Integer Programming Notes / Winter 22/23

MARTIN KOUTECKÝ, Charles University, Czech Republic

1 INTEGER PROGRAMMING IN GENERAL

Our focus is on the integer (linear) programming problem in standard form

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

$$\min\{\mathbf{w}\mathbf{x} \mid A\mathbf{x} = \mathbf{b},\ \mathbf{l} \le \mathbf{x} \le \mathbf{u},\ \mathbf{x} \in \mathbb{Z}^n\}, \tag{ILP}$$

with A an integer $m \times n$ matrix, $f : \mathbb{R}^n \to \mathbb{R}$ a separable convex function, $\mathbf{b} \in \mathbb{Z}^m$, and $\mathbf{l}, \mathbf{u} \in (\mathbb{Z} \cup \{\pm\infty\})^n$. (IP) is well-known to be strongly NP-hard already in the special case (ILP) when $f(\mathbf{x}) = \mathbf{w}\mathbf{x}$ is a linear objective function for some vector $\mathbf{w} \in \mathbb{Z}^n$. (E.g. it is easy to encode Vertex Cover as ILP.) In this course, we will cover some important, broad, natural, and useful conditions under which (IP) can be solved in polynomial time.

Notation

We write vectors in boldface (e.g., $\mathbf{x}, \mathbf{y}$) and their entries in normal font (e.g., the i-th entry of $\mathbf{x}$ is x_i). For positive integers $m \le n$ we set $[m, n] := \{m, \dots, n\}$ and $[n] := [1, n]$, and we extend this notation for vectors: for $\mathbf{l}, \mathbf{u} \in \mathbb{Z}^n$ with $\mathbf{l} \le \mathbf{u}$, $[\mathbf{l}, \mathbf{u}] := \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{l} \le \mathbf{x} \le \mathbf{u}\}$. If A is a matrix, $A_{i,j}$ denotes the j-th coordinate of the i-th row, $A_{i,\bullet}$ denotes the i-th row and $A_{\bullet,j}$ denotes the j-th column. We use $\log := \log_2$. For an integer $a \in \mathbb{Z}$, we denote by $\langle a \rangle := 1 + \lceil \log(|a| + 1) \rceil$ the binary encoding length of a; we extend this notation to vectors, matrices and tuples of these objects. For example, $\langle A, \mathbf{b} \rangle = \langle A \rangle + \langle \mathbf{b} \rangle$, and $\langle A \rangle = \sum_{i,j} \langle A_{i,j} \rangle$.

2 FIXED DIMENSION

(IP) can be solved in time $g(n)poly(n, L)$ for some function g, and this goes back to the work of Lenstra [17]. The best current bound is $g(n) = O(n)^n$ and is due to Dadush [3]. The algorithm even applies to the case where f is general convex (non-separable), and where $\mathbf{x}$ belongs to some convex body $K \subseteq \mathbb{R}^n$.

We will sketch the main ideas of the proof, but don't pretend to give all details. Define the *width of K along a direction* $\mathbf{d} \in \mathbb{Z}^n$ to be

$$w_{\mathbf{d}}(K) = \max\{\mathbf{d}\mathbf{x} \mid \mathbf{x} \in K\} - \min\{\mathbf{d}\mathbf{x} \mid \mathbf{x} \in K\} \ .$$

If the max or min does not exist, we define the width to be infinity. The *width of K* is defined as the smallest width over all non-zero directions; notice that we are taking the directions over all integer vectors in order to avoid silly issues like being able to get very small width by taking very small (non-integral) $\mathbf{d}$:

$$w(K) = \min_{\mathbf{d} \in \mathbb{Z}^d \setminus \{\mathbf{0}\}} w_{\mathbf{d}}(K) \ .$$

A $\mathbf{d}$ which attains the minimum above is called a *flat direction of K*. The algorithm relies on the following deep and famous result:

Proposition 1 (Khinchine's Flatness Theorem). *Let $K \subseteq \mathbb{R}^n$ be a convex body. Then either K contains a lattice point (i.e., $K \cap \mathbb{Z}^n \neq \emptyset$), or $w(K) \le \omega(n)$ where $\omega(n)$ is some constant only depending on n.*

Author's address: Martin Koutecký, Charles University, Prague, Czech Republic, koutecky@iuuk.mff.cuni.cz.

(It could be that K is flat *and* contains an integer point, and this doesn't bother us.)

We focus on solving feasibility, that is, deciding $K \cap \mathbb{Z}^n \neq \emptyset$; optimization can be handled by doing a binary search over the objective and then adding this objective bound into the set of constraints. Specifically, if our guess on the objective is T, we want to enforce a constraint $f(\mathbf{x}) \leq T$, and because f is convex, this is a convex constraint and thus $K' = K \cap \{\mathbf{x} \mid f(\mathbf{x}) \leq T\}$ is a convex set and we solve feasibility for K' instead of optimization over K.

The main idea of the algorithm is this. We compute a flat direction $\mathbf{d}$ of K (which is not an easy problem but is known to be doable so we treat it here as an oracle call). If we see that $w(K) > \omega(n)$, we know that K contains an integer point and we are done. Otherwise, $w(K) \leq \omega(n)$. This means we can branch into at most $\omega(n)$ lower-dimensional slices of K and solve the problem inductively in each of them. Because the dimension drops by at least one in each branching, the branching tree has at most n levels, and because we branch into at most $\omega(n)$ slices, the degree of the tree is at most $\omega(n)$, so altogether the tree has at most $\omega(n)^n$ nodes.

What does this branching look like in detail? If K contains an integer point, then it must lie on one of the hyperplanes

$$\mathbf{dx} = \delta, \text{ where } \delta \in [\min\{\mathbf{dx} \mid \mathbf{x} \in K\}, \max\{\mathbf{dx} \mid \mathbf{x} \in K\}] \ .$$

The rest of the work is that we need to transform the set $K \cap \{\mathbf{x} \mid \mathbf{dx} = \delta\}$ which is less than n-dimensional but lives in n dimensions into a set $K' \subseteq \mathbb{R}^{n-1}$ which is integer feasible iff K is, and then call the algorithm on K'.

3 VARIABLE DIMENSION

Some more preliminaries are in order now.

For a function $f : \mathbb{Z}^n \to \mathbb{Z}$ and two vectors $\mathbf{l}, \mathbf{u} \in \mathbb{Z}^n$, we define $f_{\max}^{[\mathbf{l},\mathbf{u}]} := \max_{\mathbf{x},\mathbf{x}' \in [\mathbf{l},\mathbf{u}]} |f(\mathbf{x}) - f(\mathbf{x}')|$; if $[\mathbf{l}, \mathbf{u}]$ is clear from the context we omit it and write just $f_{\max}$. We assume that $f : \mathbb{R}^n \to \mathbb{R}$ is a separable convex function, i.e., it can be written as $f(\mathbf{x}) = \sum_{i=1}^n f_i(x_i)$ where f_i is a convex function of one variable, for each $i \in [n]$. Moreover, we require that for each $\mathbf{x} \in \mathbb{Z}^n$, $f(\mathbf{x}) \in \mathbb{Z}$. We assume f is given by a comparison oracle. We use ω to denote the smallest number such that matrix multiplication of $n \times n$ matrices can be performed in time $O(n^\omega)$. We say that a system of equations $A\mathbf{x} = \mathbf{b}$ is *pure* if the rows of A are linearly independent. The next statement follows easily by Gaussian elimination, hence we assume $m \leq n$ throughout the paper.

Proposition 2 (Purification [11, Theorem 1.4.8]). *Given $A \in \mathbb{Z}^{m \times n}$ and $\mathbf{b} \in \mathbb{Z}^m$ one can in time $O(\min\{n, m\}nm)$ either declare $A\mathbf{x} = \mathbf{b}$ infeasible, or output a pure equivalent subsystem $A'\mathbf{x} = \mathbf{b}'$.*

The goal of this section is to prove the following theorem:

Theorem 3. *There is a computable function g such that (IP) can be solved in time*

$$g(\|A\|_\infty, \min\{\mathrm{td}_P(A), \mathrm{td}_D(A)\}) \cdot n^2 \log \|\mathbf{u} - \mathbf{l}, \mathbf{b}\|_\infty \log(2f_{\max}) + O(n^\omega \langle A \rangle)$$

In Sections 3.1-3.2 we shall develop the necessary ingredients to prove this theorem. Then, we will conclude in Section 3.7 by providing its proof which puts these ingredients together.

3.1 Introduction to Iterative Augmentation

Let us introduce Graver bases and discuss how they are used for optimization. We define a partial order $\sqsubseteq$ on $\mathbb{R}^n$ as follows: for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ we write $\mathbf{x} \sqsubseteq \mathbf{y}$ and say that $\mathbf{x}$ is *conformal* to $\mathbf{y}$ if, for each $i \in [n]$, $x_i y_i \geq 0$ (that is, $\mathbf{x}$ and $\mathbf{y}$ lie in the same orthant) and $|x_i| \leq |y_i|$. For a matrix $A \in \mathbb{Z}^{m \times n}$ we write $\ker_{\mathbb{Z}}(A) = \{\mathbf{x} \in \mathbb{Z}^n \mid A\mathbf{x} = \mathbf{0}\}$. It is well known that every subset of $\mathbb{Z}^n$ has finitely many $\sqsubseteq$-minimal elements [9].

Definition 4 (Graver basis [10]). The *Graver basis* of an integer $m \times n$ matrix A is the finite set $\mathcal{G}(A) \subset \mathbb{Z}^n$ of $\sqsubseteq$-minimal elements in $\ker_{\mathbb{Z}}(A) \setminus \{\mathbf{0}\}$.

One important property of $\mathcal{G}(A)$ is as follows:

Lemma 5 (Positive Sum Property [18, Lemma 3.4]). *Let $A \in \mathbb{Z}^{m\times n}$. For any $\mathbf{x} \in \ker_{\mathbb{Z}}(A)$, there exists an $n' \leq 2n-1$ and a decomposition $\mathbf{x} = \sum_{j=1}^{n'} \lambda_j \mathbf{g}_j$ with $\lambda_j \in \mathbb{N}$ and $\mathbf{g}_j \in \mathcal{G}(A)$ for each $j \in [n']$, and with $\mathbf{g}_j \sqsubseteq \mathbf{x}$, i.e., all $\mathbf{g}_j$ belonging to the same orthant as $\mathbf{x}$.*

Proof. Let G be a matrix whose columns are $\mathbf{g} \in \mathcal{G}(A)$ such that $\mathbf{g} \sqsubseteq \mathbf{x}$. Consider the following LP in variables $\mathbf{y} \in \mathbb{R}^{|G|}$:

$$\begin{aligned} \max \sum_{\mathbf{g}} y_{\mathbf{g}} \\ G\mathbf{y} &= \mathbf{x} \\ \mathbf{y} &\geq \mathbf{0} \end{aligned}$$

There is a basic optimal solution $\mathbf{y}^*$ and from LP theory we know that, because there are n equality constraints and only non-negativity constraints besides that, $|\operatorname{supp}(\mathbf{y}^*)| \leq n$. We will define the coefficient vector $\boldsymbol{\lambda}$ in two phases. In the first phase, let $\boldsymbol{\lambda} = \lfloor \mathbf{y}^* \rfloor$. Recall that $\{\mathbf{y}^*\}$ is the fractional part of $\mathbf{y}^*$. Observe that $G\{\mathbf{y}^*\}$ is an integer vector, because it is $G\mathbf{y}^* - G\lfloor \mathbf{y}^* \rfloor$ which is a difference of two integer vectors. Thus, $\{\mathbf{y}^*\}$ describes a decomposition of $\bar{\mathbf{x}} := \mathbf{x} - G\lfloor \mathbf{y}^* \rfloor \in \operatorname{Ker}_{\mathbb{Z}}(A)$. Moreover, $\{\mathbf{y}^*\}$ is a decomposition maximizing the ℓ_1-norm, which is the objective of the above LP. (The fact that $\{\mathbf{y}^*\}$ is a fractional decomposition of $\bar{\mathbf{x}}$ maximizing ℓ_1-norm is easy to see by contradiction: if there was a better decomposition $\mathbf{y}'$ of $\bar{\mathbf{x}}$, one could use it to get a better decomposition $\mathbf{y}' + \lfloor \mathbf{y}^* \rfloor$ of $\mathbf{x}$, but $\mathbf{y}^*$ was assumed to be maximum.) Finally, we have that $\|\{\mathbf{y}^*\}\|_1 < n$ because it is a sum of at most n numbers, each strictly smaller than 1. Now consider an optimal *integer* decomposition of $\bar{\mathbf{x}}$, i.e., a non-negative vector $\bar{\mathbf{y}} \in \mathbb{Z}^n$ satisfying $G\bar{\mathbf{y}} = \bar{\mathbf{x}}$. It cannot have a larger ℓ_1-norm than $\{\mathbf{y}^*\}$ because $\{\mathbf{y}^*\}$ is an optimum of the continuous relaxation, thus $\|\bar{\mathbf{y}}\|_1 \leq \|\{\mathbf{y}^*\}\|_1 < n$, that is, $\|\bar{\mathbf{y}}\|_1 \leq n-1$. This is the second phase: update $\boldsymbol{\lambda} := \boldsymbol{\lambda} + \bar{\mathbf{y}}$. We have $|\operatorname{supp}(\boldsymbol{\lambda})| \leq |\operatorname{supp}(\mathbf{y}^*)| + |\operatorname{supp}(\bar{\mathbf{y}})| \leq n + (n-1) = 2n-1$. □

In fact, the Lemma holds with a better constant $2n-2$, and we will use this bound in the sequel, although this has no asymptotic significance for us.

Proposition 6 (Positive Sum Property [18, Lemma 3.4]). *For any $\mathbf{x} \in \ker_{\mathbb{Z}}(A)$, there exists an $n' \leq 2n-2$ and a decomposition $\mathbf{x} = \sum_{j=1}^{n'} \lambda_j \mathbf{g}_j$ with $\lambda_j \in \mathbb{N}$ and $\mathbf{g}_j \in \mathcal{G}(A)$ for each $j \in [n']$, and with $\mathbf{g}_j \sqsubseteq \mathbf{x}$, i.e., all $\mathbf{g}_j$ belonging to the same orthant as $\mathbf{x}$.*

We say that $\mathbf{x} \in \mathbb{Z}^n$ is *feasible* for (IP) if $A\mathbf{x} = \mathbf{b}$ and $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$. Let $\mathbf{x}$ be a feasible solution for (IP). We call $\mathbf{g}$ a *feasible step* if $\mathbf{x} + \mathbf{g}$ is feasible for (IP). Further, call a feasible step $\mathbf{g}$ *augmenting* if $f(\mathbf{x}+\mathbf{g}) < f(\mathbf{x})$. An important implication of Proposition 6 is that if *any* augmenting step exists, then there exists one in $\mathcal{G}(A)$ [4, Lemma 3.3.2].

An augmenting step $\mathbf{g}$ and a *step length* $\lambda \in \mathbb{N}$ form an $\mathbf{x}$-*feasible step pair* with respect to $\mathbf{x}$ if $\mathbf{l} \leq \mathbf{x} + \lambda\mathbf{g} \leq \mathbf{u}$. An augmenting step $\mathbf{h}$ is a *Graver-best step* for $\mathbf{x}$ if $f(\mathbf{x}+\mathbf{h}) \leq f(\mathbf{x}+\lambda\mathbf{g})$ for all $\mathbf{x}$-feasible step pairs $(\mathbf{g}, \lambda) \in \mathcal{G}(A) \times \mathbb{N}$. A slight relaxation of a Graver-best step is a halfling: an augmenting step $\mathbf{h}$ is a *halfling* for $\mathbf{x}$ if $f(\mathbf{x}) - f(\mathbf{x}+\mathbf{h}) \geq \frac{1}{2}(f(\mathbf{x}) - f(\mathbf{x}+\lambda\mathbf{g}))$ for all $\mathbf{x}$-feasible step pairs $(\mathbf{g}, \lambda) \in \mathcal{G}(A) \times \mathbb{N}$. A *halfling augmentation procedure* for (IP) with a given feasible solution $\mathbf{x}_0$ works as follows. Let $i := 0$.

(1) If there is no halfling for $\mathbf{x}_i$, return it as optimal.
(2) If a halfling $\mathbf{h}_i$ for $\mathbf{x}_i$ exists, set $\mathbf{x}_{i+1} := \mathbf{x}_i + \mathbf{h}_i$, $i := i+1$, and go to 1.

We assume that the bounds $\mathbf{l}, \mathbf{u}$ are finite.

Lemma 7 (Halfling convergence). *Given a feasible solution $\mathbf{x}_0$ for (IP), the halfling augmentation procedure finds an optimum of (IP) in at most $3n \log\left(f(\mathbf{x}_0) - f(\mathbf{x}^*)\right) \leq 3n \log\left(f_{\max}^{[\mathbf{l},\mathbf{u}]}\right)$ steps.*

Before we prove the lemma we need a useful proposition about separable convex functions:

Proposition 8 (Separable convex superadditivity [4, Lemma 3.3.1]). *Let $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i)$ be separable convex, let $\mathbf{x} \in \mathbb{R}^n$, and let $\mathbf{g}_1, \ldots, \mathbf{g}_k \in \mathbb{R}^n$ be vectors that are pairwise conformal. Then*

$$f\left(\mathbf{x} + \sum_{j=1}^{k} \alpha_j \mathbf{g}_j\right) - f(\mathbf{x}) \geq \sum_{j=1}^{k} \alpha_j \left(f(\mathbf{x} + \mathbf{g}_j) - f(\mathbf{x})\right) \tag{1}$$

for arbitrary integers $\alpha_1, \ldots, \alpha_k \in \mathbb{N}$.

The essence of this proposition is that if multiple steps take us from $\mathbf{x}_0$ to $\mathbf{x}^*$, then the sum of their improvements considered individually with respect to $\mathbf{x}_0$ is *at least* the improvement when we take them together, i.e., the difference $f(\mathbf{x}) - f(\mathbf{x}^*)$. (This is tricky to read correctly because "improvement" is a negative term, because we are minimizing.) An illustrative example is $f(x) = x^2$: if we are at a point $x = 2$, then moving by 1 closer to the origin improves the objective by 3 (decrease from 4 to 1), but moving by another 1 only improves it by 1 (decrease from 1 to 0). So if we think of the path from 2 to 0 as two steps by 1, when we consider the total of the progress each step would achieve individually with respect to the initial point 2, we get $3 + 3 = 6$, but taken together, the steps only achieve the progress of 4. Essentially, the contribution of each step considered in the sequence is at most the contribution of each step considered individually with respect to the initial point.

Proof of Lemma 7. Let $\mathbf{x}^*$ be an optimal solution of (IP). By Proposition 6 we may write $\mathbf{x}^* - \mathbf{x}_0 = \sum_{j=1}^{n'} \lambda_j \mathbf{g}_j$ such that $\mathbf{g}_j \sqsubseteq \mathbf{x}^* - \mathbf{x}_0$ for all $j \in [n']$, and $n' \leq 2n - 2$. We apply Proposition 8 to $\mathbf{x}_0$ and the n' vectors $\lambda_j \mathbf{g}_j$ with $\alpha_j := 1$, so by (1) we have

$$0 \geq f(\mathbf{x}^*) - f(\mathbf{x}_0) = f\left(\mathbf{x}_0 + \sum_{j=1}^{n'} \lambda_j \mathbf{g}_j\right) - f(\mathbf{x}_0) \geq \sum_{j=1}^{n'} \left(f(\mathbf{x}_0 + \lambda_j \mathbf{g}_j) - f(\mathbf{x}_0)\right),$$

and multiplying by -1 gives $0 \leq f(\mathbf{x}_0) - f(\mathbf{x}^*) \leq \sum_{j=1}^{n'} (f(\mathbf{x}_0) - f(\mathbf{x}_0 + \lambda_j \mathbf{g}_j))$. By an averaging argument, there must exist an index $\ell \in [n']$ such that

$$f(\mathbf{x}_0) - f(\mathbf{x}_0 + \lambda_\ell \mathbf{g}_\ell) \geq \frac{1}{n'} \left(f(\mathbf{x}_0) - f(\mathbf{x}^*)\right) \geq \frac{1}{n'} f_{\max} \quad . \tag{2}$$

Consider a halfling $\mathbf{h}$ for $\mathbf{x}_0$: by definition, it satisfies $f(\mathbf{x}_0) - f(\mathbf{x}_0 + \mathbf{h}) \geq \frac{1}{2}(f(\mathbf{x}_0) - f(\mathbf{x}_0 + \lambda_i \mathbf{g}_i))$. Say that the halfling augmentation procedure required s iterations. For $i \in [s - 1]$ we have that

$$f(\mathbf{x}_i) - f(\mathbf{x}^*) \leq \left(1 - \frac{1}{4n - 4}\right) \left(f(\mathbf{x}_{i-1}) - f(\mathbf{x}^*)\right) = \frac{4n - 5}{4n - 4} \left(f(\mathbf{x}_{i-1}) - f(\mathbf{x}^*)\right)$$

and, by repeated application of the above,

$$f(\mathbf{x}_i) - f(\mathbf{x}^*) \leq \left(\frac{4n - 5}{4n - 4}\right)^i \left(f(\mathbf{x}_0) - f(\mathbf{x}^*)\right) \quad .$$

Since i is not the last iteration, $f(\mathbf{x}_i) - f(\mathbf{x}^*) \geq 1$ by the integrality of f. Take $t := 4n - 4$ and compute

$$\begin{aligned} 1 &\leq \left(\frac{t-1}{t}\right)^i (f(\mathbf{x}_0) - f(\mathbf{x}^*)) && / \ln(\cdot) \\ 0 &\leq i \ln\left(\frac{t-1}{t}\right) + \ln\left(f(\mathbf{x}_0) - f(\mathbf{x}^*)\right) && / -i \ln\left(\frac{t-1}{t}\right) \\ -i \ln\left(\frac{t-1}{t}\right) = i \ln\left(\frac{t}{t-1}\right) &\leq \ln\left(f(\mathbf{x}_0) - f(\mathbf{x}^*)\right) && / : \ln\left(\frac{t-1}{t}\right) \\ i &\leq \left(\ln\left(\frac{t}{t-1}\right)\right)^{-1} \ln\left(f(\mathbf{x}_0) - f(\mathbf{x}^*)\right) \end{aligned}$$

Now Taylor expansion gives for $t \geq 3$

$$\ln\left(1 + \frac{1}{t-1}\right) \geq \frac{1}{t-1} - \frac{1}{2(t-1)^2} = \frac{2t-3}{2t^2-4t+2},$$

so

$$\left(\ln\left(1 + \frac{1}{t-1}\right)\right)^{-1} \leq \frac{2t^2-4t+2}{2t-3},$$

which is bounded above by t for all $t \geq 2$ since $t(2t-3) = 2t^2 - 3t \geq 2t^2 - 4t + 2$ for all $t \geq 2$. Plugging back $t := 4n - 4$ we get that for all $n \geq 2$ we have $t \geq 3$ and hence

$$i \leq (4n-4) \ln\left(f(\mathbf{x}_0) - f(\mathbf{x}^*)\right) = (4n-4) \cdot \ln 2 \cdot \log_2\left(f(\mathbf{x}_0) - f(\mathbf{x}^*)\right),$$

and the number of iterations is at most one unit larger. Since $f(\mathbf{x}_0) - f(\mathbf{x}^*) \leq f_{\max}$ and $\ln(2) = 0.693147\cdots \leq 3/4$, we have that the number of iterations is at most $3n \log(f_{\max})$. □

Clearly it is now desirable to show how to find halflings quickly. The following lemma will be helpful in that regard.

Lemma 9 (Powers of Two). *Let $\Gamma_2 = \{1, 2, 4, 8, \dots\}$ and $\mathbf{x}$ be a feasible solution of (IP). If $\mathbf{h}$ satisfies $f(\mathbf{x}+\mathbf{h}) \leq f(\mathbf{x}+\lambda\mathbf{g})$ for each $\mathbf{x}$-feasible step pair $(\mathbf{g}, \lambda) \in \mathcal{G}(A) \times \Gamma_2$, then $\mathbf{h}$ is a halfling.*

Proof. Consider any Graver-best step pair $(\mathbf{g}^*, \lambda^*) \in \mathcal{G}(A) \times \mathbb{N}$, let $\lambda := 2^{\lfloor \log \lambda^* \rfloor}$, and choose $\frac{1}{2} < \gamma \leq 1$ in such a way that $\lambda = \gamma\lambda^*$. Convexity of f yields

$$\begin{aligned} f(\mathbf{x}_0) - f(\mathbf{x}_0 + \lambda\mathbf{g}^*) &\geq f(\mathbf{x}_0) - \left[(1-\gamma) f(\mathbf{x}_0) + \gamma f(\mathbf{x}_0 + \lambda^*\mathbf{g}^*)\right] \\ &= \gamma\left(f(\mathbf{x}_0) - f(\mathbf{x}_0 + \lambda^*\mathbf{g}^*)\right) \\ &\geq \frac{1}{2}\left(f(\mathbf{x}_0) - f(\mathbf{x}_0 + \lambda^*\mathbf{g}^*)\right) \quad . \end{aligned}$$

This shows that $\lambda\mathbf{g}^*$ is a halfling, and by the definition of $\mathbf{h}$, $f(\mathbf{x}+\mathbf{h}) \leq f(\mathbf{x}+\lambda\mathbf{g}^*)$ and thus $\mathbf{h}$ is a halfling as well. □

This makes it clear that the main task is to find, for each $\lambda \in \Gamma_2$, a step $\mathbf{h}$ which is at least as good as any feasible $\lambda\mathbf{g}$ with $\mathbf{g} \in \mathcal{G}(A)$. We need the notion of a *best* solution:

Definition 10 (S-best solution). Let $S, P \subseteq \mathbb{R}^n$. We say that $\mathbf{x}^* \in P$ is a solution of

$$S\text{-best } \{f(\mathbf{x}) \mid \mathbf{x} \in P\} \tag{S-best}$$

if $f(\mathbf{x}^*) \leq \min\{f(\mathbf{x}) \mid \mathbf{x} \in P \cap S\}$. If $P \cap S$ is empty, we say S-best $\{f(\mathbf{x}) \mid \mathbf{x} \in P\}$ has no solution.

In other words, $\mathbf{x}^*$ has to belong to P and be at least as good as any point in $P \cap S$. Note that to define the notion of an S-best solution to be a "no solution" if $P \cap S = \emptyset$ might look unnatural as one might require *any* $\mathbf{x} \in P$ if $P \cap S = \emptyset$. However, this would make (S-best) as hard as finding some $\mathbf{x} \in P$ (just take $S = \emptyset$), but intuitively (S-best) should be an easier problem. The following is a central notion.

Definition 11 (Augmentation IP). For an (IP) instance $(A, f, \mathbf{b}, \mathbf{l}, \mathbf{u})$, its feasible solution $\mathbf{x} \in \mathbb{Z}^n$, and an integer $\lambda \in \mathbb{N}$, the *Augmentation IP* problem is to solve

$$\mathcal{G}(A)\text{-best}\{f(\mathbf{x} + \lambda\mathbf{g}) \mid A\mathbf{g} = \mathbf{0},\ \mathbf{l} \leq \mathbf{x} + \lambda\mathbf{g} \leq \mathbf{u},\ \mathbf{g} \in \mathbb{Z}^n\}\ . \tag{AugIP}$$

Let $(A, f, \mathbf{b}, \mathbf{l}, \mathbf{u})$ be an instance of (IP), $\mathbf{x}$ a feasible solution, and $\lambda \in \mathbb{N}$. We call the pair $(\mathbf{x}, \lambda)$ an (AugIP) *instance* for $(A, f, \mathbf{b}, \mathbf{l}, \mathbf{u})$. If clear from the context, we omit the (IP) instance $(A, f, \mathbf{b}, \mathbf{l}, \mathbf{u})$.

By Lemma 9 we obtain a halfling by solving (AugIP) for each $\lambda \in \Gamma_2$ and picking the best solution. Given an initial feasible solution $\mathbf{x}_0$ and a fast algorithm for (AugIP), we can solve (IP) quickly:

Lemma 12 (((AugIP) and $\mathbf{x}_0$) $\implies$ (IP)). *Given an initial feasible solution $\mathbf{x}_0$ to (IP), (IP) can be solved by solving*

$$3n(\log \|\mathbf{u} - \mathbf{l}\|_\infty + 1) \log(f(\mathbf{x}_0) - f(\mathbf{x}^*)) \leq 3n(\log \|\mathbf{u} - \mathbf{l}\|_\infty + 1) \log\left(f_{\max}^{[\mathbf{l},\mathbf{u}]}\right)$$

instances of (AugIP), where $\mathbf{x}^*$ is any optimum of (IP).

Proof. Observe that no $\lambda \in \Gamma_2 = \{1, 2, 4, \dots\}$ greater than $\|\mathbf{u} - \mathbf{l}\|_\infty$ results in a non-zero $\mathbf{x}$-feasible step pair. Thus, by Lemma 9, to compute a halfling for $\mathbf{x}$ it suffices to solve (AugIP) for all $\lambda \in \Gamma_2$, $\lambda \leq \|\mathbf{u} - \mathbf{l}\|_\infty$, and there are at most $\log \|\mathbf{u} - \mathbf{l}\|_\infty + 1$ of these. By Lemma 7, $3n \log(f(\mathbf{x}_0) - f(\mathbf{x}^*)) \leq 3n \log(f_{\max})$ halfling augmentations suffice and we are done. □

Feasibility. Our goal now is to satisfy the requirement of an initial solution $\mathbf{x}_0$.

Lemma 13 ((AugIP) $\implies$ $\mathbf{x}_0$). *Given an instance of (IP), it is possible to compute a feasible solution $\mathbf{x}_0$ for (IP) or decide that (IP) is infeasible by solving $O(n \log(\|A, \mathbf{b}, \mathbf{l}, \mathbf{u}\|_\infty)^2)$ many (AugIP) instances, plus $O(n^\omega)$ time needed to compute an integral solution of $A\mathbf{z} = \mathbf{b}$. Moreover, $\langle \mathbf{x}_0 \rangle \leq \text{poly}\langle \mathbf{b} \rangle$.*

Proof. We first compute an integer solution to the system of equations $A\mathbf{z} = \mathbf{b}$. This can be done by computing the Hermite normal form of A in time $O(n^{\omega-1}m) \leq O(n^\omega)$ [22] (using $m \leq n$). Then either we conclude that there is no integer solution to $A\mathbf{z} = \mathbf{b}$ and hence (IP) is infeasible, or we find a solution $\mathbf{z} \in \mathbb{Z}^n$ with encoding length polynomially bounded in the encoding length of $A, \mathbf{b}$.

Next, we will solve an auxiliary IP. Define new relaxed bounds by

$$\hat{l}_i := \min\{l_i, z_i\},\quad \hat{u}_i := \max\{u_i, z_i\},\qquad i \in [n],$$

and define an objective function $\hat{f} := \sum_{i=1}^n \hat{f}_i$ as, for each $i \in [n]$, $\hat{f}_i(x_i) := \text{dist}(x_i, [l_i, u_i])$, which is 0 if $x_i \in [l_i, u_i]$ and $\max\{l_i - x_i, x_i - u_i\}$ otherwise. This function has at most three linear pieces, the first decreasing, the second constantly zero, and the third increasing, and thus each $\hat{f}_i$ is convex and $\hat{f}$ is separable convex. Moreover, a solution $\mathbf{x}$ has $\hat{f}(\mathbf{x}) = 0$ if and only if $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$.

By Lemma 7, an optimum $\mathbf{x}_0$ of $\min\left\{\hat{f}(\mathbf{x}) \,\middle|\, A\mathbf{x} = \mathbf{b},\ \hat{\mathbf{l}} \leq \mathbf{x} \leq \hat{\mathbf{u}},\ \mathbf{x} \in \mathbb{Z}^n\right\}$ can be computed by solving $3n(\log \|\hat{\mathbf{u}} - \hat{\mathbf{l}}\| + 1) \log\left(\hat{f}_{\max}^{[\hat{\mathbf{l}},\hat{\mathbf{u}}]}\right)$ instances of (AugIP). Since $\|\hat{\mathbf{l}}, \hat{\mathbf{u}}\|_\infty$ is polynomially bounded in $\|A, \mathbf{b}\|_\infty$ and $\|\mathbf{l}, \mathbf{u}\|_\infty$ and, by definition of $\hat{f}$, $\hat{f}_{\max}^{[\hat{\mathbf{l}},\hat{\mathbf{u}}]}$ is bounded by $n \cdot \|\hat{\mathbf{l}}, \hat{\mathbf{u}}\|_\infty$, we have that the number of times we have to solve (AugIP) is bounded by $O(n \log(\|A, \mathbf{b}, \mathbf{l}, \mathbf{u}\|_\infty)^2)$. Finally, if $\hat{f}(\mathbf{x}_0) = 0$ then $\mathbf{x}_0$ is a feasible solution of (IP) and otherwise (IP) is infeasible. □

As a corollary of Lemmas 13 and 7, we immediately obtain that a polynomial (AugIP) algorithm is sufficient for solving (IP) in polynomial time:

Corollary 14 ((AugIP) $\implies$ (IP)). *Problem (IP) can be solved by solving $O(nL^2)$ instances of (AugIP), where $L := \log(\|A, f_{\max}, \mathbf{b}, \mathbf{l}, \mathbf{u}\|_\infty)$, plus time $O(n^\omega + \min\{n, m\}nm)$.*

3.2 Bounding The Norm

We begin by using the Steinitz Lemma to obtain a basic bound on $g_1(A)$.

Proposition 15 (Steinitz [21], Sevastjanov, Banaszczyk [20]). *Let $\|\cdot\|$ be any norm, and let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ be such that $\|\mathbf{x}_i\| \le 1$ for $i \in [n]$ and $\sum_{i=1}^n \mathbf{x}_i = \mathbf{0}$. Then there exists a permutation $\pi \in S_n$ such that for each $k \in [n]$, $\|\sum_{i=1}^k \mathbf{x}_{\pi(i)}\| \le d$.*

Proof. We will inductively construct sets $A_n \supset A_{n-1} \supset A_{n-2} \supset \cdots A_m$ where $A_n = [n]$, $|A_k| = k$, and $\{\pi(i)\} = A_i \setminus A_{i-1}$, that is, $\pi(i)$ is the index which is removed as we go from A_i to A_{i-1}. (This also means we construct the permutation "backwards", first defining $\pi(n)$, then $\pi(n-1)$ etc. We stop at $\pi(n-m+1)$ because the sum of m vectors of norm at most 1 has norm at most m, regardless of the order.) The key is the following LP in variables $\boldsymbol{\lambda}$ (*not* $\mathbf{x}$; those are the input vectors, so they act as constants below):

$$\sum_{i \in A_k} \lambda_i \mathbf{x}_i = \mathbf{0}, \tag{3}$$

$$\sum_{i \in A_k} \lambda_i = k - m \tag{4}$$

$$\mathbf{0} \le \boldsymbol{\lambda} \le \mathbf{1} \tag{5}$$

Call it LP_k. First consider $k = n$. Observe that the LP is feasible: taking $\boldsymbol{\lambda} = \alpha\mathbf{1}$ satisfies (3) for any scalar $\alpha \in \mathbb{R}$, and choosing $\alpha = (n-m)/n$ scales the solution so that (4) is satisfied. Now assume that we are in step k of the construction, that is, we have a solution $\boldsymbol{\lambda}$ satisfying LP_k, and we wish to find an index to remove from A_k in order to construct A_{k-1} and define $\pi(k)$. A solution $\boldsymbol{\lambda}$ satisfying LP_k can be scaled down to satisfy $\sum_{i \in A_k} \lambda_i = k - m - 1$. Since there is a solution satisfying (3) and $\sum_{i \in A_k} \lambda_i = k - m - 1$, there is also a basic solution $\boldsymbol{\lambda}^*$ satisfying these constraints. Moreover, those are $m+1$ equality constraints and the rest are inequality constraints $\mathbf{0} \le \boldsymbol{\lambda} \le \mathbf{1}$. LP theory says that $\boldsymbol{\lambda}^*$ has at most $m+1$ fractional entries and the rest are integral, that is, either 0 or 1. We claim that at least one of the integer entries in $\boldsymbol{\lambda}^*$ is 0, and this is the index we will drop in order to define A_k. We have k variables in $\boldsymbol{\lambda}^*$, and they sum up to $k - m - 1$. Say that there are $\varphi \le m+1$ fractional variables: they contribute strictly less than φ, say $\varphi - \epsilon$, with $\epsilon < m+1$. The remaining $k - \varphi$ integer variables must sum up to $k - m - 1 - (\varphi - \epsilon) = k - m - 1 - \varphi + \epsilon$, but this is impossible if they were all 1 because $\epsilon < m+1$ and $k - \varphi > k - m - 1 - \varphi + \epsilon$, equivalently, $k > k - m - 1 + \epsilon$. So there is an index i such that $\lambda_i^* = 0$ and we set $A_{k-1} = A_k \setminus \{i\}$.

Now it remains to verify that all prefix sums are indeed bounded by m. Consider:

$$\left\| \sum_{i=1}^{k} \mathbf{x}_{\pi(i)} \right\| = \left\| \sum_{i \in A_k} \mathbf{x}_i \right\| = \left\| \sum_{i \in A_k} (1 - \lambda_i)\mathbf{x}_i \right\| \le \sum_{i \in A_k} (1 - \lambda_i) = m$$

The first equality follows by the definition of π; the second follows by the fact that subtracting $\sum_i \lambda_i \mathbf{x}_i$ equates to subtracting $\mathbf{0}$ (by constraint (3)); the next inequality follows by assumption $\|\mathbf{x}_i\| \le 1$ for each i, and the final equality is by constraint (4). □

Lemma 16 (Base bound). *Let $A \in \mathbb{Z}^{m \times n}$. Then $g_1(A) \le (2m\|A\|_\infty + 1)^m$.*

Proof. Let $\mathbf{g} \in \mathcal{G}(A)$. We define a sequence of vectors in the following manner. If $g_i \geq 0$, we add g_i copies of the i-th column of A to the sequence, if $g_i < 0$ we add $|g_i|$ copies of the negation of column i to the sequence, either way obtaining vectors $\mathbf{v}^i_1, \ldots, \mathbf{v}^i_{|g_i|}$.

Clearly, the vectors $\mathbf{v}^i_j$ sum up to $\mathbf{0}$ as $\mathbf{g} \in \ker_{\mathbb{Z}}(A)$ and their ℓ_∞-norm is bounded by $\|A\|_\infty$. Using the Steinitz Lemma, there is a reordering $\mathbf{u}^1, \ldots, \mathbf{u}^{\|\mathbf{g}\|_1}$ (i.e., $\mathbf{v}^i_j = \mathbf{u}^{\pi(i,j)}$ for some permutation π) of this sequence such that each prefix sum $\mathbf{p}_k \coloneqq \sum_{j=1}^{k} \mathbf{u}^j$ is bounded by $m\|A\|_\infty$ in the l_∞-norm. Clearly

$$|\{\mathbf{x} \in \mathbb{Z}^m \mid \|\mathbf{x}\|_\infty \leq m\|A\|_\infty\}| = (2m\|A\|_\infty + 1)^m \quad .$$

Assume for contradiction that $\|\mathbf{g}\|_1 > (2m\|A\|_\infty + 1)^m$. Then two of these prefix sums are the same, say, $\mathbf{p}_\alpha = \mathbf{p}_\beta$ with $1 \leq \alpha < \beta \leq \|\mathbf{g}\|_1$. Obtain a vector $\mathbf{g}'$ from the sequence $\mathbf{u}^1, \ldots, \mathbf{u}^\alpha, \mathbf{u}^{\beta+1}, \ldots, \mathbf{u}^{\|\mathbf{g}\|_1}$ as follows: begin with $g'_i \coloneqq 0$ for each $i \in [n]$, and for every $\mathbf{u}^\ell$ in the sequence, set

$$g'_i \coloneqq \begin{cases} g'_i + 1 & \text{if } \pi^{-1}(\ell) = (i,j) \text{ and } g_i \geq 0 \\ g'_i - 1 & \text{if } \pi^{-1}(\ell) = (i,j) \text{ and } g_i < 0 \end{cases} .$$

Similarly obtain $\mathbf{g}''$ from the sequence $\mathbf{u}^{\alpha+1} \ldots, \mathbf{u}^\beta$. We have $A\mathbf{g}'' = \mathbf{0}$ because $\mathbf{p}_\alpha - \mathbf{p}_\beta = \mathbf{0}$ and thus $\mathbf{g}'' \in \ker_{\mathbb{Z}}(A)$, and thus also $\mathbf{g}' \in \ker_{\mathbb{Z}}(A)$. Moreover, both $\mathbf{g}'$ and $\mathbf{g}''$ are non-zero and satisfy $\mathbf{g}', \mathbf{g}'' \sqsubseteq \mathbf{g}$. This is a contradiction with $\sqsubseteq$-minimality of $\mathbf{g}$, hence $\|\mathbf{g}\|_1 \leq (2m\|A\|_\infty + 1)^m$, finishing the proof. □

3.3 A Dynamic Programming Algorithm

Lemma 17 (Basic DP). *Problem* (AugIP) *can be solved in time* $(2\|A\|_\infty g_1(A) + 1)^m n$.

Proof. We solve an auxiliary problem: given $\rho \in \mathbb{N}$ and a separable convex function f, solve

$$B_1(\rho)\text{-best}\{f(\mathbf{g}) \mid A\mathbf{g} = \mathbf{0},\ \mathbf{l} \leq \mathbf{g} \leq \mathbf{u},\ \mathbf{g} \in \mathbb{Z}^n\} \quad . \tag{aux1}$$

The lemma then follows by the following substitution. For a given (AugIP) instance $(\mathbf{x}, \lambda)$, solve the auxiliary problem (aux1) above with $\rho \coloneqq g_\infty(A)$, $f(\mathbf{g}) \coloneqq f(\mathbf{x} + \lambda\mathbf{g})$, $\mathbf{l} \coloneqq \left\lceil \frac{\mathbf{l}-\mathbf{x}}{\lambda} \right\rceil$, and $\mathbf{u} \coloneqq \left\lfloor \frac{\mathbf{u}-\mathbf{x}}{\lambda} \right\rfloor$. If f of (IP) was separable convex, then the newly defined f is also separable convex. The returned solution is a solution of (AugIP) because $\mathcal{G}(A) \subseteq B_1(g_1(A))$.

Let $\hat{A}_i \coloneqq A_{\bullet,i}$ for all $i \in [n]$, be the columns of A. The crucial observation is that for every solution $\mathbf{g}$ of $A\mathbf{g} = \mathbf{0}$ with $\|\mathbf{g}\|_1 \leq \rho$ and each $i \in [n]$, both $\hat{A}_i g_i$ and $\sum_{j=1}^{i} \hat{A}_j g_j$ belong to $R \coloneqq [-\rho\|A\|_\infty, \rho\|A\|_\infty]^m$. We will now construct a directed acyclic graph, designate a start and target vertex, and explain how a shortest path between them corresponds to an optimum of (aux1).

There are $n+1$ layers of vertices. The first layer contains a single vertex s, which is the m-dimensional 0-vector, and this is the starting vertex. Each of the following $n-1$ layers is a copy of the set R. Finally, layer $n+1$ again only contains $\mathbf{0} \in \mathbb{Z}^m$, which is the target vertex t. A vertex v in layer $i \in [n]$ is some m-dimensional vector $\mathbf{r}$, and it has an edge to every $\mathbf{r}'$ in layer $i+1$ such that $\mathbf{r}' = \mathbf{r} + \hat{A}_i g_i$, $g_i \in [-\rho, \rho] \cap [l_i, u_i]$. Note that for some g_i perhaps $\mathbf{r}' \notin R$; then there is no edge. This edge has a label g_i and a length $f_i(g_i)$ (recall here that this is f from (aux1) and corresponds to $f(\mathbf{x} + \lambda\mathbf{g})$ in which $\mathbf{x}$ is a constant).

Consider a path P from s to t. Define $\mathbf{g}$ coordinate-wise by defining g_i to be the label on the edge from layer i to layer $i+1$. Observe that for every $i \in [n]$, the prefix sum $\sum_{j=1}^{i} \hat{A}_j g_j$ is exactly the vertex $\mathbf{r}$ on P in layer i; consequently, $A\mathbf{g} = \mathbf{0}$. Moreover, every $\mathbf{g} \in \ker_{\mathbb{Z}}(A) \cap B_1(\rho)$ is represented by a (unique) path P. Next, observe that $f(\mathbf{g})$ is exactly the length of the path P corresponding to $\mathbf{g}$. Thus, a vector $\mathbf{g}$ corresponding to the shortest path is a solution of (aux1).

Next, regarding the complexity of this algorithm. The shortest path can be computed layer by layer, where processing each layer takes time at most $|R|$ times the maximum out-degree, which is

$|R| \cdot (2\rho + 1) \leq (2|A|_\infty \rho + 1)^m$, and there are n layers to process. Plugging in $\rho := g_1(A)$ yields the claimed time. □

3.4 The Graphs of A and Treedepth

Definition 18 (Primal and dual graph). Given a matrix $A \in \mathbb{Z}^{m \times n}$, its *primal graph* $G_P(A) = (V, E)$ is defined as $V = [n]$ and $E = \left\{ \{i, j\} \in \binom{[n]}{2} \;\middle|\; \exists k \in [m] : A_{k,i}, A_{k,j} \neq 0 \right\}$. In other words, its vertices are the columns of A and two vertices are connected if there is a row with non-zero entries at the corresponding columns. The *dual graph of* A is defined as $G_D(A) := G_P(A^\intercal)$, that is, the primal graph of the transpose of A.

From this point on we always assume that $G_P(A)$ and $G_D(A)$ are connected, otherwise A has (up to row and column permutations) a diagonal structure $A = \begin{pmatrix} A_1 & & \\ & \ddots & \\ & & A_d \end{pmatrix}$ and solving (IP) amounts to solving d smaller (IP) instances independently.

Definition 19 (Treedepth). The *closure* $\mathrm{cl}(F)$ of a rooted tree F is the graph obtained from F by making every vertex adjacent to all of its ancestors. We consider both F and $\mathrm{cl}(F)$ as undirected graphs. The *height* of a tree F denoted $\mathrm{height}(F)$ is the maximum number of vertices on any root-leaf path. The *treedepth* $\mathrm{td}(G)$ of a connected graph G is the minimum height of a tree F such that $G \subseteq \mathrm{cl}(F)$. A td-*decomposition of* G is a tree F such that $G \subseteq \mathrm{cl}(F)$. A td-decomposition F of G is *optimal* if $\mathrm{height}(F) = \mathrm{td}(G)$.

Computing $\mathrm{td}(G)$ is NP-hard, but fortunately can be done quickly when $\mathrm{td}(G)$ is small:

Proposition 20 ([19]). *The treedepth* $\mathrm{td}(G)$ *of a graph* G *with an optimal* td-*decomposition* F *can be computed in time* $2^{\mathrm{td}(G)^2} \cdot |V(G)|$.

We define the *primal treedepth of* A to be $\mathrm{td}_P(A) := \mathrm{td}(G_P(A))$ and the *dual treedepth of* A to be $\mathrm{td}_D(A) := \mathrm{td}(G_D(A))$.

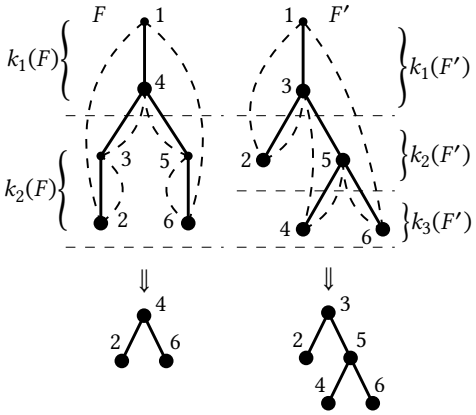
We often assume that an optimal td-decomposition is given since the time required to find it is dominated by other terms. Moreover, in many applications a small td-decomposition of $G_P(A)$ or $G_D(A)$ is clear from the way A was constructed and does not have to be computed as part of the algorithm.

It is clear that a graph G has at most $\mathrm{td}(G)^2|V(G)|$ edges because the closure of each root-leaf path of a td-decomposition of G contains at most $\mathrm{td}(G)^2$ edges, and there are at most n leaves. Thus, constructing $G_P(A)$ or $G_D(A)$ can be done in linear time if A is given in its sparse representation. Throughout we shall assume that $G_P(A)$ or $G_D(A)$ are given.

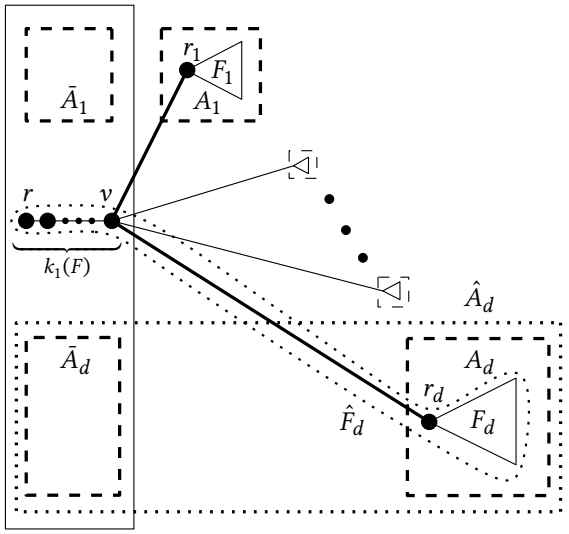
To facilitate our proofs and to provide more refined complexity bounds we introduce a new parameter called topological height. This notion is useful in our analysis and proofs, and we later show that it plays a crucial role in complexity estimates of (IP). It has not been studied elsewhere to the best of our knowledge.

Definition 21 (Topological height). A vertex of a rooted tree F is *degenerate* if it has exactly one child, and *non-degenerate* otherwise (i.e., if it is a leaf or has at least two children). The *topological height of* F, denoted $\mathrm{th}(F)$, is the maximum number of non-degenerate vertices on any root-leaf path in F. Equivalently, $\mathrm{th}(F)$ is the height of F after contracting each edge from a degenerate vertex to its unique child. Clearly, $\mathrm{th}(F) \leq \mathrm{height}(F)$.

We shall now define the level heights of F, which relate to lengths of paths between non-degenerate vertices. For a root-leaf path $P = (v_{b(0)}, \ldots, v_{b(1)}, \ldots, v_{b(2)}, \ldots, v_{b(e)})$ with e non-degenerate vertices $v_{b(1)}, \ldots, v_{b(e)}$ (potentially $v_{b(0)} = v_{b(1)}$), define $k_1(P) := |\{v_{b(0)}, \ldots, v_{b(1)}\}|$,



(a) Two optimal td-decompositions F and F' of the cycle on six vertices (in dashed edges). Non-degenerate vertices are enlarged. The trees obtained by contracting edges outgoing from vertices with only one child are pictured below. Notice that even though both F and F' are optimal td-decompositions, their topological height differs. Dashed lines depict "levels" of F and F', and we have $k_1(F) = k_2(F) = k_1(F') = 2$ and $k_2(F') = k_3(F') = 1$.



(b) The situation of Lemma 23: a td-decomposition F of $G_P(A)$ pictured in the matrix A, the decomposition into smaller blocks $\bar{A}_1, \ldots, \bar{A}_d, A_1, \ldots, A_d$ derived from F and their td-decompositions $F_1, \ldots, F_d$, and a td-decomposition $\hat{F}_d$ of $G_P(\hat{A}_d)$ (Lemma 26).

Fig. 1. Illustration of Definitions 19 and 21 (part 1a) and Lemmas 23 and 26 (part 1b).

$k_i(P) \coloneqq |\{v_{b(i-1)}, \ldots, v_{b(i)}\}| - 1$ for all $i \in [2, e]$, and $k_i(P) \coloneqq 0$ for all $i > e$. For each $i \in [\mathrm{th}(F)]$, define $k_i(F) \coloneqq \max_{P:\text{root-leaf path}} k_i(P)$. We call $k_1(F), \ldots, k_{\mathrm{th}(F)}(F)$ the *level heights of* F. See Figure 1a.

Definition 22 (Block-structured Matrix). v Let $A \in \mathbb{Z}^{m \times n}$ and F be a td-decomposition of $G_P(A)$. We say that A is *block-structured along* F if either $\mathrm{th}(F) = 1$, or if $\mathrm{th}(F) > 1$ and the following holds. Let v be the first non-degenerate vertex in F on a path from the root, $r_1, \ldots, r_d$ be the children of v, F_i be the subtree of F rooted in r_i, and $n_i \coloneqq |V(F_i)|$, for $i \in [d]$, and

$$A = \begin{pmatrix} \bar{A}_1 & A_1 & & \\ \vdots & & \ddots & \\ \bar{A}_d & & & A_d \end{pmatrix}, \qquad \text{(block-structure)}$$

where, for $i \in [d]$, $\bar{A}_i \in \mathbb{Z}^{m_i \times k_1(F)}$ where $k_1(F)$ is the first level height of F and $m_i \in \mathbb{N}$, $A_i \in \mathbb{Z}^{m_i \times n_i}$, and A_i is block-structured along F_i. Note that $\mathrm{th}(F_i) \leq \mathrm{th}(F) - 1$, $\mathrm{height}(F_i) \leq \mathrm{height}(F) - k_1(F)$, for $i \in [d]$.

Whenever A and F are given, we will assume throughout this paper that A is block-structured along F. The following Lemma shows that this is without loss of generality as we can always efficiently put A in this format.

Lemma 23 (Primal Decomposition). *Let $A \in \mathbb{Z}^{m\times n}$, $G_P(A)$, and a* td*-decomposition F of $G_P(A)$ be given, where $n, m \geq 1$. There exists an algorithm which in time $O(n)$ permutes the rows and columns of A such that the resulting matrix A' is block-structured along F.*

Proof of Lemma 23. Denote by r the root of F. Take any root-leaf path in F and let v be the first non-degenerate vertex (i.e., a leaf or a vertex with at least two children) on this path; note that possibly $v = r$. Denote by P the path from r to v and observe that $k_1(F)$ is the number of vertices of P. Let $d := \deg(v)$ be the number of children of v and denote them $r_1, \ldots, r_d$, and let $F_1, \ldots, F_d$ be the subtrees of F rooted in $r_1, \ldots, r_d$. Clearly $\mathrm{th}(F_i) \leq \mathrm{th}(F) - 1$ for each $i \in [d]$ since v is a non-degenerate vertex. For a set of column indices $S \subseteq [n]$ denote by $A_{\bullet,S}$ the submatrix of A consisting of exactly the columns indexed by S. For each $i \in [d]$, we obtain A_i from $A_{\bullet,V(F_i)}$ by deleting zero rows, and we obtain $\bar{A}_i$ from $A_{\bullet,V(P)}$ by only keeping rows which are non-zero in $A_{\bullet,V(F_i)}$. For every row of A whose support is contained in $V(P)$, append its restriction to $V(P)$ to $\bar{A}_1$, and append a zero row to A_1. If A has ζ zero rows, then append ζ zero rows to $\bar{A}_1$ and A_1; this accounts for zero rows of A. Now, apply the same procedure recursively to $F_1, \ldots, F_d$; the base case is when F_i has $\mathrm{th}(F_i) = 1$ and A_i is already block-structured by definition.

To finish the proof we need to argue that A has the form (block-structure), in particular, that there is no overlap between the blocks $A_1, \ldots, A_d$. This follows simply from the fact that by the definition of treedepth there are no edges between any two $u \in F_i$, $w \in F_j$ for $i \neq j$, and thus, by definition of $G_P(A)$, there is no row containing a non-zero at indices u and v, cf. Figure 1b. □

Note that given an (IP), the primal decomposition naturally partitions the right hand side $\mathbf{b} = (\mathbf{b}^1, \ldots, \mathbf{b}^d)$ according to the rows of $A_1, \ldots, A_d$, and each object of length n (such as bounds $\mathbf{l}, \mathbf{u}$, a solution $\mathbf{x}$, any step $\mathbf{g}$, or the objective function f) into $d+1$ objects according to the columns of $\bar{A}_1, A_1, \ldots, A_d$. For example, we write $\mathbf{x} = (\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^d)$.

By considering the transpose of A we get an analogous notion and a corollary for the dual case:

Definition 24 (Block-structured Matrix (dual)). Let $A \in \mathbb{Z}^{m\times n}$ and F be a td-decomposition of $G_D(A)$. We say that A is *dual block-structured along F* if either $\mathrm{th}(F) = 1$, or if $\mathrm{th}(F) > 1$ and the following holds. Let v be the first non-degenerate vertex in F on a path from the root, $r_1, \ldots, r_d$ be the children of v, F_i be the subtree of F rooted in r_i, and $m_i := |V(F_i)|$, for $i \in [d]$, and

$$A = \begin{pmatrix} \bar{A}_1 & \bar{A}_2 & \cdots & \bar{A}_d \\ A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_d \end{pmatrix}. \qquad \text{(dual-block-structure)}$$

where $d \in \mathbb{N}$, and for all $i \in [d]$, $\bar{A}_i \in \mathbb{Z}^{k_1(F)\times n_i}$, and $A_i \in \mathbb{Z}^{m_i \times n^i}$, $n_i \in \mathbb{N}$, and A_i is block-structured along F_i. Note that $\mathrm{th}(F_i) \leq \mathrm{th}(F) - 1$, $\mathrm{height}(F_i) \leq \mathrm{height}(F) - k_1(F)$, for $i \in [d]$.

Corollary 25 (Dual decomposition). *Let $A \in \mathbb{Z}^{m\times n}$, $G_D(A)$, and a* td*-decomposition F of $G_D(A)$ be given, where $n, m \geq 1$. There exists an algorithm which in time $O(n)$ permutes the rows and columns of A such that the resulting matrix A' is dual block-structured along F.*

Again, the dual decomposition naturally partitions the right hand side $\mathbf{b} = (\mathbf{b}^0, \mathbf{b}^1, \ldots, \mathbf{b}^d)$ according to the rows of $\bar{A}_1, A_1, \ldots, A_d$, and each object of length n into d objects according to the columns of $A_1, \ldots, A_d$.

Lemma 26. *Let $A \in \mathbb{Z}^{n\times m}$, a* td*-decomposition F of $G_P(A)$ (or $G_D(A)$), and $\bar{A}_i, A_i, F_i$, for all $i \in [d]$, be as in Definitions 22 (or 24). Let $\hat{A}_i := (\bar{A}_i\ A_i)$ (or $\hat{A}_i := \begin{pmatrix} \bar{A}_i \\ A_i \end{pmatrix}$, respectively) and let $\hat{F}_i$ be obtained*

from F_i by appending a path on $k_1(F)$ new vertices to the root of F_i, and the other endpoint of the path is the new root. Then $\hat{F}_i$ is a td*-decomposition of $\hat{A}_i$,* $\mathrm{th}(\hat{F}_i) < \mathrm{th}(F)$*, and* $\mathrm{height}(\hat{F}_i) \leq \mathrm{height}(F)$.

Proof. Consider Figure 1b. The construction of $\hat{F}_i$ can be equivalently described as taking F and deleting all F_j, $j \neq i$. Thus, $\hat{F}_i$ has the claimed properties, in particular $\mathrm{th}(\hat{F}_i) < \mathrm{th}(F)$ because v was non-degenerate in F but is degenerate in $\hat{F}_i$. The dual case follows by transposition. □

3.4.1 Pedagogical Advice. The case of $\mathrm{th}(F) = 2$ is the simplest non-trivial case; in the primal case called "2-stage stochastic IP" and in the dual case called "n-fold IP". It turns out that usually whatever arguments work in this simplest case can be inductively extended to the general case. Thus it is usually helpful to first read the inductive proofs which follow with the assumption that the blocks $A_1, \ldots, A_d$ are small, and once the reader is confident with the arguments in this case, consider what needs to be done to generalize the argument.

Similarly, it might be helpful to first assume that the blocks $\bar{A}_1, \ldots, \bar{A}_d$ are identical, and that the blocks $A_1, \ldots, A_d$ are identical, and later check that this had essentially no impact on the argument and that it holds in general.

Finally, we strive to give quantitative bounds rather than just showing that some quantity is bounded by a parameter. But on a first reading, it may be helpful to simply convince oneself of the fact that a bound or an algorithm is indeed qualitatively as claimed, and not be distracted by expressions and computations which appear complicated. We choose to give the proofs already for the general case in order to avoid repetition.

3.5 Bounding The Norms

3.5.1 Norm of Primal Treedepth.

Lemma 27 (Primal Norm). *Let $A \in \mathbb{Z}^{m \times n}$, and F be a* td*-decomposition of $G_P(A)$. Then there exists a constant $\alpha \in \mathbb{N}$ such that*

$$g_\infty(A) \leq 2^{\overbrace{2^{\cdot^{\cdot^{\cdot^{2^{(2\|A\|_\infty)^{2^{\mathrm{th}(F)} \cdot \alpha \cdot \mathrm{td}_P(A)^2}}}}}}^{\mathrm{th}(F)-1}}$$

We will need a powerful lemma due to Klein [14].

Proposition 28 (Klein [14]). *Let $T_1, \ldots, T_n \subseteq \mathbb{Z}^d$ be multisets all belonging to one orthant where all elements $\mathbf{t} \in T_i$ have bounded size $\|\mathbf{t}\|_\infty \leq C$ and where*

$$\sum_{\mathbf{t} \in T_1} \mathbf{t} = \sum_{\mathbf{t} \in T_2} \mathbf{t} = \cdots = \sum_{\mathbf{t} \in T_n} \mathbf{t} \ .$$

Then there exists a constant $\alpha \in \mathbb{N}$ and non-empty submultisets $S_1 \subseteq T_1, \ldots, S_n \subseteq T_n$ of bounded size $|S_i| \leq (dC)^{dC^{\alpha d^2}}$ such that

$$\sum_{\mathbf{s} \in S_1} \mathbf{s} = \sum_{\mathbf{s} \in S_2} \mathbf{s} = \cdots = \sum_{\mathbf{s} \in S_n} \mathbf{s} \ .$$

Proof of Lemma 27. We will proceed by induction on $\mathrm{th}(F)$. In the base case when $\mathrm{th}(F) = 1$, $G_P(A)$ is a path and thus A has $\mathrm{td}_P(A)$ columns. Observe that the number of rows of A is bounded by $\mathrm{td}_P(A)$ as we assume purity. By Lemma 16 we then have that

$$g_\infty(A) \leq \mathrm{td}_P(A) g_1(A) \leq \mathrm{td}_P(A)(2\|A\|_\infty \mathrm{td}_P(A) + 1)^{\mathrm{td}_P(A)} \leq 2^{2\alpha \cdot \mathrm{td}_P(A)^2 + \log 2\|A\|_\infty} \ .$$

In the inductive step, we assume A is block-structured along F (otherwise apply Lemma 23). Let $\hat{A}_i = (\bar{A}_i \ A_i) \in \mathbb{Z}^{m_i \times k' + n_i}$ and $\hat{F}_i$ as in Lemma 26, and let $\hat{g}_\infty := \max_{i \in [d]} g_\infty(\hat{A}_i)$. Note that

$\mathrm{td}_P(\hat{A}_i) \le \mathrm{td}_P(A)$. Since $\hat{F}_i$ is a td-decomposition of $G_P(\hat{A}_i)$ and $\mathrm{th}(\hat{F}_i) < \mathrm{th}(F)$, we may apply induction on $\hat{A}_i$, showing

$$\hat{g}_\infty \le 2^{2^{\cdots^{2^{(2\|A\|_\infty)^{2^{\mathrm{th}(F)}\cdot\alpha\cdot\mathrm{td}_P(A)^2}}}}} \quad \text{(th}(F)-2 \text{ twos)} \tag{6}$$

Consider $\mathbf{g} = (\mathbf{g}^0, \mathbf{g}^1, \dots, \mathbf{g}^d) \in \mathcal{G}(A)$. For each $i \in [d]$, decompose $(\mathbf{g}^0, \mathbf{g}^i) = \sum_{j=1}^{N_i} (\mathbf{h}^0_j, \mathbf{h}^i_j)$ with $(\mathbf{h}^0_j, \mathbf{h}^i_j) \in \mathcal{G}(\hat{A}_i)$ by the Positive Sum Property (Proposition 6). Let $T_i := \left\{\mathbf{h}^0_j \;\middle|\; j \in [N_i]\right\}$ and observe that $\max_{\mathbf{t}\in T_i} \|\mathbf{t}\|_\infty \le g_\infty(\hat{A}_i) \le \hat{g}_\infty$. If applying Proposition 28 to $T_1, \dots, T_d$ yielded sets $S_1, \dots, S_d$ such that $S_i \subsetneq T_i$ for some $i \in [d]$ then $\mathbf{g}$ was not $\sqsubseteq$-minimal, a contradiction. Let $k_1 := k_1(F)$. Thus Proposition 28 implies, for each $i \in [d]$,

$$|T_i| \le (k_1\hat{g}_\infty)^{k_1\hat{g}_\infty^{\alpha k_1^2}} = 2^{2^{\alpha k_1^2 + \log(k_1\hat{g}_\infty) + \log\log(k_1\hat{g}_\infty)}} \le 2^{2^{2\alpha k_1^2 + \log\hat{g}_\infty}}$$

and $\|(\mathbf{g}^0, \mathbf{g}^i)\|_\infty \le \hat{g}_\infty |T_i|$, which in turn means that $\|\mathbf{g}\|_\infty \le \hat{g}_\infty \max_{i\in[d]} |T_i|$. Note that $2^{2^{2\alpha k_1^2 + \log\hat{g}_\infty}}\hat{g}_\infty \le 2^{2^{2\alpha k_1^2 + \log\hat{g}_\infty + \log\log\hat{g}_\infty}}$. To simplify, let $\zeta := 2\alpha k_1^2 + \log\hat{g}_\infty + \log\log\hat{g}_\infty$ so that the expression reads 2^{2^ζ}. Plugging in the bound (6) for $\hat{g}_\infty$ then gives

$$\zeta = 2\alpha k_1^2 + \log\hat{g}_\infty + \log\log\hat{g}_\infty \le 2\alpha k_1^2 + 2\log\hat{g}_\infty \le$$
$$2\alpha k_1^2 + 2\cdot 2^{2^{\cdots^{2^{(2\|A\|_\infty)^{2^{\mathrm{th}(F)-3}\cdot\alpha\cdot\mathrm{td}_P(A)^2}}}}} \quad \text{(th}(F)-3 \text{ twos)} \le 2^{2^{\cdots^{2^{(2\|A\|_\infty)^{2^{\mathrm{th}(F)}\cdot\alpha\cdot\mathrm{td}_P(A)^2}}}}} \quad \text{(th}(F)-3 \text{ twos)}, \quad \text{and thus,}$$

$$2^{2^\zeta} \le 2^{2^{\cdots^{2^{(2\|A\|_\infty)^{2^{\mathrm{th}(F)}\cdot\alpha\cdot\mathrm{td}_P(A)^2}}}}} \quad \text{(th}(F)-3 \text{ twos)} \le g_\infty(A) \le 2^{2^{\cdots^{2^{(2\|A\|_\infty)^{2^{\mathrm{th}(F)}\cdot\alpha\cdot\mathrm{td}_P(A)^2}}}}} \quad \text{(th}(F)-1 \text{ twos)}$$

□

3.5.2 *Norm of Dual Treedepth.*

Lemma 29 (Dual Norm). *Let $A \in \mathbb{Z}^{m\times n}$, F be a* td*-decomposition of $G_D(A)$, and let $K := \max_{P:\text{root-leaf path in } F} \prod_{i=1}^{\mathrm{th}(F)} (k_i(P)+1)$. Then $g_1(A) \le (3\|A\|_\infty K)^{K-1}$.*

Proof. The proof will proceed by induction over $\mathrm{th}(F)$. In the base case we have $\mathrm{th}(F) = 1$ and thus $G_D(A)$ is a path with height(F) vertices, meaning A has height(F) rows. Now we use the Base bound of Lemma 16 to get that $g_1(A) \le (2\|A\|_\infty \mathrm{height}(F)+1)^{\mathrm{td}_D(A)}$, which is at most $(3\|A\|_\infty K)^{K-1}$, where $K = \mathrm{height}(F)+1 = k_1(F)+1$. (Note that $k_1(F) = k_1(P)$ for all root-leaf paths P in F since all paths share an identical segment from the root to the first non-degenerate vertex.)

For the inductive step, assume that the claim holds for all trees of topological height less than $\mathrm{th}(F)$. Let $\mathbf{g} \in \mathcal{G}(A)$ and $K' := \max_{P:\text{root-leaf path in } F} \prod_{i=2}^{\mathrm{th}(F)} (k_i(P)+1)$. For each $i \in [d]$, $\mathbf{g}^i$ has a decomposition into elements $\mathbf{g}^i_j$ of $\mathcal{G}(A_i)$, and by induction we have $\|\mathbf{g}^i_j\|_1 \le g_1(A_i) \le (3\|A\|_\infty K')^{K'-1} =: \hat{g}_1$. Construct a sequence of vectors as follows: for each $i \in [d]$ and each $\mathbf{g}^i_j$ in the decomposition of $\mathbf{g}^i$, insert $\mathbf{v}^i_j := \bar{A}_i\mathbf{g}^i_j$ into the sequence. Note that $\|\mathbf{v}^i_j\|_\infty \le \|A\|_\infty\hat{g}_1$. Denote the resulting sequence $\mathbf{u}^1, \dots, \mathbf{u}^N$.

Applying the Steinitz Lemma (Proposition 15) to this sequence, we obtain its permutation $\mathbf{u}^{\pi(1)}, \dots, \mathbf{u}^{\pi(N)}$ such that the ℓ_∞-norm of each of its prefix sums is bounded by $k_1(F)\|A\|_\infty\hat{g}_1$. As in the proof of Lemma 16, we will prove that no two prefix sums are the same, thus $N \le$

$(2k_1(F)\|A\|_\infty\hat{g}_1+1)^{k_1(F)}$ and subsequently $\|\mathbf{g}\|_1 \leq N\hat{g}_1 \leq \hat{g}_1(2k_1(F)\|A\|_\infty\hat{g}_1+1)^{k_1(F)}$. Plugging in $\hat{g}_1 = (3\|A\|_\infty K')^{K'-1} \leq (3\|A\|_\infty K)^{K'-1}$ and simplifying yields

$$\|\mathbf{g}\|_1 \leq (3\|A\|_\infty K)^{K'-1}\cdot(3\|A\|_\infty K)^{k_1(F)K'} = (3\|A\|_\infty K)^{K-1} \ .$$

Assume to the contrary that some two prefix sums $\mathbf{p}_\alpha$ and $\mathbf{p}_\beta$, for $\alpha<\beta$, are identical. Then the sequence $\mathbf{u}^{\alpha+1},\ldots,\mathbf{u}^\beta$ sums up to zero and we may "work backward" from it to obtain an integer vector $\bar{\mathbf{g}} \sqsubset \mathbf{g}$, which is a contradiction to $\mathbf{g}\in\mathcal{G}(A)$. Specifically, $\bar{\mathbf{g}}$ can be obtained by initially setting $\bar{\mathbf{g}}=\mathbf{0}$ and then, for each $\gamma\in[\alpha+1,\beta]$, if $\pi^{-1}(\gamma)=(i,j)$, setting $\bar{\mathbf{g}}^i := \bar{\mathbf{g}}^i+\mathbf{g}^i_j$. □

Remark. Our definition of K allows us to recover the currently best known upper bounds on $g_1(A)$ from Lemma 29. Specifically, Knop et al. [16, Lemma 10] show that $g_1(A) \leq (2\|A\|_\infty+1)^{2^{\mathrm{td}_D(A)}-1}$. This pertains to the worst case when $\mathrm{th}(F)=\mathrm{height}(F)=\mathrm{td}_D(A)$. Then, we have $K=\prod_{i=1}^{\mathrm{th}(F)}(k_i(P)+1)=2^{\mathrm{td}_D(A)}$ and our bound essentially matches theirs. On the other hand, our bound is better in scenarios when $\mathrm{th}(F)<\mathrm{height}(F)$ and K is attained by some path with $k_i(P)>1$ for some $i\in\mathrm{th}(F)$. A particular example of this are N-fold and tree-fold matrices.

3.6 Solving Augmentation IP

3.6.1 *Primal Treedepth.*

Lemma 30 (Primal Lemma). *Problem* (AugIP) *can be solved in time* $\mathrm{td}_P(A)^2(2g_\infty(A)+1)^{\mathrm{td}_P(A)}n$.

Proof. Let F be an optimal td-decomposition of $G_P(A)$. The proof proceeds by induction on $\mathrm{th}(F)\leq\mathrm{td}_P(A)$. For that, we prove a slightly more general claim:

Claim. Given $\rho\in\mathbb{N}$, there is an algorithm running in time $\mathrm{td}_P(A)^2(2\rho+1)^{\mathrm{td}_P(A)}n$ which solves

$$B_\infty(\rho)\text{-best}\{f(\mathbf{g}) \mid A\mathbf{g}=\mathbf{b},\ \mathbf{l}\leq\mathbf{g}\leq\mathbf{u},\ \mathbf{g}\in\mathbb{Z}^n\}$$

for any separable-convex function f.

The statement of the Lemma is obtained by the following substitution. For a given (AugIP) instance $(\mathbf{x},\lambda)$, solve the auxiliary problem above with $\rho := g_\infty(A)$, $f(\mathbf{g}) := f(\mathbf{x}+\lambda\mathbf{g})$, $\mathbf{b}:=\mathbf{0}$, $\mathbf{l}:=\left\lceil\frac{\mathbf{l}-\mathbf{x}}{\lambda}\right\rceil$, and $\mathbf{u}:=\left\lfloor\frac{\mathbf{u}-\mathbf{x}}{\lambda}\right\rfloor$. If f of (IP) was separable convex, then the newly defined f is also separable convex. The returned solution is a solution of (AugIP) because $\mathcal{G}(A)\subseteq B_\infty(g_\infty(A))$.

As the base case, if $\mathrm{th}(F)=1$, then F is a path, meaning that A has $\mathrm{td}_P(A)$ columns. An optimal solution is found simply by enumerating all $(2\rho+1)^{\mathrm{td}_P(A)}$ integer vectors $\mathbf{g}\in[-\rho,\rho]^{\mathrm{td}_P(A)}\cap[\mathbf{l},\mathbf{u}]$, for each checking $A\mathbf{g}=\mathbf{b}$ and evaluating f, and returning the best feasible one. Since the number of rows of A is at most its number of columns, which is $\mathrm{td}_P(A)$, checking whether $A\mathbf{g}=\mathbf{b}$ takes time at most $\mathrm{td}_P(A)^2$ for each $\mathbf{g}$.

As the induction step, we assume A is block-structured along F (otherwise apply Lemma 23), hence we have matrices $\bar{A}_1,\ldots,\bar{A}_d,A_1,\ldots,A_d$ for some d and td-decompositions $F_1,\ldots,F_d$ for $G_P(A_1),\ldots,G_P(A_d)$, respectively, with, for each $i\in[d]$, $\bar{A}_i$ having $k_1(F)$ columns, F_i having $\mathrm{th}(F_i)<\mathrm{th}(F)$, and $\mathrm{td}_P(A_i)\leq\mathrm{td}_P(A)-k_1(F)$. Now iterate over all vectors $\mathbf{g}^0\in\mathbb{Z}^{k_1(F)}$ in $[-\rho,\rho]^{k_1(F)}\cap[\mathbf{l}^0,\mathbf{u}^0]$ and for each use the algorithm which exists by induction to compute d vectors $\mathbf{g}^i$, $i\in[d]$, such that $\mathbf{g}^i$ is a solution to

$$B_\infty(\rho)\text{-best}\{f(\mathbf{g}^i) \mid A_i\mathbf{g}^i=-\bar{A}_i\mathbf{g}^0+\mathbf{b}^i,\ \mathbf{l}^i\leq\mathbf{g}^i\leq\mathbf{u}^i,\ \mathbf{g}^i\in\mathbb{Z}^{n_i}\} \ . \tag{7}$$

Finally return the vector $(\mathbf{g}^0,\ldots,\mathbf{g}^d)$ which minimizes $\sum_{i=0}^d f(\mathbf{g}^i)$. If $\mathbf{g}^i$ is undefined for some $i\in[d]$ because the subproblem (7) has no solution, report that the problem has no solution.

Let $k:=\mathrm{td}_P(A)-k_1(F)$. There are $(2\rho+1)^{k_1(F)}$ choices of $\mathbf{g}^0$, and computing the solution $(\mathbf{g}^1,\ldots,\mathbf{g}^d)$ for each takes time at most $\sum_{i=1}^d k^2(2\rho+1)^k n_i = k^2(2\rho+1)^k n$. For each choice we also

need to compute the product $-\bar{A}_i \mathbf{g}^0$, which is possible in time $k_1(F) \cdot \mathrm{td}_P(A)$ since the number of rows of $\bar{A}_i$ is at most $\mathrm{td}_P(A)$. The total time needed is thus $(2\rho+1)^{k_1(F)} \cdot \left(\mathrm{td}_P(A) \cdot k_1(F) + k^2(2\rho+1)^k\right) n \leq \mathrm{td}_P(A)^2 (2\rho+1)^{\mathrm{td}_P(A)} n$. □

Lemma 31 (Dual Lemma). *Problem* (AugIP) *can be solved in time* $(2\|A\|_\infty g_1(A) + 1)^{O(\mathrm{td}_D(A))} n$.

Proof. We solve an auxiliary problem analogous to the one in Lemma 30: given $\rho \in \mathbb{N}$ and a separable convex function f, solve

$$B_1(\rho)\text{-best}\{f(\mathbf{g}) \mid A\mathbf{g} = \mathbf{b},\ \mathbf{l} \leq \mathbf{g} \leq \mathbf{u},\ \mathbf{g} \in \mathbb{Z}^n\} \ .$$

The lemma then follows by the same substitution described at the beginning of the proof of Lemma 30. We assume that $\|\mathbf{b}\|_\infty \leq \rho\|A\|_\infty$ since otherwise there is no solution within $B_1(\rho)$.

Let F be an optimal td-decomposition of $G_D(A)$. We define the algorithm recursively over $\mathrm{th}(F)$. If $\mathrm{th}(F) \geq 2$, we assume A is dual block-structured along F (otherwise apply Corollary 25) and we have, for every $i \in [d]$, matrices $A_i, \bar{A}_i, \hat{A}_i$ and a tree $\hat{F}_i$ (see Lemma 26) with the claimed properties, and a corresponding partitioning of $\mathbf{b}, \mathbf{l}, \mathbf{u}, \mathbf{g}$ and f. If $\mathrm{th}(F) = 1$, let $d := n$ and $\hat{A}_i := A_{\bullet,i}$, for all $i \in [d]$, be the columns of A, and let $\mathbf{b}^1, \ldots, \mathbf{b}^n$ be empty vectors (i.e., vectors of dimension zero).

The crucial observation is that for every solution $\mathbf{g}$ of $A\mathbf{g} = \mathbf{b}$ with $\|\mathbf{g}\|_1 \leq \rho$ and each $i \in [d]$, both $\bar{A}_i \mathbf{g}^i$ and $\sum_{j=1}^{i} \bar{A}_j \mathbf{g}^j$ belong to $R := [-\rho\|A\|_\infty, \rho\|A\|_\infty]^{k_1(F)}$. For every $i \in [d]$ and every $\mathbf{r} \in R$, solve

$$B_1(\rho)\text{-best}\{f^i(\mathbf{g}^i) \mid \hat{A}_i \mathbf{g}^i = (\mathbf{r}, \mathbf{b}^i),\ \mathbf{l}^i \leq \mathbf{g}^i \leq \mathbf{u}^i,\ \mathbf{g}^i \in \mathbb{Z}^{n_i}\} \ . \tag{8}$$

In the base case when $\hat{A}_i$ has only one column, we simply enumerate all $\mathbf{g}^i \in [\mathbf{l}^i, \mathbf{u}^i] \cap [-\rho, \rho]$, check whether the equality constraints are satisfied, and return the best feasible choice. If $\mathrm{th}(F) > 1$, then we use recursion to solve (8). The recursive call is well-defined, since, for all $i \in [d]$, $\mathrm{th}(\hat{F}_i) < \mathrm{th}(F)$ and $\hat{F}_i$ is a td-decomposition of $G_D(\hat{A}_i)$. Next, we show how to "glue" these solutions together.

Let $\mathbf{r} \in R$ and denote by $\mathbf{g}^i_{\mathbf{r}}$ a solution to the subproblem (8); by slight abuse of notation, when the subproblem has no solution, we define $f^i(\mathbf{g}^i_{\mathbf{r}}) := +\infty$. Now we need to find such $\mathbf{r}_1, \ldots, \mathbf{r}_d \in R$ that $\sum_{i=1}^{d} \mathbf{r}_i = \mathbf{b}^0$ and $\sum_{i=1}^{d} f^i(\mathbf{g}^i_{\mathbf{r}_i})$ is minimized. This is actually a form of the $(\min, +)$-convolution problem, a fact which we will use later. For now it suffices to say that this problem can be easily solved using dynamic programming in d stages: our DP table D shall have an entry $D(i, \mathbf{r})$ for $i \in [d]$ and $\mathbf{r} \in R$ whose meaning is the minimum $\sum_{j=1}^{i} f^j(\mathbf{g}^j_{\mathbf{r}_j})$ where $\sum_{j=1}^{i} \mathbf{r}_j = \mathbf{r}$. To compute D, set $D(0, \mathbf{r}) := 0$ for $\mathbf{r} = \mathbf{0}$ and $D(0, \mathbf{r}) := +\infty$ otherwise, and for $i \in [d]$, set

$$D(i, \mathbf{r}) := \min_{\substack{\mathbf{r}', \mathbf{r}'' \in R: \\ \mathbf{r}' + \mathbf{r}'' = \mathbf{r}}} D(i-1, \mathbf{r}') + f^i(\mathbf{g}^i_{\mathbf{r}''}) \ .$$

The value of the solution is $D(d, \mathbf{b}^0)$ and the solution $\mathbf{g} = (\mathbf{g}^1, \ldots, \mathbf{g}^d)$ itself can be computed easily with a bit more bookkeeping in the table D. If $D(d, \mathbf{b}^0) = +\infty$, report that the problem has no solution. Another important observation is this: in the DP above we computed the solution of the auxiliary problem not only for the right hand side $\mathbf{b}$, but for *all* right hand sides of the form $(\mathbf{r}, \mathbf{b}^1, \ldots, \mathbf{b}^d)$ where $\mathbf{r} \in R$ and $\mathbf{b}^1, \ldots, \mathbf{b}^d$ are fixed. We store all of these intermediate results in an array (an approach also known as "memoization"). When the algorithm asks for solutions of such instances, we simply retrieve them from the array of intermediate results instead of recomputing them. This is important for the complexity analysis we will describe now.

The recursion tree has $\mathrm{th}(F)$ levels, which we number $1, \ldots, \mathrm{th}(F)$, with level 1 being the base of the recursion. Let us compute the time required at each level. In the base case $\mathrm{th}(F) = 1$, recall that the matrix $\hat{A}_i$ in subproblem (8) is a single column with height(F) rows, and solving (8) amounts to trying at most $2\rho + 1$ feasible valuations of $\mathbf{g}^i$ (which is a scalar variable) satisfying $\mathbf{l}^i \leq \mathbf{g}^i \leq \mathbf{u}^i$ and returning the best feasible one. Since there are n columns in total, computing the solutions of (8)

takes time $(2\rho+1)n$. Let N_1 be the number of leaves of F, and let α_j, $j \in [N_1]$, denote the number of columns corresponding to the j-th leaf. "Gluing" the solutions is done by solving N_1 DP instances with $\alpha_1, \ldots, \alpha_{N_1}$ stages, where $\sum_{i=1}^{N_1} \alpha_i = n$. This takes time $\sum_{i=1}^{N_1} |R|^2 \cdot \alpha_i \leq (2\|A\|_\infty\rho+1)^{\mathrm{td}_D(A)} \cdot n$, since a td-decomposition of each column is a path on $\mathrm{td}_D(A)$ vertices. In total, computing the first level of recursion takes time $(2\|A\|_\infty\rho+1)^{\mathrm{td}_D(A)}n$.

Consider a recursion level $\ell \in [2, \mathrm{th}(F)]$ and subproblem (8). The crucial observation is that when the algorithm asks for the answer to (8) for one specific $\mathbf{r}' \in R$, an answer for *all* $\mathbf{r} \in R$ is computed; recall that the last step of the DP is to return $D(d, \mathbf{r}')$ but the table contains an entry $D(d, \mathbf{r})$ for all $\mathbf{r} \in R$. Thus the time needed for the computation of all $\mathbf{g}^i_{\mathbf{r}}$ has been accounted for in lower levels of the recursion and we only have to account for the DP at the level ℓ. Let R' be the analogue of R for a specific subproblem at level ℓ, and let A' be the corresponding submatrix of A and $F' \subseteq F$ be a td-decomposition of $G_D(A')$. We have that $|R'| \leq (2\|A\|_\infty\rho+1)^{k_1(F')}$, with $k_1(F') \leq \mathrm{td}_D(A)$. Note that the levels here are defined bottom up, hence all leaves are at level 1, and an inner node of F is at level ℓ if $\ell-1$ is the largest level of its children; in particular a level does *not* correspond to the distance from the root. Let N_ℓ be the number of vertices of F at level ℓ. The number of subproblems on level ℓ is exactly N_ℓ, so computation of the ℓ-th level takes time at most $|R|^2 \cdot N_\ell \leq (2\|A\|_\infty\rho+1)^{\mathrm{td}_D(A)}N_\ell$. Adding up across all levels we get that the total complexity is at most $\left(n + \sum_{\ell=2}^{\mathrm{th}(F)} N_\ell\right) \cdot (2\|A\|_\infty\rho+1)^{\mathrm{td}_D(A)}$ where $\sum_{\ell=2}^{\mathrm{th}(F)} N_\ell < n$ since F has n leaves and each level corresponds to a vertex with degree at least 2. The lemma follows. □

3.7 The Proof

Proof of Theorem 3. We run two algorithms in parallel, terminate when one of them terminates, and return its result. In the *primal* algorithm, let $G(A) = G_P(A)$, $\mathrm{td}(A) = \mathrm{td}_P(A)$ and $p = \infty$. In the *dual algorithm*, let $G(A) = G_D(A)$, $\mathrm{td}(A) = \mathrm{td}_D(A)$ and $p = 1$. The description of both algorithms is then identical.

First, run the algorithm of Proposition 20 on $G(A)$ to obtain its optimal td-decomposition. By Lemmas 27 and 29 there is a computable function g' such the maximum ℓ_p-norm of elements of $\mathcal{G}(A)$ is bounded by $g'(\|A\|_\infty, \mathrm{td}(A))$. By Lemmas 30 and 31, there is a computable function g'' such that (AugIP) is solvable in time $g''(g'(\mathrm{td}(A), \|A\|_p), \|A\|_p, \mathrm{td}(A))$ and thus in time $g(\|A\|_p, \mathrm{td}(A))$ for some computable function g. Then, solve (IP) using the algorithm of Corollary 14 in the claimed time. □

4 STRONGLY POLYNOMIAL ALGORITHMS

Parametric, arithmetic, numeric input. To clearly state our results and compare them to previous work we introduce the following terminology. The input to a problem will be partitioned into three parts (α, β, γ), where α is the *parametric input*, β is the *arithmetic input*, and γ is the *numeric input*. The *time* an algorithm takes to solve a problem is the number of arithmetic operations and oracle queries, and all numbers involved in the computation are required to have length polynomial in (β, γ). A *polynomial algorithm* for the problem is one that solves it in time $\mathrm{poly}(\beta, \gamma)$, while a *strongly-polynomial algorithm* solves it in time $\mathrm{poly}(\beta)$, i.e., independent of the numeric input. Similarly, a *fixed-parameter tractable* (FPT) *algorithm* solves the problem in time $g(\alpha)\mathrm{poly}(\beta, \gamma)$, while a *strongly* FPT *algorithm* solves it in time $g(\alpha)\mathrm{poly}(\beta)$, where g is a computable function. If such an algorithm exists, we say that the problem is *(strongly) fixed-parameter tractable* (FPT) *parameterized by* α. Having multiple parameters $\alpha_1, \ldots, \alpha_k$ simultaneously is understood as taking the *aggregate parameter* $\alpha = \alpha_1 + \cdots + \alpha_k$. When we want to highlight the fact that an oracle is involved (e.g., when the oracle calls are expected to take a substantial portion of the time), we say that the algorithm works in certain (polynomial, strongly polynomial, FPT, etc.) *oracle time*. Each

part of the input may have several entities, which may be presented in unary or binary. For the parametric input the distinction between unary and binary is irrelevant, but it defines the function g.

We will use the notion of centering an instance at a vector $\mathbf{v} \in \mathbb{Z}^n$. For an (IP) instance $\mathcal{I}$, let $\mathrm{Sol}(\mathcal{I}) := \{\mathbf{x} \in \mathbb{Z}^n \mid A\mathbf{x} = \mathbf{b},\ \mathbf{l} \le \mathbf{x} \le \mathbf{u}\}$ denote the set of feasible solutions of $\mathcal{I}$. The next lemma follows by a simple translation, hence we omit its proof.

Lemma 32 (Equivalent centered instance). *Let an (IP) instance $\mathcal{I}$ be given, and $\mathbf{v} \in \mathbb{Z}^n$. Define an (IP) instance $\bar{\mathcal{I}} = (A, \bar{f}, \bar{\mathbf{b}}, \bar{\mathbf{l}}, \bar{\mathbf{u}})$ by*

$$\bar{\mathbf{b}} := \mathbf{b} - A\mathbf{v},\ \bar{\mathbf{l}} := \mathbf{l} - \mathbf{v},\ \bar{\mathbf{u}} := \mathbf{u} - \mathbf{v},\ \bar{f}(\mathbf{x}) := f(\mathbf{x} - \mathbf{v}) \ .$$

The translation $\tau(\mathbf{x}) = \mathbf{x} + \mathbf{v}$ is a bijection from $\mathrm{Sol}(\bar{\mathcal{I}})$ to $\mathrm{Sol}(\mathcal{I})$. Moreover, $\mathbf{x}$ is an optimal solution of $\bar{\mathcal{I}}$ if and only if $\tau(\mathbf{x})$ is an optimal solution of $\mathcal{I}$. If $\mathbf{v} \in \mathrm{Sol}(\mathcal{I})$, then $\bar{\mathbf{b}} = \mathbf{0}$, and $\mathbf{0}$ is feasible for $\bar{\mathcal{I}}$.

We say that the translated instance $\bar{\mathcal{I}}$ was obtained from $\mathcal{I}$ by *centering it at* $\mathbf{v}$. When beneficial, we will move to the centered instance $\bar{\mathcal{I}}$, recovering an optimum of $\mathcal{I}$ eventually.

Theorem 33. *Problem (ILP) with arithmetic input $\langle A \rangle$ and numeric input $\langle \mathbf{w}, \mathbf{b}, \mathbf{l}, \mathbf{u} \rangle$, endowed with an oracle solving (AugIP), is solvable in strongly polynomial oracle time.*

Note that the partition of the input to the arithmetic input $\langle A \rangle$ and the numeric input $\langle \mathbf{w}, \mathbf{b}, \mathbf{l}, \mathbf{u} \rangle$ is the same as in the classical results for linear programming [8, 23]. Together with the algorithms we showed for solving (AugIP), Theorem 33 immediately yields that:

Corollary 34. *There exists a computable function g such that problem (ILP) can be solved in time*

$$g(a, d)\mathrm{poly}(n), \qquad \text{where } d := \min\{\mathrm{td}_P(A), \mathrm{td}_D(A)\} \ .$$

Let $\mathcal{C}(A)$ be the set of *circuits* of A, which are those $\mathbf{c} \in \ker_{\mathbb{Z}}(A)$ whose support is a circuit of the linear matroid of A and whose entries are coprime. Let $c_\infty(A) := \max_{\mathbf{c} \in \mathcal{C}(A)} \|\mathbf{c}\|_\infty$. It is known that $\mathcal{C}(A) \subseteq \mathcal{G}(A)$ [18, Definition 3.1] and thus $c_\infty(A) \le g_\infty(A)$.

Proposition 35 (Onn [18, Lemma 2.17]). *For any $\mathbf{x} \in \ker(A)$, $\mathbf{x}$ may be written as $\sum_{i=1}^{n'} \lambda_i \mathbf{g}_i$ where $n' \le n - r$ with $r := \mathrm{rank}(A)$, and for all $i \in [n']$, $\lambda_i > 0$, $\mathbf{g}_i \in \mathcal{C}(A)$, and $\lambda_i \mathbf{g}_i \sqsubseteq \mathbf{x}$, i.e., the sum is sign-compatible.*

Proof of Theorem 33. The algorithm which demonstrates the theorem has several steps.

Step 1: Relaxation oracle and proximity bound (i.e., reducing $\mathbf{b}, \mathbf{l}, \mathbf{u}$.) Apply the strongly polynomial algorithm of Tardos [23] to the linear programming relaxation $\min\{\mathbf{wy} \mid \mathbf{y} \in \mathbb{R}^n,\ A\mathbf{y} = \mathbf{b},\ \mathbf{l} \le \mathbf{y} \le \mathbf{u}\}$; the algorithm performs $\mathrm{poly}(\langle A \rangle)$ arithmetic operations. If the relaxation is infeasible then so is (ILP) and we are done. If it is unbounded then (ILP) is either infeasible or unbounded too, and in this case we set $\mathbf{w} := \mathbf{0}$ so that all solutions are optimal, and we proceed as below and terminate at the end of step 3. Suppose then that we obtain an optimal solution $\mathbf{y}^* \in \mathbb{R}^n$ to the relaxation, with round down $\lfloor \mathbf{y}^* \rfloor \in \mathbb{Z}^n$. By Lemma 16 we have $c_\infty(A) \le g_\infty(A) \le g_1(A) \le (2m\|A\|_\infty + 1)^m$.

We now use the proximity results of [12, 13] (see Theorem 40) which assert that either (ILP) is infeasible or it has an optimal solution $\mathbf{x}^*$ with $\|\mathbf{x}^* - \mathbf{y}^*\|_\infty \le nc_\infty(A)$ and hence $\|\mathbf{x}^* - \lfloor \mathbf{y}^* \rfloor\|_\infty < n(2m\|A\|_\infty + 1)^m + 1$, where the "+1" is due to the round-down of $\mathbf{y}^*$. Since both sides are integers, we have $\|\mathbf{x}^* - \lfloor \mathbf{y}^* \rfloor\|_\infty \le n(2m\|A\|_\infty + 1)^m$. Thus, making the variable transformation $\mathbf{x} = \mathbf{z} + \lfloor \mathbf{y}^* \rfloor$ (i.e., recentering the instance at $\lfloor \mathbf{y}^* \rfloor$), problem (ILP) reduces to the following, FIX

$$\min\ \{\mathbf{w}(\mathbf{z} + \lfloor \mathbf{y}^* \rfloor) \mid \mathbf{z} \in \mathbb{Z}^n,\ A(\mathbf{z} + \lfloor \mathbf{y}^* \rfloor) = \mathbf{b},\ \mathbf{l} \le \mathbf{z} + \lfloor \mathbf{y}^* \rfloor \le \mathbf{u},\ \|\mathbf{z}\|_\infty \le n(2m\|A\|_\infty + 1)^m\},$$

which is equivalent to the program

$$\min\ \left\{\mathbf{wz} \mid \mathbf{z} \in \mathbb{Z}^n,\ A\mathbf{z} = \bar{\mathbf{b}},\ \bar{\mathbf{l}} \le \mathbf{z} \le \bar{\mathbf{u}}\right\}, \quad \text{where} \tag{9}$$

$\bar{\mathbf{b}} := \mathbf{b} - A\lfloor \mathbf{y}^* \rfloor$, $\bar{l}_i := \max\left\{l_i - \lfloor y_i^* \rfloor, -n(2m\|A\|_\infty + 1)^m\right\}$, $\bar{u}_i := \min\left\{u_i - \lfloor y_i^* \rfloor, n(2m\|A\|_\infty + 1)^m\right\}$.

If some $\bar{l}_i > \bar{u}_i$ then (9) is infeasible and hence so is (ILP), so we may assume that

$$-n(2m\|A\|_\infty + 1)^m \leq \bar{l}_i \leq \bar{u}_i \leq n(2m\|A\|_\infty + 1)^m, \quad \text{for all } i \in [n] \ .$$

This implies that for every point $\mathbf{z}$ feasible for (9), $\|A\mathbf{z}\|_\infty \leq n^2\|A\|_\infty(2m\|A\|_\infty + 1)^m$ holds and so we may assume that $\|\bar{\mathbf{b}}\|_\infty \leq n^2\|A\|_\infty(2m\|A\|_\infty + 1)^m$ else there is no feasible solution. By $m \leq n$ (see Proposition 2) we have

$$\|\bar{\mathbf{b}}, \bar{\mathbf{l}}, \bar{\mathbf{u}}\|_\infty \leq 2^{O(n \log n)}\|A\|_\infty^{O(n)} \text{ and hence } \langle \bar{\mathbf{b}}, \bar{\mathbf{l}}, \bar{\mathbf{u}} \rangle \text{ is polynomial in } \langle A \rangle \ .$$

Step 2: Feasibility oracle. The next step is to find an integer solution to the system of equations $A\mathbf{z} = \bar{\mathbf{b}}$, and then to use this solution in an auxiliary problem with relaxed bounds to find an initial feasible solution to (9). This is exactly the purpose of Lemma 13. Crucially, its bound on the number of calls to an (AugIP) oracle and the time to compute an integral solution of $A\mathbf{z} = \bar{\mathbf{b}}$ only depends on $\langle A, \bar{\mathbf{b}}, \bar{\mathbf{l}}, \bar{\mathbf{u}} \rangle$ and *not* on the objective function $\mathbf{w}$.

Step 3: Reducibility bound (i.e., reducing $\mathbf{w}$). Let $N := 2n(2m\|A\|_\infty + 1)^m$. Now apply the strongly polynomial algorithm of Frank and Tardos [8], which on arithmetic input $n, \langle N \rangle$ and numeric input $\langle \mathbf{w} \rangle$, outputs $\bar{\mathbf{w}} \in \mathbb{Z}^n$ with $\|\bar{\mathbf{w}}\|_\infty \leq 2^{O(n^3)}N^{O(n^2)}$ such that $\operatorname{sign}(\mathbf{w}\mathbf{x}) = \operatorname{sign}(\bar{\mathbf{w}}\mathbf{x})$ for all $\mathbf{x} \in \mathbb{Z}^n$ with $\|\mathbf{x}\|_1 < N$. Since $\langle N \rangle = 1 + \lceil \log N \rceil = O(n \log n + n \log \|A\|_\infty)$ is polynomial in $\langle A \rangle$, this algorithm is also strongly polynomial in our original input. Now, for every two points $\mathbf{x}, \mathbf{z}$ feasible in (9) we have $\|\mathbf{x} - \mathbf{z}\|_1 < 2n(2m\|A\|_\infty + 1)^m = N$, so that for any two such points we have $\mathbf{w}\mathbf{x} \leq \mathbf{w}\mathbf{z}$ if and only if $\bar{\mathbf{w}}\mathbf{x} \leq \bar{\mathbf{w}}\mathbf{z}$, and therefore we can replace (9) by the equivalent program

$$\min\left\{\bar{\mathbf{w}}\mathbf{z} \mid \mathbf{z} \in \mathbb{Z}^n, A\mathbf{z} = \bar{\mathbf{b}}, \bar{\mathbf{l}} \leq \mathbf{z} \leq \bar{\mathbf{u}}\right\}, \quad \text{where} \tag{10}$$

$$\|\bar{\mathbf{w}}\|_\infty = 2^{O(n^3 \log n)}\|A\|_\infty^{O(n^3)} \text{ and hence } \langle \bar{\mathbf{w}}, \bar{\mathbf{b}}, \bar{\mathbf{l}}, \bar{\mathbf{u}} \rangle \text{ is polynomial in } \langle A \rangle.$$

Step 4: Augmentation oracle. Starting from the point $\mathbf{z}$ which is feasible in (10) and using the (AugIP) oracle, we can solve program (10) using Lemma 12 in polynomial time and in a number of arithmetic operations and oracle queries which is polynomial in n and in $\log(f_{\max})$, which is bounded by $\log\left(n\|\bar{\mathbf{w}}\|_\infty\|\bar{\mathbf{u}} - \bar{\mathbf{l}}\|_\infty\right)$, which is polynomial in $\langle A \rangle$, and hence strongly polynomially. □

5 PROXIMITY, SENSITIVITY, SCALING

We consider the following fractional relaxation of (IP).

$$\min\{f(\mathbf{x}) \mid A\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{R}^n\} \ . \tag{P}$$

In the context of non-linear functions we run into the possibility of irrational optima. Hochbaum and Shanthikumar [13, Section 1.2] argue in favor of the notion of an *ϵ-accurate* optimum, which is a solution of (P) close to some optimum in terms of distance, not necessarily objective value. Moreover, they show that under reasonable assumptions on the objective such an optimum is also close in terms of objective value.

Definition 36 (ϵ-accuracy [13]). Let $\mathbf{x}_\epsilon$ be a feasible solution of (P). We say that $\mathbf{x}_\epsilon$ is an *ϵ-accurate solution* if there exists an optimum $\mathbf{x}^*$ of (P) with $\|\mathbf{x}^* - \mathbf{x}_\epsilon\|_\infty \leq \epsilon$.

Proximity bounds relate an optimum of (P) (or an ϵ-accurate solution) to an optimum of (IP), for example in order to reduce the bounds $\mathbf{l}, \mathbf{u}$ and subsequently the right hand side $\mathbf{b}$. We will use a general definition of proximity from Cslovjecsek et al. [2] which is independent of the objective function, and which gives proximity guarantees for linear and separable convex functions (Lemma 38 below):

Definition 37 (Conformal proximity bound). Let $1 \leq p \leq +\infty$. We say that $\mathcal{P}_p(A) \geq 0$ is the *conformal ℓ_p-proximity bound of A* if it is the infimum of reals $\rho \geq 0$ satisfying the following: for every IP of the form (IP) with A as the constraint matrix, for every fractional solution $\mathbf{x}$ of (P) and every integer solution $\mathbf{z}$ of (IP), there is an integer solution $\mathbf{z}'$ of (IP) such that

$$\|\mathbf{x} - \mathbf{z}'\|_p \leq \rho \qquad \text{and} \qquad \mathbf{z}' \sqsubseteq \mathbf{x} - \mathbf{z} \ .$$

The condition that $\mathbf{z}' \sqsubseteq \mathbf{x} - \mathbf{z}$ is equivalent to saying that $\mathbf{z}'$ is contained in an axis-parallel box spanned by $\mathbf{x}$ and $\mathbf{z}$. Let us show that this implies proximity of fractional and integer optima for IPs with separable convex objectives:

Lemma 38. *Let $\hat{\mathbf{x}}$ be an optimum of (P) and $\hat{\mathbf{z}}$ be an optimum of (IP). There exist $\mathbf{x}^* \in \mathbb{R}^n$ and $\mathbf{z}^* \in \mathbb{Z}^n$ optima of (P) and (IP), respectively, such that*

$$\|\hat{\mathbf{x}} - \mathbf{z}^*\|_p = \|\mathbf{x}^* - \hat{\mathbf{z}}\|_p \leq \mathcal{P}_p(A) \ .$$

We will use a straight-forward proposition which follows from Proposition 8:

Proposition 39. *Let $\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^n$, $\mathbf{y}_1, \mathbf{y}_2$ be from the same orthant, and f be a separable convex function. Then*

$$f(\mathbf{x} + \mathbf{y}_1 + \mathbf{y}_2) - f(\mathbf{x} + \mathbf{y}_1) \geq f(\mathbf{x} + \mathbf{y}_2) - f(\mathbf{x}) \ .$$

Proof. Apply Proposition 8 with $\mathbf{x} := \mathbf{x}$, $\mathbf{g}_1 := \mathbf{y}_1$, $\mathbf{g}_2 := \mathbf{y}_2$, and $\lambda_1, \lambda_2 := 1$, to get

$$f(\mathbf{x}+\mathbf{y}_1+\mathbf{y}_2) - f(\mathbf{x}) \geq \big(f(\mathbf{x}+\mathbf{y}_1) - f(\mathbf{x})\big) + \big(f(\mathbf{x}+\mathbf{y}_2) - f(\mathbf{x})\big) = f(\mathbf{x}+\mathbf{y}_1) + f(\mathbf{x}+\mathbf{y}_2) - 2f(\mathbf{x}) \ .$$

Adding $2f(\mathbf{x})$ to both sides and rearranging then yields the statement. □

Proof. By Definition 37, there exists a $\mathbf{z}^* \in \mathbb{Z}^n$ with $\mathbf{z}^* \sqsubseteq \hat{\mathbf{x}} - \hat{\mathbf{z}}$ and $\|\mathbf{z}^* - \hat{\mathbf{x}}\|_p \leq \mathcal{P}_p(A)$. Let $\mathbf{g} = \hat{\mathbf{x}} - \mathbf{z}^*$ and $\bar{\mathbf{g}} = \mathbf{z}^* - \hat{\mathbf{z}}$ and note that $\hat{\mathbf{x}} = \hat{\mathbf{z}} + \mathbf{g} + \bar{\mathbf{g}}$. Now define

$$\mathbf{x}^* := \hat{\mathbf{z}} + \mathbf{g}, \qquad\qquad \mathbf{z}^* := \hat{\mathbf{z}} + \bar{\mathbf{g}} = \hat{\mathbf{x}} - \mathbf{g} \ .$$

By the fact that both $\hat{\mathbf{z}}$ and $\hat{\mathbf{x}}$ lie within the bounds $\mathbf{l}$ and $\mathbf{u}$ and that both $\mathbf{g}$ and $\bar{\mathbf{g}}$ are conformal to $\hat{\mathbf{x}} - \hat{\mathbf{z}}$, we see that both $\mathbf{x}^*$ and $\mathbf{z}^*$ also lie within the bounds $\mathbf{l}$ and $\mathbf{u}$. Thus $\mathbf{x}^*$ is a feasible solution of (P) and $\mathbf{z}^*$ is a feasible solution of (IP). We can also write

$$\hat{\mathbf{x}} - \hat{\mathbf{z}} = (\mathbf{x}^* - \hat{\mathbf{z}}) + (\mathbf{z}^* - \hat{\mathbf{z}}),$$

which, using Proposition 39 with values $\mathbf{x} = \hat{\mathbf{z}}$, $\mathbf{y}_1 = \mathbf{g}$, $\mathbf{y}_2 = \bar{\mathbf{g}}$, gives

$$f(\hat{\mathbf{x}}) - f(\mathbf{x}^*) \geq f(\mathbf{z}^*) - f(\hat{\mathbf{z}}) \ .$$

Since $\hat{\mathbf{x}}$ is a continuous optimum and $\mathbf{x}^*$ is a feasible solution to (P), the left hand side is non-positive, and so is $f(\mathbf{z}^*) - f(\hat{\mathbf{z}})$. But since $\hat{\mathbf{z}}$ is an integer optimum it must be that $\mathbf{z}^*$ is another integer optimum and thus $f(\hat{\mathbf{z}}) = f(\mathbf{z}^*)$, and subsequently $f(\mathbf{x}^*) = f(\hat{\mathbf{x}})$ and thus $\mathbf{x}^*$ is another continuous optimum. □

Basic Proximity Theorem. For any separable convex function f, $\mathcal{P}_\infty(A, f) \leq ng_\infty(A)$ due to Hemmecke, Köppe and Weismantel [12]. Now, we show that a careful analysis of a proof of Hemmecke, Köppe and Weismantel [12] allows us to extend their theorem to additionally provide an ℓ_p-norm bound, for any p with $1 \leq p \leq +\infty$. Moreover, it is a consequence of Lemma 38 that for each integer optimum there is a continuous optimum nearby, which extends the result of [12] in the spirit of Hochbaum and Shanthikumar [13]. Note that obviously any p-norm proximity bound implies an ℓ_∞-norm proximity bound, i.e., $\mathcal{P}_\infty(A, f) \leq \mathcal{P}_p(A, f)$ for any $1 \leq p < \infty$.

Theorem 40 (Basic Proximity). $\mathcal{P}_p(A) \leq nc_p(A)$.

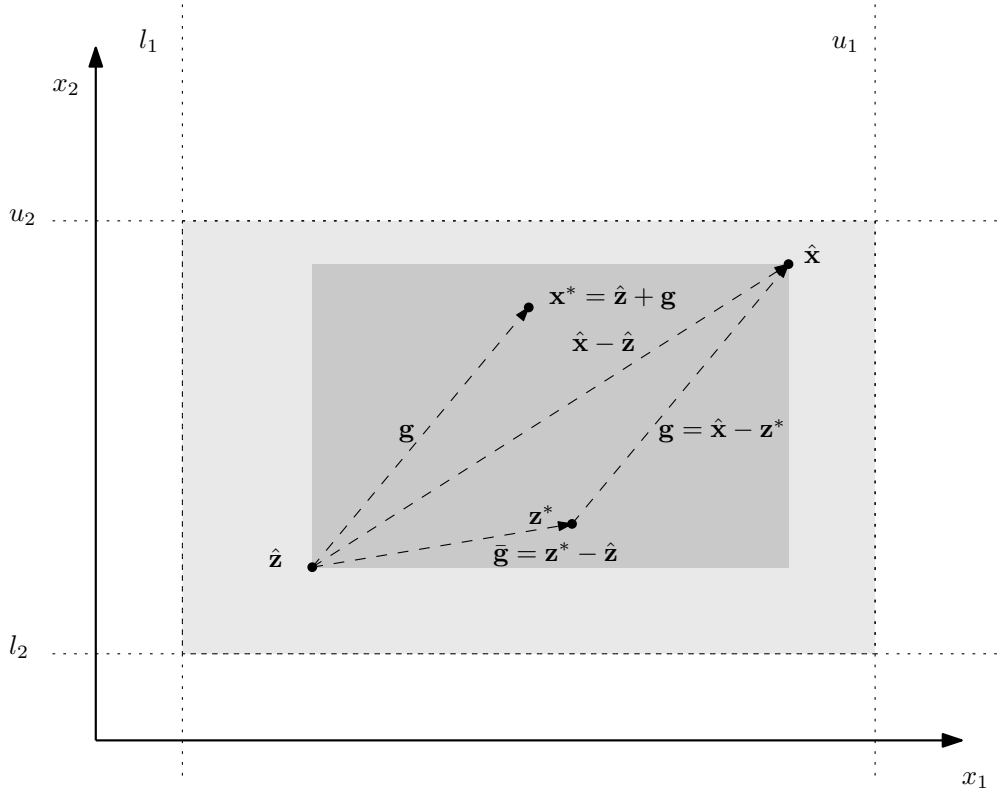


Fig. 2. The situation of Lemma 38: the feasible region between lower and upper bounds is the light grey rectangle; the dark grey rectangle marks the region of vectors which, when translated to $\hat{\mathbf{z}}$, are conformal to $\hat{\mathbf{x}} - \hat{\mathbf{z}}$. The picture makes it clear that $\mathbf{g}$ and $\bar{\mathbf{g}}$ are conformal to $\hat{\mathbf{x}} - \hat{\mathbf{z}}$, and that the identity $\hat{\mathbf{x}} = \hat{\mathbf{z}} + \mathbf{g} + \bar{\mathbf{g}}$ holds.

Proof of Theorem 40. Let $\mathbf{z}$ be a solution of (IP) and $\mathbf{x}$ a solution of (P). By Proposition 35, we may write $\mathbf{x} - \mathbf{z}$ as a sign-compatible sum $\sum_{i=1}^{n'} \lambda_i \mathbf{g}_i$ where $n' \leq n - r$ with $r = \operatorname{rank}(A)$, and, for all $i \in [n']$, $\mathbf{g}_i \in C(A) \subseteq \mathcal{G}(A)$, $\lambda_i \in \mathbb{R}_{>0}$, and $\lambda_i \mathbf{g}_i \sqsubseteq \mathbf{x} - \mathbf{z}$. Write $\mathbf{x} - \mathbf{z} = \sum_{i=1}^{n'} \lfloor \lambda_i \rfloor \mathbf{g}_i + \sum_{i=1}^{n'} \{\lambda_i\} \mathbf{g}_i$, where $\{\lambda\} \coloneqq \lambda - \lfloor \lambda \rfloor$ denotes the fractional part of λ. Now define $\mathbf{z}' \coloneqq \mathbf{z} + \sum_{i=1}^{n'} \lfloor \lambda_i \rfloor \mathbf{g}_i = \mathbf{x} - \sum_{i=1}^{n'} \{\lambda_i\} \mathbf{g}_i$. Clearly $\mathbf{z}'$ is conformal to $\mathbf{x} - \mathbf{z}$ and integer.

Let us now compute the proximity. The following derivation is invariant under the norm bound on the elements of $C(A)$:

$$\|\mathbf{x} - \mathbf{z}'\| = \|\sum_{i=1}^{n'} \{\lambda_i\} \mathbf{g}_i\| \leq n' \max_{i=1,\ldots,n} \|\mathbf{g}_i\| \leq n \max_{\mathbf{g} \in C(A)} \|\mathbf{g}\| = n c_p(A),$$

where the first equality follows by definition of $\mathbf{z}'$, and the claim follows. □

5.1 Scaling and Proximity

The goal of this section is to describe an algorithm which iteratively refines a solution, eventually leading to an optimal solution. In each iteration, the algorithm solves an instance whose bounding box is potentially much smaller than in the original instance, specifically $O(\rho)$ where $\rho \geq \mathcal{P}_\infty(A)$, for example $\rho = n c_\infty(A)$. The motivation for these results was taken from the techniques in [13].

We lay out the approach more specifically. Assume for now that $\mathbf{0}$ is a feasible solution, i.e. we are concerned with the following IP:

$$\min\{f(\mathbf{x}) : A\mathbf{x} = \mathbf{0},\ \mathbf{l} \le \mathbf{x} \le \mathbf{u},\ \mathbf{x} \in \mathbb{Z}^n\}. \tag{11}$$

Instead of looking for a solution in the lattice $\mathbf{x} \in \mathbb{Z}^n$, we first look for a solution in the scaled lattice $\mathbf{z} \in s\mathbb{Z}^n = \{s\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^n\}$, for some $s \in \mathbb{Z}_{\ge 1}$.

$$\min\{f(\mathbf{x}) : A\mathbf{x} = \mathbf{0},\ \mathbf{l} \le \mathbf{x} \le \mathbf{u},\ \mathbf{x} \in s\mathbb{Z}^n\}. \tag{s-lattice IP}$$

Observe that both systems have the same continuous relaxation. Hence, an optimal solution $\mathbf{x}^*$ to the continuous relaxation of (11) is also an optimal solution to the continuous relaxation of (s-lattice IP). With this, we can relate an optimal solution $\mathbf{z}^*$ of (11) to an optimal solution $\hat{\mathbf{z}}$ of (s-lattice IP).

Theorem 41 (Scaling proximity). *Let an IP (11) be given, $s \in \mathbb{Z}_{\ge 1}$ and $1 \le p \le +\infty$. For every optimal solution $\mathbf{z}^*$ of (11), there exists an optimal solution $\hat{\mathbf{z}}$ of (s-lattice IP) such that*

$$\|\mathbf{z}^* - \hat{\mathbf{z}}\|_p \le (s+1)\mathcal{P}_p(A)\ .$$

Vice versa, for every $\hat{\mathbf{z}}$ optimum of (s-lattice IP), there is an optimum $\mathbf{z}^*$ of (11) satisfying $\|\mathbf{z}^* - \hat{\mathbf{z}}\|_p \le (s+1)\mathcal{P}_p(A)$.

Proof. We will show one direction, as the other direction is symmetric. Let $\mathbf{z}^*$ be an optimum solution of (11). By Definition 37 and Lemma 38, there exists an optimum $\mathbf{x}^*$ to the continuous relaxation (P) with

$$\|\mathbf{z}^* - \mathbf{x}^*\|_p \le \mathcal{P}_p(A)\ . \tag{12}$$

As the continuous relaxations coincide, $\mathbf{x}^*$ is also an optimum to the continuous relaxation of (s-lattice IP). Substituting $\mathbf{x}' := \frac{1}{s}\mathbf{x}$, we obtain an objective-value preserving bijection between the solutions of (s-lattice IP) and the solutions of the *s-scaled IP*

$$\min\left\{f(s\mathbf{x}') : A\mathbf{x}' = \mathbf{0},\ \tfrac{\mathbf{l}}{s} \le \mathbf{x}' \le \tfrac{\mathbf{u}}{s},\ \mathbf{x}' \in \mathbb{Z}^n\right\}. \tag{13}$$

In particular, $\bar{\mathbf{x}} := \frac{1}{s}\mathbf{x}^*$ is an optimum solution to the continuous relaxation of (13).

Again by proximity, the instance (13) has an optimal solution $\bar{\mathbf{z}}$ with

$$\|\bar{\mathbf{x}} - \bar{\mathbf{z}}\|_p \le \mathcal{P}_p(A) \quad \Leftrightarrow \quad \|s\bar{\mathbf{x}} - s\bar{\mathbf{z}}\|_p \le s \cdot \mathcal{P}_p(A) \tag{14}$$

Substituting back, $\hat{\mathbf{z}} := s\bar{\mathbf{z}}$ is an optimal solution to (s-lattice IP). The claim follows by triangle inequality: $\|\mathbf{z}^* - \hat{\mathbf{z}}\|_p \le \|\mathbf{z}^* - \mathbf{x}^*\|_p + \|\mathbf{x}^* - \hat{\mathbf{z}}\|_p \le (s+1)\mathcal{P}_p(A)$. □

5.2 Scaling algorithm.

As a 2^k-scaled instance is also a 2-scaled instance of a 2^{k-1}-scaled instance, this gives the following idea for an algorithm to solve a general (IP). The algorithm is parameterized by some upper bound ρ on the proximity bound $\mathcal{P}_\infty(A)$.

(1) Find an initial feasible solution $\mathbf{x}_0$, and recenter the instance at $\mathbf{x}_0$, obtaining (11) where $\mathbf{0}$ is feasible. Let $k = \log_2(\max(\|\mathbf{u}\|_\infty, \|\mathbf{l}\|_\infty)) + 1$, and set $\mathbf{x}_k := 0$. As it is the only feasible solution of a 2^k-scaled instance, $\mathbf{x}_k$ is also its optimal solution.
(2) If $k > 0$, intersect the box constraints of the 2^{k-1}-scaled instance with the proximity bounds $\|\mathbf{x}_k - \mathbf{x}\|_\infty \le 3\rho$. Solve this instance with initial solution $\mathbf{x}_k$ to optimality, and let $\mathbf{x}_{k-1}$ denote the optimum found. Update $k \leftarrow k-1$ and repeat.
(3) Output $\mathbf{x}_0$.

The bound of 3ρ in step (2) follows from the fact that we are viewing the 2^k-scaled instance as an $s = 2$-scaled instance of the 2^{k-1}-scaled instance, and the $3 = s + 1$ in the bound of Theorem 41. Analyzing the running time, we obtain the following corollary.

Corollary 42. *Given an instance of* (IP) *with finite bounds, a* $\rho \geq \mathcal{P}_\infty(A)$*, and an initial feasible solution* $\mathbf{x}_0$*, we can find an optimum of* (IP) *by solving* $2\log\|\mathbf{u}-\mathbf{l}\|_\infty$ *instances of* (IP) *with right-hand side* $\bar{\mathbf{b}} = \mathbf{0}$*, the lower and upper bounds* $\bar{\mathbf{l}}, \bar{\mathbf{u}}$ *of any instance satisfy* $\left\|\bar{\mathbf{u}}-\bar{\mathbf{l}}\right\|_\infty \leq 6\rho$*, and each instance is given with an initial feasible solution.*

The feasibility problem can be solved by one of the methods described in [**?** , Section 3.4], for example, by solving a feasibility ILP with a constraint matrix $(A\ I)$ using the same scaling algorithm.

Proof. Given an initial feasible solution $\mathbf{x}_0$, we center the original instance at $\mathbf{x}_0$ using the translation $\tau(\mathbf{x}) = \mathbf{x} + \mathbf{x}_0$, which transforms its right hand side to $\mathbf{b} - A\mathbf{x}_0 = \mathbf{0}$ as required (see Lemma 32). It is clear that we need at most $\log(\|\mathbf{u}-\mathbf{l}\|_\infty)$ iterations, in each of which we apply Theorem 41 with $s = 2$ to the bounds $\bar{\mathbf{l}}, \bar{\mathbf{u}}$. □

5.3 Solving (P) by Scaling

Once we reached an optimal solution in (IP), we could continue scaling, and look for a solution in the superlattice $\frac{1}{2}\mathbb{Z}^n$. Continuing further, this can be used to find a 2^{-k}-accurate solution for the continuous relaxation of (IP).

The following is a simple corollary of Theorem 40, applied to the s-scaled instance (13) with the respective parameters.

Corollary 43 (Solving (P) by solving (s-lattice IP)). *Let* $\epsilon > 0$*. Then an optimal solution for* (s-lattice IP) *is an* ϵ*-accurate solution of* (P) *for any* $s \leq \frac{\epsilon}{\mathcal{P}_\infty(A)}$.

This corollary immediately motivates an algorithm for finding an ϵ-accurate solution for the continuous problem (P).

Theorem 44. *Let* $\epsilon > 0$ *and* $\rho \geq \mathcal{P}_\infty(A)$*. We can find an* ϵ*-accurate solution to* (P) *by making*

$$O\left(\log(\|\mathbf{u}-\mathbf{l}\|_\infty) + \log\tfrac{\rho}{\epsilon}\right)$$

steps of the scaling algorithm. Each step means solving an (IP) *instance with parameters bounded as in Corollary 42.*

Proof. Let $k := 2\log(\|\mathbf{u}-\mathbf{l}\|_\infty + 1)$. The first k iterations of the algorithm refine the grid from $2^k\mathbb{Z}^n$ to $\mathbb{Z}^n$. The following $2\log\frac{\rho}{\epsilon}$ iterations further refine the grid from $\mathbb{Z}^n$ to $\frac{\rho}{\epsilon}\mathbb{Z}^n$. By Corollary 43, the optimum of the last instance is an ϵ-approximate solution of (IP). □

5.4 Fast Primal Algorithm

In the following we will describe an instantiation of the scaling algorithm which is efficient when A has small $\mathrm{td}_P(A)$ and $\|A\|_\infty$, e.g. when it is a 2-stage or a multi-stage stochastic matrix with small coefficients. Formally,

Theorem 45. *There is an algorithm which solves* (IP) *in time* $g(\mathrm{td}_P(A), \|A\|_\infty)n\log\|\mathbf{u}-\mathbf{l}\|_\infty$ *for some computable function* g*.*

The function g depends on $g_\infty(A)$ and the proximity bound $\mathcal{P}_\infty(A)$. The currently best known bounds for both are triple-exponential in terms of $\mathrm{td}_P(A)$ [15, Corollary 2.1, Lemma 2.3], and this
TODO seems to be optimal for $g_\infty(A)$.. Since the exact dependence of g on $\mathrm{td}_P(A)$ and $\|A\|_\infty$ is not the focus of this work, we shall simply focus on proving the theorem in its stated form.

We will use a recent result of Klein and Reuter [15] (improving on Cslovjecsek et al. [2]) which gives a bound on $\mathcal{P}_\infty(A)$ independent of n:

Proposition 46 ([15, Lemma 2.3]). *There is a computable function* g' *such that*

$$\mathcal{P}_\infty(A) \leq g'(\mathrm{td}_P(A), \|A\|_\infty) \ .$$

Proof of Theorem 45. A run of the scaling algorithm consists of $\log \|\mathbf{u}-\mathbf{l}\|_\infty$ iterations, each of which requires solving an IP with bounds $\bar{\mathbf{l}}, \bar{\mathbf{u}}$ satisfying $\|\bar{\mathbf{u}}-\bar{\mathbf{l}}\|_\infty \leq 6\rho$, with a right hand side $\bar{\mathbf{b}} = \mathbf{0}$, with the constraint matrix A, and with an objective function $\bar{f}$ which is defined as $\bar{f}(\mathbf{x}) = f(s\mathbf{x})$ for some scaling factor s. Specifically, $\bar{f}$ is separable convex if f was.

[**?** , Lemma 25] shows that such an IP can be solved in time $\mathrm{td}_P(A)^2(12\rho+1)^{\mathrm{td}_P(A)}n$ by a simple branching algorithm. Taking $g(\mathrm{td}_P(A), \|A\|_\infty) = \mathrm{td}_P(A)^2(12\rho+1)^{\mathrm{td}_P(A)}$ and the fact that $\rho = g'(\mathrm{td}_P(A), \|A\|_\infty)$ for some computable function g' (by Proposition 46,), the claim is shown. □

Applications: Computational Social Choice

6 INTRO TO COMPUTATIONAL SOCIAL CHOICE

6.1 Voting, Elections, Bribing

Elections. An election (C, V) consists of a set C of candidates and a set V of voters, who indicate their preferences over the candidates in C. There are many ways in which a voter's preferences can be modeled; here we use a variant of the ordinal model, where each voter v's preferences are represented via a *preference order* $\succ_v$ which is a total order over C unless stated otherwise. In some problems we study voters who indicate their preferences only for their "top candidates"; we model this with "truncated orders". For an integer $t \in \mathbb{N}$, a preference order $\succ_v$ is *t-top-truncated* if there is a permutation π over $\{1, \ldots, |C|\}$ such that $\succ_v$ is of the form $c_{\pi(1)} \succ_v \cdots \succ_v c_{\pi(t)} \succ_v \{c_{\pi(t+1)}, \ldots, c_{\pi(|C|)}\}$; that is, v is indifferent among the members of the set $\{c_{\pi(t+1)}, \ldots, c_{\pi(|C|)}\}$ which we call *unranked candidates*; we refer to $\{c_{\pi(1)}, \ldots, c_{\pi(t)}\}$ as to the *ranked candidates*. For a ranked candidate c we denote by $\mathrm{rank}(c, v)$ their rank in $\succ_v$; then v's most preferred candidate has rank 1 and their least preferred candidate has rank $|C|$. Also, for t-top-truncated preference orders $\succ_v$ it holds that $\mathrm{rank}(c, v) \leq t$ for all *ranked* candidates $c \in C$. We note here that if $\succ_v$ is a *weak order* (or *bucket order*), i.e., when it is a linear order over disjoint groups of candidates with the voter having no preference over candidates in one group, we may replace it with any linear extension of $\succ_v$ and set the cost of swapping (see below) two candidates c, c' to 0 whenever $\mathrm{rank}(c, v) = \mathrm{rank}(c', v)$ in the original order. Independently of voters, for the set of candidates C we also refer to a linear order $\succ_C$ over C as to a *ranking* of C (i.e., ranking is a shorthand for a linear order on candidates). For distinct candidates $c, c' \in C$, we write $c \succ_v c'$ if voter v prefers c over c'. To simplify notation, we sometimes identify the candidate set C with the set $\{1, \ldots, |C|\}$, in particular when expressing permutations over C. All studied problems designate a candidate in C; we always denote it by $c^\star$.

Next, we describe the actions by which we perturb a given election (C, V). Applying a set Γ of actions to (C, V) yields a *perturbed* election that we denote by $(C, V)^\Gamma$. Performing an action incurs a cost; we specify these costs by functions that for each voter $v \in V$ specify their individual cost of performing the action.

6.2 Actions for Manipulation

Swaps. Let (C, V) be an election, let $v \in V$ be a voter, and let $\succ_v$ be their preference order. For candidates $c, c' \in C$, a *swap* $s = (c, c')_v$ means to exchange the positions of c and c' in $\succ_v$; denote the perturbed order by $\succ_v^s$. A swap $(c, c')_v$ is *admissible in* $\succ_v$ if $\mathrm{rank}(c, v) = \mathrm{rank}(c', v) - 1$. A set S of swaps is *admissible in* $\succ_v$ if they can be applied sequentially in $\succ_v$, one after the other, in some order, such that each one of them is admissible. Note that the perturbed vote, denoted by $\succ_v^S$, is independent from the order in which the swaps of S are applied. We also extend this notation for applying swaps in several votes and denote it V^S. We specify v's cost of swaps by a function $\sigma^v \colon C \times C \to \mathbb{Z}$. A special case of swaps are "shifts", where we want to make $c^\star$ win the perturbed election by shifting them forward in some of the votes, at an appropriate cost, without exceeding a given budget. Shifts can be modeled by swaps only involving $c^\star$.

Push actions. Let (C, V) be an election. In certain voting rules, such as SP-AV or Fallback (to be defined below), each voter $v \in V$ additionally has an *approval count* $a^v \in \{0, \ldots, |C|\}$. Voter v's approval count[1] a^v indicates that they approve the top-ranked a^v many candidates in their preference order, and disapprove all others. A "push action" can change a voter's approval count:

[1] See Scoring protocols for the definition.

formally, for voter v and $t \in \{-a^v, \dots, |C| - a^v\}$, a *push action* $p^v = t$ changes their approval count to $a^v + t$. We specify the cost of push actions by a function $\pi^v \colon \{-a^v, \dots, |C| - a^v\} \to \mathbb{Z}$; we stipulate that $\pi^v(0) = 0$. If a voter v is involved in a swap or a push action, a one-time *influence cost* ι^v occurs.

Control changes. Let (C, V) be an election. We partition the set V into a set V_a of *active* voters and a set V_ℓ of *latent* voters. Only active voters participate in an election, but through a "control change" latent voters can become active or active voters can become latent. (If no partition of V into V_a and V_ℓ is specified, then we implicitly assume that $V = V_a$.)

Formally, a *control change* γ activates some latent voters from V_ℓ and deactivates some active voters from V_a; denote the changed set of voters by $(V_\ell \cup V_a)^\gamma$. We denote the cost of activating voter $v \in V_\ell$ by α^v and the cost of deactivating voter $v \in V_a$ by δ^v.

6.3 Voting rules

A voting rule $\mathcal{R}$ is a function that maps an election (C, V) to a subset $W \subseteq C$, called the *winners*. We study the following voting rules:

Scoring protocols. A scoring protocol is defined through a vector $\mathbf{s} = (s_1, \dots, s_{|C|})$ of integers with $s_1 \geq \cdots \geq s_{|C|} \geq 0$. For each position $p \in \{1, \dots, |C|\}$, the value s_p specifies the number of points that each candidate c receives from each voter that ranks c as p^{th} best. Any candidate with the maximum number of points is a winner. Examples of scoring protocols include the Plurality rule with $\mathbf{s} = (1, 0, \dots, 0)$, the d-Approval rule with $\mathbf{s} = (1, \dots, 1, 0, \dots, 0)$ with d ones, and the Borda rule with $\mathbf{s} = (|C| - 1, |C| - 2, \dots, 1, 0)$. Throughout, we consider only *natural* scoring protocols for which $s_1 \leq |C|$; this is the case for the aforementioned popular rules.

Bucklin. The *Bucklin winning round* is the (unique) number k such that using the k-approval rule yields a candidate with more than $\frac{n}{2}$ points, but the $(k-1)$-approval rule does not. A *Bucklin winner* is then any candidate with the maximum points (over all candidates) when the k-approval rule is applied.

Condorcet-consistent rules. A candidate $c \in C$ is a *Condorcet winner* if any other $c' \in C \setminus \{c\}$ satisfies $|\{v \in V \mid c \succ_v c'\}| > |\{v \in V \mid c' \succ_v c\}|$. A voting rule is *Condorcet-consistent* if it selects the Condorcet winner in case there is one. Fishburn [7] classified voting rules as C1, C2, or C3, depending on the kind of information needed to determine the winner[2]. For candidates $c, c' \in C$ let $v(c, c')$ be the number of voters who prefer c over c', that is, $v(c, c') = |\{v \in V \mid c \succ_v c'\}|$; we write $c <_M c'$ if c beats c' in a head-to-head contest, that is, if $v(c, c') > v(c', c)$.

C1: $\mathcal{R}$ *is C1* if knowing $<_M$ suffices to determine the winner, that is, for each pair of candidates c, c' we know whether $v(c, c') > v(c', c)$, $v(c, c') < v(c', c)$ or $v(c, c') = v(c', c)$. An example is the Copeland$^\alpha$ rule for a rational number $\alpha \in [0, 1]$, which specifies that for each head-to-head contest between two distinct candidates, if some candidate is preferred by a majority of voters, then they obtain one point and the other candidate obtains zero points, and if a tie occurs, then both candidates obtain α points; the candidate with largest sum of points is a winner.

C2: $\mathcal{R}$ *is C2* if it is not C1 and knowing the *exact value of* $v(c, c')$ for all $c, c' \in C$ suffices to determine the winner. Examples are the *Maximin* rule which declares any candidate $c \in C$ a winner who maximizes $v_*(c) = \min\{v(c, c') \mid c' \in C \setminus \{c\}\}$; and the *Kemeny* rule which declares any candidate $c \in C$ a winner for whom there exists a ranking $\succ_R$ of C that ranks c

[2] Sometimes the classification (C1, C2, C3) applies to Condorcet-consistent rules only, i.e., voting rules that guarantee to select a Condorcet winner as a winner if such a candidate exists. Here, we follow [24] where this is not required.

first and maximizes the total agreement with voters

$$\sum_{v \in V} \left| \{(c', c'') \mid ((c' \succ_R c'') \Leftrightarrow (c' \succ_v c'')) \ \ \forall c', c'' \in C\} \right|$$

among all rankings.

C3: $\mathcal{R}$ *is C3* if it is neither C1 nor C2. Examples are the *Dodgson* rule which declares any candidate $c \in C$ a winner for whom a minimum number of swaps make them the Condorcet winner of the manipulated election; and the *Young* rule which declares any candidate $c \in C$ a winner for whom removing a minimum number of voters from the election makes c the Condorcet winner of the perturbed election.

Additionally, if approval counts are given for each voter, other voting rules are possible:

Sincere-strategy preference-based approval voting (SP-AV).. Each candidate c receives a point from every voter v with $\text{rank}(c, v) \leq a^v$. A candidate with maximum number of points is a winner in the election.

Fallback. Delete, for each voter $v \in V$, their unranked candidates (i.e., all c with $\text{rank}(c, v) > a^v$) from its order. Then, use the Bucklin rule, which might fail to determine a winner due to the deletion of unranked candidates; in that case, use the SP-AV rule.

6.4 Voter Types, Societies, and Moves

Partition the voters into *types* such that two voters of the same type are indistinguishable, i.e., they have the same preference order, bribery costs, etc. For example, if all bribery costs are unit, there are $\tau \leq m!$ types present in a given election since there are at most $m!$ distinct preference orders. We order the types arbitrarily so that we can speak of "the j-th type" for a given $j \in [\tau]$. By the *weight* of voters with type j, denoted either w_j or $w(j)$, as is more convenient, we mean the number of voters of type j. It will be convenient to allow some types to have weight 0. Sometimes we represent an election as a vector $\mathbf{w} \in \mathbb{N}^\tau$, whose j-th entry represents the weight of type j. We refer to such vectors as *societies*. Formally, any non-negative vector $\mathbf{w} \in \mathbb{N}^\tau$ represents a society.

Societies and Moves. In most problems, we are interested in modifying a society by moving people among types. A *move* is a vector $\mathbf{m} = (m_{1,1}, \ldots, m_{\tau,\tau}) \in \mathbb{Z}^{\tau^2}$. Intuitively, $m_{i,j}$ is the number of people of type i turning type j.

Definition 47. A *change* is a vector $\mathbf{\Delta} = (\Delta_1, \ldots, \Delta_\tau) \in \mathbb{Z}^\tau$ whose elements sum up to 0. We say that $\mathbf{\Delta}$ *is the change associated with a move* $\mathbf{m}$ if, for all $i \in [\tau]$, $\Delta_i = \sum_{j=1}^{\tau} m_{j,i} - m_{i,j}$, and we write $\mathbf{\Delta} = \Delta(\mathbf{m})$. A change $\mathbf{\Delta}$ is *feasible wrt. society* $\mathbf{w}$ if $\mathbf{w} + \mathbf{\Delta} \geq \mathbf{0}$, i.e., if applying the change $\mathbf{\Delta}$ to $\mathbf{w}$ results in a society (i.e., as long as there are enough voters from each type to move to other types).

A useful notion is the *move costs vector*, which is a vector $\mathbf{c} = (c_{1,1}, \ldots, c_{\tau,\tau})$ in $(\mathbb{N} \cup \{+\infty\})^{\tau^2}$ satisfying the triangle inequality, i.e., $c_{i,k} \leq c_{i,j} + c_{j,k}$ for all distinct i, j, k.

Remark. We will mainly focus on moves that correspond to swap bribery actions. However, the actions of bribery, manipulation, and control are all expressible as moves in societies. E.g., one may create, for each voter type $t \in [\tau]$, an "inactive" variant t', and moving a voter from t to t' corresponds to deleting this voter while moving a voter from t' to t corresponds to adding it – which are the actions considered in constructive control by adding/deleting voters. Hence, we encourage the reader to keep in mind that whenever we talk about swaps and bribery, many other types of actions may be substituted or added in that place.

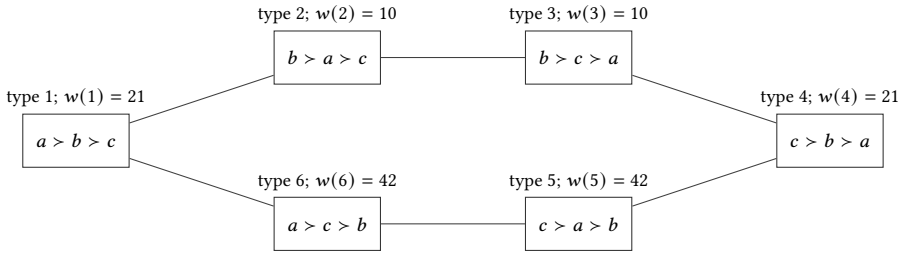


Fig. 3. A society graph with three candidates and six types (corresponding to the six possible preference orders on those three candidates). In this graph there are, e.g., 42 voters of type 6, each with preference order $a \succ c \succ b$; this graph corresponds to a society $\mathbf{w} = [21, 10, 10, 21, 42, 42]$.

6.5 Society Graphs

As we are interested in diffusion processes operating on the voter types, we associate a given election with a vertex-weighted graph $G = (V, \mathbf{w}, E)$, termed the *society graph*. The society graph contains τ vertices, where τ is the number of types in the election (specifically, $V = \{v_1, \ldots, v_\tau\}$, where vertex v_j corresponds to voter type j, and its weight w_j is equal to the number of voters of that type in the given election). There is an edge between vertices v_j and $v_{j'}$ if the preference orders corresponding to types j and j' differ by the ordering of a single pair of adjacent candidates (in other words, if it is possible to transform one into the other with a single swap of two consecutive candidates). We show an example of a society graph in Figure 3.[3]

6.6 Diffusion of Preferences

Given a society graph (which encodes a given election), we consider two variants of the diffusion process, namely asynchronous and synchronous. In the asynchronous variant, in each step of the process some vertex v of the society graph G is picked and, then, the following occurs (we do not specify which vertex is selected and, as we will see in Example 48 below, different orders of selecting the vertices may lead to different outcomes of the process). We consider the closed neighborhood $N[v]$ of v in G (the *closed neighborhood* of a vertex is the set containing the vertex and its neighbors) and check whether there is a neighbor x of v for which $w_x > 1/2 \sum_{u \in N[v]} w_u$; that is, a neighbor whose weight exceeds the sum of the weights of all other vertices in the closed neighborhood of v. If such a neighbor x exists, then we add the current weight w_v of v to that of x and change the weight w_v to be 0. Intuitively, the voters of type represented at v look at all the voters with similar or identical preferences and if there is a majority support among these voters for some preference order, then they switch to it. In the synchronous variant we proceed in the same way, but simultanously for all vertices. The diffusion process halts whenever it stabilizes (i.e., whenever there is no change between an iteration and its successive one).

Example 48. Consider the society graph depicted in Figure 3 and asynchronous diffusion. Assume that we first choose type 3. As type 3 has as neighbors types 2 and 4, together there are 41 voters of these types, and 21 of them have preference order $c \succ b \succ a$. So, the 10 voters with type 3 move to have type 4. If we then select type 4, type 6, and then type 2, then the diffusion converges with 115 voters of type 5 (with preference order $c \succ a \succ b$) and 31 voters of type 1 (with preference order $a \succ b \succ c$); thus, Plurality selects c. However, if we select first type 2, then 1, then 5, and then 3, then we reach convergence with 115 voters of type 6 (with preference order $a \succ c \succ b$) and 31

[3]Graphs of this form are quite popular in the study of permutations. Later we will also consider other graphs.

voters of type 5 (with preference order $c \succ b \succ a$); thus, Plurality selects a. This shows that the asynchronous diffusion process can lead to different outcomes, depending on the order in which vertices are considered.

Let us now consider the same society graph and synchronous diffusion. After the first round, we have 10 voters of type 1 (voters of type 2 moved to have type 1, whereas original type 1 voters moved to have type 6), no voters of types 2 and 3, 10 voters of type 4, 63 voters of type 5, and 63 voters of type 6. After the next round there are 73 voters of type 5 and 73 voters of type 6. No further changes are possible and the process converges; Plurality selects a and c as two tied winners.

6.7 Bribery in Society Graphs

Besides issues related to the diffusion of preferences, we are mainly interested in understanding the possibility of manipulating election outcomes. Thus we assume that there is an external briber who has some budget and, using this budget, can affect the original preference orders of some voters (i.e., the preference orders they have prior to the diffusion). Specifically, in a single bribery action the briber chooses a single voter and, at unit cost, shifts the briber's preferred candidate p up by one position in this voter's preference order (in effect, changing this voter's type; see the work of Elkind et al. [5, 6] and Bredereck et al. [1] for a detailed discussion of shift bribery and its various cost models). The briber performs as many bribery actions as he wants, up to the budget limit, and then the diffusion process takes place. The goal of the briber is to have his preferred candidate $c^\star$ win the resulting election (under a given, predetermined voting rule). Formally, we are interested in the following general problem.

> $\mathcal{R}$-Bribery in society graphs ($\mathcal{R}$-BSG)
> **Input:** A society graph G (given directly as a graph), a preferred candidate $c^\star$, and a budget b.
> **Question:** Are there at most b (unit-cost, shift-) bribery actions, such that after performing them on G and then running the diffusion process, $c^\star$ is an $\mathcal{R}$-winner of the resulting election?

Corresponding to the synchronous and asynchronous diffusion processes, we consider both *sync-$\mathcal{R}$-BSG* and *async-$\mathcal{R}$-BSG* problems. For the asynchronous diffusion, we further consider the *optimistic* and *pessimistic* variants of the problem. In the former, we ask whether the briber's preferred candidate wins for *some* order of the diffusion steps. In the latter, we require that p wins for *every* order of diffusion steps that leads to convergence.

Remark. The input to $\mathcal{R}$-BSG is a labeled graph with weighted vertices, a preferred candidate $c^\star$, and a budget b. Thus the size of the input encoding is linear in the number of voter types and only logarithmic in the number of voters.

7 PA IS FPT

Let m, n be integers. We wish to develop efficient algorithms that find optimal strategies for agents that are manipulating a given election. Our approach is to write a formula in Presburger arithmetic (here we shorten to PA; this is not to be confused with Peano arithmetic) with a vector $\mathbf{m}$ of free variables such that the satisfying assignments are bribery actions corresponding to a first move in a winning strategy. Here we first provide a brief introduction to PA, and later show that optimizing over the satisfying assignments of a PA formula can be done efficiently if some of its parameters are bounded, as will be the case for the formulas modeling winning strategies.

PA is a useful logic to reason about numbers. Intuitively, PA can be viewed as Integer Linear Programming (ILP) enriched with logical connectives and quantifiers. For two formulas Φ and Ψ, we denote their equivalence by $\Phi \cong \Psi$.

Definition 49 (Extended Presburger Arithmetic (PA)). An *atom* (or *atomic formula*) is a linear inequality $\mathbf{ax} \leq b$ or a congruence $\mathbf{ax} \equiv b \mod p$, with $\mathbf{a} \in \mathbb{Z}^n$ and $b, p \in \mathbb{Z}$. We call $\mathbf{t} \equiv \mathbf{ax} + b$ for some $\mathbf{a}$ and b a *term*. A *formula* is obtained by taking Boolean combinations of atoms using the standard logical connectives ($\wedge, \vee, \implies, \neg$, etc.) and by existential and general quantifiers $\exists, \forall$, respectively. Denote by PA the set of all PA formulas. A *literal* is an atom or its negation. A variable is *bound* in a formula φ if it appears in a quantifier, and it is *free* otherwise. If $\mathbf{x}$ is a vector of the free variables of a formula φ, we write $\varphi(\mathbf{x})$.

Remark. The term "extended" in the definition above refers to the congruence atoms that are not present in the original language as defined by Presburger [**?**]; however, it is typical to speak of PA as this extended language because it allows for quantifier elimination, unlike PA without congruence atoms. For detailed definitions see Klaedtke [**?**].

We provide some further useful notation below. For $\varphi \in$ PA we define $\mathcal{T}(\varphi)$ to be the set of all atoms of φ of the form $\mathbf{ax} \leq b$, $\mathsf{D}(\varphi)$ to be the set of all atoms of the form $\mathbf{ax} \equiv b \mod p$, $\mathsf{L}(\varphi)$ to be the number of symbols of φ (i.e., the number of atoms, logical connectives, and quantifiers),[4] the *maximum coefficient* $\alpha(\varphi)$ to be the maximum $\|\mathbf{a}\|_\infty$ and p contained in any of its atoms, and the *maximum constant* $\beta(\varphi)$ to be the largest right hand side b in any of its atoms.

In this section, we aim at solving the following problem.

Presburger arithmetic minimization	
Input:	A PA formula $\varphi(\mathbf{x})$ with d free variables, and a function $f : \mathbb{R}^d \to \mathbb{R}$.
Task:	Find an assignment $\mathbf{x} \in \mathbb{Z}^d$ satisfying $\varphi(\mathbf{x})$ and minimizing $f(\mathbf{x})$ (among satisfying assignments), or reports unsatisfiability.

The main algorithmic result regarding Presburger arithmetic we prove here is the following.

Theorem 50. *Presburger arithmetic minimization is fixed-parameter tractable parameterized by* $\mathsf{L}(\varphi) + \alpha(\varphi)$ *for any convex function* f.

Proof. Assume that $\varphi(\mathbf{y}) \equiv Q_1 x_1 Q_2 x_2 \dots Q_{k-1} x_{k-1} Q_k x_k \zeta(\mathbf{x}, \mathbf{y})$, with $Q_1, \dots, Q_k \in \{\exists, \forall\}$ and $\zeta(\mathbf{x}, \mathbf{y})$ containing no quantifiers. Here, $\mathbf{y}$ is the vector of free variables of φ. The proof proceeds by *quantifier elimination*: if we show that the innermost quantifier ($\exists x_k$ in our example) can be eliminated, i.e., if we can construct an equivalent formula $\varphi'(\mathbf{y}) \equiv Q_1 x_1 \dots Q_{k-1} x_{k-1} \zeta'(x_1, \dots, x_{k-1}, \mathbf{y})$, then repeatedly applying this procedure reduces φ down to a formula with no quantifiers and only containing variables $\mathbf{y}$ (but no variables $\mathbf{x}$). Optimizing over the satisfying assignments of such a formula then (with some more work) reduces to optimization over linear constraints in small dimension. Already the original proof of Presburger that PA is decidable worked by quantifier elimination. We shall now describe an algorithm of Cooper, which achieves better complexity. Let us stress that our goal is not to prove the correctness of the algorithm, only to describe it in sufficient detail so as to analyze its complexity, and still convey the main underlying intuition; for correctness, we refer the reader to existing textbooks [**?**].

Consider a formula $\varphi(x_k) \equiv \exists x_k \zeta(x_k)$, where $\zeta(x_k)$ is quantifier-free. Here, φ stands for the suffix of the whole formula to be decided; we sometimes disregard the prefix $Q_1 x_1 \cdots Q_{k-1} x_{k-1}$ and the free variables $\mathbf{y}$ for brevity. Note that if the last quantifier was $\forall$, we negate the formula, obtain $\exists$ as the last quantifier, and in the end negate the (eventually quantifier-free) formula again. The algorithm proceeds in three steps. **First**, we put $\varphi(x_k)$ into negation normal form (pushing all negations inward as much as possible) using De Morgan's rules, yielding an equivalent formula

[4] Note that this definition is different than the standard definition of the length of a formula, which uses unary encoding of numbers.

$\varphi_1(x_k)$. **Second**, we normalize $\varphi_1(x_k)$ so that the coefficients of x_k are all 1 or -1, yielding $\varphi_2(x_k)$. This is done as follows. Let A be the set of coefficients of x_k in $\varphi_1(x_k)$, and let $M = \text{lcm}(A)$, where lcm is the least common multiple and hence $M \leq (\max_{a \in A} a)^{|A|}$. Replace every atom $\mathbf{a}(\mathbf{x}, \mathbf{y}) \leq b$ in $\mathcal{T}(\varphi_1)$ with $(M/a_k) \cdot \mathbf{a}(\mathbf{x}, \mathbf{y}) \leq (M/a_k)b$ (recall a_k is the coefficient of x_k in $\mathbf{ax}$), replace every atom $\mathbf{a}(\mathbf{x}, \mathbf{y}) \equiv b \mod p$ in $\mathsf{D}(\varphi_1)$ with $(M/a_k)\mathbf{a}(\mathbf{x}, \mathbf{y}) \equiv (M/a_k)b \mod (M/a_k)p$ and call $\varphi_2(x_k)$ the resulting formula. Now we perform the substitution $x'_k = Mx_k$, hence let $\varphi_3(x'_k) \equiv (\varphi_2(Mx_k) \wedge x'_k \equiv 0 \mod M)$. Now, all coefficients of x'_k in $\varphi_3(x'_k)$ are 1 or -1.

The **third** step is the most involved. Denote by $\bar{\mathbf{x}} = (x_1, \ldots, x_{k-1})$. Notice that all literals of $\varphi_3(x'_k)$ are (perhaps after simple rearranging) of one of the following types, where $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3$ are terms over the variables $\bar{\mathbf{x}}$:

a) $x'_k \leq \mathbf{t}_1$,
b) $\mathbf{t}_2 \leq x'_k$,
c) $x'_k = \mathbf{t}_3 \mod p$,
d) $\neg(x'_k = \mathbf{t}_3 \mod p)$.

We distinguish two cases. Either $\varphi_3(x'_k)$ has arbitrarily small satisfying assignments (i.e., for any $t \in \mathbb{Z}$, there exists a satisfying assignment to x'_k smaller than t). Then, for a sufficiently small satisfying assignment, literals of type a) are satisfied and can be replaced by $\top$, and literals of type b) are falsified and can be replaced by $\bot$. Call $\varphi_{-\infty}(x'_k)$ a formula obtained from $\varphi_3(x'_k)$ by the aforementioned replacements. Let M' be the least common multiple of all the moduli p in literals of types c) and d), and let $\varphi_{41} \equiv \bigvee_{j=1}^{M'} \varphi_{-\infty}(j)$. Then φ_{41} is satisfiable iff $\varphi(x_k)$ has arbitrarily small satisfying assignments.

Now we construct a formula φ_{42} which is satisfied in the converse case when $\varphi(x_k)$ has a least satisfying assignment. For such an assignment some type-b) literal is satisfied and for smaller assignments it is not. We let $B = \{\mathbf{t}_2 \mid \mathbf{t}_2 \leq x'_k \text{ is a type-b) literal}\}$ and define $\varphi_{42} \equiv \bigvee_{j=1}^{M'} \bigvee_{\mathbf{t}_2 \in B} \varphi_3(\mathbf{t}_2 + j)$ (note here that $\varphi_3(\mathbf{t}_2 + j)$ is φ_3 with x'_k substituted by $\mathbf{t}_2 + j$). Then $\varphi_4 \equiv \varphi_{41} \vee \varphi_{42}$ and it does not contain any occurrence of x_k, finishing the elimination.

Let us now bound the length, maximum coefficient, and maximum constant of the formula resulting after eliminating all quantifiers. For this, it suffices to bound the blow-up caused by one quantifier elimination. Step one does not consider the coefficients or constants in any way and hence $\mathsf{L}(\varphi_1)$ is bounded by a function of $\mathsf{L}(\varphi)$, $\alpha(\varphi_1) = \alpha(\varphi)$, and $\beta(\varphi_1) = \beta(\varphi)$. Step two increases the coefficients and constants by a number only depending on the previously largest coefficient and the number of literals and adds two symbols, specifically, $\alpha(\varphi_3) \leq \alpha(\varphi_1)^{\mathsf{L}(\varphi_1)}$, $\beta(\varphi_3) \leq \beta(\varphi_1) \cdot \alpha(\varphi_1)^{\mathsf{L}(\varphi_1)}$, and $\mathsf{L}(\varphi_3) = \mathsf{L}(\varphi_1) + 2$. In step three again the length, largest coefficient and largest constant only grows by a factor of the initial length and largest coefficient, specifically, $\mathsf{L}(\varphi_4) \leq \alpha(\varphi_3)^{\mathsf{L}(\varphi_3)} \cdot \mathsf{L}(\varphi_3)^2$, $\alpha(\varphi_4) \leq \alpha(\varphi_3)^{\mathsf{L}(\varphi_3)} \cdot \alpha(\varphi_3) = \alpha(\varphi_3)^{\mathsf{L}(\varphi_3)+1}$, and $\beta(\varphi_4) \leq \beta(\varphi_3) \cdot \alpha(\varphi_3)^{\mathsf{L}(\varphi_3)}$. Applying the bounds derived above inductively, we get the following intermediate claim:

Lemma 51. *There exists a computable function g such that given a formula $\varphi(\mathbf{y}) \in \mathsf{PA}$, an equivalent quantifier-free formula $\psi(\mathbf{y})$ can be obtained in time $O(g(\mathsf{L}(\varphi), \alpha(\varphi)))$ and it satisfies:*

- $\mathsf{L}(\psi), \alpha(\psi) \leq g(\mathsf{L}(\varphi), \alpha(\varphi))$,
- $\beta(\psi) \leq g(\mathsf{L}(\varphi), \alpha(\varphi)) \cdot \beta(\varphi)$.

Now, we come to the second step, where we wish to optimize over the satisfying assignments of $\psi(\mathbf{y})$. We transform ψ to disjunctive normal form (so that it is a disjunction of conjunctions of atoms), which may increase its length exponentially, but that is still bounded by a function of the parameters. Let us now assume that $\psi \equiv \psi_1 \vee \psi_2 \cdots \psi_K$ for some $K \in \mathbb{N}$, where each ψ_i is a conjunction of literals. Clearly, an assignment $\mathbf{y}$ minimizing $f(\mathbf{y})$ over the satisfying assignments of ψ satisfies some conjunction ψ_i, so we may instead minimize separately over the satisfying assignments for each ψ_i, $i \in [K]$. This can be done by algorithms for convex integer minimization

whenever ψ_i is a conjunction of linear atoms [3, 11], so our next task is to linearize the congruence atoms.

This is easy for positive literals: say we have $\mathbf{a}\mathbf{y} = b \mod p$; then we introduce a new variable z, and add a linear constraint $pz = \mathbf{a}\mathbf{y} - b$, which is satisfied iff $\mathbf{a}\mathbf{y} - b$ is divisible by p. For the negative literals, this is a little bit trickier: say we have $\mathbf{a}\mathbf{y} \neq b \mod p$. This is equivalent to saying that $\mathbf{a}\mathbf{y}$ mod p is between $b+1$ and $b+p-1$. We introduce two variables z, z' and add the following three linear constraints: $pz = \mathbf{a}\mathbf{y} - z'$ and $b+1 \leq z' < b+p$. Since we have introduced a constant number of new variables and constraints for each congruence atom, the system resulting from ψ_i is still of length bounded by a function of the parameters, and evaluating f over its integer assignments can be done in fixed-parameter tractable time (specifically, in time $\delta^{O(\delta)}\text{poly}\log(\alpha(\psi_i), \beta(\psi_i))$ where δ is the number of variables, which is at most $3d$, the dimension of $\mathbf{y}$, i.e., the number of free variables of φ) [3, 11]. This concludes the proof. □

8 ALGORITHMS: PUTTING IT TOGETHER

Putting together the algorithms we know for n-fold IPs and Presburger Arithmetic, we are able to solve many of the problems outlined above (bribery, bribery with opinion diffusion, etc.)

8.1 Borda-Bribery is FPT

This is just as sketch of the algorithm Say there are τ' types of voters on input, and we consider bribery/control actions which can only create T new types for each input type. This captures all the actions defined above; for example, swaps can only produce $m! - 1$ new types for one input type; control (the activation/deactivation of a voter) only creates 1 more type for each input type, etc. Thus, we know that there are at most $\tau \leq (T+1) \cdot \tau'$ "potential types", those types which could appear because of the bribery/control etc. Also assume that we can construct a moves cost vector $\mathbf{c}$ from the given bribery/control costs, so that c_{ij} is the cost of moving a voter of type i to type j.

Consider for example Condorcet's voting rule. Then we construct an n-fold IP as follows. The variables will represent the move vector of the bribery: m_{ij} is the number of voters of type i which are to be bribed to become of type j. Thus, $\mathbf{w}' = \mathbf{w} + \mathbf{\Delta}(\mathbf{m})$ is the new society, and ensuring that $c^\star$ is the Condorcet winner amounts to the following set of constraints:

$$\sum_{i : c^\star >_i c'} w'_i > \sum_{i : c' >_i c^\star} w'_i \ . \qquad \forall c \in C \setminus \{c^\star\}$$

Notice that w'_i above is just syntactic sugar for $w_i - (\sum_j m_{ij}) + (\sum_i m_{ij})$, where w_i is a constant, so the above is a set of $|C| - 1$ constraints, each only involving the $\mathbf{m}$ variables. We also need another set of constraints:

$$\sum_{j \in [\tau]} m_{ij} = w_i \qquad \forall i \in [\tau']$$

which ensure that $\mathbf{m}$ is indeed a valid move, i.e. $\mathbf{w} + \mathbf{\Delta}(\mathbf{m}) \geq \mathbf{0}$.

Now observe that the first type of constraints may involve all the variables in an unrestricted way, but there are only $|C| - 1$ of those constraints. On the other hand, there are many (τ') constraints of the second type, but they don't "overlap" in the variables they involve – only the i-th constraint involves variables $m_{i\bullet}$. Thus, the ILP above is an n-fold IP with parameters $r = |C| - 1$, $s = 1$, $t = \tau$, and with $n = \tau'$ many blocks, thus applying our FPT algorithms for it will give an FPT algorithm for Bribery parameterized by the number of candidates $|C|$.

8.2 Other Voting Rules

What if we wanted to consider other voting rules? For many of them, one could construct the corresponding constraints by hand. However, for some more involved voting rules, such as STV, or even worse, Young and Dodgson, this is not at all obvious. A neat trick is the following. Write a PA formula $\varphi(\mathbf{w})$ which is true iff $\mathbf{w}$ is a society in which $c^\star$ wins under some voting rule $\mathcal{R}$ (this could be STV, Dodgson, Young, etc.) and so that the length of φ and its largest coefficient are bounded by a parameter. We have proven that there exists an equivalent DNF formula $\psi(\mathbf{w})$ whose each clause corresponds to an ILP with some congruence predicates, which we can turn into a "pure" ILP (no congruence predicates) with the use of some extra variables. Because the length of ψ is bounded by a function of the length of φ and its largest coefficient, the number of constraints corresponding to each clause of ψ is small. Thus, if we use them as winning constraints in the n-fold IP from the previous section, we can still solve the resulting IP quickly. Now do this separately for each clause of ψ, compute the optimum of the resulting n-fold, and return the minimum of these.

8.3 Diffusion Process

Also using our algorithm for PA, we can solve the $\mathcal{R}$-BSG. One just needs to notice that the whole diffusion process can be expressed as a short PA formula.

REFERENCES

[1] R. Bredereck, J. Chen, P. Faliszewski, A. Nichterlein, and R. Niedermeier. 2016. Prices matter for the parameterized complexity of shift bribery. *Information and Computation* 251 (2016), 140–164.

[2] Jana Cslovjecsek, Friedrich Eisenbrand, Michal Pilipczuk, Moritz Venzin, and Robert Weismantel. 2021. Efficient Sequential and Parallel Algorithms for Multistage Stochastic Integer Programming Using Proximity. In *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference) (LIPIcs)*, Petra Mutzel, Rasmus Pagh, and Grzegorz Herman (Eds.), Vol. 204. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 33:1–33:14. https://doi.org/10.4230/LIPIcs.ESA.2021.33

[3] Daniel Dadush, Chris Peikert, and Santosh Vempala. 2011. Enumerative Lattice Algorithms in any Norm Via M-ellipsoid Coverings.. In *FOCS*, Rafail Ostrovsky (Ed.). IEEE, 580–589. http://dblp.uni-trier.de/db/conf/focs/focs2011.html#DadushPV11;http://dx.doi.org/10.1109/FOCS.2011.31;http://www.bibsonomy.org/bibtex/2eb00db06f2ab0db87494018ddc765f1e/dblp

[4] Jesús A. De Loera, Raymond Hemmecke, and Matthias Köppe. 2013. *Algebraic and Geometric Ideas in the Theory of Discrete Optimization*. MOS-SIAM Series on Optimization, Vol. 14. SIAM.

[5] E. Elkind and P. Faliszewski. 2010. Approximation Algorithms for Campaign Management. In *Proceedings of WINE '10*. 473–482.

[6] E. Elkind, P. Faliszewski, and A. Slinko. 2009. Swap Bribery. In *Proceedings of SAGT '09*. 299–310.

[7] Peter C. Fishburn. 1977. Condorcet social choice functions. *SIAM J. Appl. Math.* 33, 3 (1977), 469–489.

[8] András Frank and Éva Tardos. 1987. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica* 7, 1 (1987), 49–65.

[9] Paul Gordan. 1873. Ueber die Auflösung linearer Gleichungen mit reellen Coefficienten. *Math. Ann.* 6, 1 (1873), 23–28.

[10] Jack E. Graver. 1975. On the foundations of linear and integer linear programming I. *Math. Program* 9, 1 (1975), 207–226.

[11] Martin Grötschel, László Lovász, and Alexander Schrijver. 1993. *Geometric algorithms and combinatorial optimization* (second ed.). Algorithms and Combinatorics, Vol. 2. Springer-Verlag, Berlin. xii+362 pages.

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

[13] Dorit S. Hochbaum and J. George Shanthikumar. 1990. Convex separable optimization is not much harder than linear optimization. *J. ACM* 37, 4 (1990), 843–862.

[14] Kim-Manuel Klein. 2019. About the Complexity of Two-Stage Stochastic IPs. *CoRR* abs/1901.01135 (2019). arXiv:1901.01135 http://arxiv.org/abs/1901.01135

[15] Kim-Manuel Klein and Janina Reuter. 2022. Collapsing the Tower - On the Complexity of Multistage Stochastic IPs. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, Joseph (Seffi) Naor and Niv Buchbinder (Eds.). SIAM, 348–358. https://doi.org/10.1137/1.

9781611977073.17

[16] Dusan Knop, Michal Pilipczuk, and Marcin Wrochna. 2018. Tight complexity lower bounds for integer linear programming with few constraints. *CoRR* abs/1811.01296 (2018). arXiv:1811.01296 http://arxiv.org/abs/1811.01296

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

[18] Shmuel Onn. 2010. Nonlinear discrete optimization. *Zurich Lectures in Advanced Mathematics, European Mathematical Society* (2010). `http://ie.technion.ac.il/~onn/Book/NDO.pdf`.

[19] Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. 2014. A Faster Parameterized Algorithm for Treedepth. In *Proceedings Part I of the 41st International Colloquium on Automata, Languages, and Programming, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014 (Lecture Notes in Computer Science)*, Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias (Eds.), Vol. 8572. Springer, 931–942.

[20] Sergey Sevast'janov and Wojciech Banaszczyk. 1997. To the Steinitz lemma in coordinate form. *Discrete Math.* 169, 1-3 (1997), 145–152.

[21] E. Steinitz. 1916. Bedingt konvergente Reihen und konvexe Systeme. *J. Reine Angew. Math.* 146 (1916), 1–52.

[22] Arne Storjohann and George Labahn. 1996. Asymptotically Fast Computation of Hermite Normal Forms of Integer Matrices. In *Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation, ISSAC '96, Zurich, Switzerland, July 24-26, 1996*, Erwin Engeler, B. F. Caviness, and Yagati N. Lakshman (Eds.). ACM, 259–266. http://dl.acm.org/citation.cfm?id=236869

[23] Éva Tardos. 1986. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research* 34, 2 (1986), 250–256.

[24] William S. Zwicker. 2016. Introduction to the theory of voting. In *Handbook of Computational Social Choice*, Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia (Eds.). Cambridge University Press, 23–56.