Math. Program., Ser. A (2013) 137:325–341
DOI 10.1007/s10107-011-0490-y

FULL LENGTH PAPER

n-Fold integer programming in cubic time

Raymond Hemmecke · Shmuel Onn · Lyubov Romanchuk

Received: 15 January 2011 / Accepted: 16 September 2011 / Published online: 30 September 2011

Abstract n-Fold integer programming is a fundamental problem with a variety of natural applications in operations research and statistics. Moreover, it is universal and provides a new, variable-dimension, parametrization of all of integer programming. The fastest algorithm for n-fold integer programming predating the present article runs in time $O\left(n^{g(A)}L\right)$ with L the binary length of the numerical part of the input and $g(A)$ the so-called Graver complexity of the bimatrix A defining the system. In this article we provide a drastic improvement and establish an algorithm which runs in time $O\left(n^3L\right)$ having cubic dependency on n regardless of the bimatrix A. Our algorithm works for separable convex piecewise affine objectives as well. Moreover, it can be used to define a hierarchy of approximations for any integer programming problem.

Mathematics Subject Classification (2000) 52B · 52C · 62H · 68Q · 68R · 90B · 90C

1 Introduction

n-Fold integer programming is the following problem in variable dimension nt,

$$\min\left\{\mathbf{wx}\ :\ A^{(n)}\mathbf{x}=\mathbf{b},\ \mathbf{l}\le\mathbf{x}\le\mathbf{u},\ \mathbf{x}\in\mathbb{Z}^{nt}\right\}, \tag{1}$$

where $\mathbf{w},\mathbf{l},\mathbf{u}\in\mathbb{Z}^{nt}$, $\mathbf{b}\in\mathbb{Z}^{r+ns}$, and where

R. Hemmecke (✉)
Technische Universität Munich, Munich, Germany
e-mail: hemmecke@ma.tum.de

S. Onn · L. Romanchuk
Technion, Israel Institute of Technology, Haifa, Israel

$$A^{(n)} \quad := \quad \begin{pmatrix} A_1 & A_1 & \cdots & A_1 \\ A_2 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_2 \end{pmatrix} \tag{2}$$

is an $(r + ns) \times nt$ matrix which is the *n-fold product* of a fixed $(r, s) \times t$*bimatrix* $A = \binom{A_1}{A_2}$, that is, of 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. Note that, while the bimatrix A is typically fixed (but arbitrary), the matrix $A^{(n)}$ defining the problem (1) varies with n and results in integer programs in large variable dimension nt. Note also that the lower and upper bound vectors $\mathbf{l}$, $\mathbf{u}$, as well as the right-hand side and objective vectors $\mathbf{b}$, $\mathbf{w}$, are assumed to be presented in their binary encoding (see Sect. 2 for details), and so the variables can take on very large values. Therefore, these programs are not amenable to standard dynamic programming techniques for integer programs with unary presented right-hand sides, such as unary (multi)-knapsack, which lead to pseudo-polynomial algorithms only. Our algorithms for problem (1) are genuinely polynomial.

The n-fold integer programming problem (1) is a fundamental problem with a variety of natural applications in operations research and statistics which, along with extensions and variations, include multiindex and multicommodity transportation problems, privacy and disclosure control in statistical databases, and stochastic integer programming. We briefly discuss some of these applications in Sect. 5 (Corollaries 5.1 and 5.2). For more information see e.g. [4,5,7,9–12,14,15,20], [17, Chapters 4, 5], and the references therein.

Moreover, n-fold integer programming is universal [6] and provides a new, variable-dimension, parametrization of all of integer programming: every program is an n-fold program for some m over the bimatrix $A := A(m)$ with first block A_1 the $3m \times 3m$ identity matrix and second block A_2 the $(3 + m) \times 3m$ incidence matrix of the complete bipartite graph $K_{3,m}$. We make further discussion of this in Sect. 7.

The fastest algorithm for n-fold integer programming predating the present article is in [10] and runs in time $O\left(n^{g(A)}L\right)$ with $L = \langle \mathbf{w}, \mathbf{b}, \mathbf{l}, \mathbf{u} \rangle$ the binary length of the numerical part of the input, and $g(A)$ the so-called *Graver complexity* of the bimatrix A. Unfortunately, the Graver complexity is typically very large [2,18]: for instance, the bimatrices $A(m)$ mentioned above have Graver complexity $g(A(m)) = \Omega(2^m)$, yielding polynomial but very large $n^{\Omega(2^m)}$ dependency of the running time on n.

In this article we provide a drastic improvement and establish an algorithm which runs in time $O\left(n^3 L\right)$ having the cubic dependency on n which is alluded to in the title, regardless of the fixed bimatrix A. So the Graver complexity $g(A)$ now drops down from the exponent of n to the constant multiplying n^3. This is established in Sect. 3 (Theorem 3.9). Moreover, our construction can be used to define a natural hierarchy of approximations for (1) for the bimatrices $A(m)$ with variable parameter m, and therefore, by the universality theorem of [6], for any integer programming problem. These approximations are currently under study, implementation and testing, and will be discussed briefly in Sect. 7 and in more detail elsewhere.

Our algorithm works, moreover, for certain nonlinear objective functions: using results of [16] on certain optimality criteria, we provide in Sect. 4 an optimality certification procedure for separable convex objectives whose time complexity is linear in n (Theorem 4.1) and an algorithm for solving problems with separable convex piecewise affine objectives whose time complexity is again cubic in n (Theorem 4.2). Furthermore, the algorithm also leads to the first polynomial time solution of n-fold integer programming problems over bimatrices with variable entries (Theorem 6.2).

2 Notation and preliminaries

We start with some notation and review of some preliminaries on Graver bases and n-fold integer programming that we need later on. See the book [17] for more details.

Graver bases were introduced in [8] as optimality certificates for integer programming. Define a partial order $\sqsubseteq$ on $\mathbb{R}^n$ by $\mathbf{x} \sqsubseteq \mathbf{y}$ if $x_i y_i \geq 0$ and $|x_i| \leq |y_i|$ for all i. So $\sqsubseteq$ extends the coordinate-wise partial order $\leq$ on the nonnegative orthant $\mathbb{R}^n_+$ to all of $\mathbb{R}^n$. By a classical lemma of Gordan, every subset $Z \subseteq \mathbb{Z}^n$ has finitely-many $\sqsubseteq$-minimal elements, that is, $\mathbf{x} \in Z$ such that no other $\mathbf{y} \in Z$ satisfies $\mathbf{y} \sqsubseteq \mathbf{x}$.

We have the following fundamental definition from [8].

Definition 2.1 The *Graver basis* of an integer $m \times n$ matrix D is defined to be the finite set $\mathcal{G}(D) \subset \mathbb{Z}^n$ of $\sqsubseteq$-minimal elements in $\{\mathbf{x} \in \mathbb{Z}^n \ : \ D\mathbf{x} = 0,\ \mathbf{x} \neq \mathbf{0}\}$.

For instance, the Graver basis of the matrix $D := (1\ \ 2\ \ 1)$ consists of 8 vectors,

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

Consider the general integer programming problem in standard form,

$$\min\left\{\mathbf{wx} \ : \ D\mathbf{x} = \mathbf{b},\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\ \mathbf{x} \in \mathbb{Z}^n\right\}. \tag{3}$$

A *feasible step* for feasible point $\mathbf{x}$ in (3) is any vector $\mathbf{v}$ such that $\mathbf{x}+\mathbf{v}$ is also feasible, that is, $D\mathbf{v} = \mathbf{0}$ so $D(\mathbf{x}+\mathbf{v}) = \mathbf{b}$, and $\mathbf{l} \leq \mathbf{x}+\mathbf{v} \leq \mathbf{u}$. An *augmenting step* for $\mathbf{x}$ is a feasible step $\mathbf{v}$ such that $\mathbf{x}+\mathbf{v}$ is better, that is, $\mathbf{wv} < 0$ so $\mathbf{w}(\mathbf{x}+\mathbf{v}) < \mathbf{wx}$.

Graver has shown that a feasible point $\mathbf{x}$ in (3) is optimal if and only if there is no element $\mathbf{g} \in \mathcal{G}(D)$ in the Graver basis of D which is an augmenting step for $\mathbf{x}$.

This suggests the following simple augmentation scheme: start from any feasible point in (3) and iteratively augment it to an optimal solution using Graver augmenting steps $\mathbf{g} \in \mathcal{G}(D)$ as long as possible. While the number of iterations in this simple scheme as is may be exponential, it was recently shown in [10] that if in each iteration the best possible augmenting step of the form $\gamma\mathbf{g}$ with positive integer γ and $\mathbf{g} \in \mathcal{G}(D)$ is taken, then the number of iterations does become polynomial. In what follows, we call an augmenting step $\mathbf{t}$ which is at least as good as the best possible augmenting step $\gamma\mathbf{g}$ with $\gamma \in \mathbb{Z}_+$ and $\mathbf{g} \in \mathcal{G}(D)$, a *Graver-best augmenting step*. Note that a Graver-best augmenting step $\mathbf{t}$ itself need not be of the form $\gamma\mathbf{g}$ with $\gamma \in \mathbb{Z}_+$ and $\mathbf{g} \in \mathcal{G}(D)$, but $\mathbf{w}(\mathbf{z}_0 + \mathbf{t}) \leq \mathbf{w}(\mathbf{z}_0 + \gamma\mathbf{g})$ must hold for all $\gamma \in \mathbb{Z}_+$ and $\mathbf{g} \in \mathcal{G}(D)$ for which $\mathbf{z}_0 + \gamma\mathbf{g}$ is feasible.

It was shown in [5] that for fixed bimatrix A, the Graver basis $\mathcal{G}\left(A^{(n)}\right)$ of the n-fold product of A can be computed in time polynomial in n. Thus, to find a Graver-best augmenting step of the form $\gamma\mathbf{g}$ for an n-fold integer program (1), it is possible, as shown in [10], to check each element $g \in \mathcal{G}\left(A^{(n)}\right)$, and for each, find the best possible step size γ. However, as we explain below, the Graver basis $\mathcal{G}\left(A^{(n)}\right)$ is very large. Therefore, in this article, we do it the other way around. For each of $O(n)$ critical positive integer potential step sizes γ, we determine an augmenting step $\gamma\mathbf{h}$ which is at least as good as the best possible augmenting step $\gamma\mathbf{g}$ with $g \in \mathcal{G}\left(A^{(n)}\right)$. We then show that the best among these steps over all such γ is a Graver-best augmenting step.

In preparation for this, we need to review some material on Graver bases of n-fold products. Let A be a fixed integer $(r, s) \times t$ bimatrix. For any n we write each vector $\mathbf{x} \in \mathbb{Z}^{nt}$ as a tuple $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^n)$ of n *bricks* $\mathbf{x}^i \in \mathbb{Z}^t$. It has been shown in [1,18], and [13], in increasing generality, that for every fixed bimatrix A, the number of nonzero bricks appearing in any element in the Graver basis $\mathcal{G}\left(A^{(n)}\right)$ for any n is bounded by a constant independent of n. So we can make the following definition.

Definition 2.2 The *Graver complexity* of an integer bimatrix A is defined to be the largest number $g(A)$ of nonzero bricks $\mathbf{g}^i$ in any element $\mathbf{g} \in \mathcal{G}\left(A^{(n)}\right)$ for any n.

This was used in [5] to show that the Graver basis $\mathcal{G}(A^{(n)})$ has a polynomial number $O(n^{g(A)})$ of elements and is computable in time $O(n^{g(A)})$ polynomial in n. Thus, the computation of a Graver-best augmenting step in [10] was done by finding the best step size γ for each of these $O(n^{g(A)})$ elements of $\mathcal{G}(A^{(n)})$, resulting in polynomial but very large $O(n^{g(A)})$ dependency of the running time on n. In Sect. 3 we show how to find a Graver-best augmenting step without constructing $\mathcal{G}(A^{(n)})$ explicitly in quadratic time $O(n^2)$ regardless of the bimatrix A and its Graver complexity. Clearly, the running time *does* depend on A and $g(A)$, but both contribute only to the constant in front of n^2 and hence they disappear in the O-notation.

We conclude this section with some remarks about complexity and finiteness. The *binary length* of an integer number z is the number of bits in its binary encoding, which is $\Theta(\log |z|)$, and is denoted by $\langle z \rangle$. The binary length $\langle \mathbf{z} \rangle$ of an integer vector $\mathbf{z}$ is the sum of binary lengths of its entries. We denote by L the binary length of all numerical part of the input. In particular $L = \langle \mathbf{w}, \mathbf{b}, \mathbf{l}, \mathbf{u} \rangle$ for problem (1). All numbers manipulated by our algorithms remain polynomial in the binary length of the input and our algorithms are polynomial time in the Turing machine model. But we are mostly interested in the number of arithmetic operations performed (additions, multiplications, divisions, comparisons), so time in our complexity statements is the number of such operations as in the real arithmetic model of computation.

For simplicity of presentation we assume throughout that all entries of the bounds $\mathbf{l}, \mathbf{u}$ in program (1) are finite and hence the set of feasible points in (1) is finite. This is no loss of generality since, as is well known, it is always possible to add suitable polynomial upper and lower bounds without excluding some optimal solution if any.

3 The algorithm

We now show how to decide if a given feasible point $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^n)$ in (1) is optimal in linear time $O(n)$, and if not, determine a Graver-best augmenting step $\gamma\mathbf{g}$ for $\mathbf{x}$ in quadratic time $O(n^2)$. This is then incorporated into an iterative algorithm for solving (1).

We begin with a lemma about elements of Graver bases of n-fold products.

Lemma 3.1 *Let $A = \binom{A_1}{A_2}$ be integer $(r, s) \times t$ bimatrix with Graver complexity $g(A)$. Let*

$$Z(A) \ := \ \left\{\mathbf{z} \in \mathbb{Z}^t \ : \ \mathbf{z} \textit{ is the sum of at most } g(A) \textit{ elements of } \mathcal{G}(A_2)\right\}. \qquad (4)$$

Then for any n, any $\mathbf{g} \in \mathcal{G}\left(A^{(n)}\right)$ and any $I \subseteq \{1, \ldots, n\}$, we have $\sum_{i \in I} \mathbf{g}^i \in Z(A)$.

Proof Consider any Graver basis element $\mathbf{g} \in \mathcal{G}\left(A^{(n)}\right)$ for some n. Then $A^{(n)}\mathbf{g} = \mathbf{0}$ and hence $\sum_{i=1}^n A_1\mathbf{g}^i = \mathbf{0}$ and $A_2\mathbf{g}^i = \mathbf{0}$ for all i. Therefore (see [17, Chapter 4]) each $\mathbf{g}^i$ can be written as the sum $\mathbf{g}^i = \sum_{j=1}^{k_i} \mathbf{h}^{i,j}$ of some elements $\mathbf{h}^{i,j} \in \mathcal{G}(A_2)$ for all i, j. Let $m := k_1 + \cdots + k_n$ and let $\mathbf{h}$ be the vector

$$\mathbf{h} \ := \ (\mathbf{h}^{1,1}, \ldots, \mathbf{h}^{1,k_1}, \ldots, \mathbf{h}^{n,1}, \ldots, \mathbf{h}^{n,k_n}) \ \in \ \mathbb{Z}^{mt}.$$

Then $\sum_{i,j} A_1\mathbf{h}^{i,j} = \mathbf{0}$ and $A_2\mathbf{h}^{i,j} = \mathbf{0}$ for all i, j and hence $A^{(m)}\mathbf{h} = \mathbf{0}$. We claim that moreover, $\mathbf{h} \in \mathcal{G}\left(A^{(m)}\right)$. Suppose indirectly this is not the case. Then there is an $\bar{\mathbf{h}} \in \mathcal{G}\left(A^{(m)}\right)$ with $\bar{\mathbf{h}} \sqsubset \mathbf{h}$. But then the vector $\bar{\mathbf{g}} \in \mathbb{Z}^{nt}$ defined by $\bar{\mathbf{g}}^i := \sum_{j=1}^{k_i} \bar{\mathbf{h}}^{i,j}$ for all i satisfies $\bar{\mathbf{g}} \sqsubset \mathbf{g}$ contradicting $\mathbf{g} \in \mathcal{G}\left(A^{(n)}\right)$. This proves the claim. Therefore, by Definition 2.2 of Graver complexity, the number of nonzero bricks $\mathbf{h}^{i,j}$ of $\mathbf{h}$ is at most $g(A)$. So for every $I \subseteq \{1, \ldots, n\}$, we have that $\sum_{i \in I} \mathbf{g}^i = \sum_{i \in I} \sum_{j=1}^{k_i} \mathbf{h}^{i,j}$ is a sum of at most $g(A)$ nonzero elements $\mathbf{h}^{i,j} \in \mathcal{G}(A_2)$ and hence $\sum_{i \in I} \mathbf{g}^i \in Z(A)$. □

Example 3.2 Let $A := A(3)$ be the $(9, 6) \times 9$ bimatrix mentioned in the introduction, which arises in the universality of n-fold integer programming discussed further in Sect. 7, having first block $A_1 = I_9$ the 9×9 identity matrix and second block the following 6×9 incidence matrix of the complete bipartite graph $K_{3,3}$,

$$A_2 \ = \ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Since A_2 is totally unimodular, its Graver basis $\mathcal{G}(A_2)$ consists of the 30 vectors in $\{0, \pm 1\}^9$ supported on circuits of $K_{3,3}$ with alternating ± 1, see [17]. Also, it is known that the Graver complexity of this bimatrix is $g(A) = 9$, see [2,18]. Therefore, the set $Z(A)$ in (4) which corresponds to A, consists of all sums of at most 9 such circuit vectors, and turns out to be comprised of 42931 vectors in $\mathbb{Z}^9$, such as

$$\begin{pmatrix} 9 & -2 & -7 & -4 & 5 & -1 & -5 & -3 & 8 \end{pmatrix}.$$

We now define a dynamic program, that is, a weighted digraph, which will enable to find a Graver-best augmenting step $\gamma\mathbf{g}$ for a feasible point $\mathbf{x}$ of (1) or detect that none exists.

Definition 3.3 (*the dynamic program*) Let A be a fixed $(r, s) \times t$ bimatrix and let $g(A)$ be its Graver complexity. Given n, $\mathbf{w}$, $\mathbf{b}$, $\mathbf{l}$, $\mathbf{u}$, feasible point $\mathbf{x}$ in (1), and positive integer γ, define a weighted digraph as follows. Its vertices are partitioned into $n+1$ stages defined in terms of the fixed finite set $Z(A) \subset \mathbb{Z}^t$ in (4), by

$$S_0 := \{\mathbf{0}\}, \quad S_1 := S_2 := \cdots := S_{n-1} := Z(A), \quad S_n := \{\mathbf{z} \in Z(A) \ : \ A_1\mathbf{z} = \mathbf{0}\}.$$

Denote the vertices of S_i by $\mathbf{h}^i \in \mathbb{Z}^t$. Introduce an arc $(\mathbf{h}^{i-1}, \mathbf{h}^i)$ from $\mathbf{h}^{i-1} \in S_{i-1}$ to $\mathbf{h}^i \in S_i$ if $\mathbf{g}^i := \mathbf{h}^i - \mathbf{h}^{i-1} \in Z(A)$ and $\mathbf{l}^i \le \mathbf{x}^i + \gamma\mathbf{g}^i \le \mathbf{u}^i$, and give it weight $\mathbf{w}^i\mathbf{g}^i$.

To each dipath $\mathbf{h} = (\mathbf{h}^0, \mathbf{h}^1, \ldots, \mathbf{h}^n)$ from S_0 to S_n in this digraph we associate a vector $\mathbf{g}(\mathbf{h}) := (\mathbf{h}^1 - \mathbf{h}^0, \ldots, \mathbf{h}^n - \mathbf{h}^{n-1}) \in \mathbb{Z}^{nt}$. Note that $\mathbf{0} \in Z(A)$ and hence the trivial path $\mathbf{h} = (\mathbf{0}, \ldots, \mathbf{0})$ with weight 0 and vector $\mathbf{g}(\mathbf{h}) = (\mathbf{0}, \ldots, \mathbf{0})$ always exists. Note also that $\mathbf{w}\mathbf{g}(\mathbf{h}) = \sum_{i=1}^n \mathbf{w}^i(\mathbf{h}^i - \mathbf{h}^{i-1})$ is precisely the weight of the dipath $\mathbf{h}$.

The following lemma relates this dynamic program to Graver augmentations.

Lemma 3.4 *For any fixed bimatrix A there is an algorithm that, given n, γ, $\mathbf{w}$, $\mathbf{b}$, $\mathbf{l}$, $\mathbf{u}$, and feasible point $\mathbf{x}$ for (1), finds a feasible step $\gamma\mathbf{g}$ for $\mathbf{x}$ which satisfies $\mathbf{w}(\mathbf{x}+\gamma\mathbf{g}) \le \mathbf{w}(\mathbf{x}+\gamma\bar{\mathbf{g}})$ for any feasible step $\gamma\bar{\mathbf{g}}$ with $\bar{\mathbf{g}} \in \mathcal{G}\left(A^{(n)}\right)$ in linear time $O(n)$.*

Proof Let $\mathbf{h}$ be a minimum weight dipath from S_0 to S_n and let $\mathbf{g} := \mathbf{g}(\mathbf{h})$ be the vector associated with $\mathbf{h}$. We claim that $\gamma\mathbf{g}$ is the desired feasible step for $\mathbf{x}$.

We begin with the complexity statement. Since A is fixed, so is $g(A)$, and hence so is each S_i. As the digraph is acyclic, the minimum weight dipath from S_0 to $\mathbf{h}^i \in S_i$ decomposes into a minimum weight dipath from S_0 to some $\mathbf{h}^{i-1} \in S_{i-1}$ plus the arc from $\mathbf{h}^{i-1}$ to $\mathbf{h}^i$. Thus, we have to check at most a constant number $|S_{i-1}| \cdot |S_i|$ of such pairs $(\mathbf{h}^{i-1}, \mathbf{h}^i)$ to find the minimum weight dipaths from S_0 to every $\mathbf{h}^i \in S_i$ given the minimum weight dipaths from S_0 to every $\mathbf{h}^{i-1} \in S_{i-1}$. Repeating this for each of the sets $S_1, \ldots, S_n$ one after the other takes $O(n)$ time.

We next show that $\gamma\mathbf{g}$ is a good feasible step. Since $\mathbf{g}^i = \mathbf{h}^i - \mathbf{h}^{i-1}$ and $(\mathbf{h}^{i-1}, \mathbf{h}^i)$ is an arc, we have $\mathbf{l}^i \le \mathbf{x}^i + \gamma\mathbf{g}^i \le \mathbf{u}^i$, and $\mathbf{g}^i \in Z(A)$ and hence $A_2\mathbf{g}^i = \mathbf{0}$, for all i. Also, $\sum_{i=1}^n \mathbf{g}^i = \mathbf{h}^n \in S_n$ and hence $\sum_{i=1}^n A_1\mathbf{g}^i = A_1\mathbf{h}^n = \mathbf{0}$. So $\mathbf{x}+\gamma\mathbf{g}$ is feasible in (1). Moreover, $\mathbf{w}\mathbf{g} = \sum_{i=1}^n \mathbf{w}^i\mathbf{g}^i$ is the weight of the minimum weight dipath $\mathbf{h}$. Now consider any feasible step $\gamma\bar{\mathbf{g}}$ for $\mathbf{x}$ with $\bar{\mathbf{g}} \in \mathcal{G}\left(A^{(n)}\right)$. Define $\bar{\mathbf{h}}^i := \sum_{j \le i} \bar{\mathbf{g}}^j$ for all i. Then $\bar{\mathbf{h}}^i \in Z(A)$ for all i by Lemma 3.1. Moreover, $A_1\bar{\mathbf{h}}^n = A_1 \sum_{j=1}^n \bar{\mathbf{g}}^j = \mathbf{0}$. Therefore $\bar{\mathbf{h}}^i \in S_i$ for all i. Furthermore, $\bar{\mathbf{h}}^i - \bar{\mathbf{h}}^{i-1} = \bar{\mathbf{g}}^i \in Z(A)$ and $\mathbf{l}^i \le \mathbf{x}^i + \gamma\bar{\mathbf{g}}^i \le \mathbf{u}^i$

and therefore $(\bar{\mathbf{h}}^{i-1}, \bar{\mathbf{h}}^{i})$ is an arc of weight $\mathbf{w}^i\bar{\mathbf{g}}^i$ for all i. So $\bar{\mathbf{h}} = (\bar{\mathbf{h}}^0, \bar{\mathbf{h}}^1, \ldots, \bar{\mathbf{h}}^n)$ is a dipath from S_0 to S_n with weight $\mathbf{w}\bar{\mathbf{g}} = \sum_{i=1}^n \mathbf{w}^i\bar{\mathbf{g}}^i$ and associated vector $\mathbf{g}(\bar{\mathbf{h}}) = \bar{\mathbf{g}}$. Since $\mathbf{h}$ is a minimum weight dipath, $\mathbf{w}\mathbf{g} \le \mathbf{w}\bar{\mathbf{g}}$ and so $\mathbf{w}(\mathbf{x} + \gamma\mathbf{g}) \le \mathbf{w}(\mathbf{x} + \gamma\bar{\mathbf{g}})$. □

Remark 3.5 (optimality certification in linear time) As noted in Sect. 2, a feasible point $\mathbf{x}$ in integer program (1) is optimal if and only if there is no Graver augmenting step $\mathbf{g} \in \mathcal{G}\left(A^{(n)}\right)$ for $\mathbf{x}$. Thus, with $\gamma := 1$, Lemma 3.4 implies that the optimality of a feasible point $\mathbf{x}$ in (1) can be determined in linear time $O(n)$.

The next lemma shows that we can quickly find a Graver-best augmentation.

Lemma 3.6 *For any fixed bimatrix A there is an algorithm that, given $n, \mathbf{w}, \mathbf{b}, \mathbf{l}, \mathbf{u}$, and feasible point $\mathbf{x}$ for (1), finds a feasible step $\gamma\mathbf{g}$ for $\mathbf{x}$ which satisfies $\mathbf{w}(\mathbf{x} + \gamma\mathbf{g}) \le \mathbf{w}(\mathbf{x} + \bar{\gamma}\bar{\mathbf{g}})$ for any feasible step $\bar{\gamma}\bar{\mathbf{g}}$ with $\bar{\gamma} \in \mathbb{Z}_+$ and $\bar{\mathbf{g}} \in \mathcal{G}\left(A^{(n)}\right)$ in quadratic time $O(n^2)$.*

Proof If $Z(A) = \{\mathbf{0}\}$ then $\mathcal{G}\left(A^{(n)}\right) = \emptyset$ by Lemma 3.1, so $\gamma\mathbf{g} := \mathbf{0}$ will do. Otherwise, construct a set Γ of $O(n)$ positive integers in $O(n)$ time as follows: for every $i = 1, \ldots, n$ and every $\mathbf{z} \in Z(A)\backslash\{\mathbf{0}\}$ determine the largest nonnegative integer γ such that $\mathbf{l}^i \le \mathbf{x}^i + \gamma\mathbf{z} \le \mathbf{u}^i$ and if it is positive, include it in Γ. Now, for each $\gamma \in \Gamma$, construct and solve the corresponding dynamic program, resulting in total of $O(n^2)$ time by Lemma 3.4. Let $\gamma\mathbf{g}$ be that feasible step for $\mathbf{x}$ which attains minimum value $\mathbf{w}\gamma\mathbf{g}$ among the best steps obtained from all these dynamic programs, and let $\bar{\gamma}\bar{\mathbf{g}}$ be that feasible step for $\mathbf{x}$ which attains minimum value $\mathbf{w}\bar{\gamma}\bar{\mathbf{g}}$ among $\bar{\gamma}\bar{\mathbf{g}}$ with $\bar{\mathbf{g}} \in \mathcal{G}\left(A^{(n)}\right)$ if any. Assume that $\mathbf{w}\bar{\gamma}\bar{\mathbf{g}} < 0$ as otherwise we are done since $\mathbf{w}\gamma\mathbf{g} \le \mathbf{w}\gamma\mathbf{0} = 0$. Then $\bar{\gamma}$ is the largest positive integer such that $\mathbf{l} \le \mathbf{x} + \bar{\gamma}\bar{\mathbf{g}} \le \mathbf{u}$ since otherwise the step $(\bar{\gamma} + 1)\bar{\mathbf{g}}$ will be feasible and better. So for some $i = 1, \ldots, n$, it must be that $\bar{\gamma}$ is the largest positive integer such that $\mathbf{l}^i \le \mathbf{x}^i + \bar{\gamma}\bar{\mathbf{g}}^i \le \mathbf{u}^i$. Since $\bar{\mathbf{g}} \in \mathcal{G}\left(A^{(n)}\right)$, it follows from Lemma 3.1 that $\bar{\mathbf{g}}^i \in Z(A)$. Therefore $\bar{\gamma} \in \Gamma$. Now let $\bar{\gamma}\hat{\mathbf{g}}$ be the best step attained from the dynamic program of $\bar{\gamma}$. Then $\mathbf{w}\gamma\mathbf{g} \le \mathbf{w}\bar{\gamma}\hat{\mathbf{g}}$ by choice of $\gamma\mathbf{g}$ and $\mathbf{w}\bar{\gamma}\hat{\mathbf{g}} \le \mathbf{w}\bar{\gamma}\bar{\mathbf{g}}$ by Lemma 3.4. Therefore $\mathbf{w}(\mathbf{x} + \gamma\mathbf{g}) \le \mathbf{w}(\mathbf{x} + \bar{\gamma}\bar{\mathbf{g}})$ as claimed. □

We next show, following [10], that repeatedly applying Graver-best augmenting steps, we can augment an initial feasible point for (1) to an optimal one efficiently.

Lemma 3.7 *For any fixed bimatrix A there is an algorithm that, given $n, \mathbf{w}, \mathbf{b}, \mathbf{l}, \mathbf{u}$, and feasible point $\mathbf{x}$ for (1), finds an optimal solution $\mathbf{x}^*$ for (1) in time $O(n^3L)$.*

Proof Iterate the following: find by the algorithm of Lemma 3.6 a Graver-best augmenting step $\gamma\mathbf{g}$ for $\mathbf{x}$; if it is augmenting then set $\mathbf{x} := \mathbf{x} + \gamma\mathbf{g}$ and repeat, else $\mathbf{x}^* := \mathbf{x}$ is optimal.

To bound the number of iterations, following [10], note that while $\mathbf{x}$ is not optimal, and $\mathbf{x}^*$ is some optimal solution, we have that $\mathbf{x}^* - \mathbf{x} = \sum_{i=1}^k \gamma_i\mathbf{g}_i$ is a nonnegative integer combination of Graver basis elements $\mathbf{g}_i \in \mathcal{G}\left(A^{(n)}\right)$ all lying in the same orthant, and hence each $\mathbf{x} + \gamma_i\mathbf{g}_i$ is feasible in (1). Moreover, by the integer Carathéodory theorem of [3,19], we can assume that $k \le 2(nt - 1)$. Letting $\gamma_i\mathbf{g}_i$ be a summand attaining minimum $\mathbf{w}\gamma_i\mathbf{g}_i$, and letting $\gamma\mathbf{g}$ be a Graver-best augmenting step for $\mathbf{x}$ obtained from the algorithm of Lemma 3.6, we find that

$$\mathbf{w}(\mathbf{x}+\gamma\mathbf{g})-\mathbf{w}\mathbf{x} \le \mathbf{w}(\mathbf{x}+\gamma_i\mathbf{g}_i)-\mathbf{w}\mathbf{x} \le \frac{1}{2(nt-1)}\left(\mathbf{w}\mathbf{x}^*-\mathbf{w}\mathbf{x}\right).$$

So the Graver-best augmenting step provides an improvement which is a constant fraction of the best possible improvement, and this can be shown to lead to a bound of $O(nL)$ on the number of iterations to optimality, see [10] for more details. Since each iteration takes $O(n^2)$ time by Lemma 3.6, the overall running time is $O(n^3L)$ as claimed. □

We next show how to find an initial feasible point for (1) with the same complexity. We follow the approach of [5] using a suitable auxiliary n-fold program.

Lemma 3.8 *For any fixed bimatrix A there is an algorithm that, given n, $\mathbf{b}$, $\mathbf{l}$, $\mathbf{u}$, either finds a feasible point $\mathbf{x}$ for (1) or asserts that none exists, in time $O(n^3L)$.*

Proof Construct an auxiliary n-fold integer program

$$\min\left\{\bar{\mathbf{w}}\mathbf{z}\ :\ \bar{A}^{(n)}\mathbf{z}=\mathbf{b}\,,\ \bar{\mathbf{l}}\le\mathbf{z}\le\bar{\mathbf{u}}\,,\ \mathbf{z}\in\mathbb{Z}^{n(t+2r+2s)}\right\} \tag{5}$$

as follows. First, construct a new fixed $(r,s)\times(t+2r+2s)$ bimatrix $\bar{A}$ with

$$\bar{A}_1 := \begin{pmatrix} A_1 & I_r & -I_r & 0_{r\times s} & 0_{r\times s}\end{pmatrix}\quad \bar{A}_2 := \begin{pmatrix} A_2 & 0_{s\times r} & 0_{s\times r} & I_s & -I_s\end{pmatrix}.$$

Now, the $n(t+2r+2s)$ variables $\mathbf{z}$ have a natural partition into nt original variables $\mathbf{x}$ and $n(2r+2s)$ new auxiliary variables $\mathbf{y}$. Keep the original lower and upper bounds on the original variables and introduce lower bound 0 and upper bound $\|\mathbf{b}\|_\infty$ on each auxiliary variable. Let the new objective $\bar{\mathbf{w}}\mathbf{z}$ be the sum of auxiliary variables. Note that the binary length of the auxiliary program satisfies $\bar{L}=O(L)$ and an initial feasible point $\bar{\mathbf{z}}$ with $\bar{\mathbf{x}}=\mathbf{0}$ for (5) with the original $\mathbf{b}$ is easy to construct. Now apply the algorithm of Lemma 3.7 and find in time $O(n^3\bar{L})=O(n^3L)$ an optimal solution $\mathbf{z}$ for (5). If the optimal objective value is 0 then $\mathbf{y}=\mathbf{0}$ and $\mathbf{x}$ is feasible in the original program (1) whereas if it is positive then (1) is infeasible. □

We can now obtain the main result of this article.

Theorem 3.9 *For every fixed integer $(r,s)\times t$ bimatrix A, there is an algorithm that, given n, vectors $\mathbf{w},\mathbf{l},\mathbf{u}\in\mathbb{Z}^{nt}$ and $\mathbf{b}\in\mathbb{Z}^{r+ns}$ having binary encoding length $L:=\langle\mathbf{w},\mathbf{b},\mathbf{l},\mathbf{u}\rangle$, solves in time $O(n^3L)$ the n-fold integer programming problem*

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

Proof Use the algorithm of Lemma 3.8 to either detect infeasibility or obtain a feasible point and augment it by the algorithm of Lemma 3.7 to optimality. □

4 Extensions to nonlinear objectives

Here we extend some of our results to programs with nonlinear objective functions,

$$\min \left\{ f(\mathbf{x}) \ : \ A^{(n)}\mathbf{x} = \mathbf{b} \, , \ \mathbf{l} \le \mathbf{x} \le \mathbf{u} \, , \ \mathbf{x} \in \mathbb{Z}^{nt} \right\} . \tag{6}$$

A function $f : \mathbb{R}^{nt} \to \mathbb{R}$ is *separable convex* if $f(\mathbf{x}) = \sum_{i=1}^{n} f^i(\mathbf{x}^i) = \sum_{i=1}^{n}\sum_{j=1}^{t} f^i_j(x^i_j)$ with each f^i_j univariate convex. In [16] it was shown that Graver bases provide optimality certificates for problem (6) with separable convex functions as well: a feasible point $\mathbf{x}$ is optimal if and only if there is no feasible Graver step $\mathbf{g}$ for $\mathbf{x}$ which satisfies $f(\mathbf{x}+\mathbf{g}) < f(\mathbf{x})$. This was used in [10] to provide polynomial time procedures for optimality certification and solution of problem (6) with separable convex functions f. However, this involved again checking each of the $O(n^{g(A)})$ elements of $\mathcal{G}\left(A^{(n)}\right)$.

Our results from Sect. 3 can be extended to provide linear time optimality certification for separable convex functions and a cubic time solution of (6) for separable convex piecewise affine functions. We discuss these respectively next.

4.1 Optimality certification for separable convex objectives

Here we assume that the objective function f is presented by a *evaluation oracle* that, when queried on a vector $\mathbf{x}$, returns the values $f^i(\mathbf{x}^i)$ for all $i = 1, \ldots, n$. The time complexity now measures the number of arithmetic operations and oracle queries.

Theorem 4.1 *For any fixed bimatrix A, there is an algorithm that, given $n, \mathbf{b}, \mathbf{l}, \mathbf{u}$, separable convex f presented by an evaluation oracle, and feasible point $\mathbf{x}$ in program (6), either asserts that $\mathbf{x}$ is optimal or finds an augmenting step $\mathbf{g}$ for $\mathbf{x}$ which satisfies $f(\mathbf{x}+\mathbf{g}) \le f(\mathbf{x}+\bar{\mathbf{g}})$ for any feasible step $\bar{\mathbf{g}} \in \mathcal{G}\left(A^{(n)}\right)$, in linear time $O(n)$.*

Proof Given the feasible point $\mathbf{x}$, set a dynamic program similar to that in Definition 3.3, with $\gamma := 1$, with the only modification that the weight of arc $(\mathbf{h}^{i-1}, \mathbf{h}^i)$ from $\mathbf{h}^{i-1} \in S_{i-1}$ to $\mathbf{h}^i \in S_i$ is now defined to be $f^i(\mathbf{x}^i + \mathbf{g}^i) - f^i(\mathbf{x}^i)$ with $\mathbf{g}^i := \mathbf{h}^i - \mathbf{h}^{i-1}$. Then, for every dipath $\mathbf{h}$ and its associated vector $\mathbf{g} := \mathbf{g}(\mathbf{h})$, we now have

$$f(\mathbf{x}+\mathbf{g}) - f(\mathbf{x}) \ = \ \sum_{i=1}^{n} \Big(f^i(\mathbf{x}^i + \mathbf{g}^i) - f^i(\mathbf{x}^i) \Big) \ = \ \text{weight of dipath } \mathbf{h}.$$

We now claim that the desired step is the vector $\mathbf{g} := \mathbf{g}(\mathbf{h})$ associated with a minimum weight dipath $\mathbf{h}$ in this dynamic program. Indeed, an argument similar to that in the proof of Lemma 3.4 now implies that for any feasible step $\bar{\mathbf{g}} \in \mathcal{G}\left(A^{(n)}\right)$ we have $f(\mathbf{x}+\mathbf{g}) - f(\mathbf{x}) \le f(\mathbf{x}+\bar{\mathbf{g}}) - f(\mathbf{x})$ and therefore $f(\mathbf{x}+\mathbf{g}) \le f(\mathbf{x}+\bar{\mathbf{g}})$. □

4.2 Optimization of separable convex piecewise affine objectives

In [10] it was shown that problem (6) can be solved for any separable convex function in polynomial time, but with very large dependency of $O(n^{g(A)})$ of the running time on n, with the exponent $g(A)$ depending on the bimatrix A. Here we restrict attention to separable convex objective functions which are piecewise affine, for which we are able to reduce the time dependency on n to $O(n^3)$ independent of A. Clearly, the constant term in front of n^3 does depend on A. However, the dependence on A is removed from the exponent of n.

So we assume again that $f(\mathbf{x}) = \sum_{i=1}^n f^i(\mathbf{x}^i) = \sum_{i=1}^n \sum_{j=1}^t f^i_j(x^i_j)$ with each $f^i_j : \mathbb{R} \to \mathbb{R}$ univariate convex. Moreover, we now also assume that for some fixed p, each f^i_j is p-piecewise affine, that is, the interval between the lower bound $\mathbf{l}^i_j$ and upper bound $\mathbf{u}^i_j$ is partitioned into at most p intervals with integer end-points, and the restriction of f^i_j to each interval k is an affine function $w^i_{j,k}x^i_j + a^i_{j,k}$ with all $w^i_{j,k}, a^i_{j,k}$ integers. We denote by $\langle f \rangle$ the binary length of f which is the sum of binary lengths of all interval end-points and $w^i_{j,k}, a^i_{j,k}$ needed to describe it. The binary length of the input for the nonlinear problem (6) is now $L := \langle f, \mathbf{b}, \mathbf{l}, \mathbf{u} \rangle$.

Theorem 4.2 *For any fixed p and bimatrix A, there is an algorithm that, given $n, \mathbf{b}, \mathbf{l}, \mathbf{u}$, and separable convex p-piecewise affine f, solves in time $O(n^3 L)$ the program*

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

Proof We need to establish analogs of some of the lemmas of Sect. 3 for such objective functions. First, for the analog of Lemma 3.4, proceed as in the proof of Theorem 4.1 above: given a feasible point $\mathbf{x}$ and positive integer γ, set again a dynamic program similar to that in Definition 3.3, with the weight of arc $(\mathbf{h}^{i-1}, \mathbf{h}^i)$ from $\mathbf{h}^{i-1} \in S_{i-1}$ to $\mathbf{h}^i \in S_i$ defined to be $f^i(\mathbf{x}^i + \gamma \mathbf{g}^i) - f^i(\mathbf{x}^i)$ with $\mathbf{g}^i := \mathbf{h}^i - \mathbf{h}^{i-1}$. A similar argument to that in the proof of Lemma 3.4 now shows that $\gamma \mathbf{g}$ with $\mathbf{g} := \mathbf{g}(\mathbf{h})$ the vector associated with the minimum weight dipath $\mathbf{h}$ is a feasible step for $\mathbf{x}$ which satisfies $f(\mathbf{x} + \gamma \mathbf{g}) \le f(\mathbf{x} + \gamma \bar{\mathbf{g}})$ for any feasible step $\gamma \bar{\mathbf{g}}$ with $\bar{\mathbf{g}} \in \mathcal{G}\left(A^{(n)}\right)$.

For the analog of Lemma 3.6, we construct again a set Γ of critical step sizes γ as follows. First, as in the proof of Lemma 3.6, we collect the critical step sizes due to the lower and upper bound constraints by finding, for every $i = 1, \ldots, n$ and every $\mathbf{z} \in Z(A) \setminus \{\mathbf{0}\}$, the largest nonnegative integer γ such that $\mathbf{l}^i \le \mathbf{x}^i + \gamma \mathbf{z} \le \mathbf{u}^i$, and if it is positive, include it in Γ. However, in contrast to Lemma 3.6, we are now dealing with the more general class of piecewise affine objective functions f^i_j. So we must add also the following values γ to Γ: for every $i = 1, \ldots, n$, every $\mathbf{z} \in Z(A) \setminus \{\mathbf{0}\}$, and every $j = 1, \ldots, t$, if $x^i_j + \gamma z_j$ and $x^i_j + (\gamma + 1) z_j$ belong to different affine pieces of f^i_j, then γ and $\gamma + 1$ are included in Γ, if the bounds $l^i_j \le x^i_j + \gamma z_j \le u^i_j$ and $l^i_j \le x^i_j + (\gamma + 1) z_j \le u^i_j$ are satisfied. Finally, for technical reasons (see below), we include $\gamma = 1$ into Γ. Since the number p of affine pieces is constant, the number of such values for each i, z and j is also constant. So the total number of elements of Γ remains linear and it can be constructed in linear time $O(n)$ again. By construction, the set Γ has the following nice property: if we order the elements in Γ by increasing value and if we then choose two consecutive values $\gamma_1 < \gamma_2$, then either $\gamma_2 = \gamma_1 + 1$ or for all values $\gamma \in [\gamma_1, \gamma_2]$ the corresponding dynamic programming programming graphs are the same (as in the pure linear case) and the piece-wise affine linear objective function is linear (as, by construction, there is no change between linear pieces in the given interval $[\gamma_1, \gamma_2]$). Hence, as in the pure linear case, we only need to solve the dynamic program for the endpoints of the interval: $\gamma = \gamma_1$ and $\gamma = \gamma_2$ (which both lie in Γ). In order to cover all such intervals, it is necessary to include $\gamma = 1$ into Γ to have definitely available also the left endpoint for the interval of smallest γ-values, a simple technicality.

Now we continue as in the proof of Lemma 3.6: for each γ in Γ, using the analog of Lemma 3.4 established in the first paragraph above, we solve the corresponding dynamic program in $O(n)$ time, resulting in total of $O(n^2)$ time again. Let $\gamma\mathbf{g}$ be that feasible step for $\mathbf{x}$ which attains minimum value $f(\mathbf{x} + \gamma\mathbf{g})$ among the best steps obtained from all these dynamic programs, and let $\bar{\gamma}\bar{\mathbf{g}}$ be that feasible step for $\mathbf{x}$ which attains minimum value $f(\mathbf{x} + \bar{\gamma}\bar{\mathbf{g}})$ among $\bar{\gamma}\bar{\mathbf{g}}$ with $\bar{\mathbf{g}} \in \mathcal{G}\left(A^{(n)}\right)$ if any. It now follows from the construction of Γ that if $\bar{\gamma}\bar{\mathbf{g}}$ is augmenting, namely, if $f(\mathbf{x}+\bar{\gamma}\bar{\mathbf{g}}) - f(\mathbf{x}) < 0$, then $\bar{\gamma} \in \Gamma$, as otherwise the step $(\bar{\gamma} + 1)\bar{\mathbf{g}}$ will be feasible and better. Let $\bar{\gamma}\hat{\mathbf{g}}$ be the best step attained from the dynamic program of $\bar{\gamma}$. Then

$$f(\mathbf{x} + \gamma\mathbf{g}) \ \leq \ f(\mathbf{x} + \bar{\gamma}\hat{\mathbf{g}}) \ \leq \ f(\mathbf{x} + \bar{\gamma}\bar{\mathbf{g}})$$

where the first inequality follows from the choice of $\gamma\mathbf{g}$ and the second inequality follows from the analog of Lemma 3.4. Therefore $f(\mathbf{x} + \gamma\mathbf{g}) \leq f(\mathbf{x} + \bar{\gamma}\bar{\mathbf{g}})$.

For the analog of Lemma 3.7, we use the results of [10] incorporating the optimality criterion of [16], which assure that the number of iterations needed when using a Graver-best augmenting step at each iteration, is bounded by $O(n\langle f\rangle) = O(nL)$, resulting again in overall time complexity $O(n^3L)$ for augmenting an initial feasible point to an optimal solution of (6). Since an initial feasible point if any can be found by Lemma 3.8 as before in the same complexity, the theorem now follows. □

5 Some consequences

Here we briefly discuss two of the many consequences of n-fold integer programming which, now with our new algorithm, can be solved drastically faster than before.

5.1 Nonlinear multicommodity transportation

The multicommodity transportation problem seeks minimum cost routing of l commodities from m suppliers to n consumers subject to supply, consumption and capacity constraints. For $l = 1$ this is the classical transportation problem which is efficiently solvable by linear programming. But already for $l = 2$ it is NP-hard. Here we consider the problem with fixed (but arbitrary) number l of commodities, fixed (but arbitrary) number m of suppliers, and variable number n of consumers. This is natural in typical applications where few facilities serve many customers.

The data is as follows. Each supplier i has a supply vector $\mathbf{s}^i \in \mathbb{Z}_+^l$ with s_k^i its supply in commodity k. Each consumer j has a consumption vector $\mathbf{c}^j \in \mathbb{Z}_+^l$ with c_k^j its consumption in commodity k. The amount of commodity k to be routed from supplier i to consumer j is an integer decision variable $x_{i,k}^j$. The total amount $\sum_{k=1}^l x_{i,k}^j$ of commodities routed on the channel from i to j should not exceed the channel capacity $u_{i,j}$, and has cost $f_{i,j}\left(\sum_{k=1}^l x_{i,k}^j\right)$ for suitable univariate functions $f_{i,j}$. We can handle standard linear costs as well as more realistic, convex piecewise affine cost functions $f_{i,j}$, which account for channel congestion under heavy routing.

As a corollary of Theorem 4.2, for any fixed numbers l of commodities and m of suppliers, the problem can be solved in time cubic in the number n of consumers.

Corollary 5.1 *For every fixed l commodities, m suppliers, and p, there exists an algorithm that, given n consumers, supplies and demands $\mathbf{s}^i, \mathbf{c}^j \in \mathbb{Z}_+^l$, capacities $u_{i,j} \in \mathbb{Z}_+$, and convex p-piecewise affine costs $f_{i,j} : \mathbb{Z} \to \mathbb{Z}$, solves in time $O(n^3 L)$, with $L := \langle \mathbf{s}^i, \mathbf{c}^j, u_{i,j}, f_{i,j} \rangle$, the integer multicommodity transportation problem*

$$\min \left\{ \sum_{i=1}^m \sum_{j=1}^n f_{i,j} \left(\sum_{k=1}^l x_{i,k}^j \right) \, : \, x_{i,k}^j \in \mathbb{Z}_+, \ \sum_j x_{i,k}^j = s_k^i, \ \sum_i x_{i,k}^j = c_k^j, \ \sum_{k=1}^l x_{i,k}^j \le u_{i,j} \right\}.$$

Proof Introduce new variables y_i^j and equations $y_i^j = \sum_{k=1}^l x_{i,k}^j$ for all i, j. Then the objective function becomes $\sum_{i,j} f_{i,j}(y_i^j)$ which is separable convex p-piecewise affine in the new variables, and the capacity constraints become $y_i^j \le u_{i,j}$ which are upper bounds on the new variables. Use $u_{i,j}$ as an upper bound on $x_{i,k}^j$ and 0 as a trivial lower bound on y_i^j for all i, j, k. As shown in [11], arranging the original and new variables in a tuple $\mathbf{z} = (\mathbf{z}^1, \ldots, \mathbf{z}^n)$ of n bricks $\mathbf{z}^j \in \mathbb{Z}^{m(l+1)}$ defined by

$$\mathbf{z}^j := (x_{1,1}^j, \ldots, x_{1,l}^j, y_1^j, \ x_{2,1}^j, \ldots, x_{2,l}^j, y_2^j, \ldots, \ x_{m,1}^j, \ldots, x_{m,l}^j, y_m^j)$$

this problem can be modeled as an n-fold program, over a suitable $(r, s) \times t$ bimatrix A with $r = lm, s = l + m$, and $t = m(l + 1)$, resulting in solution in large time $O\left(n^{g(A)} L\right)$, with exponent depending on l, m. Theorem 4.2 now enables solution in cubic time independent of the numbers l of commodities and m of suppliers. □

5.2 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, see [7,20] and the references therein. In particular, if this set is small or consists of a unique value, that of the source table, then this entry can be exposed. Thus, it is desirable to compute the minimum and maximum integer values that can occur in an entry, which in particular are equal if and only if the entry value is unique, before margin disclosure is enabled.

Consider $(d+1)$-way tables of format $m_0 \times \cdots \times m_d$, that is, arrays $v = (v_{i_0,\ldots,i_d})$ indexed by $1 \le i_j \le m_j$ for all j, with all entries $v_{i_0,\ldots,i_d}$ nonnegative integers. Our results hold for arbitrary hierarchical margins, but for simplicity we restrict attention to disclosure of d-margins, that is, the $d+1$ many d-way tables $(v_{i_0,\ldots,i_{j-1},*,i_{j+1},\ldots,i_d})$ obtained from v by collapsing one factor $0 \le j \le d$ at a time, with entries given by

$$v_{i_0,\ldots,i_{j-1},*,i_{j+1},\ldots,i_d} := \sum_{i_j=1}^{m_j} v_{i_0,\ldots,i_{j-1},i_j,i_{j+1},\ldots,i_d}, \quad 1 \le i_k \le m_k, \;\; 0 \le k \le d, \;\; k \ne j.$$

The problem is then to compute the minimum and maximum integer values that can occur in an entry subject to the margins of the source table in the database.

This problem is NP-hard already for 3-way tables of format $n \times m \times 3$, see [6]. However, as a corollary of Theorem 3.9, if only one side n of the table is variable, the problem can be solved in cubic time regardless of the other sides m_i as follows.

Corollary 5.2 *For every fixed $d, m_1, \ldots, m_d$, there is an algorithm that, given $n := m_0$, integer d-margins $(v_{*,i_1,\ldots,i_d}), \ldots, (v_{i_0,\ldots,i_{d-1},*})$, and index $(k_0, \ldots, k_d)$, determines, in time $O(n^3 L)$, with L the binary length of the given margins, the minimum and maximum values of entry $x_{k_0,\ldots,k_d}$ among all tables with these margins, that is, solves*

$$\min / \max \left\{ x_{k_0,\ldots,k_d} : x \in \mathbb{Z}_+^{n \times m_1 \times \cdots \times m_d}, \; (x_{i_0,\ldots,i_{j-1},*,i_{j+1},\ldots,i_d}) = (v_{i_0,\ldots,i_{j-1},*,i_{j+1},\ldots,i_d}) \; \forall j \right\}.$$

Proof Let u be the maximum value of any entry in the given margins and use it as an upper bound on every variable. As shown in [5], this problem can be modeled as an n-fold integer programming problem, over a suitable $(r,s) \times t$ bimatrix A with $t = r = m_1 m_2 \ldots m_d$ and $s = \sum_{k=1}^{d} \prod_{i \ne k} m_i$, resulting in solution in large running time $O\left(n^{g(A)} L\right)$, with exponent which depends on $m_1, \ldots, m_d$. Theorem 3.9 now enables to solve it in cubic time independent of the table dimensions $m_1, \ldots, m_d$. □

We note that long tables, with one side much larger than the others, often arise in practical applications. For instance, in health statistical tables, the long factor may be the age of an individual, whereas other factors may be binary (yes-no) or ternary (subnormal, normal, and supnormal). Moreover, it is always possible to merge categories of factors, with the resulting coarser tables approximating the original ones, making the algorithm of Corollary 5.2 applicable.

We also note that, by repeatedly incrementing a lower bound and decrementing an upper bound on the entry $x_{k_0,\ldots,k_d}$, and computing its new minimum and maximum values subject to these bounds, we can produce the entire set of values that can occur in that entry in time proportional to the number of such values.

6 Solvability over bimatrices with variable entries

The drop of the Graver complexity from the exponent of n to the constant multiple also leads to the first polynomial time solution of n-fold integer programming with variable bimatrices. Of course, by the universality of n-fold integer programming, the variability of the bimatrices must be limited. In what follows, we fix the dimensions r, s, t of the input bimatrix A, and let the entries vary. We show that, given as part of the input an upper bound a on the absolute value of every entry of A, we can solve the problem in time polynomial in a, that is, polynomial in the unary length of a. This holds for linear as well as separable convex piecewise affine objectives. Before we state our complexity result, let us prove a useful bound on the Graver complexity of A in this case.

Lemma 6.1 *For any fixed r, s, t, and fixed $(r, s) \times t$ bimatrix A with all entries bounded by a, the Graver complexity $g(A)$ of A is bounded by $g(A) = O(a^{rs+st+r})$.*

Proof Let $\mathcal{G}(A_2)$ be the Graver basis of the $s \times t$ second block A_2 of A, let $q := |\mathcal{G}(A_2)|$ be its cardinality, and arrange its elements as the columns of a $t \times q$ matrix G_2. Since r, s, t are fixed, it follows from bounds on Graver bases (see e.g. [17, Section 3.4]) that every $g \in \mathcal{G}(A_2)$ satisfies $\|g\|_\infty = O(a^s)$ and hence $q = O(a^{st})$.

Now, it is known (see [13,18] or [17, Section 4.1]) that the Graver complexity $g(A)$ of A is equal to the maximum value $\|v\|_1$ of any element v in the Graver basis $\mathcal{G}(A_1G_2)$ of the $r \times q$ matrix A_1G_2. Since the entries of A_1G_2 are bounded in absolute value by $O(a^{s+1})$, the bounds on Graver bases (see again [17, Section 3.4]) imply that $\|v\|_1 = O(q \cdot (a^{s+1})^r)$ for every $v \in \mathcal{G}(A_1G_2)$ and hence $g(A) = O(a^{rs+st+r})$. □

We now have the following theorem, with $L := \langle f, a, \mathbf{b}, \mathbf{l}, \mathbf{u}\rangle$ the length of the input.

Theorem 6.2 *For any fixed r, s, t and p, there is an algorithm that, given n, a, $(r, s) \times t$ bimatrix A with all entries bounded by a in absolute value, $\mathbf{b}, \mathbf{l}, \mathbf{u}$, and separable convex p-piecewise affine f, in polynomial time $O(a^{3t(rs+st+r+s)}n^3L)$, solves*

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

Proof Let $\mathcal{G}(A_2)$ again denote the Graver basis of the $s \times t$ second block A_2 of A. From the proof of Lemma 6.1 we get that the maximum value of $\|\mathbf{g}\|_\infty$ over all $\mathbf{g} \in \mathcal{G}(A_2)$ is $O(a^s)$. Now, consider the following set defined in (4) in Lemma 3.1,

$$Z(A) \;:=\; \left\{\mathbf{z} \in \mathbb{Z}^t \;:\; \mathbf{z} \text{ is the sum of at most } g(A) \text{ elements of } \mathcal{G}(A_2)\right\}.$$

For each $\mathbf{z} \in Z(A)$ we have that $\|\mathbf{z}\|_\infty$ is bounded by $g(A)$ times the maximum value of $\|\mathbf{g}\|_\infty$ over all $\mathbf{g} \in \mathcal{G}(A_2)$, and therefore, by Lemma 6.1, $\|\mathbf{z}\|_\infty = O(g(A) \cdot a^s) = O(a^{rs+st+r+s})$. So the cardinality of $Z(A)$ satisfies $|Z(A)| = O((a^{rs+st+r+s})^t) = O(a^{t(rs+st+r+s)})$.

Now, suitable analogs of some of the lemmas of Sect. 3 go through, except that the complexities now depend on the variable size of the set $Z(A)$, as follows. The time complexity of the algorithm of Lemma 3.4 becomes $O(|Z(A)|^2 n)$. The size of the set Γ of critical values is now $O(|Z(A)|n)$ and therefore the time complexity of the algorithm of Lemma 3.6 now becomes $O(|Z(A)|^3 n^2)$. The number of iterations needed to augment an initial feasible point to an optimal solution remains $O(nL)$ as before and therefore the time complexity of the algorithm of Lemma 3.7 now become $O(|Z(A)|^3 n^3 L)$. To find an initial feasible point, one can use the algorithm of Lemma 3.8, but then t would have to be replaced by $t + 2r + 2s$ for the auxiliary bimatrix $\bar{A}$, resulting in a somewhat larger exponent for a in the running time. However, it is possible to find an initial feasible point in an alternative, somewhat more involved way, keeping the original system with the bimatrix A, as follows. First find an integer solution to the system of equations only (without the lower and upper bounds) using the Hermite normal forms of the blocks A_1 and A_2. Second, relax the bounds so as to make that point feasible. Third, minimize the following auxiliary objective function which is separable convex 3-piecewise affine, with

$$f_j^i(x_j^i) := \begin{cases} l_j^i - x_j^i, & \text{if } x_j^i \le l_j^i, \\ 0, & \text{if } l_j^i \le x_j^i \le u_j^i, \\ x_j^i - u_j^i, & \text{if } x_j^i \ge u_j^i. \end{cases}$$

If the optimal value is zero then the optimal auxiliary solution is feasible in the original problem, whereas if it is positive then the original problem is infeasible. Since this minimization can be done using the separable convex piecewise affine analog of Lemma 3.7 described in the proof of Theorem 4.2 in the same complexity $O(|Z(A)|^3 n^3 L)$, the overall running time is $O(a^{3t(rs+st+r+s)} n^3 L)$ as claimed. □

7 Parametrization and approximation hierarchy

We conclude with a short discussion of the universality of n-fold integer programming and the resulting parametrization and simple approximation hierarchy for all of integer programming. As mentioned in the introduction, every integer program is an n-fold program for some m over the bimatrix $A(m)$ having first block the identity matrix I_{3m} and second block the $(3+m) \times 3m$ incidence matrix of $K_{3,m}$. It is convenient and illuminating to introduce also the following description.

Consider the following special form of the n-fold product operator. For an $s \times t$ matrix D, let $D^{[n]} := A^{(n)}$ where A is the $(t, s) \times t$ bimatrix $A = \binom{A_1}{A_2}$ with first block $A_1 := I_t$ the $t \times t$ identity matrix and second block $A_2 := D$. We consider such m-fold products of the 1×3 matrix $(1\ 1\ 1)$. Note that $(1\ 1\ 1)^{[m]}$ is precisely the $(3+m) \times 3m$ incidence matrix of the complete bipartite graph $K_{3,m}$. For instance,

$$(1\ 1\ 1)^{[2]} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

The following theorem was established in [6].

The Universality Theorem [6] *Every (bounded) integer programming problem* $\min\{\mathbf{cy} : \mathbf{y} \in \mathbb{Z}_+^k,\ V\mathbf{y} = \mathbf{v}\}$ *is polynomial time equivalent to some integer program*

$$\min\left\{\mathbf{wx} : \mathbf{x} \in \mathbb{Z}_+^{3mn}, (1\ 1\ 1)^{[m][n]}\mathbf{x} = \mathbf{b}\right\} \cong \min\left\{\mathbf{wx} : \mathbf{x} \in \mathbb{Z}_+^{3mn}, A(m)^{(n)}\mathbf{x} = \mathbf{b}\right\}.$$

This theorem provides a new, variable dimension, parametrization of integer programming: for each fixed value of the parameter m, the resulting programs above with variable parameter n live in variable dimension $3mn$ and include natural models such as those described in Sect. 5, and can be solved in cubic time $O(n^3 L)$ by Theorem 3.9; and when the parameter m varies, every integer program appears for some m.

Our new algorithm suggests a natural simple approximation hierarchy for integer programming, parameterized by degree d, as follows. Fix any d. Then given any m, let $A := A(m)$, so $t = 3m$, $A_1 = I_{3m}$, and A_2 is the incidence matrix of $K_{3,m}$. Define the approximation at degree d of the set $Z(A)$ in Eq. (4) in Lemma 3.1 by

$$Z_d(m) := \left\{\mathbf{z} \in \mathbb{Z}^{3m} : \mathbf{z} \text{ is the sum of at most } d \text{ elements of } \mathcal{G}(A_2)\right\}. \tag{7}$$

Since A_2 is totally unimodular, $\mathcal{G}(A_2)$ consists of the $O(m^3)$ vectors in $\{0, \pm 1\}^{3m}$ supported on circuits of $K_{3,m}$ with alternating ± 1 and hence $|Z_d(m)| = O(m^{3d})$.

Now, given a feasible point $\mathbf{x}$ in the universal program above, and positive integer γ, set a dynamic program similar to that in Definition 3.3, with the only modification that the sets S_i are defined using the approximation $Z_d(m)$ of $Z(A)$. Weaker forms of Lemmas 3.4 and 3.6 now assert that in time $O(|Z_d(m)|^2 n^2) = O(m^{6d} n^2)$, which is polynomial in both m and n, we can find a good feasible step $\gamma\mathbf{g}$. We use this iteratively to augment an initial feasible point to one which is as good as possible and output it. However, Lemma 3.1 no longer holds and not all brick sums of elements of the Graver basis $\mathcal{G}(A^{(n)})$ lie in $Z_d(m)$. So the bounds on the number of iterations and total running time are no longer valid and the output point may be non optimal.

By increasing the degree d we can get better approximations at increasing running times, and when $d = g(A)$ we get the true optimal solution. These approximations are currently under study, implementation and testing. They show promising behavior already at degree $d = 3$ and will be discussed in more detail elsewhere.

For $m = 3$, discussed in Example 3.2, for which the universal problem is equivalent to optimization over 3-way $n \times 3 \times 3$ tables, the approximation $Z_3(3)$ at degree $d = 3$ contains only 811 vectors out of the 42931 vectors in the true $Z(A)$, such as

$$\begin{pmatrix} -3 & 2 & 1 & 2 & -3 & 1 & 1 & 1 & -2 \end{pmatrix}.$$

Acknowledgments Shmuel Onn is supported in part by the Israel Science Foundation. Lyubov Romanchuk is supported in part by the Technion Graduate School.

References

1. Aoki, S., Takemura, A.: Minimal basis for connected Markov chain over $3 \times 3 \times K$ contingency tables with fixed two-dimensional marginals. Aust. New Zeal. J. Stat. **45**, 229–249 (2003)
2. Berstein, Y., Onn, S.: The Graver complexity of integer programming. Ann. Combin. **13**, 289–296 (2009)
3. Cook, W., Fonlupt, J., Schrijver, A.: An integer analogue of Carathéodory's theorem. J. Combin. Theory Ser. B **40**, 63–70 (1986)
4. De Loera, J., Hemmecke, R., Onn, S., Rothblum, U.G., Weismantel, R.: Convex integer maximization via Graver bases. J. Pure Appl. Algebra **213**, 1569–1577 (2009)
5. De Loera, J.A., Hemmecke, R., Onn, S., Weismantel, R.: N-fold integer programming. Discret. Optim. **5**, 231–241 (2008). (Volume in memory of George B. Dantzig)
6. De Loera, J.A., Onn, S.: All linear and integer programs are slim 3-way transportation programs. SIAM J. Optim. **17**, 806–821 (2006)
7. Dobra, A., Fienberg, S.E., Rinaldo, A., Slavković, A., Zhou, Y.: Algebraic statistics and contingency table problems: log-linear models, likelihood estimation, and disclosure limitation. In: Emerging Applications of Algebraic Geometry: IMA Volumes in Mathematics and its Applications, vol. 148, pp. 63–88. Springer (2009)
8. Graver, J.E.: On the foundation of linear and integer programming I. Math. Program. **9**, 207–226 (1975)
9. Hemmecke, R., Köppe, M., Weismantel, R.: A polynomial-time algorithm for optimizing over N-fold 4-block decomposable integer programs. In: IPCO, vol. 14 (2010)
10. Hemmecke, R., Onn, S., Weismantel, R.: A polynomial oracle-time algorithm for convex integer minimization. Math. Program. **126**, 97–117 (2011)
11. Hemmecke, R., Onn, S., Weismantel, R.: N-fold integer programming and nonlinear multi-transshipment. Optim. Lett. **5**, 13–25 (2011)
12. Hemmecke, R., Schultz, R.: Decomposition of test sets in stochastic integer programming. Math. Program. **94**, 323–341 (2003)
13. Hoşten, S., Sullivant, S.: Finiteness theorems for Markov bases of hierarchical models. J. Combin. Theory Ser. A **114**, 311–321 (2007)
14. Kobayashi, Y., Murota, K., Weismantel, R.: Cone superadditivity of discrete convex functions. Math. Program. (to appear). doi:10.1007/s10107-011-0447-1
15. Louveaux, F.V., Schultz, R.: Stochastic integer programming. In: Handbooks in Operations Research and Management Science, vol. 10 pp. 213–266. Elsevier (2003)
16. Murota, K., Saito, H., Weismantel, R.: Optimality criterion for a class of nonlinear integer programs. Oper. Res. Lett. **32**, 468–472 (2004)
17. Onn, S.: Nonlinear Discrete Optimization. Zurich Lectures in Advanced Mathematics. Eur. Math. Soc. x+137 pp. (September 2010)
18. Santos, F., Sturmfels, B.: Higher Lawrence configurations. J. Combin. Theory Ser. A **103**, 151–164 (2003)
19. Sebö, A.: Hilbert Bases, Carathéodory's Theorem and Combinatorial Optimization. pp. 431–607. vol. 1, Waterloo University Press, IPCO, Waterloo (1990)
20. Slavković, A.B., Zhu, X., Petrović, S.: A sample space of k-way tables given conditionals and their relations to marginals: Implications for cell bounds and Markov bases. Preprint, 35 pp. (2009)