Combinatorial n-fold Integer Programming and Applications*†

Dušan Knop[1], Martin Koutecký[2], and Matthias Mnich[3]

1 **Charles University, Prague, Czech Republic, and Department of Informatics, University of Bergen, Bergen, Norway**
knop@kam.mff.cuni.cz

2 **Charles University, Prague, Czech Republic**
koutecky@kam.mff.cuni.cz

3 **Universität Bonn, Bonn, Germany, and Maastricht University, Maastricht, The Netherlands**
mmnich@uni-bonn.de

Abstract

Many fundamental NP-hard problems can be formulated as integer linear programs (ILPs). A famous algorithm by Lenstra allows to solve ILPs in time that is exponential only in the dimension of the program. That algorithm therefore became a ubiquitous tool in the design of fixed-parameter algorithms for NP-hard problems, where one wishes to isolate the hardness of a problem by some parameter. However, it was discovered that in many cases using Lenstra's algorithm has two drawbacks: First, the run time of the resulting algorithms is often doubly-exponential in the parameter, and second, an ILP formulation in small dimension can not easily express problems which involve many different costs.

Inspired by the work of Hemmecke, Onn and Romanchuk [Math. Prog. 2013], we develop a single-exponential algorithm for so-called *combinatorial n-fold integer programs*, which are remarkably similar to prior ILP formulations for various problems, but unlike them, also allow variable dimension. We then apply our algorithm to a few representative problems like Closest String, Swap Bribery, Weighted Set Multicover, and obtain exponential speedups in the dependence on the respective parameters, the input size, or both.

Unlike Lenstra's algorithm, which is essentially a bounded search tree algorithm, our result uses the technique of augmenting steps. At its heart is a deep result stating that in combinatorial n-fold IPs an existence of an augmenting step implies an existence of a "local" augmenting step, which can be found using dynamic programming. Our results provide an important insight into many problems by showing that they exhibit this phenomenon, and highlights the importance of augmentation techniques.

1998 ACM Subject Classification F2.2 Nonnumerical Algorithms and Problems

Keywords and phrases integer programming, closest strings, fixed-parameter algorithms

Digital Object Identifier 10.4230/LIPIcs.ESA.2017.54

* D.K. supported by the CE-ITI grant project P202/12/G061 of GA ČR; M.K. supported by the GA UK grant project 1784214. M.M. supported by ERC Starting Grant 305465 (BeyondWorstCase).
† A full version of the paper is available at https://arxiv.org/abs/1705.08657.

1 Introduction

The Integer Linear Programming (ILP) problem is fundamental as it models many combinatorial optimization problems. Since it is NP-complete, we naturally ask about the complexity of special cases. A fundamental algorithm by Lenstra from 1983 shows that ILPs can be solved in polynomial time when their number of variables (the dimension) d is fixed [30]; that algorithm is thus a natural tool to prove that the complexity of some special cases of other NP-hard problems is also polynomial.

A systematic way to study the complexity of "special cases" of NP-hard problems has been developed in the past 25 years in the field of parameterized complexity. There, the problem input is augmented by some integer parameter k, and one then measures the problem complexity in terms of both the instance size n as well as k. Of central importance are algorithms with run times of the form $f(k)n^{O(1)}$ for some computable function f, which are called *fixed-parameter algorithms*; the key idea is that the degree of the polynomial does not grow with k. For background on parameterized complexity, we refer to the monograph [7].

Kannan's improvement [23] of Lenstra's algorithm runs in time $d^{O(d)}n$, which is thus a fixed-parameter algorithm for parameter d. Gramm et al. [17] pioneered the application of Lenstra's and Kannan's algorithm in parameterized complexity, giving a fixed-parameter algorithm for the Closest String problem [17]. This led Niedermeier [34] to propose:

> *[...] It remains to investigate further examples besides* Closest String *where the described ILP approach turns out to be applicable. More generally, it would be interesting to discover more connections between fixed-parameter algorithms and (integer) linear programming.*

Since then, many more applications of Lenstra's and Kannan's algorithm for parameterized problems have been proposed. However, essentially all of them [5, 9, 10, 21, 33, 29] share a common trait with the algorithm for Closest String: they have a doubly-exponential dependence on the parameter. Moreover, it is difficult to find ILP formulations with small dimension for problems whose input contains many objects with varying cost functions, such as in Swap Bribery [4, Challenge #2].

Our contributions. We show that a certain form of ILP, which is closely related to the previously used formulations for Closest String and other problems, can be solved in single-exponential time and in variable dimension. For example, Gramm et al.'s [17] algorithm for Closest String runs in time $2^{2^{O(k \log k)}} \log L$ and has not been improved since 2003, while our algorithm runs in time $k^{O(k^2)} \log L$. Moreover, our algorithm has a strong combinatorial flavor and is based on different notions than are typically encountered in parameterized complexity, most importantly augmenting steps.

As an example of our form of ILP, consider the following ILP formulation of the Closest String problem. We are given k strings $s_1, \ldots, s_k$ of length L that come (after some preprocessing) from alphabet $[k] := \{1, \ldots, k\}$, and an integer d. The goal is to find a string $y \in [k]^L$ such that, for each s_i, the Hamming distance $d_H(y, s_i)$ is at most d, if such y exists. For $i \in [L]$, $(s_1[i], \ldots, s_k[i])$ is the i-th column of the input. Clearly there are at most k^k different column types in the input, and we can represent the input succinctly with multiplicities $b^{\mathbf{f}}$ of each column type $\mathbf{f} \in [k]^k$. Moreover, there are k choices for the output string y in each column. Thus, we can encode the solution by, for each column type $\mathbf{f} \in [k]^k$ and each output character $e \in [k]^k$, describing how many solution columns are of type $(\mathbf{f}, e)$.

This is the basic idea behind the formulation of Gramm et al. [17], as depicted on the left:

$$\begin{array}{rcll} \sum_{e\in[k]}\sum_{\mathbf{f}\in[k]^k} d_H(e,f_j)x_{\mathbf{f},e} & \leq & d & \\ \sum_{e\in[k]} x_{\mathbf{f},e} & = & b^{\mathbf{f}} & \\ x_{\mathbf{f},e} & \geq & 0 & \end{array} \quad \left| \quad \begin{array}{rcll} \sum_{\mathbf{f}\in[k]^k}\sum_{(\mathbf{f}',e)\in[k]^{k+1}} d_H(e,f_j)x^{\mathbf{f}}_{\mathbf{f}',e} & \leq & d & \forall j\in[k] \\ \sum_{(\mathbf{f}',e)\in[k]^{k+1}} x^{\mathbf{f}}_{\mathbf{f}',e} & = & b^{\mathbf{f}} & \forall \mathbf{f}\in[k]^k \\ & & & \forall(\mathbf{f},e)\in[k]^{k+1} \\ x^{\mathbf{f}'}_{\mathbf{f},e} & = & 0 & \forall \mathbf{f}'\neq\mathbf{f}, \forall e\in[k] \\ 0\leq x^{\mathbf{f}}_{\mathbf{f},e} & \leq & b^{\mathbf{f}} & \forall \mathbf{f}\in[k]^k \end{array}$$

Let $(1\ \cdots\ 1) = \mathbf{1}^\intercal$ be a row vector of all ones. Then we can view the above as

$$\begin{array}{cccc|c} D_1 & D_2 & \cdots & D_{k^k} & \leq d \\ \mathbf{1}^\intercal & 0 & \cdots & 0 & = b^1 \\ 0 & \mathbf{1}^\intercal & \cdots & 0 & = b^2 \\ \vdots & \vdots & \ddots & \vdots & = \vdots \\ 0 & 0 & \cdots & \mathbf{1}^\intercal & = b^{k^k} \end{array} \quad \left| \quad \begin{array}{cccc|c} D & D & \cdots & D & \leq d \\ \mathbf{1}^\intercal & 0 & \cdots & 0 & = b^1 \\ 0 & \mathbf{1}^\intercal & \cdots & 0 & = b^2 \\ \vdots & \vdots & \ddots & \vdots & = \vdots \\ 0 & 0 & \cdots & \mathbf{1}^\intercal & = b^{k^k}, \end{array}\right.$$

where $D = (D_1\ D_2\ \ldots\ D_{k^k})$. The formulation on the right is clearly related to the one on the left, but contains "dummy" variables which are always zero. This makes it seem unnatural at first, but notice that it has the nice form

$$\min\left\{f(\mathbf{x}) \mid E^{(n)}\mathbf{x}=\mathbf{b},\ \mathbf{l}\leq\mathbf{x}\leq\mathbf{u},\ \mathbf{x}\in\mathbb{Z}^{nt}\right\},\ \text{where}\ E^{(n)} := \begin{pmatrix} D & D & \cdots & D \\ A & 0 & \cdots & 0 \\ 0 & A & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A \end{pmatrix}. \quad (1)$$

Here, $r, s, t, n \in \mathbb{N}$, $\mathbf{u}, \mathbf{l} \in \mathbb{Z}^{nt}$, $\mathbf{b} \in \mathbb{Z}^{r+ns}$ and $f : \mathbb{Z}^{nt} \to \mathbb{Z}$ is a separable convex function, $E^{(n)}$ is an $(r+ns)\times nt$-matrix, $D \in \mathbb{Z}^{r\times t}$ is an $r\times t$-matrix and $A \in \mathbb{Z}^{s\times t}$ is an $s\times t$-matrix. We call $E^{(n)}$ the *n-fold product of* $E = \binom{D}{A}$. This problem (1) is known as *n-fold integer programming* $(IP)_{E^{(n)},\mathbf{b},\mathbf{l},\mathbf{u},f}$. Building on a dynamic program of Hemmecke, Onn and Romanchuk [19] and a so-called proximity technique of Hemmecke, Köppe and Weismantel [18], Knop and Koutecký [25] prove that:

▶ **Proposition 1** ([25, Theorem 7]). *There is an algorithm that, given $(IP)_{E^{(n)},\mathbf{b},\mathbf{l},\mathbf{u},f}$ encoded with L bits, solves*[1] *it in time $a^{O(trs+t^2s)}\cdot n^3L$, where $a = \max\{\|D\|_\infty, \|A\|_\infty\}$.*

However, since the ILP on the bottom right of the previous page has $t = k^k$, applying Proposition 1 gives no advantage over applying Lenstra to solve the Closest String problem. We overcome this by focusing on a special case with $A = (1\ \cdots\ 1) = \mathbf{1}^\intercal \in \mathbb{Z}^{1\times t}$, $(b^1,\ldots,b^n) \geq \mathbf{0}$, $u^i_j \in \{0, \|\mathbf{b}\|_\infty\}$ for all $i \in [n]$ and $j \in [t]$[2], and $f(\mathbf{x}) = \mathbf{w}^\intercal\mathbf{x}$, i.e., the objective is linear. We denote $f^i(\mathbf{x}^i) = \mathbf{w}^{i\intercal}\mathbf{x}^i$. We call this form *combinatorial n-fold IP*[3], and achieve an exponential speed-up in t:

[1] Given an (IP), we say that to *solve* it is to either (i) declare it infeasible or unbounded or (ii) find a minimizer of it.

[2] More precisely, $\mathbf{x} \geq \mathbf{0}$ and $\mathbf{1}^\intercal\mathbf{x}^i = b^i$ imply $x^i_j \leq b^i$ and thus we just need either $u^i_j = 0$ or $u^i_j \geq b^i$.

[3] We deliberately use the term "n-fold IP" even if our objective is linear, making it an ILP, in order to be consistent with the previous literature [19, 31, 35].

▶ **Theorem 2.** *Let $(IP)_{E^{(n)},\mathbf{b},\mathbf{0},\mathbf{u},\mathbf{w}}$ be a combinatorial n-fold IP instance with $L = \langle \mathbf{b}, \mathbf{0}, \mathbf{u}, \mathbf{w} \rangle$ and $a = \|D\|_\infty$. Then it can be solved in time $t^{O(r)}(ar)^{O(r^2)}n^3L$.*

Observe that, when applicable, our algorithm is not only faster than Lenstra's, but works even if the number n is variable (not parameter).

By applying this result to a few selected problems we obtain exponential improvements in the dependence on the parameter, the length of the input, or both, as presented in Table 1. Statements whose proofs are omitted due to space constraints are marked with $\star$.

Stringology. A typical problem from stringology is to find a string y satisfying certain distance properties with respect to k strings $s_1, \ldots, s_k$. All previous fixed-parameter algorithms for such problems we are aware of for parameter k rely on Lenstra's algorithm, or their complexity status was complexity open (e.g., the complexity of Optimal Consensus [1] was unknown for all $k \geq 4$). Interestingly, Boucher and Wilkie [3] show the counterintuitive fact that Closest String is easier to solve when k is large, which makes the parameterization by k even more significant. Finding an algorithm with run time only single-exponential in k was a repeatedly posed open problem, e.g. [6, Challenge #1] and [2, Problem 7.1]. By applying our result, we close this gap for a wide range of problems.

▶ **Theorem 3** ($\star$). *The problems*

- Closest String, Farthest String, Distinguishing String Selection, Neighbor String, Closest String with Wildcards, Closest to Most Strings, c-HRC *and* Optimal Consensus *are solvable in time $k^{O(k^2)} \log L$, and,*
- d-Mismatch *is solvable in time $k^{O(k^2)}L^2 \log L$,*

where k is the number of input strings, L is their length, and we are assuming that the input is presented succinctly by multiplicities of identical columns.

Computational Social Choice. A typical problem in computational social choice involves an election with voters (V) and candidates (C). A natural and much studied parameter is the number of candidates $|C|$. For a long time, only algorithms double-exponential in $|C|$ were known, and improving upon them was posed as a challenge [4, Challenge #1]. Recently, Knop et al. [27] solved the challenge using Proposition 1. However, Knop et al.'s result has a cubic dependence $O(|V|^3)$ on the number of voters, and the dependence on the number of candidates is still quite large, namely $|C|^{O(|C|^6)}$. We improve their result as follows:

▶ **Theorem 4** ($\star$). $\mathcal{R}$-Swap Bribery *can be solved in time*

- $|C|^{O(|C|^2)}T^3(\log |V| + \log \sigma_{\max})$ *for $\mathcal{R}$ any natural scoring protocol, and,*
- $|C|^{O(|C|^4)}T^3(\log |V| + \log \sigma_{\max})$ *for $\mathcal{R}$ any C1 rule,*

where $T \leq |V|$ is the number of voter types and $\sigma_{\max}$ is the maximum cost of a swap.

Weighted Set Multicover. Bredereck et al. [5] define the Weighted Set Multicover (WSM) problem, which is a significant generalization of the classical Set Cover problem. Their motivation to study WSM was that it captures several problems from computational social choice and optimization problems on graphs [11, 13, 29, implicit in]. Bredereck et al. [5] design an algorithm for WSM that runs in time $2^{2^{O(k \log k)}} \log n$, using Lenstra's algorithm.

Again, applying our result yields an exponential improvement over that of Bredereck et al. [4] both in the dependence on the parameter and the size of the instance:

▶ **Theorem 5.** *There is an algorithm that solves the* Weighted Set Multicover *problem and runs in time $k^{O(k^2)}W^3(\log n + \log w_{\max})$, where k is the size of the universe, n denotes the number of sets, W is the number of different weights and $w_{\max}$ is the maximum weight.*

Table 1 Complexity improvements for a few representative problems.

Problem	Previous best runtime	Our result
Closest String	$2^{2^{O(k \log k)}} \log L$ [17]	$k^{O(k^2)} \log L$
Optimal Consensus	FPT for $k \leq 3$, open for $k \geq 4$ [1]	$k^{O(k^2)} \log L$
Score-Swap Bribery	$2^{2^{O(\lvert C\rvert \log \lvert C\rvert)}} \log \lvert V\rvert$ [9] / $\lvert C\rvert^{O(\lvert C\rvert^6)} \lvert V\rvert^3$ [27]	$\lvert C\rvert^{O(\lvert C\rvert^2)} T^3 \log \lvert V\rvert$, with $T \leq \lvert V\rvert$
C1-Swap Bribery	$2^{2^{O(\lvert C\rvert \log \lvert C\rvert)}} \log \lvert V\rvert$ [9] / $\lvert C\rvert^{O(\lvert C\rvert^6)} \lvert V\rvert^3$ [27]	$\lvert C\rvert^{O(\lvert C\rvert^4)} T^3 \log \lvert V\rvert$, with $T \leq \lvert V\rvert$
Weighted Set Multicover	$2^{2^{O(k \log k)}} n$ [5]	$k^{O(k^2)} \log n$
Huge n-fold IP	FPT with $D = I$ and A totally unimodular	FPT with parameter-sized domains

Huge n-fold IP. Onn [36] introduces a high-multiplicity version of the standard n-fold IP problem (1). It is significant because of its connection to the Bin Packing problem in the case of few item sizes, as studied by Goemans and Rothvoss [16]. Previously, Huge n-fold IP was shown to be fixed-parameter tractable when $D = I$ and A is totally unimodular; using our result, we show that it is also fixed-parameter tractable when D and A are arbitrary, but the size of variable domains is bounded by a parameter.

A summary of our results is given in Table 1; this list is not meant to be exhaustive. In fact, we believe that for any Lenstra-based result in the literature which only achieves double-exponential run times, there is a good chance that it can be sped up using our algorithm. The only significant obstacle seem to be large coefficients in the constraint matrix. We provide further insights and discussion in the full version of the paper [26].

Related work. Our main inspiration are augmentation methods based on Graver bases, especially a fixed-parameter algorithm for n-fold IP of Hemmecke, Onn and Romanchuk [19]. Our result improves the runtime of their algorithm for a special case. All the following related work is orthogonal to ours in either the achieved result, or the parameters used for it.

In fixed dimension, Lenstra's algorithm [30] was generalized for arbitrary convex sets and quasiconvex objectives by Khachiyan and Porkolab [24]. The currently fastest algorithm of this kind is due to Dadush et al. [8]. The first notable fixed-parameter algorithm for a non-convex objective is due to Lokshtanov [32], who shows that optimizing a quadratic function over the integers of a polytope is fixed-parameter tractable if all coefficients are small. Ganian and Ordyniak [14] and Ganian et al. [15] study the complexity of ILP with respect to structural parameters such as treewidth and treedepth, and introduce a new parameter called *torso-width*.

Besides fixed-parameter tractability, there is interest in the (non)existence of kernels of ILPs, which formalize the (im)possibility of various preprocessing procedures. Jansen and Kratsch [22] show that ILPs containing parts with simultaneously bounded treewidth and bounded domains are amenable to kernelization, unlike ILPs containing totally unimodular parts. Kratsch [28] studies the kernelizability of sparse ILPs with small coefficients.

2 Preliminaries

For positive integers m, n we set $[m : n] = \{m, \ldots, n\}$ and $[n] = [1 : n]$. For a graph G we denote by $V(G)$ the set of its vertices. We write vectors in boldface (e.g., $\mathbf{x}, \mathbf{y}$ etc.) and their entries in normal font (e.g., the i-th entry of $\mathbf{x}$ is x_i). Given an matrix $A \in \mathbb{Z}^{m \times n}$, vectors

$\mathbf{b} \in \mathbb{Z}^m$, $\mathbf{l}, \mathbf{u} \in \mathbb{Z}^n$ and a function $f : \mathbb{Z}^n \to \mathbb{Z}$, we denote by $(IP)_{A,\mathbf{b},\mathbf{l},\mathbf{u},f}$ the problem

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

We say that $\mathbf{x}$ is feasible for $(IP)_{A,\mathbf{b},\mathbf{l},\mathbf{u},f}$ if $A\mathbf{x} = \mathbf{b}$ and $\mathbf{l} \le \mathbf{x} \le \mathbf{u}$. If we want to talk about any such IP, we simply denote it as (IP).

Graver Bases and Augmentation. Let us now introduce Graver bases, how they can be used for optimization, and also the special case of n-fold IPs. For background, we refer to the books of Onn [35] and De Loera et al. [31].

Given two n-dimensional integer vectors $\mathbf{x}$ and $\mathbf{y}$, we say they are *sign-compatible* if they lie in the same orthant, or equivalently, if for each $i \in [n]$, the sign of x_i and y_i is the same. We say $\sum_i \mathbf{g}^i$ is a *sign-compatible sum* if all $\mathbf{g}^i$ are pair-wise sign-compatible. Moreover, we write $\mathbf{y} \sqsubseteq \mathbf{x}$ if $\mathbf{x}$ and $\mathbf{y}$ are sign-compatible and $|y_i| \le |x_i|$ for each $i \in [n]$, and write $\mathbf{y} \sqsubset \mathbf{x}$ if at least one of the inequalities is strict. Clearly, $\sqsubseteq$ imposes a partial order called "conformal order" on n-dimensional vectors. For an integer matrix $A \in \mathbb{Z}^{m\times n}$, its *Graver basis* $\mathcal{G}(A)$ is the set of $\sqsubseteq$-minimal non-zero elements of the *lattice* of A, $\ker_{\mathbb{Z}}(A) = \{\mathbf{z} \in \mathbb{Z}^n \mid A\mathbf{z} = \mathbf{0}\}$. An important property of $\mathcal{G}(A)$ is the following.

▶ **Proposition 6** ([35, Lemma 3.2]). *Every integer vector $\mathbf{x} \neq \mathbf{0}$ with $A\mathbf{x} = \mathbf{0}$ is a sign-compatible sum $\mathbf{x} = \sum_i \mathbf{g}^i$ of Graver basis elements $\mathbf{g}^i \in \mathcal{G}(A)$, with some elements possibly appearing with repetitions.*

Given a feasible solution $\mathbf{x}$ to an (IP), we call $\mathbf{g}$ a *feasible step* if $\mathbf{x} + \mathbf{g}$ is feasible in (IP). Moreover, we call a feasible step $\mathbf{g}$ *augmenting* if $f(\mathbf{x} + \mathbf{g}) < f(\mathbf{x})$. Given a feasible solution $\mathbf{x}$ to (IP), we call a tuple $(\mathbf{g}, \alpha)$ with $\alpha \in \mathbb{Z}$ a *Graver-best step* if $\mathbf{g}$ is an augmenting step and $\forall \tilde{\mathbf{g}} \in \mathcal{G}(A)$ and $\forall \alpha' \in \mathbb{Z}$, $f(\mathbf{x} + \alpha\mathbf{g}) \le f(\mathbf{x} + \alpha'\tilde{\mathbf{g}})$. We call α the *step length*. The *Graver-best augmentation procedure* for an (IP) and a given feasible solution $\mathbf{x}_0$ works as follows:

1. If there is no Graver-best step for $\mathbf{x}_0$, return it as optimal.
2. If a Graver-best step $(\alpha, \mathbf{g})$ for $\mathbf{x}_0$ exists, set $\mathbf{x}_0 := \mathbf{x}_0 + \alpha\mathbf{g}$ and go to 1.

▶ **Proposition 7** ([31, implicit in Theorem 3.4.1]). *Given a feasible solution $\mathbf{x}_0$ and a separable convex function f, the Graver-best augmentation procedure finds an optimum in at most $2n - 2\log M$ steps, where $M = f(\mathbf{x}_0) - f(\mathbf{x}^*)$ and $\mathbf{x}^*$ is any minimizer.*

***n*-fold IP.** The structure of $E^{(n)}$ (in problem (1)) allows us to divide the nt variables of $\mathbf{x}$ into n *bricks* of size t. We use subscripts to index within a brick and superscripts to denote the index of the brick, i.e., x^i_j is the j-th variable of the i-th brick with $j \in [t]$ and $i \in [n]$.

3 Combinatorial n-fold IPs

This section is dedicated to proving Theorem 2. We fix an instance of combinatorial n-fold IP, that is, a tuple $(n, D, \mathbf{b}, \mathbf{u}, \mathbf{w})$.

3.1 Graver complexity of combinatorial n-fold IP

The key property of the n-fold product $E^{(n)}$ is that, for any $n \in \mathbb{N}$, the number of nonzero bricks of any $\mathbf{g} \in \mathcal{G}(E^{(n)})$ is bounded by some constant $g(E)$ called the *Graver complexity of E*. A proof is given for example by Onn [35, Lemma 4.3]; it goes roughly as follows. Consider any $\mathbf{g} \in \mathcal{G}(E^{(n)})$ and take its restriction to its nonzero bricks $\bar{\mathbf{g}}$. By Proposition 6,

each brick $\bar{\mathbf{g}}^j$ can be decomposed into elements from $\mathcal{G}(A)$, giving a vector $\mathbf{h}$ whose bricks are elements of $\mathcal{G}(A)$. Then, consider a compact representation $\mathbf{v}$ of $\mathbf{h}$ by counting how many times each element from $\mathcal{G}(A)$ appears. Since $\mathbf{g} \in \mathcal{G}\big(E^{(n)}\big)$ and $\mathbf{h}$ is a decomposition of its nonzero bricks, we have that $\sum_j D\mathbf{h}^j = 0$. Let G be a matrix with the elements of $\mathcal{G}(A)$ as columns. It is not difficult to show that $\mathbf{v} \in \mathcal{G}(DG)$. Since $\|v\|_1$ is an upper bound on the number of bricks of $\mathbf{h}$ and thus of nonzero bricks of $\mathbf{g}$ and clearly does not depend on n, $g(E) = \max_{v \in \mathcal{G}(DG)} \|v\|_1$ is finite. Let us make precise two observations from this proof.

▶ **Lemma 8** ([20, Lemma 3.1], [35, implicit in proof of Lemma 4.3])**.** *Let* $(\mathbf{g}^1, \ldots, \mathbf{g}^n) \in \mathcal{G}\big(E^{(n)}\big)$. *Then, for all* $i \in [n]$ *there exist vectors* $\mathbf{h}^{i,1}, \ldots, \mathbf{h}^{i,n_i} \in \mathcal{G}(A)$ *such that* $\mathbf{g}^i = \sum_{k=1}^{n_i} \mathbf{h}^{i,k}$, *and* $\sum_{i=1}^{n} n_i \leq g(E)$.

▶ **Lemma 9** ([20, Lemma 6.1], [35, implicit in proof of Lemma 4.3])**.** *Let* $D \in \mathbb{Z}^{r \times t}$, $A \in \mathbb{Z}^{s \times t}$, $G \in \mathbb{Z}^{t \times p}$ *be the matrix whose columns are elements of* $\mathcal{G}(A)$ *and* $p = |\mathcal{G}(A)| \leq \|A\|_\infty^{st}$, *and let* $E = \binom{D}{A}$. *Then* $g(E) \leq \max_{v \in \mathcal{G}(DG)} \|v\|_1 \leq \|A\|_\infty^{st} \cdot (r\|DG\|_\infty)^r$.

Notice that this bound on $g(E)$ is exponential in t. Our goal now is to exploit the fact that the matrix A in a combinatorial n-fold IP is very simple and thus get a better bound.

▶ **Lemma 10.** *Let* $D \in \mathbb{Z}^{r \times t}$, $E = \binom{D}{\mathbf{1}^\intercal}$, *and* $a = \|D\|_\infty$. *Then,* $g(E) \leq t^2(2ra)^r$.

To see this, we will need to understand the structure of $\mathcal{G}(\mathbf{1}^\intercal)$:

▶ **Lemma 11.** *It holds that* $\mathcal{G}(\mathbf{1}^\intercal) = \{\mathbf{g} \mid \mathbf{g}$ *has one* 1 *and one* -1 *and* 0 *otherwise*$\} \subseteq \mathbb{Z}^t$, $|\mathcal{G}(\mathbf{1}^\intercal)| = t(t-1)$, *and for all* $\mathbf{g} \in \mathcal{G}(\mathbf{1}^\intercal)$, $\|\mathbf{g}\|_1 = 2$.

Proof. Observe that the claimed set of vectors is clearly $\sqsubseteq$-minimal in $\ker_{\mathbb{Z}}(\mathbf{1}^\intercal)$. We are left with proving there is no other non-zero $\sqsubseteq$-minimal vector in $\ker_{\mathbb{Z}}(\mathbf{1}^\intercal)$. For contradiction assume there is such a vector $\mathbf{h}$. Since it is non-zero, it must have a positive entry h_i. On the other hand, since $\mathbf{1}^\intercal\mathbf{h} = \mathbf{0}$, it must also have a negative entry h_j. But then $\mathbf{g}$ with $g_i = 1$, $g_j = -1$ and $g_k = 0$ for all $k \notin \{i, j\}$ is $\mathbf{g} \sqsubset \mathbf{h}$, a contradiction. The rest follows. ◀

Proof of Lemma 10. We simply plug into the bound of Lemma 9. By Lemma 11, $p = t(t-1) \leq t^2$. Also, $\|DG\|_\infty \leq \max_{\mathbf{g} \in \mathcal{G}(\mathbf{1}^\intercal)} \{\|D\|_\infty \cdot \|\mathbf{g}\|_1\} \leq 2a$ where the last inequality follows from $\|\mathbf{g}\|_1 = 2$ for all $\mathbf{g} \in \mathcal{G}(\mathbf{1}^\intercal)$, again by Lemma 11. ◀

3.2 Dynamic programming

Hemmecke, Onn and Romanchuk [19] devise a clever dynamic programming algorithm to find augmenting steps for a feasible solution of an n-fold IP. Lemma 8 is key in their approach, as they continue by building a set $Z(E)$ of all sums of at most $g(E)$ elements of $\mathcal{G}(A)$ and then use it to construct the dynamic program. However, such a set $Z(E)$ would clearly be of size exponential in t, which we cannot afford. Our insight here is to build a different dynamic program. In [20], the layers of the dynamic program correspond to partial sums of elements of $\mathcal{G}(A)$; in our dynamic program, the layers will correspond directly to elements of $\mathcal{G}(A)$. This makes it impossible to enforce feasibility with respect to lower and upper bounds in the same way as done in [20]; however, we work around this by exploiting the special structure of $\mathcal{G}(A) = \mathcal{G}(\mathbf{1}^\intercal)$ and simpler lower and upper bounds and enforce them by varying the number of layers of given types. Additionally, we also differ in how we enforce feasibility with respect to the upper rows $(D\ D\ \cdots\ D)$.

Given a brick $i \in [n]$ and $j \in [t]$, let $\mathcal{H}_j^i = \big\{\mathbf{h} \in \mathcal{G}(\mathbf{1}^\intercal) \mid h_j = -1, \mathbf{h} \leq \mathbf{u}^i\big\} \cup \{\mathbf{0} \in \mathbb{Z}^t\}$; here $\mathcal{H}_j^i$ represents the steps which can decrease coordinate x_j^i. Observe that $|\mathcal{H}_j^i| \leq t$. Let $\Sigma(E) = \prod_{j=1}^{r} [-2g(E)a : 2g(E)a]$ be the *signature set of* E whose elements are *signatures*.

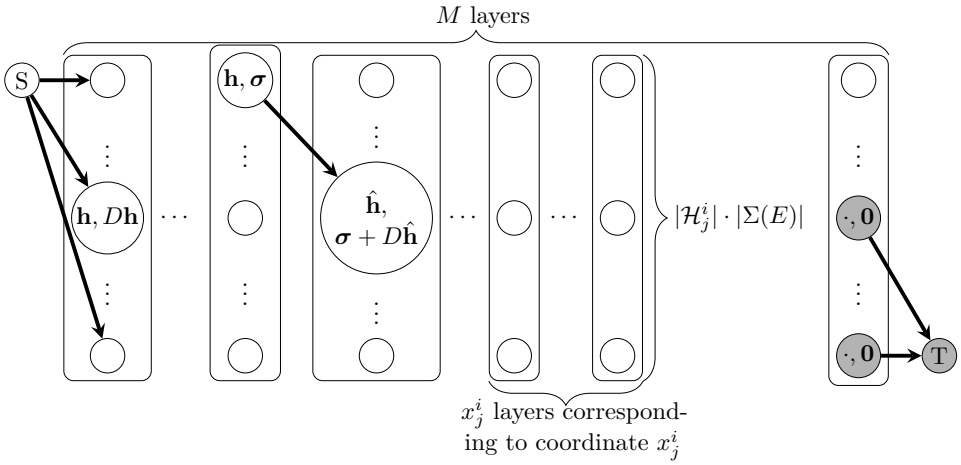

Figure 1 A schema of the augmentation graph $DP(\mathbf{x})$.

Essentially, we will use the signature set to keep track of partial sums of selected elements from $\mathbf{h} \in \mathcal{G}(\mathbf{1}^\intercal)$ to ensure that a resulting vector $\mathbf{g}$ satisfies $D\mathbf{g} = \mathbf{0}$. However, we notice that to ensure $D\mathbf{g} = \mathbf{0}$, it is sufficient to remember the partial sum of elements $D\mathbf{h}$ for $\mathbf{h} \in \mathcal{G}(\mathbf{1}^\intercal)$, thus shrinking them to dimension r. This is another insight which allows us to avoid the exponential dependence on t. Note that $|\Sigma(E)| \leq (1 + 4g(E)a)^r$. Given $\mathbf{x}$ with $\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}$, we define an index function μ: for $i \in [n]$, $j \in [t]$ and $\ell \in [x_j^i]$ let $\mu(i,j,\ell) := \left(\sum_{k=1}^{i-1} \|\mathbf{x}^k\|_1\right) + \left(\sum_{j'=1}^{j-1} x_{j'}^i\right) + \ell$. In the following text, we consider any vector $\mathbf{x}$ satisfying $\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}$ even though it would be natural to consider a feasible solution. This is deliberate, as we will later show that we need these claims to hold also for vectors $\mathbf{x}$ derived from feasible solutions which, however, need not be feasible solutions themselves.

▶ **Definition 12** (Augmentation Graph). Given a vector $\mathbf{x}$ with $\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}$, we define the *augmentation graph* $DP(\mathbf{x})$ to be the following vertex weighted directed layered graph.

There are two distinguished vertices S and T in $DP(\mathbf{x})$, called the *source* and the *sink*. We split the remaining vertices of $DP(\mathbf{x})$ into $M = \|\mathbf{x}\|_1$ layers, denoted $\mathcal{L}(1), \ldots, \mathcal{L}(M)$. With $i \in [n], j \in [t]$ and $\ell \in [\mathbf{x}_j^i]$ we associate the layer $\mathcal{L}(1) = \{(1, \mathbf{h}, D\mathbf{h}) \mid \mathbf{h} \in \mathcal{H}_1^1\}$ if $\mu(i,j,\ell) = 1$ and $\mathcal{L}\big(\mu(i,j,\ell)\big) = \big\{\mu(i,j,\ell)\big\} \times \mathcal{H}_j^i \times \Sigma(E)$ otherwise. Let $L = \max_{\ell=1,\ldots,M} |\mathcal{L}(\ell)|$. A vertex $\big(\mu(i,j,\ell), \mathbf{h}, \boldsymbol{\sigma}\big)$ has weight $f^i(\mathbf{h} + \mathbf{x}^i) - f^i(\mathbf{x}^i)$.

There are the following edges in $DP(\mathbf{x})$. From S to every vertex in the first layer $\mathcal{L}(1)$. Let $u \in \mathcal{L}(\ell)$ and $v \in \mathcal{L}(\ell+1)$ be vertices in consecutive layers with $u = (\ell, \mathbf{h}^\ell, \boldsymbol{\sigma}^\ell)$ and $v = (\ell+1, \mathbf{h}^{\ell+1}, \boldsymbol{\sigma}^{\ell+1})$. If $\boldsymbol{\sigma}^{\ell+1} = \boldsymbol{\sigma}^\ell + D\mathbf{h}^{\ell+1}$, then there is an edge oriented from u to v. Finally, there is an edge from every vertex $u \in \mathcal{L}(M)$ to T if $u = (M, \mathbf{h}, \mathbf{0})$.

Note that by the bounds on $|\mathcal{G}(\mathbf{1}^\intercal)|$ (Lemma 11) and $g(E)$ (Lemma 10), there are at most $L \leq t\left(t^2(2ra)^r\right)^r$ vertices in each layer of $DP(\mathbf{x})$. For an overview of the augmentation graph refer to Fig. 1.

Let P be an S–T path in $DP(\mathbf{x})$ and let $\mathbf{h}^\ell \in \mathcal{G}(\mathbf{1}^\intercal)$ be such that $(\ell, \mathbf{h}^\ell, \boldsymbol{\sigma})$ is its $(\ell+1)$-st vertex. For each $i \in [n]$, let $\mathbf{g}^i = \sum_{j=1}^t \sum_{\ell=1}^{x_j^i} \mathbf{h}^{\mu(i,j,\ell)}$. We say that $\mathbf{h} = (\mathbf{h}^1, \ldots, \mathbf{h}^M)$ is the *P-augmentation vector* and that $\mathbf{g} = (\mathbf{g}^1, \ldots, \mathbf{g}^n)$ is the *compression of* $\mathbf{h}$ (denoted by $\mathbf{g} = \mathbf{h}^\downarrow$). Conversely let $\mathbf{g} \in \mathcal{G}\big(E^{(n)}\big)$ and recall that M is the number of layers of $DP(\mathbf{x})$. By Lemma 8, for all $i \in [n]$ there exist vectors $\mathbf{h}^{i,1}, \ldots, \mathbf{h}^{i,n_i} \in \mathcal{G}(\mathbf{1}^\intercal)$ such that $\mathbf{g}^i = \sum_{k=1}^{n_i} \mathbf{h}^{i,k}$, and $\sum_{i=1}^n n_i \leq g(E)$. For each $i \in [n]$ and $j \in [t]$, let m_j^i be the number of $\mathbf{h}^{i,k}$ with $h_j^{i,k} = -1$. The *expansion* of $\mathbf{g}$ is $\mathbf{h} = (\mathbf{h}^1, \ldots, \mathbf{h}^M)$ defined as follows (we denote this as

$\mathbf{h} = \mathbf{g}^{\uparrow}$). Fix $i \in [n]$ and $j \in [t]$. Assign the distinct m^i_j vectors $\mathbf{h}^{i,k}$ with $h^{i,k}_j = -1$ to $\mathbf{h}^{\mu(i,j,\ell)}$ for $\ell \in [m^i_j]$, and let $\mathbf{h}^{\mu(i,j,\ell)} = \mathbf{0}$ for $\ell \in [m^i_j + 1 : x^i_j]$. Essentially, we pad the vector $\mathbf{h}$ obtained by Lemma 8 with $\mathbf{0}$ bricks to construct an $\mathbf{h} = \mathbf{g}^{\uparrow}$. Also notice that an S–T path P such that $\mathbf{h}$ is a P-augmentation vector can be constructed by choosing appropriate $\boldsymbol{\sigma} \in \Sigma(E)$ for each brick of $\mathbf{h}$.

Let $\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}$. We say that $\mathbf{g}$ is a *solution of* $DP(\mathbf{x})$ if $\mathbf{0} \leq \mathbf{x} + \mathbf{g} \leq \mathbf{u}$ and there exists an S–T path P with P-augmentation vector $\mathbf{h}$ and $\mathbf{g} = \mathbf{h}^{\downarrow}$; the weight $w(\mathbf{g})$ is then defined as the weight of the path P; note that $w(\mathbf{g}) = f(\mathbf{x} + \mathbf{g}) - f(\mathbf{x})$. A solution $\mathbf{g}$ is called a *minimal solution of* $DP(\mathbf{x})$ if it is a solution of minimal weight. The following lemma relates solutions of $DP(\mathbf{x})$ to potential feasible steps in $\mathcal{G}\big(E^{(n)}\big)$.

▶ **Lemma 13.** *Let $\mathbf{x} \in \mathbb{Z}^{nt}$ satisfy $\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}$ and let $\mathbf{g}$ be a solution of $DP(\mathbf{x})$. It holds that $\mathbf{0} \leq \mathbf{x} + \mathbf{g} \leq \mathbf{u}$ and $E^{(n)}\mathbf{g} = \mathbf{0}$.*

Proof. It follows from the definition that there are exactly x^i_j layers in which it is possible to select a vector $\mathbf{h}$ such that $h^i_j = -1$. Observe further that all other layers that are derived from the i-th brick can only increase the value of g^i_j. It follows that $\mathbf{x} + \mathbf{g} \geq \mathbf{0}$.

Recall that $u^i_j \in \{0, \|\mathbf{b}\|_\infty\}$. If $u^i_j = 0$, then we have excluded all vectors $\mathbf{h}$ with $h^i_j = 1$ from $\mathcal{H}^i_k$ for all $k \in [t]$. Thus $x^i_j + g^i_j = x^i_j = 0 \leq 0$ as claimed. On the other hand, if $u^i_j = \|\mathbf{b}\|_\infty$, then observe that $\sum_{k=1}^{t} g^i_k = 0$ and because $\mathbf{x} + \mathbf{g} \geq \mathbf{0}$, we conclude that $x^i_j + g^i_j \leq b^i \leq \|\mathbf{b}\|_\infty = u^i_j$.

Let $(\mathbf{h}_k, \boldsymbol{\sigma}_k)$, for each $k \in [M]$, be the vertex from the k-th layer of path P corresponding to $\mathbf{g}^{\uparrow}$. Note that $\boldsymbol{\sigma}_M = \mathbf{0}$. If follows that $\boldsymbol{\sigma}_{\ell+1} = \left(\sum_{k=1}^{\ell} D\mathbf{h}_k\right) + D\mathbf{h}_{\ell+1}$ for all $1 \leq \ell \leq M-1$. Thus $\sum_{k=1}^{M} D\mathbf{h}_k = \mathbf{0}$ and because we have $A\mathbf{h}_k = \mathbf{0}$ from the definition of $\mathcal{H}^i_j$ we conclude that $E^{(n)}\mathbf{g} = \mathbf{0}$. ◀

▶ **Lemma 14.** (⋆) *Let $\mathbf{x} \in \mathbb{Z}^{nt}$ satisfy $\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}$. Every $\tilde{\mathbf{g}} \in \mathcal{G}\big(E^{(n)}\big)$ with $\mathbf{0} \leq \mathbf{x} + \tilde{\mathbf{g}} \leq \mathbf{u}$ is a solution of $DP(\mathbf{x})$.*

We define the $g(E)$-*truncation* of $\mathbf{x}$ as the vector $\overline{\mathbf{x}}$ given by $\overline{x}^i_j = \min\{x^i_j, g(E)\}$.

▶ **Lemma 15.** (⋆) *Let $\mathbf{x} \in \mathbb{Z}^{nt}$ satisfy $\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}$. Every $\tilde{\mathbf{g}} \in \mathcal{G}\big(E^{(n)}\big)$ with $\mathbf{0} \leq \mathbf{x} + \tilde{\mathbf{g}} \leq \mathbf{u}$ is a solution of $DP(\overline{\mathbf{x}})$.*

Clearly our goal is then to find the lightest S–T path in the graph $DP(\overline{\mathbf{x}})$. However, there will be edges with negative weights. Still, finding the lightest path can be done in a layer by layer manner (see e.g. [19, Lemma 3.4]) in time $O(|V(DP(\overline{\mathbf{x}}))| \cdot L) = O(\|\overline{\mathbf{x}}\|_1 \cdot L^2)$. The following lemma is then an immediate consequence of Lemmas 14 and 15.

▶ **Lemma 16** (Optimality certification). *Given $\mathbf{x} \in \mathbb{Z}^{nt}$ with $\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}$, it is possible to find a vector $\mathbf{g}$ such that $E^{(n)}\mathbf{g} = \mathbf{0}$, $\mathbf{0} \leq \mathbf{x} + \mathbf{g} \leq \mathbf{u}$, and $f(\mathbf{x} + \mathbf{g}) < f(\mathbf{x})$, or decide there is none such $\mathbf{g}$, in time $\|\overline{\mathbf{x}}\|_1 \cdot L^2 \leq t^{O(r)}(ar)^{O(r^2)}n$.*

Proof. It follows from Lemma 13 that all solutions of $DP(\mathbf{x})$ fulfill the first two conditions. Observe that if we take $\mathbf{g}$ to be a minimal solution of $DP(\overline{\mathbf{x}})$, then either $f(\mathbf{x}) = f(\mathbf{x} + \mathbf{g})$ or $f(\mathbf{x}) < f(\mathbf{x} + \mathbf{g})$. Due to Lemma 15 the set of solutions of $DP(\overline{\mathbf{x}})$ contains all $\tilde{\mathbf{g}} \in \mathcal{G}\big(E^{(n)}\big)$ with $\mathbf{0} \leq \tilde{\mathbf{g}} \leq \mathbf{u}$. Thus, by Proposition 7, if $f(\mathbf{x}) = f(\mathbf{x} + \mathbf{g})$, no $\mathbf{g}$ satisfying all three conditions exist.

Now simply plug in our bounds on $\|\overline{\mathbf{x}}\|_1$ and L and compute a minimal S–T path:

$$L \leq |\mathcal{G}(\mathbf{1}^{\intercal})| \cdot |\Sigma(E)| \leq t^2 \cdot \left(1 + 4t^2 a(2ra)^r\right)^r \leq t^{O(r)}(ar)^{O(r^2)}$$

is the maximum size of a layer and $\|\overline{\mathbf{x}}\|_1 \leq nt \cdot g(E) \leq nt \cdot t^2(2ra)^r \leq O(t^2)(ar)^{O(r)}n$ is the number of layers. ◀

3.3 Long steps

So far, we are able to *find* an augmenting step in time independent of M; however, each step might only bring an improvement of $O(1)$ and thus possibly many improving steps would be needed. Now, given a step length $\alpha \in \mathbb{N}$, we will show how to find a feasible step $\mathbf{g}$ such that $f(\mathbf{x}+\alpha\mathbf{g}) \leq f(\mathbf{x}+\alpha\tilde{\mathbf{g}})$ for any $\tilde{\mathbf{g}} \in \mathcal{G}\big(E^{(n)}\big)$. Moreover, we will show that there are not too many step lengths that need to be considered in order to find a Graver-best step which, by Proposition 7, leads to a good bound on the required number of steps.

Let $\alpha \in \mathbb{N}$ and let $\mathbf{x}$ with $\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}$. We define $\mathbf{x}_\alpha$ to be the *α-reduction* of $\mathbf{x}$, $\mathbf{x}_\alpha = \left\lfloor \frac{\mathbf{x}}{\alpha} \right\rfloor$. This operation takes priority over the truncation operation, that is, by $\overline{\mathbf{x}_\alpha}$ we mean the $g(E)$-truncation of vector $\mathbf{x}_\alpha$ (i.e., $\overline{\mathbf{x}_\alpha} = \overline{(\mathbf{x}_\alpha)}$). Note that for large enough α, $DP(\mathbf{x}_\alpha)$ contains only two vertices S and T and no arcs and thus there is no S–T path and no solutions.

▶ **Lemma 17.** (⋆) *Let $\alpha \in \mathbb{N}$ and let $\mathbf{x}$ with $\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}$. Every $\tilde{\mathbf{g}} \in \mathcal{G}\big(E^{(n)}\big)$ with $\mathbf{0} \leq \mathbf{x}+\alpha\tilde{\mathbf{g}} \leq \mathbf{u}$ is a solution of $DP(\overline{\mathbf{x}_\alpha})$.*

However, a Graver-best step might still be such that its step length α is large and thus we cannot afford to find a minimal solution of $DP(\mathbf{x}_\alpha)$ for all possible step lengths. We need another observation to see that many step lengths need not be considered. Let the *state of* $\mathbf{x}_\alpha$, $\psi(\mathbf{x}_\alpha) \in \{0,1,2\}^{[n]\times[t]}$, be defined by:

- $\psi(\mathbf{x}_\alpha)^i_j = 0$ if $(\mathbf{x}_\alpha)^i_j = 0$,
- $\psi(\mathbf{x}_\alpha)^i_j = 1$ if $1 \leq (\mathbf{x}_\alpha)^i_j < g(E)$, and,
- $\psi(\mathbf{x}_\alpha)^i_j = 2$ if $(\mathbf{x}_\alpha)^i_j \geq g(E)$.

Given a feasible solution $\mathbf{x}$, we call a step length α *interesting* if $\overline{\mathbf{x}_\alpha} \neq \overline{\mathbf{x}_{\alpha+1}}$ and *boring* otherwise. Moreover, α is *irrelevant* if there is no Graver-best step with step length α.

▶ **Lemma 18.** (⋆) *If α is boring, then it is irrelevant.*

▶ **Definition 19** (Candidate step lengths Γ)**.** Let Γ be a set of candidate step lengths constructed iteratively as follows:

Input: vector $\mathbf{x}$ with $\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}$ and $g(E)$
Computes: set of candidate steps Γ
$\Gamma \leftarrow \{1\}$ and $\gamma \leftarrow 2$
while $\mathbf{x}_\gamma > \mathbf{0}$ **do**
  foreach i,j *with* $\psi(\mathbf{x}_\gamma)^i_j = 1$ **do**
    $\Gamma^i_j \leftarrow \left\{\left(k, \lfloor x^i_j/k \rfloor\right) \mid k \in \mathbb{N}, 0 < \lfloor x^i_j/k \rfloor < (\mathbf{x}_{\gamma-1})^i_j\right\}$
    $\gamma^i_j \leftarrow k$ such that $(k,q) \in \Gamma^i_j$, q is maximal, and secondary to this k also maximal
  $\tilde{\gamma}_1 \leftarrow \min\left\{\gamma^i_j \mid \psi(\mathbf{x}_\gamma)^i_j = 1\right\}$
  $\tilde{\gamma}_2 \leftarrow \min\left\{\left\lfloor \frac{x^i_j}{g(E)} \right\rfloor \mid \psi(\mathbf{x}_\gamma)^i_j = 2\right\}$
  add $\min\{\tilde{\gamma}_1, \tilde{\gamma}_2\}$ to Γ
  $\gamma \leftarrow \max\Gamma + 1$
return Γ

▶ **Lemma 20.** *If α is the step length of a Graver-best step, then $\alpha \in \Gamma$.*

Proof. We will prove that Γ contains all interesting step lengths. Consider an $\alpha \notin \Gamma$. Either $\mathbf{x}_\alpha = \mathbf{0}$ and clearly in that case $DP(\mathbf{x}_\alpha)$ does not yield an augmenting step since it has no layers and thus no weighted vertices, and thus α is irrelevant.

Otherwise, take $\gamma := \min\{\gamma' \mid \gamma' \in \Gamma, \gamma' \geq \alpha\}$. Because of the minimality of γ with respect to all of the $\min\{\cdot\}$ clauses of the algorithm of Definition 19, we have that $\overline{\mathbf{x}_\alpha} = \overline{\mathbf{x}_\gamma}$ and thus α is boring and by Lemma 18 irrelevant.

Since Γ contains all remaining step lengths, it also contains all interesting steps and must contain the step length for any Graver-best step. ◀

▶ **Lemma 21.** *$|\Gamma| \leq O(nt \cdot g(E))$ and Γ can be constructed in time $O(|\Gamma| \cdot \log \|\mathbf{x}\|_\infty)$.*

Proof. Fix a coordinate x^i_j of $\mathbf{x}$ and consider a run of the algorithm of Definition 19. If $x^i_j > g(E)$, $\tilde{\gamma}_2 := \left\lfloor \frac{x^i_j}{g(E)} \right\rfloor$ is added to Γ at some point. For every $\gamma > \tilde{\gamma}_2$ we have that $(\mathbf{x}_\gamma)^i_j < g(E)$ and thus we need not consider the $\min\{\cdot\}$ clause for $\psi(\mathbf{x}_\gamma)^i_j = 2$.

Consider a step of the algorithm which adds $\tilde{\gamma}_1$, and observe that $\tilde{\gamma}_1$ is chosen such that $(\mathbf{x}_{\tilde{\gamma}_1})^i_j > (\mathbf{x}_{\tilde{\gamma}_1+1})^i_j$. But since $(\mathbf{x}_{\tilde{\gamma}_2+1})^i_j < g(E)$, such situation can occur at most $g(E)$ times.

Thus we have added at most $O(g(E))$ different step lengths to Γ per coordinate, $O(nt \cdot g(E))$ step lengths in total.

Regarding the time it takes to construct Γ, we perform $O(|\Gamma|)$ arithmetic operations, and since we are dealing with numbers of size at most $\|\mathbf{x}\|_\infty$, each operation takes time $O(\log \|\mathbf{x}\|_\infty)$, concluding the proof. ◀

▶ **Lemma 22** (Graver-best computation)**.** *Given a feasible solution $\mathbf{x}$ of a combinatorial n-fold IP, in time $t^{O(r)}(ar)^{O(r^2)}n^2$ one can either find a Graver-best step $(\alpha, \mathbf{g})$ or decide that none exists.*

Proof. For $\gamma \in \Gamma$ let $\mathbf{g}^\gamma$ be a minimal solution of $DP(\overline{\mathbf{x}_\gamma})$ and let $\alpha := \arg\min_{\gamma \in \Gamma} \left(f(\mathbf{x} + \gamma \mathbf{g}^\gamma)\right)$. Finally, let $\mathbf{g} := \mathbf{g}^\alpha$. Then we claim that $(\alpha, \mathbf{g})$ is a Graver-best step.

By Lemma 17 for all $\tilde{\mathbf{g}} \in \mathcal{G}\left(E^{(n)}\right)$ it holds that $f(\mathbf{x} + \alpha \mathbf{g}) \leq f(\mathbf{x} + \alpha \tilde{\mathbf{g}})$. Moreover, by Lemma 20, if there exists a Graver-best step with step length γ, then $\gamma \in \Gamma$, and thus by the construction of α, $(\alpha, \mathbf{g})$ is Graver-best step.

Regarding the time complexity, to obtain $\mathbf{g}$ we need to solve $DP(\overline{\mathbf{x}_\gamma})$ for each $\gamma \in \Gamma$ by Lemma 16, requiring time $|\Gamma| \cdot t^{O(r)}(ar)^{O(r^2)}n \leq t^{O(r)}(ar)^{O(r^2)}n^2$. ◀

3.4 Finishing the proof

Proof of Theorem 2. In order to prove Theorem 2 we need to put the pieces together. First, let us assume that we have an initial feasible solution $\mathbf{x}_0$. In order to reach the optimum, by Proposition 7 we need to make at most $(2nt - 2) \cdot O(L)$ Graver-best steps, where $L = \langle \mathbf{b}, \mathbf{0}, \mathbf{u}, \mathbf{w} \rangle$; this is because $O(L)$ is an upper bound on $f(\mathbf{x}^0) - f(\mathbf{x}^*)$ for some minimum $\mathbf{x}^*$. By Lemma 22, it takes time $t^{O(r)}(ar)^{O(r^2)}n^2$ to find a Graver-best step.

Now we are left with the task of finding a feasible solution. We follow along the lines of [19, Lemma 3.8] and solve an auxiliary combinatorial n-fold IP given by the bimatrix $\hat{E} = \begin{pmatrix} \hat{D} \\ \hat{A} \end{pmatrix}$ with $\hat{D} := (D\ I_r\ -I_r\ \mathbf{0})$ and $\hat{A} := (A\ \mathbf{1}_{2r+1}) = \mathbf{1}^\intercal \in \mathbb{Z}^{t+2r+1}$, where I_r is the identity matrix of dimension r, $\mathbf{0}$ is a column vector of length r and $\mathbf{1}_{2r+1}$ is the vector of all 1s of length $2r + 1$.

The variables $\hat{\mathbf{x}}$ of this problem have a natural partition into nt variables $\mathbf{x}$ corresponding to the original problem and $n(2r + 1)$ new auxiliary variables $\tilde{\mathbf{x}}$. Keep the original lower and upper bounds on $\mathbf{x}$ and introduce a lower bound 0 and upper bound $\|\mathbf{b}\|_\infty$ on each auxiliary variable. Finally, let the new linear objective $\hat{\mathbf{w}}^\intercal \hat{\mathbf{x}}$ be the sum of the auxiliary

variables. Observe that it is easy to construct an initial feasible solution by setting $\mathbf{x} = \mathbf{0}$ and computing $\tilde{\mathbf{x}}$ accordingly, as $\tilde{\mathbf{x}}$ serve the role of slack variables.

Then, applying the algorithm described previously either finds a solution with objective value 0, implying $\tilde{\mathbf{x}} = \mathbf{0}$, and thus $\mathbf{x}$ is feasible for the original problem, or no such solution exists, meaning that the original problem is infeasible. ◀

4 An Application to Weighted Set Multicover

In applications, it is practical to use combinatorial n-fold IP formulations which contain inequalities. Given an n-fold IP (in particular a combinatorial n-fold IP), we call the upper rows $(D\ D \cdots\ D)\mathbf{x} = \mathbf{b}^0$ *globally uniform constraints*, and the lower rows $A\mathbf{x}^i = \mathbf{b}^i$, for all $i \in [n]$, *locally uniform constraints*. In the full version [26] we show that introducing inequalities into a combinatorial n-fold IP is possible, however we need a slightly different approach than in a standard n-fold IP to keep the rigid format of a combinatorial n-fold IP.

Weighted Set Multicover. We demonstrate Theorem 2 on the following problem:

Weighted Set Multicover

Input: A universe of size k, $U = [k]$, a set system represented by a multiset $\mathcal{F} = \{F_1, \ldots, F_n\} \subseteq 2^U$, weights $w_1, \ldots, w_n \in \mathbb{N}$, demands $d_1, \ldots, d_k \in \mathbb{N}$.

Find: A multisubset $\mathcal{F}' \subseteq \mathcal{F}$ minimizing $\sum_{F_i \in \mathcal{F}'} w_i$ and satisfying $\left|\{i \mid F_i \in \mathcal{F}', j \in F_i\}\right| \geq d_j$ for all $j \in [k]$.

Proof of Theorem 5. Observe that since W is the number of different weights, and there can be at most 2^k different sets $F \in 2^U$, each pair (F, w) on the input is of one of $T \leq W2^k$ different types; let $n_1, \ldots, n_T \in \mathbb{N}$ be a succinct representation of the instance.

We shall construct a combinatorial n-fold IP to solve the problem. Let $x^\tau_{\mathbf{f}}$ for each $\mathbf{f} \in 2^U$ and each $\tau \in [T]$ be a variable. Let $u^\tau_{\mathbf{f}} = 0$ for each $\mathbf{f} \in 2^U$ such that $\mathbf{f} \neq F^\tau$, and let $u^\tau_{\mathbf{f}} = \max n_\tau$ for $\mathbf{f} = F^\tau$. The variable $x^\tau_{\mathbf{f}}$ with $\mathbf{f} = F^\tau$ represents the number of sets of type τ in the solution. The formulation is straightforward and reads

$$\begin{aligned}
\min \quad & \sum_{\tau=1}^{T} \sum_{\mathbf{f} \in 2^U} w^\tau x^\tau_{\mathbf{f}} \\
\text{s.t.} \quad & \sum_{\tau=1}^{T} \sum_{\mathbf{f} \in 2^U} f_i x^\tau_{\mathbf{f}} \geq d_i, \qquad \text{for all } i \in [k] \\
& \sum_{\mathbf{f} \in 2^U} x^\tau_{\mathbf{f}} \leq n_\tau \qquad \text{for all } \tau \in [T];
\end{aligned}$$

note that f_i is 1 if $i \in \mathbf{f}$ and 0 otherwise. Let us determine the parameters $\hat{a}, \hat{r}, \hat{t}, \hat{n}$ and $\hat{L}$ of this combinatorial n-fold IP instance. Clearly, the largest coefficient $\hat{a}$ is 1, the number of globally uniform constraints $\hat{r}$ is k, the number of variables per brick $\hat{t}$ is 2^k, the number of bricks $\hat{n}$ is T, and the length of the input $\hat{L}$ is at most $\log n + \log w_{\max}$. ◀

5 Open problems

Can our result be extended to minimizing a separable convex function f? Is Huge n-fold IP fixed-parameter tractable for parameters r, s, t and a? It is not difficult to see that optimality certification is fixed-parameter tractable using ideas similar to Onn [36]; however, one possibly needs exponentially (in the input size) many augmenting steps.

For most of our applications, complexity lower bounds are not known to us. Our algorithms yield complexity upper bounds of $k^{O(k^2)}$ on the dependence on parameter k for various problems, such as Closest String, Weighted Set Multicover, Score-Swap Bribery or even Makespan Minimization [25]. Is this just a common feature of our algorithm, or are there hidden connections between some of these problems? And what are their actual complexities? All we know so far is a trivial ETH-based $2^{o(k)}$ lower bound for Closest String based on its reduction from Satisfiability [12].

References

1. Amihood Amir, Gad M. Landau, Joong Chae Na, Heejin Park, Kunsoo Park, and Jeong Seop Sim. Efficient algorithms for consensus string problems minimizing both distance sum and radius. *Theoret. Comput. Sci.*, 412(39):5239–5246, 2011.
2. Liliana Félix Avila, Alina Garcıa, Marıa José Serna, and Dimitrios M Thilikos. A list of parameterized problems in bioinformatics. Technical report, Technical University of Catalonia, 2006. Technical report LSI-06-24-R.
3. Christina Boucher and Kathleen Wilkie. Why large closest string instances are easy to solve in practice. In *Proc. SPIRE 2010*, volume 6393 of *Lecture Notes Comput. Sci.*, pages 106–117, 2010.
4. Robert Bredereck, Jiehua Chen, Piotr Faliszewski, Jiong Guo, Rolf Niedermeier, and Gerhard J. Woeginger. Parameterized algorithmics for computational social choice: Nine research challenges. *Tsinghua Sci. Tech.*, 19(4):358–373, 2014.
5. Robert Bredereck, Piotr Faliszewski, Rolf Niedermeier, Piotr Skowron, and Nimrod Talmon. Elections with few candidates: Prices, weights, and covering problems. In *Proc. ADT 2015*, volume 9346 of *Lecture Notes Comput. Sci.*, pages 414–431, 2015.
6. Laurent Bulteau, Falk Hüffner, Christian Komusiewicz, and Rolf Niedermeier. Multivariate algorithmics for NP-hard string problems. *Bulletin of the EATCS*, 114, 2014.
7. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, 2015.
8. Daniel Dadush, Chris Peikert, and Santosh Vempala. Enumerative lattice algorithms in any norm via M-ellipsoid coverings. In *Proc. FOCS 2011*, pages 580–589. 2011.
9. Britta Dorn and Ildikó Schlotter. Multivariate complexity analysis of swap bribery. *Algorithmica*, 64(1):126–151, 2012.
10. Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Frances A. Rosamond, and Saket Saurabh. Graph layout problems parameterized by vertex cover. In *Proc. ISAAC 2008*, volume 5369 of *Lecture Notes Comput. Sci.*, pages 294–305. Springer, Berlin, 2008.
11. Jiří Fiala, Tomáš Gavenčiak, Dušan Knop, Martin Koutecký, and Jan Kratochvíl. Parameterized complexity of distance labeling and uniform channel assignment problems. *Discrete Appl. Math.*, 2017. to appear.
12. M. Frances and A. Litman. On covering problems of codes. *Theory Comput. Syst.*, 30(2):113–119, 1997.
13. Jakub Gajarský, Michael Lampis, and Sebastian Ordyniak. Parameterized algorithms for modular-width. In *Proc. IPEC 2013*, volume 8246 of *Lecture Notes Comput. Sci.*, pages 163–176. 2013.
14. Robert Ganian and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP. In *Proc. AAAI 2016*, pages 710–716, 2016.
15. Robert Ganian, Sebastian Ordyniak, and M. S. Ramanujan. Going beyond primal treewidth for (M)ILP. In *Proc. AAAI 2017*, pages 815–821, 2017.
16. Michel X. Goemans and Thomas Rothvoß. Polynomiality for bin packing with a constant number of item types. In *Proc. SODA 2014*, pages 830–839. ACM, New York, 2014.

17 Jens Gramm, Rolf Niedermeier, and Peter Rossmanith. Fixed-parameter algorithms for closest string and related problems. *Algorithmica*, 37(1):25–42, 2003.

18 Raymond Hemmecke, Matthias Köppe, and Robert Weismantel. Graver basis and proximity techniques for block-structured separable convex integer minimization problems. *Math. Program.*, 145(1-2, Ser. A):1–18, 2014.

19 Raymond Hemmecke, Shmuel Onn, and Lyubov Romanchuk. n-fold integer programming in cubic time. *Math. Program.*, 137(1-2, Ser. A):325–341, 2013.

20 Raymond Hemmecke, Shmuel Onn, and Robert Weismantel. A polynomial oracle-time algorithm for convex integer minimization. *Math. Program.*, 126(1, Ser. A):97–117, 2011.

21 Danny Hermelin and Liat Rozenberg. Parameterized complexity analysis for the closest string with wildcards problem. *Theoret. Comput. Sci.*, 600:11–18, 2015.

22 Bart M. P. Jansen and Stefan Kratsch. A structural approach to kernels for ILPs: Treewidth and total unimodularity. In *Proc. ESA 2015*, volume 9294 of *Lecture Notes Comput. Sci.*, pages 779–791, 2015.

23 Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.

24 Leonid Khachiyan and Lorant Porkolab. Integer optimization on convex semialgebraic sets. *Discrete Comput. Geom.*, 23(2):207–224, 2000.

25 Dušan Knop and Martin Koutecký. Scheduling meets n-fold integer programming. *Journal of Scheduling*, page to appear, 2017.

26 Dušan Knop, Martin Koutecký, and Matthias Mnich. Combinatorial n-fold integer programming and applications. Technical report, 2017. URL: `https://arxiv.org/abs/1705.08657`.

27 Dušan Knop, Martin Koutecký, and Matthias Mnich. Voting and bribing in single-exponential time. In *Proc. STACS 2017*, volume 66 of *Leibniz Int. Proc. Informatics*, pages 46:1–46:14, 2017.

28 Stefan Kratsch. On polynomial kernels for sparse integer linear programs. *J. Comput. System Sci.*, 82(5):758–766, 2016.

29 Michael Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012.

30 Hendrik W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

31 Jesus A. De Loera, Raymond Hemmecke, and Matthias Köppe. *Algebraic and Geometric Ideas in the Theory of Discrete Optimization*, volume 14 of *MOS-SIAM Series on Optimization*. SIAM, 2013.

32 Daniel Lokshtanov. Parameterized integer quadratic programming: Variables and coefficients. Technical report, 2015. `http://arxiv.org/abs/1511.00310`.

33 Matthias Mnich and Andreas Wiese. Scheduling and fixed-parameter tractability. *Math. Program.*, 154(1-2, Ser. B):533–562, 2015.

34 Rolf Niedermeier. Ubiquitous parameterization—invitation to fixed-parameter algorithms. In *Proc. MFCS 2004*, volume 3153 of *Lecture Notes in Comput. Sci.*, pages 84–103. Springer, Berlin, 2004.

35 Shmuel Onn. Nonlinear discrete optimization. *Zurich Lectures in Advanced Mathematics, European Mathematical Society*, 2010.

36 Shmuel Onn. Huge multiway table problems. *Discrete Optim.*, 14:72–77, 2014.