define $h : \mathbb{Z}^{n+d} \rightarrow \mathbb{Z}$ 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 (13) can be rewritten as

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

and the statement follows at once by applying Theorem 3.11 to this problem. \square

Lemma 3.22 *For every fixed integer $(r, s) \times t$ bimatrix A and $(p, q) \times t$ bimatrix W , there is an algorithm that, given any positive integer 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 whose blocks are defined by

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

Apply the algorithm of Theorem 3.20 and compute in polynomial time the Graver basis $\mathcal{G}(D^{(n)})$ of the n -fold product of D , which is the following matrix:

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

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

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

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 (the permutation of the rows does 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 $\mathcal{G}(C)$, indexed, according to the block structure, as $(x^1, x^2, \dots, x^n, y^1, y^2, \dots, y^n, z^1, z^2, \dots, z^n)$. Clearly, if $y^k = 0$ for $k = 2, \dots, n$ then the restriction $(x^1, x^2, \dots, x^n, y^1, z^1, z^2, \dots, z^n)$ of this element is in the Graver basis of B . On the other hand, if $(x^1, x^2, \dots, x^n, y^1, z^1, z^2, \dots, z^n)$ is any element in $\mathcal{G}(B)$ then its extension $(x^1, x^2, \dots, x^n, y^1, 0, \dots, 0, z^1, z^2, \dots, z^n)$ is clearly in $\mathcal{G}(C)$. So the Graver basis of B can be obtained in polynomial time by

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

This completes the proof. \square

Theorem 1.5 [13] *For each fixed integer $(r, s) \times t$ bimatrix A and integer $(p, q) \times t$ bimatrix W , there is an algorithm that, given $n, l, u \in \mathbb{Z}_\infty^{nt}, \hat{l}, \hat{u} \in \mathbb{Z}_\infty^{p+qn}, b \in \mathbb{Z}^{r+ns}$, and separable convex functions $f : \mathbb{Z}^{p+qn} \rightarrow \mathbb{Z}, g : \mathbb{Z}^{nt} \rightarrow \mathbb{Z}$ presented by evaluation oracles, solves in time polynomial in n and $\langle l, u, \hat{l}, \hat{u}, b, \hat{f}, \hat{g} \rangle$, the generalized program*

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

Proof. Use the algorithm of Lemma 3.22 to compute the Graver basis $\mathcal{G}(B)$ of

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

Now apply the algorithm of Lemma 3.21 and solve the nonlinear integer program. \square

4 Discussion

We conclude with a short discussion of the universality of n -fold integer programming and the Graver complexity of (directed) graphs, a new important invariant which controls the complexity of our multiway table and multicommodity flow applications.

4.1 Universality of N-Fold Integer Programming

Let us introduce the following notation. For an integer $s \times t$ matrix D , let $\boxplus D$ denote the $(t, s) \times t$ bimatrix whose first block is the $t \times t$ identity matrix and whose

second block is D . Consider the following special form of the n -fold product, defined for a matrix D , by $D^{[n]} := (\boxminus D)^{(n)}$. We consider such m -fold products of the 1×3 matrix $\mathbf{1}_3 := [1, 1, 1]$. Note that $\mathbf{1}_3^{[m]}$ is precisely the $(3 + m) \times 3m$ incidence matrix of the complete bipartite graph $K_{3,m}$. For instance, for $m = 3$, it is the matrix

$$\mathbf{1}_3^{[3]} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

We can now rewrite Theorem 2.1 in the following compact and elegant form.

The Universality Theorem [7] *Every rational polytope $\{y \in \mathbb{R}_+^d : Ay = b\}$ stands in polynomial time computable integer preserving bijection with some polytope*

$$\left\{ x \in \mathbb{R}_+^{3mn} : \mathbf{1}_3^{[m][n]}x = a \right\}. \quad (14)$$

The bijection constructed by the algorithm of this theorem is, moreover, a simple projection from \mathbb{R}^{3mn} to \mathbb{R}^d that erases all but some d coordinates (see [7]). For $i = 1, \dots, d$ let $x_{\sigma(i)}$ be the coordinate of x that is mapped to y_i under this projection. Then any linear or nonlinear integer program $\min\{f(y) : y \in \mathbb{Z}_+^d, Ay = b\}$ can be lifted in polynomial time to the following integer program over a simple $\{0, 1\}$ -valued matrix $\mathbf{1}_3^{[m][n]}$ which is completely determined by two parameters m and n only,

$$\min \left\{ f(x_{\sigma(1)}, \dots, x_{\sigma(d)}) : x \in \mathbb{Z}_+^{3mn}, \mathbf{1}_3^{[m][n]}x = a \right\}. \quad (15)$$

This also shows the universality of n -fold integer programming: every linear or nonlinear integer program is equivalent to an n -fold integer program over some bimatrix $\boxminus \mathbf{1}_3^{[m]}$ which is completely determined by a single parameter m .

Moreover, for every fixed m , program (15) can be solved in polynomial time for linear forms and broad classes of convex and concave functions by Theorems 1.1–1.5.

4.2 Graver Complexity of Graphs and Digraphs

The significance of the following new (di)-graph invariant will be explained below.

Definition 4.1 [1] 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 .

One major task done by our algorithms for linear and nonlinear n -fold integer programming over a bimatrix A is the construction of the Graver basis $\mathcal{G}(A^{(n)})$ in time $O(n^{g(A)})$ with $g(A)$ the Graver complexity of A (see proof of Theorem 3.20).

Since the bimatrix underlying the universal n -fold integer program (15) is precisely $\boxminus D$ with $D = \mathbf{1}_3^{[m]}$ the incidence matrix of $K_{3,m}$, it follows that the complexity of computing the relevant Graver bases for this program for fixed m and variable n is $O(n^{g(K_{3,m})})$ where $g(K_{3,m})$ is the Graver complexity of $K_{3,m}$ as just defined.

Turning to the many-commodity transshipment problem over a digraph G discussed in §2.2.1, the bimatrix underlying the n -fold integer program (2) in the proof of Theorem 2.6 is precisely $\boxminus D$ with D the incidence matrix of G , and so it follows that the complexity of computing the relevant Graver bases for this program is $O(n^{g(G)})$ where $g(G)$ is the Graver complexity of the digraph G as just defined.

So the Graver complexity of a (di)-graph controls the complexity of computing the Graver bases of the relevant n -fold integer programs, and hence its significance.

Unfortunately, our present understanding of the Graver complexity of (di)-graphs is very limited and much more study is required. Very little is known even for the complete bipartite graphs $K_{3,m}$: while $g(K_{3,3}) = 9$, already $g(K_{3,4})$ is unknown. See [1] for more details and a lower bound on $g(K_{3,m})$ which is exponential in m .

Acknowledgements

I thank Jon Lee and Sven Leyffer for inviting me to write this article. I am indebted to Jesus De Loera, Raymond Hemmecke, Uriel Rothblum and Robert Weismantel for their collaboration in developing the theory of n -fold integer programming, and to Raymond Hemmecke for his invaluable suggestions. The article was written mostly while I was visiting and delivering the Nachdiplom Lectures at ETH Zürich during Spring 2009. I thank the following colleagues at ETH for useful feedback: David Adjiashvili, Jan Foniok, Martin Fuchsberger, Komei Fukuda, Dan Hefetz, Hans-Rudolf Künsch, Hans-Jakob Lüthi, and Philipp Zumstein. I also thank Dorit Hochbaum, Peter Malkin, and a referee for useful remarks.

References

- [1] Berstein, Y., Onn, S.: The Graver complexity of integer programming. *Annals Combin.* 13 (2009) 289–296
- [2] Cook, W., Fonlupt, J., Schrijver, A.: An integer analogue of Carathéodory’s theorem. *J. Comb. Theory Ser. B* 40 (1986) 63–70

- [3] Cox L.H.: On properties of multi-dimensional statistical tables. *J. Stat. Plan. Infer.* 117 (2003) 251–273
- [4] De Loera, J., Hemmecke, R., Onn, S., Weismantel, R.: N-fold integer programming. *Disc. Optim.* 5 (Volume in memory of George B. Dantzig) (2008) 231–241
- [5] De Loera, J., Hemmecke, R., Onn, S., Rothblum, U.G., Weismantel, R.: Convex integer maximization via Graver bases. *J. Pure App. Algeb.* 213 (2009) 1569–1577
- [6] De Loera, J., Onn, S.: The complexity of three-way statistical tables. *SIAM J. Comp.* 33 (2004) 819–836
- [7] De Loera, J., Onn, S.: All rational polytopes are transportation polytopes and all polytopal integer sets are contingency tables. In: Proc. IPCO 10 – Symp. on Integer Programming and Combinatorial Optimization (Columbia University, New York). *Lec. Not. Comp. Sci.*, Springer 3064 (2004) 338–351
- [8] De Loera, J., Onn, S.: Markov bases of three-way tables are arbitrarily complicated. *J. Symb. Comp.* 41 (2006) 173–181
- [9] Fienberg, S.E., Rinaldo, A.: Three centuries of categorical data analysis: Log-linear models and maximum likelihood estimation. *J. Stat. Plan. Infer.* 137 (2007) 3430–3445
- [10] Gordan, P.: Über die Auflösung linearer Gleichungen mit reellen Coefficienten. *Math. Annalen* 6 (1873) 23–28
- [11] Graver, J.E.: On the foundations of linear and linear integer programming I. *Math. Prog.* 9 (1975) 207–226
- [12] Hemmecke, R., Onn, S., Weismantel, R.: A polynomial oracle-time algorithm for convex integer minimization. *Math. Prog.* To appear
- [13] Hemmecke, R., Onn, S., Weismantel, R.: Multicommodity flow in polynomial time. Submitted
- [14] Hochbaum, D.S., Shanthikumar, J.G.: Convex separable optimization is not much harder than linear optimization. *J. Assoc. Comp. Mach.* 37 (1990) 843–862
- [15] Hoffman, A.J., Kruskal, J.B.: Integral boundary points of convex polyhedra. In: *Linear inequalities and Related Systems*, Ann. Math. Stud. 38 223–246, Princeton University Press, Princeton, NJ (1956)
- [16] Hosten, S., Sullivant, S.: Finiteness theorems for Markov bases of hierarchical models. *J. Comb. Theory Ser. A* 114 (2007) 311–321

- [17] Irving, R., Jerrum, M.R.: Three-dimensional statistical data security problems. *SIAM J. Comp.* 23 (1994) 170–184
- [18] Lenstra Jr., H.W.: Integer programming with a fixed number of variables. *Math. Oper. Res.* 8 (1983) 538–548
- [19] Motzkin, T.S.: The multi-index transportation problem. *Bull. Amer. Math. Soc.* 58 (1952) 494
- [20] Onn, S.: Entry uniqueness in margined tables. In: Proc. PSD 2006 – Symp. on Privacy in Statistical Databases (Rome, Italy). Lec. Not. Comp. Sci., Springer 4302 (2006) 94–101
- [21] Onn, S.: Convex discrete optimization. In: *Encyclopedia of Optimization*, Springer (2009) 513–550
- [22] Onn, S.: Nonlinear discrete optimization. *Nachdiplom Lectures*, ETH Zürich, Spring 2009, available online at <http://ie.technion.ac.il/~onn/Nachdiplom>
- [23] Onn, S., Rothblum, U.G.: Convex combinatorial optimization. *Disc. Comp. Geom.* 32 (2004) 549–566
- [24] Santos, F., Sturmfels, B.: Higher Lawrence configurations. *J. Comb. Theory Ser. A* 103 (2003) 151–164
- [25] Schrijver, A.: *Theory of Linear and Integer Programming*. Wiley, New York (1986)
- [26] Sebö, A.: Hilbert bases, Carathéodory's theorem and combinatorial optimization. In: Proc. IPCO 1 - 1st Conference on Integer Programming and Combinatorial Optimization (R. Kannan and W.R. Pulleyblank Eds.) (1990) 431–455
- [27] Vlach, M.: Conditions for the existence of solutions of the three-dimensional planar transportation problem. *Disc. App. Math.* 13 (1986) 61–78
- [28] Yemelichev, V.A., Kovalev, M.M., Kravtsov, M.K.: *Polytopes, Graphs and Optimisation*. Cambridge University Press, Cambridge (1984)

Shmuel Onn

Technion - Israel Institute of Technology, 32000 Haifa, Israel

and

ETH Zürich, 8092 Zürich, Switzerland

onn@ie.technion.ac.il, http://ie.technion.ac.il/~onn