Available online at www.sciencedirect.com
ScienceDirect
Discrete Optimization 5 (2008) 231–241
DISCRETE OPTIMIZATION
www.elsevier.com/locate/disopt

N-fold integer programming

Jesús A. De Loera[a], Raymond Hemmecke[b], Shmuel Onn[c,*], Robert Weismantel[b]

[a] *University of California at Davis, Davis, CA 95616, USA*
[b] *Otto-von-Guericke Universität Magdeburg, D-39106 Magdeburg, Germany*
[c] *Technion - Israel Institute of Technology, 32000 Haifa, Israel*

Received 27 September 2005; accepted 10 June 2006
Available online 31 October 2007

Abstract

In this article we study a broad class of integer programming problems in variable dimension. We show that these so-termed *n-fold integer programming problems* are polynomial time solvable. Our proof involves two heavy ingredients discovered recently: the equivalence of linear optimization and the so-called directed augmentation, and the stabilization of certain Graver bases.

We discuss several applications of our algorithm to multiway transportation problems and to packing problems. One important consequence of our results is a polynomial time algorithm for the d-dimensional integer transportation problem for long multiway tables. Another interesting application is a new algorithm for the classical cutting-stock problem.

Keywords: Integer programming; Graver basis; Graver complexity; Markov basis; Markov complexity; Hilbert basis; Contingency table; Multiway table; Transportation polytope; Transportation problem; Bin packing; Cutting stock; Augmentation; Test set; Primal method; n-fold; Confidentiality; Privacy; Combinatorial optimization; Computational complexity

1. Introduction

The integer programming problem is the following discrete optimization problem, where $\mathbb{N}$ denotes the set of nonnegative integers, A is an integer matrix and b, c are integer vectors of suitable dimensions:

$$\min\{cx\colon Ax = b, x \in \mathbb{N}^q\}.$$

It is well-known to be generally NP-hard but polynomial time solvable in fixed dimension q, see [18].

In this article, motivated by the applications to high-dimensional transportation problems and contingency tables and by the recently discovered striking universality theorem for rational polytopes [6] (see Section 2), we study the following class of integer programming problems in variable dimension.

* Corresponding author.
E-mail addresses: deloera@math.ucdavis.edu (J.A. De Loera), hemmecke@imo.math.uni-magdeburg.de (R. Hemmecke), onn@ie.technion.ac.il (S. Onn), weismantel@imo.math.uni-magdeburg.de (R. Weismantel).
URLs: http://www.math.ucdavis.edu/~deloera (J.A. De Loera), http://www.math.uni-magdeburg.de/~hemmecke (R. Hemmecke), http://ie.technion.ac.il/~onn (S. Onn), http://www.math.uni-magdeburg.de/~weismant (R. Weismantel).

1572-5286/$ - see front matter
doi:10.1016/j.disopt.2006.06.006

The n-fold integer programming problem. Fix a $p \times q$ integer matrix A. Given positive integer n and integer vectors $b = (b^0, b^1, \ldots, b^n)$ and $c = (c^1, \ldots, c^n)$, where $b^0 \in \mathbb{Z}^q$, and $b^k \in \mathbb{Z}^p$ and $c^k \in \mathbb{Z}^q$ for $k = 1, \ldots, n$, find a nonnegative integer vector $x = (x^1, \ldots, x^n)$, where $x^k \in \mathbb{N}^q$ for $k = 1, \ldots, n$, which minimizes $cx = \sum_{k=1}^n c^k x^k$ subject to the equations $\sum_{k=1}^n x^k = b^0$ and $Ax^k = b^k$ for $k = 1, \ldots, n$.
The term "n-fold integer programming" refers to the problem being almost separable into n similar programs $\min\{c^k x : Ax^k = b^k, x^k \in \mathbb{N}^q\}$ in fixed dimension; however, the constraint $\sum_{k=1}^n x^k = b^0$ binds these programs together, and the result is an integer program in large variable dimension nq.
Let the *n-fold matrix* of A be the following $(q + np) \times nq$ matrix, with I_q the $q \times q$ identity matrix:

$$A^{(n)} := (\mathbf{1}_n \otimes I_q) \oplus (I_n \otimes A) = \begin{pmatrix} I_q & I_q & I_q & \cdots & I_q \\ A & 0 & 0 & \cdots & 0 \\ 0 & A & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A \end{pmatrix}.$$

Then the n-fold integer programming problem can be conveniently written in matrix form as

$$\min\{cx : A^{(n)} x = b, x \in \mathbb{N}^{nq}\}.$$

In this article we establish the following theorem. Naturally, the input size is n plus the bit size of the integer objective vector $c \in \mathbb{Z}^{nq}$ and the integer right-hand side vector $b \in \mathbb{Z}^{q+np}$.

Theorem 1.1. *Fix any integer matrix A. Then there is a polynomial time algorithm that, given any n and any integer vectors b and c, solves the corresponding n-fold integer programming problem.*

The proof of this theorem involves two heavy ingredients. First, it makes use of the equivalence of the linear optimization problem and the directed augmentation problem, recently introduced and studied in [23]. Second, it uses recent results of [15,21] on the stabilization of certain Graver bases.

One important consequence of Theorem 1.1 is a polynomial time algorithm for the 3-way transportation problem for long tables, settling its computational complexity; see Section 2 for details.

Corollary 2.2. *Fix any r, s. Then there is a polynomial time algorithm that, given l, integer objective vector c, and integer line sums $(u_{i,j})$, $(v_{i,k})$ and $(w_{j,k})$, solves the integer transportation problem*

$$\min\left\{cx : x \in \mathbb{N}^{r \times s \times l}, \sum_i x_{i,j,k} = w_{j,k}, \sum_j x_{i,j,k} = v_{i,k}, \sum_k x_{i,j,k} = u_{i,j}\right\}.$$

The n-fold integer programming problem and theorem can be generalized as follows.
Generalized n-fold integer programming. Fix integer matrices A, B of sizes $r \times q$ and $s \times q$, respectively. Given positive integer n and integer vectors $b = (b^0, b^1, \ldots, b^n)$ and $c = (c^1, \ldots, c^n)$, with $b^0 \in \mathbb{Z}^s$, and $b^k \in \mathbb{Z}^r$ and $c^k \in \mathbb{Z}^q$ for $k = 1, \ldots, n$, find $x = (x^1, \ldots, x^n)$ with $x^k \in \mathbb{N}^q$ for $k = 1, \ldots, n$, which minimizes $cx = \sum_{k=1}^n c^k x^k$ subject to $\sum_{k=1}^n Bx^k = b^0$ and $Ax^k = b^k$ for $k = 1, \ldots, n$.
We have the following more general result, from which Theorem 1.1 is deduced in the case $B = I_q$.

Theorem 1.2. *Fix any pair of integer matrices A, B of compatible sizes. Then there is a polynomial time algorithm that solves the generalized n-fold integer programming problem on any input n, b, c.*

The article is organized as follows. In Section 2 we discuss applications of Theorem 1.1 to multiway transportation problems and to some packing problems, as follows. In Section 2.1 we obtain the aforementioned Corollary 2.2 which provides a polynomial time solution to 3-way integer transportation problems for long tables, contrasting the recent universality theorem of [6,8] for slim tables. We also extend this result to d-way transportation problems for long tables of any dimension (Corollary 2.4). In Section 2.2 we describe applications to a certain shipment problem (Corollary 2.5) and to the classical cutting-stock problem of [10] (Corollary 2.6). In Sections 3 and 4 we develop the

necessary ingredients for our n-fold integer programming algorithm, as follows. In Section 3 we discuss Graver bases and augmentation. We show (Lemma 3.2) that the Graver basis allows to solve the so-called directed augmentation problem, and, combining this with the results of [23], shows that any feasible solution to an integer program can be augmented to an optimal one in polynomial time provided the Graver basis is part of the input (Theorem 3.3). In Section 4 we discuss the stabilization of Graver bases discovered recently in [15,21], and use it to show that Graver bases of n-fold matrices can be computed in polynomial time (Theorem 4.2). Finally, in Section 5, we combine all the ingredients from Section 3 and 4, and prove our main result Theorem 1.2 and its specialization Theorem 1.1.

2. Applications

2.1. High-dimensional transportation problems

A *d-way transportation polytope* is the set of all $m_1 \times \cdots \times m_d$ nonnegative arrays $x = (x_{i_1,\ldots,i_d})$ such that the sums of the entries over some of their lower-dimensional subarrays (*margins*) are specified. For simplicity of exposition, we shall concentrate here only on d-way *line-sum polytopes*, of the form

$$T = \left\{ x \in \mathbb{R}_+^{m_1 \times \cdots \times m_d} : \sum_{i_1} x_{i_1,\ldots,i_d} = u_{i_2,\ldots,i_d}, \sum_{i_2} x_{i_1,\ldots,i_d} = u_{i_1,i_3,\ldots,i_d}, \ldots, \sum_{i_d} x_{i_1,\ldots,i_d} = u_{i_1,\ldots,i_{d-1}} \right\}.$$

Transportation polytopes and their integer points (called contingency tables by statisticians), have been studied and used extensively in the operations research literature and in the context of secure statistical data disclosure by public agencies such as the census bureau and the national center for health statistics. In the operations research literature, one is typically interested in the integer and linear transportation problems, which are the integer and linear programming problems over the transportation polytope, see e.g. [2,17,19,20,24,25] and references therein. In the statistics community, one is often interested in the values an entry can attain in all tables with fixed margins, related to the security of the entry under margin disclosure, and in the construction of a Markov basis allowing a random walk on the set of tables with fixed margins, see e.g. [1,3,4,7,9,16] and references therein.

It is well-known that the system defining a 2-way transportation polytope is totally unimodular. This implies that all the above problems are easy in this case. However, already 3-way transportation problems are much harder. Consider the problem of deciding if a given 3-way line-sum polytope of $r \times s \times l$ arrays (with r rows, s columns and l layers) contains an integer point: the computational complexity of this problem provides useful indication about the difficulty of the problems mentioned above. If r, s, l are all fixed, then the problem is solvable in polynomial time by integer programming in fixed dimension rsl. On the other hand, if r, s, l are all variable part of the input, then the problem is NP-complete [16]. The in-between cases are much more delicate. The case of two parameters r, s variable and one parameter l fixed was recently resolved in [5], where it was shown to be NP-complete, strengthening [16]. Moreover, very recently, in [6,8], the following universality result was shown.

Proposition 2.1. *Any rational polytope $P = \{y \in \mathbb{R}_+^n : Ay = b\}$ is polynomial time representable as a 3-way line-sum transportation polytope of size $r \times s \times 3$ for some (polynomially bounded) r and s,*

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

Here representable means that there is a coordinate-erasing projection from $\mathbb{R}^{r \times s \times 3}$ onto $\mathbb{R}^n$ providing a bijection between T and P and between the sets of integer points $T \cap \mathbb{Z}^{r \times s \times 3}$ and $P \cap \mathbb{Z}^n$. Thus, *any* rational polytope is an $r \times s \times 3$ line-sum polytope, and *any* integer (respectively, linear) programming problem is equivalent to an integer (respectively, linear) $r \times s \times 3$ line-sum transportation problem. This result led to the solution of several open problems from [24,25] and had several implications on the complexity of Markov bases and the entry security problem, see [6–8] for more details.

However, the last case, of two parameters r, s fixed and one parameter l variable, has remained open and intriguing. Here, as a consequence of Theorem 1.1, we are able to resolve this problem and show that both the decision and optimization problems are polynomial time solvable.

Corollary 2.2. *Fix any r, s. Then there is a polynomial time algorithm that, given l, integer objective vector c, and integer line sums $(u_{i,j})$, $(v_{i,k})$ and $(w_{j,k})$, solves the integer transportation problem*

$$\min\left\{cx : x \in \mathbb{N}^{r\times s\times l}, \sum_i x_{i,j,k} = w_{j,k}, \sum_j x_{i,j,k} = v_{i,k}, \sum_k x_{i,j,k} = u_{i,j}\right\}.$$

Proof. We formulate the 3-way integer transportation problem as an n-fold integer program with $n := l$, $p := r + s$, and $q := r \cdot s$. Reindex the variables as $x^k_{i,j} := x_{i,j,k}$ so that the variables vector is $x = (x^1, \ldots, x^n)$ with $x^k = (x^k_{i,j}) \in \mathbb{N}^{r\times s}$ a 2-way $r \times s$ table, the kth layer of the 3-way table x. Similarly write $c = (c^1, \ldots, c^n)$ with $c^k \in \mathbb{Z}^{r\times s}$ for the objective vector. Next, put $b := (b^0, b^1, \ldots, b^n)$, with $b^0 \in \mathbb{N}^{rs}$ defined by $b^0 := (u_{i,j})$, and $b^k \in \mathbb{N}^{r+s}$ defined by $b^k := ((v_{i,k}), (w_{j,k}))$ for $k = 1, \ldots, n$. Finally, let A be the $p \times q = (r + s) \times r \cdot s$ matrix of equations for the usual 2-way transportation polytope, forcing row sums and column sums on each of the $r \times s$ layers x^k by $Ax^k = b^k$, $k = 1, \ldots, n$. Then the equation $Ax^k = b^k$ force the line sums $v_{i,k}$ and $w_{j,k}$, and the additional n-fold integer program binding constraint $\sum_{k=1}^n x^k = b^0$ forces the "long" line sums $u_{i,j}$. This completes the encoding. Since r, s are fixed, so are p, q, and A, and therefore, the corollary follows from Theorem 1.1. □

Example 2.3. Consider the case $r = s = 3$ (the smallest where the problem is genuinely 3-dimensional). Then $p = 6$, $q = 9$, and writing $x^k = (x^k_{1,1}, x^k_{1,2}, x^k_{1,3}, x^k_{2,1}, x^k_{2,2}, x^k_{2,3}, x^k_{3,1}, x^k_{3,2}, x^k_{3,3})$, the matrix A which defines the n-fold program providing the formulation of the $3 \times 3 \times l$ transportation problem is

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Already for this case, of $3 \times 3 \times l$ tables, the only polynomial time algorithm for the corresponding line-sum integer transportation problem we are aware of is the one guaranteed by Corollary 2.2 above.

Corollary 2.2 extends to transportation problems of any dimension d, for *long* tables, namely, of size $m_1 \times \cdots \times m_{d-1} \times l$, where $m_1, \ldots, m_{d-1}$ are fixed and only the length l is variable, as follows.

Corollary 2.4. *Fix $d, m_1, \ldots, m_{d-1}$. Then there is a polynomial time algorithm that, given l, integer objective c, and line sums $(u_{i_2,\ldots,i_d}), \ldots, (u_{i_1,\ldots,i_{d-1}})$, solves the long multiway transportation problem*

$$\min\left\{cx : x \in \mathbb{N}^{m_1\times\cdots\times m_{d-1}\times l} : \sum_{i_1} x_{i_1,\ldots,i_d} = u_{i_2,\ldots,i_d}, \ldots, \sum_{i_d} x_{i_1,\ldots,i_d} = u_{i_1,\ldots,i_{d-1}}\right\}.$$

Proof. The long multiway transportation problem can be encoded as an n-fold integer program with $n := l$, $p := \sum_{i=1}^{d-1} m_i$, and $q := \prod_{i=1}^{d-1} m_i$, by reindexing the variables as $x^{i_d}_{i_1,\ldots,i_{d-1}} := x_{i_1,\ldots,i_d}$, letting A be the matrix of equations of line sums of $(d-1)$-way transportation polytope of $m_1 \times \cdots \times m_{d-1}$ arrays, and proceeding in direct analogy to the proof of Corollary 2.2. The details are omitted. □

2.2. Some packing problems

2.2.1. Minimum cost shipment

The minimum cost shipment problem concerns the shipment of a large stock of items of several types, using various vessels, with minimum possible cost. More precisely, the data is as follows. There are t types of items. The weight of each item of type j is w_j and there are n_j items of type j to be shipped. There are v available vessels, where vessel k has maximum weight capacity u_k. The cost of shipping one item of type j on vessel k is $p_{j,k}$. We now formulate this as an n-fold integer programming problem. We set $n := v$, $p := 1$, $q := t + 1$. The defining matrix is the row vector $A = (A_j) := (w_1, w_2, \ldots, w_t, 1)$. The variables vector is $x = (x^1, \ldots, x^n)$ with $x^k = (x^k_1, \ldots, x^k_t, x^k_q)$, where x^k_j

represents the number of items of type j to be shipped on vessel k for $j = 1, \ldots, t$, and x_q^k is an extra slack variable representing the unused weight capacity in vessel k. The cost vector is $c = (c^1, \ldots, c^n)$ with $c^k = (c_1^k, \ldots, c_t^k, c_q^k)$, where $c_j^k := p_{j,k}$ for $j = 1, \ldots, t$, and $c_q^k := 0$. Finally, the demand vector is $b = (b^0, b^1, \ldots, b^n)$ with $b^k := u_k$ for $k = 1, \ldots, n$, and $b^0 := (n_1, \ldots, n_t, \sum_{k=1}^{v} u_k - \sum_{j=1}^{t} n_j w_j)$. Then the resulting n-fold integer programming problem, $\min\{cx : A^{(n)}x = b, x \in \mathbb{N}^{nq}\}$, can be written in scalar form as follows:

$$\min \sum_{j=1}^{q}\sum_{k=1}^{n} c_j^k x_j^k = \sum_{j=1}^{t}\sum_{k=1}^{v} p_{j,k} x_j^k$$

$$\text{s.t.} \sum_{k=1}^{n} x_j^k = b_j^0 = n_j, \quad j = 1, \ldots, t$$

$$\sum_{k=1}^{n} x_q^k = b_q^0 = \sum_{k=1}^{v} u_k - \sum_{j=1}^{t} n_j w_j$$

$$\sum_{j=1}^{q} A_j x_j^k = \sum_{j=1}^{t} w_j x_j^k + x_q^k = b^k = u_k, \quad k = 1, \ldots, n$$

$$x_j^k \in \mathbb{N}, \quad j = 1, \ldots, q, k = 1, \ldots, n.$$

Assume that the number t of types is fixed, but the numbers n_j of items of each type may be very large: this is a reasonable assumption in applications (for instance, several types of automobiles to be shipped overseas, or several types of appliances to be shipped on ground). Then we obtain the following corollary of Theorem 1.1, showing that the problem is polynomial time solvable, where the input size is v plus the bit size of the integer numbers n_j, u_k, $p_{j,k}$ constituting the data. Note that this result is *much stronger* than the standard results on the *pseudo-polynomial time* solvability of this kind of packing and knapsack-type problems using dynamic programming: our algorithm can handle *very large* n_j and u_k, possibly exponential in the dimensional parameter v.

Corollary 2.5. *For any fixed number t of types and type weights w_j, the minimum cost shipment problem is solvable in time which is polynomial in the number v of vessels and in the bit size of the integer numbers n_j of items of each type to be shipped, vessel capacities u_k, and shipment costs $p_{j,k}$.*

2.2.2. The cutting-stock problem

This is a classical manufacturing problem [10], where the usual setup is as follows: a manufacturer supplies rolls of material (such as scotch-tape or band-aid) in one of t different widths $w_1, \ldots, w_t$. The rolls are all cut out from standard rolls of common large width u, coming out of the production line. Given orders by customers for n_j rolls of width w_j, the problem facing the manufacturer is to meet the orders using the smallest possible number of standard rolls. This is almost a direct special case of the minimum cost shipment problem discussed above, with sufficiently many identical vessels, say $v := \sum_{j=1}^{t} \lceil n_j / \lfloor u/w_j \rfloor \rceil$, of capacity $u_k := u$ each, playing the role of the standard rolls, and with cost $p_{j,k} := w_j$ for each roll of width w_j regardless of the standard roll from which it is being cut out. The only correction needed is that each slack variable $x_q^k = x_{t+1}^k$, measuring the unused width of the kth standard roll, has cost of one unit instead of zero, so that the total cost becomes the number of standard rolls used. Thus the formulation as an n-fold program is with $n := \sum_{j=1}^{t} \lceil n_j / \lfloor u/w_j \rfloor \rceil$, $p := 1$, $q := t + 1$, $A := (w_1, w_2, \ldots, w_t, 1)$, variables x_j^k representing the number of rolls of width w_j cut out of the kth roll for $j = 1, \ldots, t$ and x_q^k representing the unused width of the kth standard roll, costs $c_j^k := w_j$ for $j = 1, \ldots, t$ and $c_q^k := 1$, and demands $b^k := u$ for $k = 1, \ldots, n$ and $b^0 := (n_1, \ldots, n_t, nu - \sum_{j=1}^{t} n_j w_j)$.

Again, quite surprisingly, we get the following useful corollary regarding this classical problem.

Corollary 2.6. *For any fixed t and widths $w_1, \ldots, w_t$, the cutting-stock problem is solvable in time polynomial in $\sum_{j=1}^{t} \lceil n_j / \lfloor u/w_j \rfloor \rceil$ and in the bit size of the numbers n_j of orders and raw roll width u.*

One common approach to the cutting-stock problem makes use of the so-called *cutting patterns*, which are feasible solutions of the knapsack problem $\{y \in \mathbb{N}^t : \sum_{j=1}^{t} w_j y_j \leq u\}$. This is useful when the width u of the standard

rolls is of the same order of magnitude as the demand widths w_j. However, when u is much larger than the w_j, the number of cutting patterns becomes prohibitively large to handle. But then the values $\lfloor u/w_j \rfloor$ are large and hence $n := \sum_{j=1}^{t} \lceil n_j / \lfloor u/w_j \rfloor \rceil$ is small, in which case the result of Corollary 2.6 using n-fold integer programming becomes particularly appealing.

3. Graver bases and directed augmentation

Consider the following family IP_A of integer programs in standard form, with arbitrary demand vector $b \in \mathbb{Z}^m$ and arbitrary objective vector $c \in \mathbb{Z}^n$, sharing the same constraint matrix $A \in \mathbb{Z}^{m \times n}$,

$$\mathrm{IP}_A(b, c)\colon \min\{cx\colon Ax = b, x \in \mathbb{N}^n\}.$$

A *universal test set* for the family IP_A is a *finite* subset G of the lattice $\mathcal{L}(A) := \{x \in \mathbb{Z}^n\colon Ax = 0\}$ of dependencies on A such that whenever x is feasible but not optimal for $\mathrm{IP}_A(b, c)$ (where $b := Ax$), there is an *improving direction* $g \in G$, namely such that $x - g$ is feasible and better, that is, $x - g \in \mathbb{N}^n$ and $cg > 0$. Thus, a universal test set enables the solution of the following augmentation problem.

Augmentation problem. Given $A \in \mathbb{Z}^{m \times n}$, $x \in \mathbb{N}^n$ and $c \in \mathbb{Z}^n$, either find an improving direction $g \in \mathbb{Z}^n$, namely one with $x - g \in \{y \in \mathbb{N}^n : Ay = Ax\}$ and $cg > 0$, or assert that no such g exists.

An *augmentation oracle for a matrix* A is one that solves the augmentation problem, that is, when queried on $x \in \mathbb{N}^n$ and $c \in \mathbb{Z}^n$, it either returns an improving direction g or asserts that none exists. Clearly, an explicit universal test set G for A enables the efficient realization of an augmentation oracle for A by simply searching for an improving direction $g \in G$. An oracle solving the augmentation problem, and in particular, an explicit universal test set G, enable the following simple iterative procedure that, for any program $\mathrm{IP}_A(b, c)$ in IP_A with bounded objective function, converts any feasible x to an optimal one: "while there exists an improving direction g set $x := x - g$ and repeat".

In 1975, Graver [11] constructed, for *every* integer matrix A, a canonical universal test set. The *Graver basis* $\mathcal{G}(A)$ of A can be defined as follows. First, we need to extend the partial ordering $\leq$ from $\mathbb{N}^n$ to $\mathbb{Z}^n$. For $u, v \in \mathbb{Z}^n$ we say that u is *conformal* to v, denoted $u \sqsubseteq v$, if $|u_i| \leq |v_i|$ and $u_i v_i \geq 0$ for $i = 1, \ldots, n$, that is, u and v lie in the same orthant of $\mathbb{R}^n$ and each component of u is bounded by the corresponding component of v in absolute value. With this, $\mathcal{G}(A)$ consists precisely of all $\sqsubseteq$-minimal vectors in $\mathcal{L}(A) \setminus \{0\}$. For a more detailed introduction of Graver bases we refer to [12]. The currently fastest algorithm to compute Graver bases, based on a completion procedure and a project-and-lift approach, is described in [13] and implemented in the software package `4ti2` [14].

The Graver basis, being a universal test set, provides an augmentation oracle and hence enables us to convert any feasible solution to an optimal one for any $\mathrm{IP}_A(b, c)$ by the iterative augmentation procedure above. But this in itself is not enough to guarantee an *efficient* (polynomial time) solution: a major remaining question is how many augmentation steps are needed to reach an optimal solution.

Recently, in [23], a directed version of the augmentation problem was introduced; quite remarkably, it was shown that the number of *directed* augmentation steps needed to reach optimality is polynomial. We discuss this next. First, we define the directed augmentation problem.

Directed augmentation problem. Given $A \in \mathbb{Z}^{m \times n}$, $x \in \mathbb{N}^n$ and $c', c'' \in \mathbb{Z}^n$, either find $g = g^+ - g^- \in \mathbb{Z}^n$ satisfying $x - g \in \{y \in \mathbb{N}^n : Ay = Ax\}$ and $c'g^+ - c''g^- > 0$, or assert that none exists.

Here and throughout, $g^+, g^- \in \mathbb{N}^n$ denote the *positive* and *negative* parts of $g \in \mathbb{Z}^n$, defined by $g_i^+ := \max\{g_i, 0\}$ and $g_i^- := -\min\{g_i, 0\}$ for $i = 1, \ldots, n$. Thus, the directed augmentation problem involves *two* objective function vectors: c' controls the cost of the positive part of g and c'' controls the cost of the negative part of g. The usual augmentation problem occurs as the special case $c' = c'' = c$. A *directed augmentation oracle for A* is one that solves the directed augmentation problem, i.e. when queried on $x \in \mathbb{N}^n$, $c', c'' \in \mathbb{Z}^n$, it either returns an improving direction g or asserts that none exists.

In [23], it was assumed that the input includes an upper bound vector $u \in \mathbb{N}^n$ on the variables, so that the actual feasible set is $\{y \in \mathbb{N}^n : Ay = Ax,\ y \leq u\}$. Under this assumption, the feasible region is always bounded and there is always an optimal solution. Further, the complexity estimates in [23] depended on the bit size of u and $b := Ax$. However, this is not really needed. Consider the integer program $\mathrm{IP}_A(b, c)\colon \min\{cy\colon Ay = b,\ y \in \mathbb{N}^n\}$ with $b = Ax$. Its objective function is bounded (and hence there is an optimal solution) if and only if it is bounded for the corresponding LP-relaxation $\min\{cy\colon Ay = b,\ y \in \mathbb{R}^n_+\}$, which can be checked in polynomial time by linear programming. Furthermore, whenever $\mathrm{IP}_A(b, c)$ has an optimal solution, it has one of bit size polynomially bounded

in the size of A and $b = Ax$; this basically follows from Cramer's rule, see e.g. [22, Section 17.1]. Therefore, it is possible to compute an upper bound u in terms of A and x only, and plug it into the analysis of [23]; for instance, $u_k := m!\,n(n+1)(\max|x_j|)(\max|A_{i,j}|)^m$ for $k = 1, \ldots, n$ will do.

With this, the results of [23] imply the following.

Proposition 3.1. *There is a polynomial oracle time algorithm that, given $A \in \mathbb{Z}^{m\times n}$, $x \in \mathbb{N}^n$, $c \in \mathbb{Z}^n$, solves the integer program $\mathrm{IP}_A(b, c)$ with $b := Ax$ by querying a directed augmentation oracle for A.*

Here, as usual, *solving* the (feasible) integer program means that the algorithm either returns an optimal solution or asserts that the objective function is unbounded; and *polynomial oracle time* means that the number of arithmetic operations, the number of calls to the oracle, and the size of the numbers occurring throughout the algorithm are polynomially bounded in the size of the input A, x, c.

It is not hard to see that if the matrix A is totally unimodular, in particular the incidence matrix of a directed graph, then a directed augmentation oracle for A can be realized using linear programming. However, in general it is not clear which matrices A admit efficient directed augmentation.

As explained above, the Graver basis of a matrix A yields an augmentation oracle for A. We now show that, moreover, it enables the realization of a directed augmentation oracle for A as well.

Lemma 3.2. *Let $\mathcal{G}(A)$ be the Graver basis of $A \in \mathbb{Z}^{m\times n}$. For any $x \in \mathbb{N}^n$ and $c', c'' \in \mathbb{Z}^n$, there is a $g \in \mathbb{Z}^n$ with $x - g \in \{y \in \mathbb{N}^n : Ay = Ax\}$ and $c'g^+ - c''g^- > 0$ if and only if there is such $g \in \mathcal{G}(A)$.*

Proof. Suppose g is an improving direction. Then $g \in \mathcal{L}(A) \setminus \{0\}$ and hence can be written as a *conformal sum* of (not necessarily distinct) elements of the Graver basis of A, that is, $g = \sum g^i$ with $g^i \sqsubseteq g$ and $g^i \in \mathcal{G}(A)$ for all i. To see this, recall that $\mathcal{G}(A)$ is the set of $\sqsubseteq$-minimal elements in $\mathcal{L}(A) \setminus \{0\}$ and note that $\sqsubseteq$ is a well-ordering; if $g \in \mathcal{G}(A)$, we are done; otherwise there is an $h \in \mathcal{G}(A)$ with $h \sqsubset g$ in which case, by induction on $\sqsubseteq$, there is a conformal sum $g - h = \sum g^i$ giving $g = h + \sum g^i$.

Now, $g^i \sqsubseteq g$ is equivalent to $(g^i)^+ \le g^+$ and $(g^i)^- \le g^-$, so the conformal sum $g = \sum g^i$ gives corresponding sums of the positive and negative parts $g^+ = \sum (g^i)^+$ and $g^- = \sum (g^i)^-$. Consequently,

$$0 < c'g^+ - c''g^- = c' \sum (g^i)^+ - c'' \sum (g^i)^- = \sum (c'(g^i)^+ - c''(g^i)^-),$$

which implies that there is some g^i in this sum with $c'(g^i)^+ - c''(g^i)^- > 0$. Now, $g^i \in \mathcal{G}(A) \subset \mathcal{L}(A)$ so $Ag^i = 0$ and hence $A(x - g^i) = Ax$. Finally we show that $x - g^i \ge 0$: if $g^i_j \le 0$ then $x_j - g^i_j \ge x_j \ge 0$; and if $g^i_j > 0$ then $g^i \sqsubseteq g$ implies $g^i_j \le g_j$ and thus $x_j - g^i_j \ge x_j - g_j \ge 0$, the last inequality holding because g is an improving direction. So $g^i \in \mathcal{G}(A)$ is an improving direction in the Graver basis. □

As an immediate corollary of Proposition 3.1 and Lemma 3.2, we get the following statement.

Theorem 3.3. *There is a polynomial time algorithm that, given any matrix $A \in \mathbb{Z}^{m\times n}$ along with its Graver basis $\mathcal{G}(A)$, and vectors $x \in \mathbb{N}^n$ and $c \in \mathbb{Z}^n$, solves the integer program $\mathrm{IP}_A(b, c)$ with $b := Ax$.*

While Theorem 3.3 holds for any matrix, its complexity bound depends on the size of the Graver basis which is part of the input. Typically, the Graver basis is very large and its cardinality may be exponential in n. However, in the next section we show that for a broad and useful class of matrices, we can tame the behavior of the Graver basis, leading to an efficient algorithm in terms of A, x, c only.

4. Graver bases of n-fold matrices

Fix any pair of integer matrices A and B with the same number of columns, of dimensions $r \times q$ and $s \times q$, respectively. The *n-fold matrix of the ordered pair A, B* is the following $(s + nr) \times nq$ matrix,

$$[A, B]^{(n)} := (\mathbf{1}_n \otimes B) \oplus (I_n \otimes A) = \begin{pmatrix} B & B & B & \cdots & B \\ A & 0 & 0 & \cdots & 0 \\ 0 & A & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A \end{pmatrix}.$$

With this, the generalized n-fold integer programming problem can be conveniently written as

$$\min\{cx : [A, B]^{(n)}x = b, x \in \mathbb{N}^{nq}\}.$$

The n-fold of a single matrix A, defined in the introduction, is the special case $A^{(n)} = [A, I_q]^{(n)}$ with $B = I_q$ the $q \times q$ identity, giving the regular (nongeneralized) n-fold integer programming problem.

We now discuss a recent result of [21] and its extension in [15] on the stabilization of Graver bases of n-fold matrices. Consider vectors $x = (x^1, \ldots, x^n)$ with $x^k \in \mathbb{N}^q$ for $k = 1, \ldots, n$. The *type* of x is the number $|\{k : x^k \neq 0\}|$ of nonzero components $x^k \in \mathbb{N}^q$ of x. The following result of [15] on the stabilization of Graver bases of $[A, B]^{(n)}$ extends the earlier result for $B = I_q$ from [21].

Proposition 4.1. *For every pair of integer matrices $A \in \mathbb{Z}^{r\times q}$ and $B \in \mathbb{Z}^{s\times q}$, there exists a constant $g(A, B)$ such that for all n, the Graver basis of $[A, B]^{(n)}$ consists of vectors of type at most $g(A, B)$.*

The smallest constant $g(A, B)$ possible in the proposition is called the *Graver complexity* of A, B.

Using Proposition 4.1, we now show that $\mathcal{G}([A, B]^{(n)})$ can be computed in polynomial time.

Theorem 4.2. *Fix any pair of integer matrices $A \in \mathbb{Z}^{r\times q}$ and $B \in \mathbb{Z}^{s\times q}$. Then there is a polynomial time algorithm that, given n, computes the Graver basis $\mathcal{G}([A, B]^{(n)})$ of the n-fold matrix $[A, B]^{(n)}$. In particular, the cardinality and the bit size of $\mathcal{G}([A, B]^{(n)})$ are bounded by a polynomial function of n.*

Proof. Let $g := g(A, B)$ be the Graver complexity of A, B and consider any $n \geq g$. We show that the Graver basis of $[A, B]^{(n)}$ is the union of $\binom{n}{g}$ suitably embedded copies of the Graver basis of $[A, B]^{(g)}$. Consider any g indices $1 \leq k_1 < \cdots < k_g \leq n$ and define a map $\phi_{k_1,\ldots,k_g}$ from $\mathbb{Z}^{gq}$ to $\mathbb{Z}^{nq}$ by sending $x = (x^1, \ldots, x^g)$ to $y = (y^1, \ldots, y^n)$ defined by $y^{k_t} := x^t$ for $t = 1, \ldots, g$, and $y^k := 0$ for all other k.

We claim that the Graver basis of $[A, B]^{(n)}$ is the union of the images of the Graver basis of $[A, B]^{(g)}$ under the $\binom{n}{g}$ maps $\phi_{k_1,\ldots,k_g}$ for all $1 \leq k_1 < \cdots < k_g \leq n$, that is,

$$\mathcal{G}([A, B]^{(n)}) = \bigcup_{1\leq k_1<\cdots<k_g\leq n} \phi_{k_1,\ldots,k_g}(\mathcal{G}([A, B]^{(g)})). \tag{1}$$

To see this, recall first that, by definition, the Graver basis of a matrix M is the set of all $\sqsubseteq$-minimal nonzero dependencies on M (where a dependency on M is a vector x satisfying $Mx = 0$). Thus, if $x = (x^1, \ldots, x^g) \in \mathcal{G}([A, B]^{(g)})$ then x is a $\sqsubseteq$-minimal nonzero dependency on $[A, B]^{(g)}$, implying that $\phi_{k_1,\ldots,k_g}(x)$ is a $\sqsubseteq$-minimal nonzero dependency on $[A, B]^{(n)}$ and hence $\phi_{k_1,\ldots,k_g}(x) \in \mathcal{G}([A, B]^{(n)})$. This establishes that the right-hand side of (1) is contained in the left-hand side. Conversely, consider any $y \in \mathcal{G}([A, B]^{(n)})$. Then, by Proposition 4.1, the type of y is at most g, so there are indices $1 \leq k_1 < \cdots < k_g \leq n$ such that all nonzero components of y are among those of the reduced vector $x := (y^{k_1}, \ldots, y^{k_g})$, and therefore $y = \phi_{k_1,\ldots,k_g}(x)$. Now, $y \in \mathcal{G}([A, B]^{(n)})$ implies that y is a $\sqsubseteq$-minimal nonzero dependency on $[A, B]^{(n)}$, and therefore x is a $\sqsubseteq$-minimal nonzero dependency on $[A, B]^{(g)}$ and hence $x \in \mathcal{G}([A, B]^{(g)})$, showing that $y \in \phi_{k_1,\ldots,k_g}(\mathcal{G}([A, B]^{(g)}))$. This establishes that the left-hand side of (1) is contained in the right-hand side. Thus, the Graver basis of $[A, B]^{(n)}$ is indeed given by (1).

Since A, B are fixed and hence $g = g(A, B)$ is constant, the g-fold matrix $[A, B]^{(g)}$ is also fixed and so the cardinality and bit size of its Graver basis $\mathcal{G}([A, B]^{(g)})$ are constant as well. It follows from (1) that $|\mathcal{G}([A, B]^{(n)})| \leq \binom{n}{g} |\mathcal{G}([A, B]^{(g)})| = O(n^g)$. Further, each element of $\mathcal{G}([A, B]^{(n)})$ is an nq-dimensional vector $\phi_{k_1,\ldots,k_g}(x)$ obtained from some $x \in \mathcal{G}([A, B]^{(g)})$ (of constant bit size) by appending zero components, and therefore is of linear bit size $O(n)$, showing that the bit size of the entire Graver basis $\mathcal{G}([A, B]^{(n)})$ is $O(n^{g+1})$. Finally, it is clear that the $\binom{n}{g} = O(n^g)$ images $\phi_{k_1,\ldots,k_g}(\mathcal{G}([A, B]^{(g)}))$ and their union $\mathcal{G}([A, B]^{(n)})$ can be computed in time polynomial in n, completing the proof. □

Example 4.3. Consider the matrices $A = [1\ 1]$ and $B = I_2$. The Graver complexity of the pair A, B is $g(A, B) = 2$. The 2-fold matrix and its Graver basis, consisting of two antipodal vectors only, are

$$[A, B]^{(2)} = A^{(2)} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \qquad \mathcal{G}([A, B]^{(2)}) = \pm \begin{pmatrix} 1 & -1 & -1 & 1 \end{pmatrix}.$$

By Theorem 4.2, the Graver basis of the 4-fold matrix $[A, B]^{(4)} = A^{(4)}$ can be computed by taking the union of the images of the $6 = \binom{4}{2}$ maps $\phi_{k_1,k_2} : \mathbb{Z}^{2\cdot 2} \longrightarrow \mathbb{Z}^{4\cdot 2}$ for $1 \le k_1 < k_2 \le 4$, and we obtain

$$[A, B]^{(4)} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

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

5. The polynomial time algorithm for n-fold integer programming

We now provide the polynomial time algorithm for the generalized n-fold integer programming problem

$$\min\{cx : [A, B]^{(n)} x = b, x \in \mathbb{N}^{nq}\}. \tag{2}$$

First, combining the results of the previous two sections, we get a polynomial time procedure for converting any feasible solution to an optimal one. We record this result in the following lemma.

Lemma 5.1. *Fix any pair of integer matrices $A \in \mathbb{Z}^{r\times q}$ and $B \in \mathbb{Z}^{s\times q}$. Then there is a polynomial time algorithm that, given n, objective vector $c \in \mathbb{Z}^{nq}$, and nonnegative integer vector $x \in \mathbb{N}^{nq}$, solves the generalized n-fold integer programming problem in which x is feasible, i.e. the one with $b := [A, B]^{(n)} x$.*

Proof. First, apply the polynomial time algorithm underlying Theorem 4.2 on input n and compute the Graver basis $\mathcal{G}([A, B]^{(n)})$ of the n-fold matrix $[A, B]^{(n)}$. Then apply the polynomial time algorithm underlying Theorem 3.3 on input $[A, B]^{(n)}$, $\mathcal{G}([A, B]^{(n)})$, c and x, solving the integer program (2). □

We now show that, moreover, given any b, we can efficiently find an initial feasible solution to (2).

Lemma 5.2. *Fix any pair of integer matrices $A \in \mathbb{Z}^{r\times q}$ and $B \in \mathbb{Z}^{s\times q}$. Then there is a polynomial time algorithm that, given n and demand vector $b \in \mathbb{Z}^{s+nr}$, either finds a feasible solution $x \in \mathbb{N}^{nq}$ to the generalized n-fold integer programming problem (2), or asserts that no feasible solution exists.*

Proof. Introduce $2n(r + s)$ auxiliary variables to the given generalized n-fold integer program, and denote by z the resulting vector of $n(2(r + s) + q)$ variables. Consider the auxiliary integer program of finding a nonnegative integer vector z that minimizes the sum of the auxiliary variables subject to the following system of equations, with I_r and I_s

the $r \times r$ and $s \times s$ identity matrices:

$$\begin{pmatrix} B & I_s & -I_s & 0 & 0 & B & I_s & -I_s & 0 & 0 & B & \cdots & B & I_s & -I_s & 0 & 0 \\ A & 0 & 0 & I_r & -I_r & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & A & 0 & 0 & I_r & -I_r & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & & & & & \ddots & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & A & 0 & 0 & I_r & -I_r \end{pmatrix} \cdot z = b.$$

This auxiliary program is in fact again a generalized n-fold integer program, with matrices $\bar{A} = (A, 0, 0, I_r, -I_r)$ and $\bar{B} = (B, I_s, -I_s, 0, 0)$. Since A and B are fixed, so are $\bar{A}$ and $\bar{B}$. Due to the special structure of the auxiliary program, a feasible solution of this program can be written down easily in terms of b. Consequently, the auxiliary program can be solved by the algorithm underlying Lemma 5.1, in time polynomial in n and the bit size of b. Since the auxiliary objective is bounded below by zero, the algorithm will output an optimal solution z. If the optimal objective value is (strictly) positive, then the original n-fold program (2) has no feasible solution, whereas if the optimal value is zero, then the restriction of z to the original variables is a feasible solution x of the original program (2). □

Combining the results of Lemmas 5.1 and 5.2, we obtain the main result of this article.

Theorem 1.2. Fix any pair of integer matrices A, B of compatible sizes. Then there is a polynomial time algorithm that solves the generalized n-fold integer programming problem on any input n, b, c.

Clearly, Theorem 1.1 is deduced from Theorem 1.2 in the special case $B = I_q$. We emphasize again that, by solving the generalized n-fold integer programming problem, we mean in the complete sense that the algorithm concludes with precisely one of the following: it either asserts that there is no feasible solution, or asserts that the objective function is unbounded, or returns an optimal solution.

Acknowledgements

The first author was supported in part by NSF grant DMS-0309694, 2003 UC-Davis Chancellor's fellow award, and the Alexander von Humboldt foundation. The second author was supported in part by the European network ADONET 504438 and by grant FOR 468 of the Deutsche Forschungsgemeinschaft. The third author was supported in part by a grant from ISF—the Israel Science Foundation, by the Technion President Fund, by the Jewish Communities of Germany Research Fund, and by an S. Langberg Research Fund and the fourth author was supported in part by the European network ADONET 504438 and by grant FOR 468 of the Deutsche Forschungsgemeinschaft.

References

[1] S. Aoki, A. Takemura, Minimal basis for connected Markov chain over $3 \times 3 \times K$ contingency tables with fixed two-dimensional marginals, Aust. N. Z. J. Stat. 45 (2003) 229–249.
[2] M.L. Balinski, F.J. Rispoli, Signature classes of transportation polytopes, Math. Program. A 60 (1993) 127–144.
[3] L.H. Cox, Bounds on entries in 3-dimensional contingency tables, in: Inference Control in Statistical Databases — From Theory to Practice, in: Lec. Not. Comp. Sci., vol. 2316, Springer, New York, 2002, pp. 21–33.
[4] L.H. Cox, On properties of multi-dimensional statistical tables, J. Statist. Plann. Inference 117 (2003) 251–273.
[5] J. De Loera, S. Onn, The complexity of three-way statistical tables, SIAM J. Comput. 33 (2004) 819–836.
[6] J. De Loera, S. Onn, All rational polytopes are transportation polytopes and all polytopal integer sets are contingency tables, in: Proc. 10th Ann. Math. Prog. Soc. Symp. Integ. Prog. Combin. Optim. (Columbia University, New York), in: Lec. Not. Comp. Sci., vol. 3064, Springer, New York, 2004, pp. 338–351.
[7] J. De Loera, S. Onn, Markov bases of three-way tables are arbitrarily complicated, J. Symbolic Comput. 41 (2006) 173–181.
[8] J. De Loera, S. Onn, All linear and integer programs are slim 3-way transportation programs, SIAM J. Optim. 17 (2006) 806–821.
[9] G.T. Duncan, S.E. Fienberg, R. Krishnan, R. Padman, S.F. Roehrig, Disclosure limitation methods and information loss for tabular data, in: P. Doyle, J.I. Land, J.M. Theeuwes, L.V. Zayatz (Eds.), Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies, North-Holland, 2001.
[10] P.C. Gilmore, R.E. Gomory, A linear programming approach to the cutting-stock problem, Oper. Res. 9 (1961) 849–859.
[11] J.E. Graver, On the foundation of linear and integer programming I, Math. Program. 9 (1975) 207–226.
[12] R. Hemmecke, On the positive sum property and the computation of Graver test sets, Math. Program. 96 (2003) 247–269.
[13] R. Hemmecke, Exploiting symmetries in the computation of Graver bases. e-print: arXiv:math.CO/0410334 (2004).

[14] R. Hemmecke, P. Malkin, 4ti2 Version 1.2 — Computation of Hilbert bases, Graver bases, toric Gröbner bases, and more. Available at: http://www.4ti2.de/, July 2005.
[15] S. Hoşten, S. Sullivant, Finiteness theorems for Markov bases, J. Combin. Theory Ser. A 114 (2007) 311–321.
[16] R. Irving, M.R. Jerrum, Three-dimensional statistical data security problems, SIAM J. Comput. 23 (1994) 170–184.
[17] V. Klee, C. Witzgall, Facets and vertices of transportation polytopes, in: Mathematics of the Decision Sciences, Part I (Stanford, CA, 1967), AMS, Providence, RI, 1968, pp. 257–282.
[18] H.W. Lenstra Jr., Integer programming with a fixed number of variables, Math. Oper. Res. 8 (1983) 538–548.
[19] S. Onn, U.G. Rothblum, Convex combinatorial optimization, Discrete Comput. Geom. 32 (2004) 549–566.
[20] M. Queyranne, F.C.R. Spieksma, Approximation algorithms for multi-index transportation problems with decomposable costs, Discrete Appl. Math. 76 (1997) 239–253.
[21] F. Santos, B. Sturmfels, Higher Lawrence configurations, J. Combin. Theory Ser. A 103 (2003) 151–164.
[22] A. Schrijver, Theory of Linear and Integer Programming, Wiley, New York, 1986.
[23] A. Schulz, R. Weismantel, The complexity of generic primal algorithms for solving general integral programs, Math. Oper. Res. 27 (2002) 681–692.
[24] M. Vlach, Conditions for the existence of solutions of the three-dimensional planar transportation problem, Discrete Appl. Math. 13 (1986) 61–78.
[25] V.A. Yemelichev, M.M. Kovalev, M.K. Kravtsov, Polytopes, Graphs and Optimisation, Cambridge University Press, Cambridge, 1984.