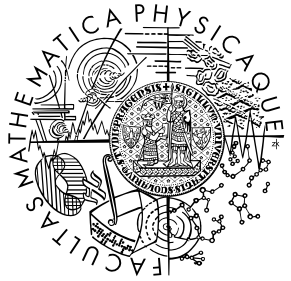


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

Parameterized Algorithms for Block Structured Integer Programs

Martin Koutecký
Computer Science Institute, MFF UK
Malostranské náměstí 25, 118 00 Praha 1
koutecky@iuuk.mff.cuni.cz

HABILITATION THESIS

IN THE FIELD COMPUTER SCIENCE –
– THEORETICAL COMPUTER SCIENCE

September 1, 2024

DEDICATION

First, I would like to thank all my collaborators. I have never hoped my job will be so much fun, and it is largely thanks to having the honor of working with you. (This is also the sole explanation of the fact that I have only one single-author paper.) I also thank all the members of KAM and IÚUK for the stimulating, supportive, and friendly environment. Many thanks also to my students, working with you is exciting and fulfilling, and I am grateful that you have chosen me. Last but not least, I would like to thank my family for their grounding presence and support. Special thanks to Christopher Robin.

The research in this thesis was supported by the following grants and projects: Czech Science Foundation Grants 14-10003S, 15-11559S, GX19-27871X, 22-22997S, and CE-ITI P202/12/G061, grant 308/18 from the Israel Science Foundation, Charles University grants GAUK 1784214 and GAUK 338216, and the projects UNCE/SCI/004 and UNCE 24/SCI/008 (Center of Modern Computer Science) of Charles University.

Soli Deo Gloria

THESIS CONTENT

This thesis consists of the following papers, grouped into corresponding sections.

Section 2: (Strongly) Fixed-Parameter Tractable Algorithms for (ILP) **and** (IP)

[H1] Martin Koutecký, Asaf Levin, and Shmuel Onn. A parameterized strongly polynomial algorithm for block structured integer programs. In *Proc. ICALP 2018*, volume 107 of *Leibniz Int. Proc. Informatics*, pages 85:1–85:14, 2018

[H2] Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. Sparse integer programming is fixed-parameter tractable. *Mathematics of Operations Research*, 0(0):null, 0. doi:10.1287/moor.2023.0162

This paper is currently *"Ahead of Print"*, but already available online at the journal's website.

Section 3: High-multiplicity n-fold (IP)

[H3] Dušan Knop, Martin Koutecký, Asaf Levin, Matthias Mnich, and Shmuel Onn. High-multiplicity N-fold IP via configuration LP. *Math. Program.*, 200(1):199–227, 2023. URL: https://doi.org/10.1007/s10107-022-01882-9, doi:10.1007/S10107-022-01882-9

Section 4: Mixed (ILP) **and** (IP)

[H4] Cornelius Brand, Martin Koutecký, and Sebastian Ordyniak. Parameterized algorithms for MILPs with small treedepth. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12249–12257. AAAI Press, 2021. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17454

[H5] Cornelius Brand, Martin Koutecký, Alexandra Lassota, and Sebastian Ordyniak. Separable convex mixed-integer optimization: Improved algorithms and lower bounds. In Timothy Chan, Johannes Fischer, and John Iacono, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, September 2-4, 2024, London, United Kingdom*, volume 308 of *LIPIcs*, pages 32:1–32:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024

Section 5: Sparsity Beyond Treedepth

[H6] Timothy F. N. Chan, Jacob W. Cooper, Martin Koutecký, Daniel Kráľ, and Kristýna Pekárková. Matrices of optimal tree-depth and a row-invariant parameterized algorithm for integer programming. *SIAM J. Comput.*, 51(3):664–700, 2022. doi:10.1137/20m1353502

[H7] Marcin Briański, Martin Koutecký, Daniel Kráľ, Kristýna Pekárková, and Felix Schröder. Characterization of matrices with bounded graver bases and depth parameters and applications to integer programming. *Mathematical Programming*, pages 1–35, 2024. doi:10.1007/s10107-023-02048-x

This paper is currently an *"Article in Press'*, not yet assigned a volume, but already available at the journal's website and indexed by Scopus.

Section 6: Beyond Small Coefficients and Graver Bases – Coming Full Circle

[H8] Jana Cslovjecsek, Martin Koutecký, Alexandra Lassota, Michał Pilipczuk, and Adam Polak. Parameterized algorithms for block-structured integer programs with large entries. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 740–751. SIAM, 2024

1 INTRODUCTION

Imagine you want to start a burrito truck business. A survey of the city gives you an estimate of roughly how much demand to expect from each neighborhood, and you know how much it costs you to produce one burrito, for how much it sells, and how much it costs to buy one food truck. Each truck captures an amount of demand which depends on its proximity to the respective neighborhoods. In order to maximize your profit, you must strike some balance: if you buy more trucks, you capture more demand, but also incur more costs. Of course, not just the number of trucks, but even more their placement matters. Moreover, the situation may develop over time, and you may want to move your trucks (after all, they can drive) to respond to changes in demand. (An interested reader may try their skill in the cute and educational Burrito Optimization Game at https://www.gurobi.com/burrito-optimization-game/.)

This is one of a myriad of problems which broadly fall into an area called *Operations Research*[1]. Precisely *because* there are so many optimization problems, each usually with several variants (e.g., if we were placing factories instead of trucks, we would *not* be able to move them around easily as demand changes), attention has historically focused on certain *meta-problems* or optimization paradigms which are capable of modeling a wide range of real-world applications. One of the most important of these is the *Integer Programming* problem, which is the main subject of this thesis.

In an integer program, a solution is described by a vector of variables, each of which can take only integer values. For example, in our burrito truck problem, we may have a variable x_p for each possible location p of a truck, and x_p would equal 1 if a truck is placed at location p and 0 otherwise. Which solution is or is not feasible is described by a system of linear inequalities. In our example, allowing only the values 0 and 1 for the x_p variables can be achieved by introducing the inequalities $x_p \geq 0$ and $x_p \leq 1$ for each $p \in P$, where P is the set of all possible locations. (If we allowed for fractional values of x_p, we would be in the realm of *linear programming*, which is a much more tractable problem.) Maybe we do not have the capacity to manage more than 5 trucks; then we would enforce $\sum_{p \in P} x_p \leq 5$. Finally, the goal of the optimization (in our case, maximizing profit) is described by an objective function which is to be minimized or maximized. Expressing profit based on the selected truck locations is a bit more involved, but can be done using the linear inequalities and additional variables.

Clearly, integer programming is a very general paradigm, capable of capturing many real-world problems. There is thus great interest both in practically efficient algorithms as well as in getting a solid theoretical understanding of the problem. Computational complexity theory is the branch of computer science which studies the inherent difficulty of computational problems, and it is the lens which we will use. The most basic distinction in computational complexity is between problems which can be solved in polynomial time and those which likely cannot. (The "likely" part of the previous sentence is the heart of the famous $\mathsf{P} \neq \mathsf{NP}$ problem.) Already since the '70s, it is known that integer programming falls into the second group and a universally effective algorithm for it is unlikely to exist.

As such, much effort has been dedicated to identifying conditions as broad as possible under which integer programming can be solved efficiently. One group of conditions studied since the very beginning of computer science are programs in which the matrix of coefficients of the inequalities has a block structure, for example, where deleting a few rows or few columns results in a matrix which is block-diagonal. A stream of papers by experts in Operations Research published between roughly 2000-2015 has shown increasingly general and effective algorithms for block-structured integer programs.

This thesis is about how we have unified, further extended, generalized, and popularized this line of study by connecting it to other areas of theoretical computer science (and the respective communities) such as parameterized complexity and graph and matroid theory. The result of our work is not only in more effective algorithms for much more general classes of integer programs, but also in a much clearer understanding of the boundaries of efficiency, and the true underlying reasons for them.

[1]We omit references in this first page of the introduction to keep its flow natural and uninterrupted. The remainder of the thesis is fully referenced, and all references that would have appeared in these first few paragraphs do appear in the later parts of the thesis.

1.1 Integer Programming

Formally, the **integer programming** problem is

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

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 (shown by Karp [71]) already in the special case when $f(\mathbf{x}) = \mathbf{w}\mathbf{x}$ is a linear objective function for some vector $\mathbf{w} \in \mathbb{Z}^n$:

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

Despite this, it can be often solved efficiently in practice and has become a central optimization paradigm in many application domains, e.g., problems related to planning [106, 109], vehicle routing [104], process scheduling [38], packing [84], and network hub location [1]. To get a feeling for the breadth of the applicability of (IP), one can for example visit the Case Studies[2] page of the Gurobi solver.

Because of its importance, (IP) has been studied from many perspectives, including heuristics [47], or practical methods such as cutting planes [86], branch-and-cut [112, Section 9.6], etc. We are interested in identifying broad subclasses of (IP) which can be solved to exact optimality efficiently. To do so, we take the perspective of *parameterized complexity* [20]: given an instance with encoding length L, we want to understand how the running time of an algorithm scales with the size of the input L and with some additional parameter k of the instance. An algorithm is said to be *fixed-parameter tractable* (FPT) if its running time is bounded by $g(k)\text{poly}(L)$ for some computable function g. A problem which is W[1]*-hard parameterized by* k is unlikely to admit an FPT algorithm, and in this case, one seeks the weaker notion of a *slice-wise polynomial* (XP) algorithm, whose complexity is bounded by $L^{g(k)}$. Note that the FPT vs. XP difference provides a finer distinction between algorithms which are both "polynomial for fixed k", a common claim in the literature, and indeed an important part of our work (not covered here) has been deciphering which algorithms in the literature are FPT, and which are (only) XP [44]. A problem is unlikely to admit an XP algorithm if it is *para*-NP*-hard*, meaning that it is NP-hard already for some constant value of the parameter. Another important notion is that of a *strongly polynomial algorithm*, which is an algorithm that makes a number of arithmetic operations independent of the magnitude of numbers on input; in the case of (IP), this means that the number of arithmetic operations is bounded by a function of n only.

Before moving on to the tractable classes of (IP) studied by us, let us briefly mention three other important "islands of tractability" studied in the literature. Already in 1956, Hoffman and Kruskal [62] have shown that if A is totally unimodular, that is, all of its subdeterminants are between -1 and 1, then (ILP) coincides with its continuous relaxation. Combined with the polynomiality of linear programming, this implies that (ILP) can be solved in polynomial time when A is totally unimodular. A natural generalization of totally unimodular matrices are Δ*-modular* matrices, whose subdeterminants are between $-\Delta$ and Δ for some fixed Δ. Artmann et al. [3] have shown that (ILP) is strongly polynomial when A is 2-modular, and other partial results are known [37, 49, 89], but the question of whether (ILP) with a Δ-modular matrix is FPT parameterized by Δ remains open.

A different tractable class of (IP) is formed by instances with a small number n of variables, as first shown by Lenstra [83]. His algorithm was subsequently improved by Kannan [68] and generalized by Grötschel, Lovász, and Schrijver [119] to optimize any convex function over the integers contained in any convex set. The state-of-the-art is due to Reis and Rothvoss [98] who have shown that (ILP) can be solved in time $(\log n)^{O(n)}\text{poly}(L)$; the important question whether an algorithm with complexity $2^{O(n)}\text{poly}(L)$ exists remains open.

About the same time as Lenstra's result, Papadimitriou [94] has shown that the textbook dynamic programming algorithm for Knapsack can be generalized to (ILP), achieving complexity $(m\|A\|_\infty)^{O(m^2)}\text{poly}(L)$. This is FPT parameterized by the combined parameter $\|A\|_\infty + m$, that is, when A has few rows and small coefficients in absolute value.

[2] https://www.gurobi.com/case_studies/

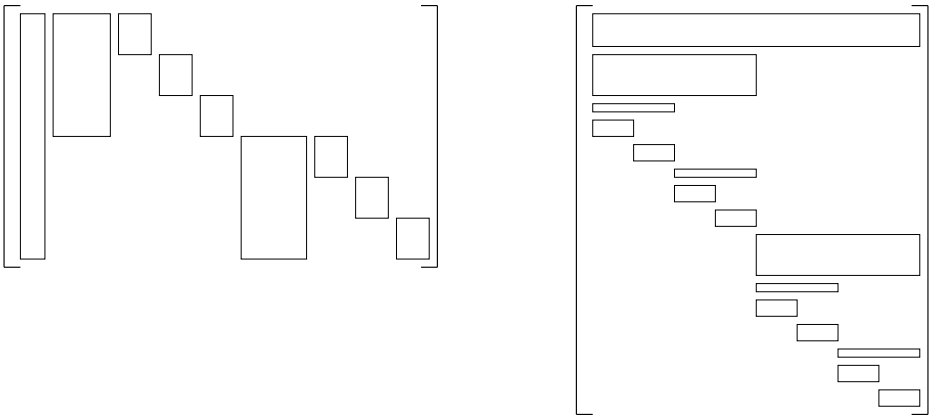


Fig. 1. On the left, a schematic multi-stage with three levels is presented. On the right, a schematic tree-fold with 4 layers is pictured. All entries within a rectangle can be non-zero, all entries outside of the rectangles must be zero.

1.2 Block-structured Integer Programs

Our main contribution is in a systematic and thorough study of algorithmic aspects of block-structured (IP). We have unified a stream of research spanning over two decades and present here time complexity improvements, gneralizations to wider classes, and complementary lower bound results. Besides the results contained in this thesis, we have also shown how our improved algorithms can be used to obtain efficient algorithms for problems in scheduling, stringology, graph theory, and computational social choice [34, 44, 74–79, 81].

The purpose of this section is to introduce the classes of block-structured IPs which we study in this thesis, and briefly explain the motivation and history of their study. Our point of departure are two classes of block-structured integer programs: n-fold and 2-stage stochastic IPs. A matrix $A^{(n)}$ is n-fold with blocks A_1, A_2 if it has the form

$$A^{(n)} = \begin{pmatrix} A_1 & A_1 & \cdots & A_1 \\ A_2 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_2 \end{pmatrix}, \tag{1}$$

for $A_1 \in \mathbb{Z}^{r \times t}$ and $A_2 \in \mathbb{Z}^{s \times t}$. To be precise, the above may be called a "uniform" n-fold matrix, as opposed to a "generalized" n-fold matrix in which the blocks may differ. We choose to deal with the uniform case here because **a)** it simplifies exposition, **b)** all presented results developed for the uniform case can be carried over to the generalized case (with one well-justified exception at the end), and, **c)** qualitatively and with respect to the relevant parameterizations (again, with one exception), the uniform case is without loss of generality, because the generalized case can be reduced to the uniform case by suitable encoding tricks. On the other hand, we should note that the general case is more appropriate when dealing with applications.

A 2-stage matrix $B^{(n)}$ is the transpose of an n-fold matrix, i.e.,

$$B^{(n)} = \begin{pmatrix} A_1 & A_2 & & \\ \vdots & & \ddots & \\ A_1 & & & A_2 \end{pmatrix}, \tag{2}$$

We will also consider generalizations of both classes, so called *tree-fold* and *multi-stage stochastic* IPs, whose matrices have a recursive structure. For example, a tree-fold matrix has the form (1), except each block A_2 is itself a tree-fold matrix. For a schematic depiction, see Figure 1

1.3 N-fold Integer Programming

N-fold (IP) was introduced under this name by De Loera et al. [24] in 2008, but programs of this form have appeared in the literature much earlier. The transportation problem, which asks for an optimal routing from several sources to several destinations, has been defined by Hitchcock [58] in 1941 and independently studied by Kantorovich [69] in 1942, and Dantzig [21] showed how the simplex method can be applied to it in 1951. The transportation problem may be seen as a *table problem* where we are given m row-sums and n column-sums and the task is to fill in non-negative integers into the table so as to satisfy these row- and column-sums. A natural generalization to higher-dimensional tables, called *multiway tables*, has been studied already in 1947 by Motzkin [88]. It also has applications in privacy in databases and confidential data disclosure of statistical tables, see a survey by Fienberg and Rinaldo [36] and the references therein.

Specifically, the three-way table problem is to decide if there exists a non-negative integer $l \times m \times n$ table satisfying given line-sums, and to find the table if there is one. Deciding the existence of such a table is NP-complete already for $l = 3$ [25]. Moreover, every bounded integer program can be isomorphically represented in polynomial time for some m and n as some $3 \times m \times n$ table problem [26]. The complexity with l, m parameters and n variable thus became an interesting problem. Let the input line-sums be given by vectors $\mathbf{u} \in \mathbb{Z}^{ml}$, $\mathbf{v} \in \mathbb{Z}^{nl}$ and $\mathbf{w} \in \mathbb{Z}^{nm}$. Observe that the problem can be formulated as an (IP) with variables $x^i_{j,k}$ for $i \in [n]$, $j \in [m]$ and $k \in [l]$, $f \equiv 0$, and the following constraints:

$$\begin{aligned}
\sum_{i=1}^{n} x^i_{j,k} &= u_{j,k} && \forall j \in [m], k \in [l], \\
\sum_{j=1}^{m} x^i_{j,k} &= v^i_k && \forall i \in [n], k \in [l], \\
\sum_{k=1}^{l} x^i_{j,k} &= w^i_j && \forall i \in [n], j \in [m], \\
\mathbf{x} &\geq \mathbf{0} && .
\end{aligned}$$

Written in matrix form, it becomes $A\mathbf{x} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$ with $\mathbf{b} = (\mathbf{u}, \mathbf{v}^1, \mathbf{w}^1, \mathbf{v}^2, \mathbf{w}^2, \ldots, \mathbf{v}^n, \mathbf{w}^n)$, I_k the $k \times k$, $k \in \mathbb{N}$, identity matrix, $\mathbf{1}_k$ the all-ones vector of dimension $k \in \mathbb{N}$, and with

$$A = \begin{pmatrix} I_{ml} & I_{ml} & \cdots & I_{ml} \\ J & & & \\ & J & & \\ & & \ddots & \\ & & & J \end{pmatrix}, \text{ where } J = \begin{pmatrix} I_l & \cdots & I_l \\ \mathbf{1}_l & & \\ & \ddots & \\ & & \mathbf{1}_l \end{pmatrix} \in \mathbb{Z}^{(l+m)\times ml} .$$

Here, J has m diagonal blocks $\mathbf{1}_l$ and A has n diagonal blocks J. Thus. A can be viewed either as an n-fold matrix, or as a tree-fold matrix with 3 levels.

A different implicit appearance of the n-fold structure is in the famous[3] 1961 and 1963 papers by Gilmore and Gomory [45, 46] on solving the Cutting Stock problem. Their work has been essential in regards to fundamental notions like column generation, cutting planes, and the *Configuration LP*, whose depiction from the 1963 paper [46] appears in Figure 2. It is clear that the presented matrix is n-fold, and we will return to it in Section 3. In fact, the significance of the n-fold structure in packing and scheduling problems seems to have been lost until a brief mention in the paper introducing n-fold IP [24], and has only been taken up in earnest [13, 14, 51, 65, 66, 73, 75, 76] in the wake of our paper from 2018 [74]. Let us now briefly describe this connection.

The problem of *uniformly related machines makespan minimization*, denoted $Q||C_{\max}$ in the standard notation, is the following. We are given m machines, each with speed $0 < s_i \leq 1$, and n jobs, where the j-th

[3] Over 2,800 and 1,700 citations, respectively, according to Google Scholar, at the time of submission.

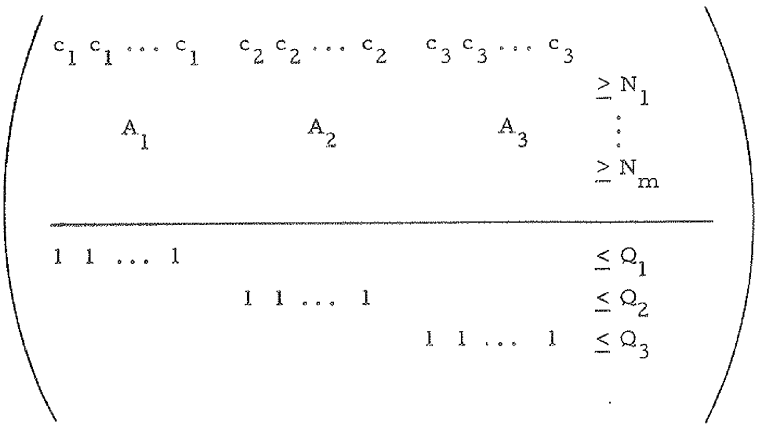


Fig. 2. A figure from "A Linear Programming Approach to the Cutting-Stock Problem—Part II" by Gilmore and Gomory [46] depicting the Configuration LP (not yet under this name) of the "Machine Balance Problem".

job has processing time $p_j \in \mathbb{N}$ and processing it on machine i takes time p_j/s_i. The task is to assign jobs to machines such that the time when the last job finishes (the *makespan*) is minimal, i.e., if M_i is the set of jobs assigned to machine i, the task is to minimize $\max_{i\in[m]} \sum_{j\in M_i} p_j/s_i$. The decision version of the problem asks whether there is a schedule of makespan $C_{\max} \in \mathbb{R}$. We consider the scenario when $p_{\max} = \max_j p_j$ is bounded by a parameter and the input is represented *succinctly* by multiplicities $n_1, \ldots, n_{p_{\max}}$ of jobs of each length, i.e., n_ℓ is the number of jobs with $p_j = \ell$. Letting x_j^i be a variable representing the number of jobs of length j assigned to machine i, Knop and Koutecký [74] give the following n-fold formulation:

$$\sum_{i=1}^{m} x_j^i = n_j \qquad \forall j \in [p_{\max}], \tag{3}$$

$$\sum_{j=1}^{p_{\max}} j \cdot x_j^i \leq \lfloor s_i \cdot C_{\max} \rfloor \qquad \forall i \in [m] \ . \tag{4}$$

Constraints (3) ensure that each job is scheduled on some machine, and constraints (4) ensure that each machine finishes before time $C_{\max}$. This corresponds to an n-fold formulation with $A_1 = I_{p_{\max}}$ and $A_2 = (1, 2, \ldots, p_{\max})$ and with $\|A^{(n)}\|_\infty = p_{\max}$.

Another scheduling problem is finding a schedule minimizing the *sum of weighted completion times* $\sum w_j C_j$. Knop and Koutecký [74] show an n-fold formulation for this problem as well, in particular one which has a separable quadratic objective. In the context of scheduling, what sets methods based on n-fold (IP) apart from other results is that they allow the handling of many "types" of machines (such as above where machines have different speeds) and also "non-linear" objectives (such as the quadratic objective in the formulation for $\sum w_j C_j$).

Another field where n-fold (IP) has had an impact is computational social choice. The problem of Bribery asks for a cheapest manipulation of voters which lets a particular candidate win an election. An FPT algorithm was known for Bribery parameterized by the number of candidates which relied on Lenstra's algorithm. However, this approach has two downsides, namely a time complexity which is doubly-exponential in the parameter, and the fact that voters have to be "uniform" and cannot each have an individual cost function. Knop et al. [79] resolved this problem using n-fold (IP) by showing a single-exponential algorithm for many Bribery-type problems, even in the case when each voter has a different cost function.

Lastly, the Dantzig-Wolfe decomposition [22] is an algorithm for solving linear programs with an n-fold structure which decomposes the problem into a master problem and a series of subproblems corresponding

to the individual blocks. One important example of a problem which can be solved by this method is the Multicommodity Flow problem, where the matrix A_2 of each block corresponds to the incidence matrix of an input graph G, and the matrix A_1 is the identity matrix. In this way, the "coupling constraints" defined by the blocks A_1 ensure that the total flow, summed up across all commodities, does not exceed the capacity of any edge.

1.4 2-stage Stochastic Integer Programming

The motivation for the study of 2-stage stochastic matrices comes from decision making under uncertainty. Here, one is asked to make a partial decision in a "first stage", and after realization of some random data, one has to complete their decision in a "second stage". The goal is minimizing the "direct" cost of the first-stage decision plus the expected cost of the second-stage decision. Random data are often modeled by a finite set of *n scenarios*, each with a given probability. Assume that the scenarios are represented by integer vectors $\mathbf{b}^1, \dots, \mathbf{b}^n \in \mathbb{Z}^t$, their probabilities by $p_1, \dots, p_n \in (0, 1]$, the first-stage decision is encoded by a variable vector $\mathbf{x}^0 \in \mathbb{Z}^r$, and the second-stage decision for scenario $j \in [n]$ is encoded by a variable vector $\mathbf{x}^j \in \mathbb{Z}^s$. Setting $\mathbf{x} \coloneqq (\mathbf{x}^0, \mathbf{x}^1, \dots, \mathbf{x}^n)$ and $\mathbf{b} \coloneqq (\mathbf{b}^1, \dots, \mathbf{b}^n)$ then makes it possible to write this problem as

$$\min \mathbf{w}^0\mathbf{x}^0 + \sum_{j=1}^{n} p_j \mathbf{w}'\mathbf{x}^j : B^{(n)}\mathbf{x} = \mathbf{b},\ \mathbf{l} \le \mathbf{x} \le \mathbf{u},\ \mathbf{x} \in \mathbb{Z}^{r+ns}, \text{ where } B^{(n)} = \begin{pmatrix} A_1 & A_2 & & \\ \vdots & & \ddots & \\ A_1 & & & A_2 \end{pmatrix}, \tag{5}$$

with $A_1 \in \mathbb{Z}^{t\times r}$, $A_2 \in \mathbb{Z}^{t\times s}$, and $\mathbf{l}, \mathbf{u} \in \mathbb{Z}^{r+ns}$ some lower and upper bounds. Problem (5) is called 2-*stage stochastic* (IP) and finds many applications in various areas, see [9, 57, 67, 96, 99] and references therein.

Similarly to the Dantzig-Wolfe decomposition method for problems with an n-fold structure, a decompositional method for problems with a 2-stage structure has been introduced by Benders [6] in 1962. Where Dantzig-Wolfe introduces new columns in each iteration (leading to "column generation"), Benders introduces new rows (leading to "row generation"). We should note that both decompositional methods have been originally introduced for *continuous* problems. Still, for both Dantzig-Wolfe [107] and Benders [103] a generalization to the (mixed) integer cases have been studied.

Integer programs with block structure have also been the subject of at least two habilitations, that of Martin [87] and Nowak [91], which contain many other interesting and relevant references. This thesis differs by taking a more theoretical perspective, in particular through the lens of parameterized complexity.

2 (STRONGLY) FIXED-PARAMETER TRACTABLE ALGORITHMS FOR (ILP) AND (IP)

This section is based on the papers **[H1]** and **[H2]**. Note that even though **[H1]** has been published in 2018 and **[H2]** only in 2024, all the results have been obtained in 2017-2018 and both logically as well as chronologically precede the papers **[H3]**-**[H8]**.

2.1 State of the Art in 2018

The developments regarding parameterized complexity of block-structured IPs until 2018 were as follows. Whenever we talk about FPT algorithms in this section, the parameter is the largest coefficient in absolute value $\|A\|_\infty$, and the sizes of the blocks composing the block-structured matrix at hand. The key notion behind most of the results reviewed here is the *Graver basis of the matrix* A, $\mathcal{G}(A)$, which is the set of non-zero integer vectors $\mathbf{g}$ satisfying $A\mathbf{g} = \mathbf{0}$ that cannot be decomposed into two non-zero integer vectors with the same sign-pattern (i.e., belonging to the same orthant) satisfying the same equation. These *Graver basis elements* are thus called *indecomposable*, *irreducible*, or *conformal-minimal*. The Graver basis allows a simple *iterative augmentation* procedure: given a feasible solution $\mathbf{x}$, either $\mathbf{x} + \mathbf{g}$ for some $\mathbf{g} \in \mathcal{G}(A)$ is feasible and improves the objective function, or $\mathbf{x}$ is already optimal.

2-stage and multi-stage stochastic (IP). Already in 2003, Hemmecke and Schultz have shown that 2-stage stochastic (IP) is FPT [56], although this FPT bound was not explicit. At the heart of their algorithm is the following finiteness result about $\mathcal{G}(A)$. For a number n, 2-stage stochastic matrix $B^{(n)}$, and a vector $\mathbf{x} = (\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^n)$, call the subvector $\mathbf{x}^0$ the *global brick*, and the subvectors $\mathbf{x}^1, \ldots, \mathbf{x}^n$ the *local bricks*. Then, there exists a number n_0 such that for every $n \geq n_0$, the set of possible global and local bricks that appear in any $\mathbf{g} \in \mathcal{G}(B^{(n)})$ is fixed and independent of n. Moreover, they provide an algorithm to compute this finite set of bricks. A simple branching algorithm then allows one to "guess" the global brick, compute, for each block, the optimal "matching" local brick, and in this way find an optimal Graver basis element. Combining this with the iterative augmentation procedure then yields an FPT algorithm. (This algorithm may take a long time to converge, but improving convergence is a separate and independent topic which we leave aside for now.) Since the aforementioned finiteness result is obtained by a saturation argument [85] and does not provide an explicit upper bound, there was no explicit upper bound on the complexity of the algorithm. This result, using an analogous technique, was extended to multi-stage stochastic (IP) by Aschenbrenner and Hemmecke [4].

N-fold and tree-fold (IP). The story was more complicated for n-fold (IP). Call the subvectors $\mathbf{x}^1, \ldots, \mathbf{x}^n$ of an nt-dimensional vector $\mathbf{x}$ *bricks*. N-fold (IP) has been introduced by De Loera et al. [24] in 2008, and it is in this paper where the fundamental structural result about $\mathcal{G}(A^{(n)})$ has been stated: the number of non-zero bricks of any $\mathbf{g} \in \mathcal{G}(A^{(n)})$ is independent of n, and so are their values. The arguments needed for this claim have been already developed in 2003 (by Santos and Sturmfels [100]) and 2007 (by Hoşten and Sullivant [63]). A particularly ingenious trick appears in the paper by Santos and Sturmfels [100], where they show that the structure of the Graver basis of $\mathcal{G}(A^{(n)})$ can be understood through what is essentially *the Graver basis of the Graver basis of* A_2*:* $\mathcal{G}(A_1\mathcal{G}(A_2))$. Quoting from [100], *"The phrase 'the Graver basis of the Graver basis' is not a typo but it is the punchline".*

Still, the 2008 algorithm of [24] only had complexity $n^{g(A_1,A_2)}\text{poly}(L)$, where g is some computable function, so this is an XP, not FPT, algorithm. A breakthrough was achieved by Hemmecke, Onn, and Romanchuk [55] in 2013, where they combined the ideas from [24] with a Knapsack-like dynamic programming algorithm (in the spirit of Papadimitriou [94]) to obtain a first FPT algorithm for n-fold (IP). This algorithm was highlighted in the textbook of Downey and Fellows [28], and it attracted much attention after we have shown its wide applicability in scheduling [74], computational social choice [79], and other areas [78]. Spurred by these results, Chen and Marx [13] have introduced tree-fold (IP) in early 2018, and have shown that it is FPT as well.

Graph Parameters. The notion of sparsity has enjoyed tremendous success in graph theory [90, 95], and it is natural to ask about its applicability to (IP). We focus on two graphs which can be associated with A:

- the *primal graph* $G_P(A)$, which has a vertex for each column and two vertices are connected if there exists a row such that both columns are non-zero, and,
- the *dual graph* $G_D(A) = G_P(A^\intercal)$, which is the above with rows and columns swapped.

Two standard parameters of structural sparsity are the *treewidth* (measuring the "tree-likeness" of a graph) and the more restrictive treedepth (measuring its "star-likeness"). We focus on the latter here: the *treedepth* of a graph G denoted $\mathrm{td}(G)$ is the smallest height of a rooted forest F such that each edge of G is between vertices which are in a descendant-ancestor relationship in F. The *primal treedepth of* A is $\mathrm{td}_P(A) := \mathrm{td}(G_P(A))$, and analogously the *dual treedepth of* A is $\mathrm{td}_D(A) := \mathrm{td}(G_D(A))$. We will not define treewidth, but we shall denote the treewidth of $G_P(A)$ and $G_D(A)$ by $\mathrm{tw}_P(A)$ and $\mathrm{tw}_D(A)$, respectively, and we note that bounded treedepth implies bounded treewidth but not vice versa.

It follows from Freuder's algorithm [40] and was reproven by Jansen and Kratsch [64] that (IP) is FPT parameterized by the primal treewidth $\mathrm{tw}_P(A)$ and the largest variable domain $\|\mathbf{u} - \mathbf{l}\|_\infty$. Regarding the dual graph $G_D(A)$, the parameters $\mathrm{td}_D(A)$ and $\mathrm{tw}_D(A)$ were only recently considered by Ganian et al. [43]. They show that even deciding feasibility of (IP) is NP-hard on instances with $\mathrm{tw}_I(A) = 3$ ($\mathrm{tw}_I(A)$ denotes the treewidth of the *incidence graph*; $\mathrm{tw}_I(A) \leq \mathrm{tw}_D(A) + 1$ always holds). Furthermore, they show that (IP) is FPT parameterized by $\mathrm{tw}_I(A)$ and parameter Γ, which is an upper bound on any prefix sum of $A\mathbf{x}$ for any feasible solution $\mathbf{x}$. Dvořák et al. [29] introduce the parameter fracture number. Having a bounded *variable fracture number* $\mathfrak{p}^V(A)$ implies that deleting a few columns of A breaks it into independent blocks of small size, similarly for *constraint fracture number* $\mathfrak{p}^C(A)$ and deleting a few rows. Having a bounded $\mathfrak{p}^V(A)$ is essentially equivalent to having a generalized n-fold structure, and similarly having a bounded $\mathfrak{p}^C(A)$ corresponds to A being a generalized 2-stage stochastic. Indeed, the algorithms presented by [29] are based on reducing the problem to the n-fold and 2-stage cases and applying the algorithms reviewed above.

2.2 Unified Approach and Treedepth

Denote by $\langle A, f, \mathbf{b}, \mathbf{l}, \mathbf{u}\rangle$ the binary encoding length of an (IP) instance. (We define the encoding length of f to be the length of f_{gap}, which is the difference between the maximum and minimum values of f on the domain.) The function f is given by an oracle. A glaring question left open by Ganian and Ordyniak [42] is that of the complexity of (IP) parameterized by $\|A\|_\infty$ and $\mathrm{td}_P(A)$ or $\mathrm{td}_D(A)$. In **[H1]**, we have fully settled this with the following result:

Theorem 1. *There exists a computable function g such that problem* (IP) *can be solved in time*

$$g(\|A\|_\infty, d)\mathrm{poly}(n, L), \qquad \textit{where } d := \min\{\mathrm{td}_P(A), \mathrm{td}_D(A)\} \textit{ and } L := \langle A, f, \mathbf{b}, \mathbf{l}, \mathbf{u}\rangle \ .$$

Note that for our algorithm to be fast it suffices if at least one of $\mathrm{td}_P(A)$ and $\mathrm{td}_D(A)$ is small. Also, all the presented results hold for (IP) whose constraints are given in the inequality form $A\mathbf{x} \leq \mathbf{b}$: introducing slack variables leads to (IP) in standard form with a constraint matrix $A_I := (A\ I)$, with $\min\{\mathrm{td}_P(A_I), \mathrm{td}_D(A_I)\} \leq \min\{\mathrm{td}_P(A) + 1, \mathrm{td}_D(A)\}$.

Our result is qualitatively optimal in the following sense. Already for $\|A\|_\infty = 1$ or $d = 1$ (ILP) is NP-hard: by the natural reduction from SAT in the former case, and similarly by the natural modeling of Knapsack in the latter. Moreover, the two arguably most important tractable classes of (IP) are formed by instances whose constraint matrix is either totally unimodular or has small number n of columns, yet our results are incomparable with either: the class of totally unimodular matrices might have large d, but has $\|A\|_\infty = 1$, and the matrices considered here have variable n.

Furthermore, treedepth cannot be replaced with the more permissive notion of treewidth, since (IP) is NP-hard already when $\min\{\mathrm{tw}_P(A), \mathrm{tw}_D(A)\} = 2$ and $a = 2$ [32]. Second, the parameterization cannot be relaxed by removing the parameter $\|A\|_\infty$: (IP) is para-NP-hard parameterized by $\mathrm{td}_P(A)$ [29, Thm 21] and strongly W[1]-hard parameterized by $\mathrm{td}_D(A)$ [79, Thm 5] alone. Third, the requirement that f is separable convex cannot be relaxed since (IP) with a non-separable convex or a separable concave function are NP-hard even for small values of our parameters [32]. Let us now sketch our approach towards proving Theorem 1.

Treedepth is Equivalent to Block Structure. The first important ingredient is showing that, assuming small $\|A\|_\infty$, matrices with small primal treedepth are essentially equivalent to multi-stage stochastic matrices, and similarly that matrices with small dual treedepth are essentially equivalent to tree-fold matrices. Thus, in a certain sense, it suffices to restrict our attention to the (more rigidly structured) classes of multi-stage stochastic and tree-fold matrices.

Graver-best Augmentation. The second ingredient is an abstraction of an iterative augmentation procedure used in [55] and elsewhere: a *Graver-best oracle* for a matrix A is one which, when queried on an (IP) instance $(A, f, \mathbf{b}, \mathbf{l}, \mathbf{u})$ and a feasible solution $\mathbf{x}$, returns a feasible step $\mathbf{h}$ which is at least as good as any feasible step $\lambda\mathbf{g}$ where $\mathbf{g} \in \mathcal{G}(A)$ and $\lambda \in \mathbb{N}$. Given such an oracle, we can solve (IP) quickly by iteratively querying it on the current solution and adding the returned step to the solution. Thus, constructing efficient Graver-best oracles is a key to obtaining efficient algorithms for (IP).

Norm Bounds and Local Search. We identify that constructing efficient Graver-best oracles requires two components. The first are bounds on the norms of Graver basis elements. Denote by $g_\infty(A) = \max_{\mathbf{g}\in\mathcal{G}(A)} \|\mathbf{g}\|_\infty$, and similarly $g_1(A) = \max_{\mathbf{g}\in\mathcal{G}(A)} \|\mathbf{g}\|_1$. We show that the structural result of Aschenbrenner and Hemmecke [4] can be interpreted as saying that $g_\infty(A)$ is bounded by a function of $\|A\|_\infty$ and $\mathrm{td}_P(A)$, and similarly that the work of Chen and Marx [13] implies that $g_1(A)$ is bounded by a function of $\|A\|_\infty$ and $\mathrm{td}_D(A)$.

Given these norm bounds, it becomes apparent that computing Graver-best steps boils down to solving a local search problem. This second component can be obtained by a relatively simple branching algorithm in the case of $\mathrm{td}_P(A)$, and by a more involved dynamic programming algorithm in the case of $\mathrm{td}_D(A)$.

2.3 Strongly Polynomial Framework

A fundamental question regarding problems involving large numbers is whether there exists an algorithm whose number of arithmetic operations does not depend on the length of the numbers involved; recall that if this number is polynomial, this is a *strongly polynomial algorithm* [102]. For example, the ellipsoid method or the interior-point method which solve LP take time which does depend on the encoding length, and the existence of a strongly polynomial algorithm for LP remains a major open problem. Until 2018, the only strongly polynomial (ILP) algorithms existed for totally unimodular (ILP) [61], bimodular (ILP) [3], so-called binet (ILP) [2], and n-fold (ILP) with constant block dimensions [23]. All remaining results, such as Lenstra's famous algorithm or the FPT algorithm for n-fold of Hemmecke et al. [55], are not strongly polynomial.

A second major contribution of **[H1]** besides Theorem 1 is the development of an algorithmic framework among others suitable for obtaining strongly polynomial algorithms, and as a consequence of Theorem 1 we show a strongly polynomial algorithm for (ILP) in the same parameter regime:

Theorem 2. *There exists a computable function g such that problem* (ILP) *can be solved with an algorithm whose number of arithmetic operations is bounded by*

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

The proof of Theorem 2 proceeds in four steps:

1. **LP relaxation:** the LP relaxation can be solved in time $\mathrm{poly}(n \cdot \langle A \rangle)$ by Tardos' algorithm [102], obtaining a fractional optimum $\mathbf{x}^*$.
2. **Proximity:** the integer optimum $\mathbf{z}^*$ which we seek is at an ℓ_∞-distance at most $O(g_\infty(A)n)$ from $\mathbf{x}^*$, thus we may reduce the lower and upper bounds $\mathbf{l}, \mathbf{u}$ to some $\mathbf{l}', \mathbf{u}'$ which are bounded by $O(g_\infty(A)n)$, and subsequently also reduce the right hand side $\mathbf{b}$ to some $\mathbf{b}'$ which is bounded by $g_\infty(A)\mathrm{poly}(n)$.
3. **Objective reduction:** since we now know that the integer optimum $\mathbf{z}^*$ lies in a "small" box $[\mathbf{l}', \mathbf{u}']$, we can apply the coefficient reduction technique of Frank and Tardos [39] to obtain an equivalent objective $\mathbf{w}'$ with $\log \|\mathbf{w}'\|_\infty = \mathrm{poly}(g_\infty(A)n)$.
4. **Convergence:** because the length of all numbers on input is now polynomial in n, the algorithm of Theorem 1 runs in strongly polynomial time.

A nice and somewhat surprising property of this approach is that, since f is given by an oracle and since step 3 reduces it to an equivalent function, this step is actually irrelevant and we may proceed to step 4 directly. Thus, the algorithm boils down to solving the LP relaxation, reducing the lower and upper bounds and right hand side in a straightforward manner, and running the algorithm of Theorem 1. The fact that step 3 is unnecessary is in fact substantial, because while the algorithm of Frank and Tardos [39] is polynomial, the degree of this polynomial is quite large and would dominate the complexity of the algorithm overall. We have expanded on these ideas in [31] by showing that if the equivalent objective need not be computed, then stronger bounds on $\|\mathbf{w}'\|_\infty$ can be obtained, leading to better bounds on the algorithm itself. Furthermore, we have shown reducibility upper bounds on not just linear but also separable convex functions, and also almost matching lower bounds.

2.4 Simplified and Self-contained

The main contribution of **[H2]** over **[H1]** is that it presents a streamlined and self-contained version of the proof of Theorem 1. With the benefit of hindsight, we have replaced Graver-best augmentation with a cheaper so-called halfling augmentation, and we have derived all results (norm bounds, local search programs) directly in the most general setting of bounded treedepth. The whole paper is only 16 pages long including references, and is an ideal entry point into the area for any newcomer, including graduate and even undergraduate students.

3 HIGH-MULTIPLICITY n-FOLD (IP)

Much research following **[H1]** has focused on reducing the run-time dependency on the number of bricks n, making it almost linear in the case of optimizing a linear objective [18, 19]. Our interest in this section is on n-fold (IP) models of applications where many bricks are of the same *type*, that is, they share the same bounds, right-hand side, and objective function. For those applications, it is natural to encode an n-fold (IP) instance *succinctly* by describing each brick type by its constraint matrix, bounds, right-hand side, and objective function, and giving a vector of brick multiplicities. When the number of brick types τ is much smaller than the number n of bricks, e.g., if $n \approx 2^\tau$, this succinct instance is (much) smaller than the "explicit" encoding of n-fold (IP), and an algorithm running in time polynomial in the size of the succinct instance may be (much) faster than current algorithms. We call the n-fold (IP) where the instance is given succinctly the *huge* (or *high multiplicity*) *n-fold* (IP) problem, and in **[H3]** we present a fast algorithm for it:

Theorem 3. *Huge n-fold* (IP) *with any separable convex objective can be solved in time*

$$(\|A\|_\infty rs)^{O(r^2s+rs^2)}\text{poly}(\tau, t, \log \|\mathbf{l}, \mathbf{u}, \mathbf{b}, n, f_{\text{gap}}\|_\infty),$$

where r, s are the numbers of rows of the blocks A_1, A_2, respectively, and t is the number of columns of A_1.

A natural application of Theorem 3 are scheduling problems. In many scheduling problems, the number N of jobs that must be assigned to machines, as well as the number m of machines, are very large, whereas the number of types of jobs and the number of kinds of machines are relatively small. An instance of such a scheduling problem can thus be compactly encoded by simply stating, for each job type and machine kind, the number of jobs with that type and machines with that kind together with their characteristics (like processing time, weight, release time, due date, etc.), respectively. This key observation was made by several researchers [17, 97], until Hochbaum and Shamir [60] coined the term *high-multiplicity scheduling problem*. Clearly, many efficient algorithms for scheduling problems, where all jobs are assumed to be distinct, become exponential-time algorithms for the corresponding high-multiplicity problem.

Let us shortly demonstrate how Theorem 3 allows designing algorithms which are efficient for the succinct high-multiplicity encoding of the input. In modern computational clusters, it is common to have several kinds of machines differing by processing unit type (high single- or multi-core performance CPUs, GPUs), storage type (HDD, SSD, etc.), network connectivity, etc. However, the number of machine kinds τ is still much smaller (perhaps 10) than the number of machines, which may be in the order of tens of thousands or more. Many scheduling problems have n-fold (IP) models [76] where τ is the number of machine kinds and n is the number of machines. On these models, Theorem 3 would likely outperform the currently fastest n-fold (IP) algorithms.

Proof Ideas. To solve a high-multiplicity problem, one needs a succinct way to argue about solutions. The fundamental and widely influential notion of Configuration IP (ConfIP) introduced by Gilmore and Gomory [45] describes a solution (e.g., a schedule) by a list of pairs "(machine schedule s, multiplicity μ of machines with schedule s)". The linear relaxation of ConfIP, called the Configuration LP (ConfLP), can often be solved efficiently, and is known to provide solutions of strikingly high quality in practice [105]; for example, the optimum of the ConfLP for Bin Packing is conjectured to have value x such that an optimal integer packing uses $\leq \lceil x \rceil + 1$ bins [101]. However, surprisingly little is known *in general* about the structure of solutions of ConfIP and ConfLP, and how they relate to each other.

We define the Configuration IP and LP of an n-fold (IP) instance, and show how to solve the ConfLP quickly using the property that the ConfLP and ConfIP have polynomial encoding length even for huge n-fold (IP). Our main technical contribution is a novel proximity theorem about n-fold (IP), showing that a solution of its relaxation corresponding to the ConfLP optimum is very close to the integer optimum. Thus, the algorithm of Theorem 3 proceeds in three steps:

(1) It solves the ConfLP via the standard approach of considering the dual and its separation problem, which in this case turns out to be an efficiently solvable (IP),

(2) It uses the novel proximity theorem to construct a "residual" n'-fold instance with n' upperbounded by $(\|A\|_\infty rs)^{O(rs)}$, and
(3) it solves the residual instance by an existing n-fold (IP) algorithm such as Theorem 1.

Precisely defining the Configuration LP of a huge n-fold (IP) is non-trivial and requires some care. However, the technical heavy lifting is done in the proof of the proximity theorem, and the theorem itself provides insight into the fundamental notion of Configuration LP. Intuitively, it means the following: there is an integer optimum of the Configuration LP which agrees with the continuous optimum on "most" configurations, and where it differs, it only deviates to configurations which are not "too far". In other words, an optimum of the Configuration LP which only uses few configurations implies the existence of an integer optimum which puts "most" weight on these configurations, and puts the remaining weight in "small" balls around these configurations. (Note that a continuous optimum using only few configurations can be found efficiently whenever *some* optimum can be found at all.)

We highlight the proximity theorem also because the strongly near-linear algorithm of Cslovjecsek et al. [18] uses a special case of our relaxation of an n-fold (IP) in order to obtain a similarly tight proximity result as we do, however, their result only holds for linear objectives, and this seems inherent. Our proof is longer (thought not necessarily more complex), but the result is more general in this way. Because our proximity theorem holds also for separable convex objectives, it allowed us in a subsequent work [75] to show efficient preprocessing algorithms (kernels) for certain scheduling problems, including those whose models involve separable convex objectives.

4 MIXED (ILP) AND (IP)

Consider again the celebrated result of Lenstra [83] that (ILP) is FPT parameterized by the number of variables n. Lenstra's brilliant (and short) paper also shows an extension of this result to Mixed ILP where both integer and non-integer variables are allowed:

$$\min\{\mathbf{wx} \mid A\mathbf{x} = \mathbf{b},\, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\,,\, \mathbf{x} \in \mathbb{Z}^z \times \mathbb{R}^q\}\,, \tag{MILP}$$

with $A \in \mathbb{Z}^{m\times(z+q)}$, $\mathbf{l}, \mathbf{u}, \mathbf{w} \in \mathbb{Z}^{z+q}$ and $\mathbf{b} \in \mathbb{Z}^m$. Specifically, Lenstra showed that (MILP) is FPT parameterized by the number of integer variables z, so the number of continuous variables q is allowed to be variable. (MILP) is a prominent modelling tool widely used in practice. For example, Bixby [10] says in his famous analysis of LP solver speed-ups, *"[I]nteger programming, and most particularly the mixed-integer variant, is the dominant application of linear programming in practice."* Given the fact that Lenstra's result extends to the mixed case, it is natural to ask whether the FPT algorithm of Theorem 1 about (IP) with small treedepth and small coefficients can too be extended to the (MILP) case. This is the subject of **[H4]**. In **[H5]**, we explore the more general setting of Mixed IP with separable convex objectives.

4.1 Linear Optimization

The main result of **[H4]** is the following:

Theorem 4. *There exists a computable function g such that problem* (MILP) *can be solved in time*

$$g(\|A\|_\infty, d)\mathrm{poly}(n, L), \qquad \textit{where } d := \min\{\mathrm{td}_P(A), \mathrm{td}_D(A)\},\ \textit{and } L := \langle A, f, \mathbf{b}, \mathbf{l}, \mathbf{u}\rangle\ .$$

We note that our result again extends to the inequality form of (MILP) with constraints of the form $A\mathbf{x} \leq \mathbf{b}$ by the fact that introducing slack variables does not increase treedepth too much. Also, by the techniques of Theorem 2, the algorithm of Theorem 4 can be made strongly FPT.

Proof Ideas. The proof goes by reducing an (MILP) instance to an (ILP) instance whose parameters do not increase too much, and then applying the existing algorithms (e.g., Theorem 2) for (ILP). A key technical result concerns the *fractionality* of an (MILP) instance, which is the minimum of the maxima of the denominators in optimal solutions. For example, it is well-known that the natural LP for the Vertex Cover problem has half-integral optima, that is, there exists an optimum with all values in $\{0, \frac{1}{2}, 1\}$.

Before we delve into the details, let us discuss alternative approaches to obtaining algorithms for (MILP). Lenstra [83] showed how to solve (MILP) with few integer variables using the fact that a projection of a polytope is again a polytope; applying this approach to our case would require us to show that if P is a polytope described by inequalities with small treedepth, then a projection of P also has an inequality description of small treedepth; this is unclear. Hemmecke [53] has studied already in 2003 a mixed-integer analogue of the Graver basis. It is unclear how to apply his approach, however, because it requires bounding the norm of elements of the mixed Graver basis, where the bound obtained by (a strengthening of) [53, Lemma 6.2],[52, Lemma 2.7.2], is polynomial in n, too much to obtain an FPT algorithm. In fact, in **[H5]** we show stronger upper bounds for the mixed Graver basis when m and $\|A\|_\infty$ are small, and for 2-stage matrices, but we also specifically rule out that the mixed Graver basis could be used to prove Theorem 4.

The usual way to go about proving fractionality bounds is via Cramer's rule and a sufficiently good bound on the determinant. As witnessed by any proper integer multiple of the identity, determinants can grow large even for matrices of very benign structure. (For example, the determinant of the $n \times n$ matrix $2I$ is 2^n.) Kotnyek [80] characterised k-integral matrices, i.e., matrices whose solutions have fractionality bounded by k, however it is unclear how his characterisation could be used to show the required bound, so we take a different route. Instead, we analyse carefully the structure of the inverse of the appearing invertible submatrices, allowing us to show that an (MILP) instance with a constraint matrix A has an optimal solution $\mathbf{x}$ whose largest denominator is bounded by $(\|A\|_\infty)^{d!}(d!)^{d!/2}$, where $d = \min\{\mathrm{td}_P(A), \mathrm{td}_D(A)\}$. Intuitively, this means that an LP or (MILP) with small treedepth and coefficients has vertices which are not "too fractional", that is, its vertices lie on the superlattice $\frac{1}{s}\mathbb{Z}^{z+q}$ (more precisely $\mathbb{Z}^z \times \frac{1}{s}\mathbb{Z}^q$) with s not too large. We are not aware of any prior work which lifts a positive result for (ILP) to a result for (MILP) in this way.

We also explore the limits of approaching the problem by bounding the fractionality of inverses: Other (ILP) classes with parameterized algorithms involve constraint matrices with small primal treewidth [64], small incidence treewidth [43], small signed clique-width [30] and 4-block n-fold matrices [54]. Here, we obtain a negative answer: For each of these parameters, there exist families of (MILP) instances with constant parameters, but unbounded fractionality. The produced families also show that our fractionality upper bound is almost optimal: there is an (MILP) instance with $\mathrm{td}_P(A), \mathrm{td}_D(A) = d$, $\|A\|_\infty = 2$, and fractionality 2^{2^d}. Compare this with our upper bound $2^{2^{d+\log d+\log\log d}}$.

The fractionality technique behind Theorem 1 cannot be used to handle separable convex objectives in general, but we show that for one important class of separable convex objectives, the fractionality does *not* increase, specifically piece-wise linear functions with integer breakpoints. For this case we show an FPT result analogous to Theorem 4.

Finally, a fascinating connection has emerged between fractionality bounds, and bounds on the norms of the circuits of a matrix A, which are a subset of the Graver basis. Ekbatani et al. [33] show that if, for any integral $\mathbf{b}$, the polytope $\{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$ is not too fractional, then the norm of the circuits of A is not too large, and vice versa. Since the circuits are a subset of $\mathcal{G}(A)$, and we already have good bounds on $\mathcal{G}(A)$ in the considered regime, the result of Ekbatani et al. implies a fractionality bound; however, our approach yields a tighter bound. The connection can be viewed from the other side as well: our upper bound on the fractionality can be used to show a circuit norm upper bound.

4.2 Separable Convex Optimization

Lenstra's famous algorithm [83] can also be extended, e.g. using [119, Theorem 6.7.9], to the setting of arbitrary convex objective functions, and, by the same arguments as with the step from (ILP) to (MILP), it can be shown that (MIP), which is the problem

$$\min\{f(\mathbf{x}) \mid A\mathbf{x} = \mathbf{b},\, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\,, \mathbf{x} \in \mathbb{Z}^z \times \mathbb{R}^q\}\,, \tag{MIP}$$

with f convex is also FPT parameterized by z. Giving attention to objective functions beyond linear is well justified: Bertsimas et al. note in their spectacular work [8] on the notorious subset selection problem in statistical learning that, over the past decades, algorithmic and hardware advances have elevated *convex* (and therefore, in particular, non-linear) mixed-integer optimization to a comparable level of relevance in applications. Given the encouraging results of **[H4]**, we ask whether the FPT algorithm of Theorem 1 can be extended to the mixed-integer, separable convex case. The *a priori* intuition leans positive: in 1990, Hochbaum and Shanthikumar [61] have published an influential paper titled "Convex separable optimization is not much harder than linear optimization" which shows that this is true in the case of (IP) with a totally unimodular A. Similarly, Chubanov's algorithm [15] reduces (purely continuous) separable convex optimization to a polynomial number of linear optimization problems.

In **[H5]**, we go directly against this intuition, in fact, we show that the mixed integer separable convex case exhibits unexpected behavior both when compared to the fully integer cases, and to the linear cases. We begin by focusing on the regime of few rows and small coefficients, which was first solved in the purely-integer case by Papadimitriou [94]. Our main positive result is an algorithm for (MIP) with few rows and small coefficients:

Theorem 5. *The problem* (MIP) *can be solved in single-exponential time* $(m\|A\|_\infty)^{O(m^2)} \cdot \mathcal{R}$*, where* $\mathcal{R}$ *is the time needed to solve the continuous relaxation of any* (MIP) *with the constraint matrix* A*.*

This improves the current state-of-the-art, double-exponential bound for mixed-integer programs with few rows and small coefficients to single-exponential, even when the target function is non-linear. Until now, the best way to solve a (MIP) with few rows and small coefficients would be to remove duplicate columns from A in a preprocessing step, and then use Lenstra's algorithm [83]. Since there are $2\|A\|_\infty + 1$ numbers of absolute value at most $\|A\|_\infty$, the preprocessing ensures that there are at most $(2\|A\|_\infty + 1)^m$ columns in A. This, however, leads to a *double-exponential* running time in terms of m.

The structural result behind Theorem 5 is an upper bound on the elements of the mixed Graver basis of A. We show this bound using a "packing lemma", powered by an old result from additive combinatorics [92] recently highlighted by Paat et al. [93]. Our bound has the useful property that it leads to a proximity result which intuitively says that, for any continuous or integer optimum, there is a mixed-integer optimum which may require several mixed-integer Graver steps to get to, but only one of those steps will be non-zero in the integer part. (Interestingly, this also means that the remaining steps will be circuits of the matrix A, but we do not use this fact in any way.) Thus, an integer or continuous optimum is always "almost correct" in the integer part. This is significant because once the integer part is correctly assigned, we get a purely continuous residual problem, which is polynomial-time solvable.

Set $\mathbb{X} = \mathbb{Z}^z \times \mathbb{R}^q$. We next use the algorithm of Theorem 5 as a starting point for developing a novel algorithm for an intermediate problem. Namely, we now allow the bounds $\mathbf{l}, \mathbf{u} \in \mathbb{X}$ and right-hand side $\mathbf{b} \in \mathbb{R}^m$ to be fractional, that is, we consider the problem

$$\min\{\mathbf{w}\mathbf{x} : E\mathbf{x} = \mathbf{b},\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\ \mathbf{x} \in \mathbb{X}\}. \tag{$\text{MILP}_{\text{frac}}$}$$

Already deciding feasibility of this variant has been shown to be NP-hard for totally unimodular matrices [16]. In a way, ($\text{MILP}_{\text{frac}}$) can be seen as a special case of (MIP) (even with integer $\mathbf{b}, \mathbf{l}, \mathbf{u}$), because a separable convex objective f can be used to model non-integer lower bounds and right-hand sides.

We are interested in the case of small treedepth and coefficients, in particular in n-fold and 2-stage stochastic matrices with small block sizes. For the case of 2-stage stochastic constraints, we give an XP algorithm:

Theorem 6. *The problem* ($\text{MILP}_{\text{frac}}$) *where A is a 2-stage stochastic matrix whose two blocks have r and s columns, and both have t rows, can be solved in time $g(r, s, \|A\|_\infty) \cdot n^r$, for some computable function g.*

This result turns out to be likely optimal, as we show that (MIP) with integral data is W[1]-hard when A is a 2-stage stochastic matrix with blocks of size bounded by a parameter and $\|A\|_\infty = 1$ already for linear objective functions. In particular, under the common parameterized complexity assumption that FPT $\neq$ W[1]-hard holds, this rules out algorithms for (MIP) with running times of the form $g(\|A\|_\infty, d) \cdot \text{poly}(n)$ with d the larger of the number of columns of the two blocks. Such a (double exponential) algorithm does exist for the pure integer case as implied by Theorem 1.

Moreover, we show that the algorithm from Theorem 6 cannot be extended to the related case of n-fold constraint structure: ($\text{MILP}_{\text{frac}}$) with integral bounds but fractional right hand sides is NP-hard when A is an n-fold matrix with blocks of constant dimensions and $\|A\|_\infty = 1$.

Interestingly, the above hardness results demonstrate that the relationship between n-fold and 2-stage stochastic programs in the mixed case is different from purely-integer case: In the purely integer case, n-fold (IP) is solvable faster than 2-stage stochastic (IP) (single- vs. double-exponential time, respectively), while in the mixed-integer case, the situation seems to be reversed (NP-hard vs W[1]-hard for n-fold and 2-stage stochastic (MIP), respectively).

Results on Mixed Graver Bases. The mixed Graver basis was introduced by Hemmecke [53] already in 2003, but not understood well enough to be used. On our way to showing Theorem 6, we prove several results about the mixed Graver basis which are of independent interest, and disprove the typical intuitions gained by studying the ordinary integral Graver basis. First, all elements of the *integral* Graver basis of an n-fold matrix with bounded block-dimension also have entries of bounded absolute value, whence they derive their algorithmic usefulness. We show that this is not true for the mixed Graver basis: there is an n-fold matrix A with constant-sized blocks and $\|A\|_\infty = 1$ such that the mixed Graver basis of A contains an element with 1-norm of size $\Omega(n)$.

On the other hand, for 2-stage stochastic matrices, the ∞-norm of its elements can be bounded by a function of the block-dimensions and $\|A\|_\infty$. This bound also implies a proximity result: for any integer optimum $\mathbf{z}^*$, there is a nearby mixed optimum $\mathbf{x}^*$. Thus, we can first find $\mathbf{z}^*$ (which can be done efficiently), and then only search in a small neighborhood around $\mathbf{z}^*$.

Until now, a bound $g(A)$ on (some) norm of the Graver elements has always led to an algorithm with a corresponding running time $g'(A)\text{poly}(n)$. However, in the mixed case, such an algorithm is ruled out by

the W[1]-hardness of 2-stage stochastic ($\text{MILP}_{\text{frac}}$). This shows that, in the mixed case, the common intuition of good bounds on the Graver norm directly leading to fast algorithms fails.

5 SPARSITY BEYOND TREEDEPTH

All of the algorithms discussed so far assume that the input matrix is sparse in the traditional sense – it only has few non-zero entries, and they are arranged in a particularly structured way. The motivation for **[H6]** and **[H7]** is the simple observation that there are matrices which are *not* sparse, but can be transformed to *become* sparse by row operations. For example, the matrix on the left below, whose dual tree-depth is 5, can be transformed by row operations to the matrix with dual tree-depth 2 given on the right.

$$\begin{pmatrix} 2 & 2 & 1 & 2 & 1 & 3 & 1 \\ 2 & 1 & 1 & 1 & 2 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 1 \\ 2 & 1 & 1 & 2 & 2 & 1 & 1 \\ 2 & 2 & 1 & 2 & 1 & 3 & 2 \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} 2 & 1 & 0 & 1 & 1 & 2 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 \end{pmatrix}$$

Could the positive results of the previous sections be extended to matrices which exhibit this kind of "hidden" sparsity? Put differently, the transformation of an input matrix A is a matrix B which is a *preconditioner* as it changes the input matrix A to an equivalent one $A' = BA$ that is computationally more tractable. The usefulness of this is that instead of solving

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

we can solve

$$\min\{\mathbf{w}\mathbf{x} \mid BA\mathbf{x} = B\mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{Z}^n\},$$

since $A\mathbf{x} = \mathbf{b}$ and $BA\mathbf{x} = B\mathbf{b}$ are satisfied by exactly the same vectors $\mathbf{x}$, and A may be dense but BA be sparse. Our question is: under what conditions does such a preconditioner to sparsity exist, and how can it be computed efficiently? **[H7]** answers such questions to a significant degree. The results contained in **[H7]** build on and subsume the results of **[H6]**, so we will not refer to **[H6]** much further.

Preconditioning a matrix to make the problem computationally simpler is a ubiquitous preprocessing step in mathematical programming solvers. In this section, we are concerned with the *existence* and *efficient computability* of preconditioners to sparsity of matrices. Let us define the natural notions of *optimal treedepth of a matrix*: the optimal primal treedepth $\mathrm{td}^*_P(A)$ is the minimum primal tree-depth of a matrix A' which can be obtained from A by elementary row operations, and similarly for optimal dual treedepth $\mathrm{td}^*_D(A)$ and optimal incidence treedepth $\mathrm{td}^*_I(A)$.

It turns out that the "deeper" notions capable of capturing the "hidden" sparsity, or, equivalently, the optimal treedepth parameters defined above, are structural parameters of the column matroid $M(A)$ of the matrix A. We deal with three parameters of $M := M(A)$:

- The *deletion-depth*, $\mathrm{dd}(M)$, introduced by DeVos et al. [27],
- the *contraction***-depth*, $\overset{*}{\mathrm{cd}}(M)$, introduced by Kardoš et al. [70] under the name *branch-depth*, however, since there was a competing notion of branch-depth [27], we decided to use a different name for this depth parameter to avoid confusion, and
- the *contraction***-deletion-depth*, $\overset{*}{\mathrm{cdd}}(M)$, introduced in **[H7]** itself.

The structural results of **[H7]** can be summarized as follows:

Theorem 7. *For every matrix A, it holds that*

- $\mathrm{td}^*_P(A) = \mathrm{dd}(M(A))$,
- $\mathrm{td}^*_D(A) = \overset{*}{\mathrm{cd}}(M(A))$, *and*
- $\mathrm{td}^*_I(A) = \overset{*}{\mathrm{cdd}}(M(A)) + 1$.

The first two (primal and dual) results can also be made efficient:

Theorem 8. *For every rational matrix $A \in \mathbb{Q}^{m \times n}$ and any integers d and a, there exists an algorithm which*

- *either decides that A is not equivalent to any matrix A' with primal or dual treedepth at most d and $\|A'\|_\infty \leq a$, or*

- *computes a matrix A' equivalent to A with primal or dual treedepth at most d and $\|A'\|_\infty \leq g(a,d)$ for some computable function g.*

The proof idea behind Theorem 8 is to use monadic second-order logic to express the existence of a matrix A' with primal or dual treedepth at most d, and to apply the algorithm of Hliněný [59] to $M(A)$ to decide this sentence. However, this is not directly possible as Hliněný's algorithm is for matroids representable over finite fields, and its complexity requires the order of the field to be a parameter; however, $M(A)$ is defined via linear independence of the columns of A over the rationals. We overcome this by showing that $M(A)$ is isomorphic to a matroid representable over a finite field, and a key ingredient in this step are our fractionality bounds from **[H4]**.

The immediate consequence of Theorem 8 is that all the thus far developed algorithms for (ILP), (IP), (MILP), and (MIP), namely Theorems 1–6, can be extended to the case where the input matrix is not necessarily of small primal or dual treedepth nor has small coefficients, but is equivalent to one which is. To summarize the most important corrolaries:

Corollary 9 (Efficient Optimization via Matroid Sparsity). *Let A be a matrix with d the smaller of the optimal primal or dual treedepth, and let a be an upper bound on $\|A'\|_\infty$ of any row-equivalent matrix A'. Then*

- (ILP) *and* (MILP) *can be solved in strongly-*FPT *time $g(a,d)\mathrm{poly}(n)$, and*
- (IP) *with a separable convex objective can be solved in* FPT *time $g(a,d)\mathrm{poly}(n,L)$,*

for some computable function g, and with $L = \langle A, f, \mathbf{b}, \mathbf{l}, \mathbf{u}\rangle$.

Another important result which we show in **[H7]** is the relationship between $\mathrm{td}^*_D(A)$, $g_1(A)$, and $c_1(A)$, which is the largest ℓ_1-norm of any circuit of A. Informally speaking, we show that the following statements are equivalent for every matrix A:

- The matrix A is equivalent to a matrix with bounded dual tree-depth and bounded entry complexity.
- The contraction*-depth of the matroid $M(A)$ is bounded, and the matrix A is equivalent to a matrix with bounded coefficients (with any dual tree-depth).
- The ℓ_1-norm of every circuit of A is bounded.
- The ℓ_1-norm of every element of the Graver basis of A is bounded.

This affirmatively resolved a question posed during the Dagstuhl workshop 19041 "New Horizons in Parameterized Complexity", about whether (IP) is FPT when parameterized by $g_1(A)$. Moreover, it shows that in the case of the ℓ_1-norm, the Graver elements cannot be much larger than the circuits, which is not known to be true in general.

We also remark that while, in the case of dual treedepth, if a matrix A has small coefficients and is equivalent to a matrix with dual tree-depth d, then there exists an equivalent matrix with dual treedepth d and coefficients bounded by a function of d and $\|A\|_\infty$, this is not true in the case of primal treedepth. The largest coefficient of every matrix with primal treedepth equal to one that is equivalent to the following matrix A is exponential in the number of rows of A, quite in a contrast to the case of dual treedepth.

$$\begin{pmatrix}
1 & 2 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\
0 & 1 & 2 & 0 & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 2 & \cdots & 0 & 0 & 0 & 0 \\
\vdots & \vdots & & \ddots & \ddots & & & \vdots & \vdots \\
\vdots & \vdots & & & \ddots & \ddots & & \vdots & \vdots \\
0 & 0 & 0 & 0 & \cdots & 1 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & \cdots & 0 & 1 & 2 & 0 \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 2
\end{pmatrix}$$

6 BEYOND SMALL COEFFICIENTS AND GRAVER BASES – COMING FULL CIRCLE

The two most prominent classes of constraint matrices we have been considering are 2-stage stochastic and n-fold matrices. Intuitively, 2-stage stochastic matrices have few "global" columns, and n-fold matrices have few "global" rows, whose deletion makes the matrix block-diagonal, respectively. It is natural to consider the combination of both: a matrix which has few rows and few columns which are "global" is called a 4-block n-fold matrix. Hemmecke et al. [54] have introduced this class and shown an XP algorithm for (IP) with such matrices, when parameterizing by the block dimensions and $\|A\|_\infty$. The arguably most important open problem in the area of block-structured integer programming is whether this algorithm is qualitatively optimal, that is, whether 4-block n-fold (IP) is W[1]-hard, or whether it can be solved in FPT time.

Let us point out that this open problem has an elegant phrasing from the perspective of the structural parameter topological height introduced in **[H2]**: (IP) parameterized by the incidence treedepth $\mathrm{td}_I(A)$ is NP-hard in general, but FPT if the topological height is 1 (because it reduces to (IP) in fixed dimension), and XP if the topological height is 2 – precisely because this corresponds to 4-block n-fold matrices.

It is not difficult to see that, by employing standard encoding tricks, any 2-stage stochastic (IP) and any n-fold (IP) instance with small block sizes but containing entries bounded by $\mathrm{poly}(n)$ in its "global" parts can be reduced to a 4-block n-fold (IP) instance with small coefficients and small block sizes. Thus, if 4-block n-fold (IP) is FPT, then at least the aforementioned two generalizations of 2-stage stochastic and n-fold (IP) are FPT as well. The goal of **[H8]** is to investigate this question.

Our Contribution. We prove that both the feasibility problem for 2-stage stochastic programs and the optimization problem for uniform n-fold programs (that is, where all the global blocks are the same) can be solved in fixed-parameter time when parameterized by the dimensions of the blocks and the maximum absolute value of any entry appearing in the diagonal blocks. That is, we allow the entries of the global blocks to be arbitrarily large, and in the case of n-fold programs, we require that all global blocks are equal. The statements below summarize our results. L is the bitsize of the input program, as usual.

Theorem 10.

(1) *The feasibility of a generalized (i.e., blocks may differ) 2-stage stochastic* (ILP) *can be solved in time* $g(k, \max_i \|A_2^i\|_\infty) \cdot L$ *for a computable function* g*, where* k *is the largest number of columns of any block, and* A_2^i*,* $i \in [n]$*, are the diagonal blocks.*

(2) *n-fold* (ILP) *with all global blocks identical can be solved in time* $g(k, \max \|A_2^i\|_\infty) \cdot \mathrm{poly}(L)$ *for a computable function* g*, where* k *is the largest number of rows and columns of any block, and* A_2^i*,* $i \in [n]$*, are the diagonal blocks.*

The uniformity condition (all global blocks identical) for n-fold (ILP) in Theorem 10 is necessary (unless P = NP), as one can very easily reduce Subset Sum to the feasibility problem for n-fold (ILP) with $k = 2$ and the diagonal blocks being $\{0, 1\}$-matrices. Indeed, given an instance of Subset Sum consisting of positive integers $a_1, \ldots, a_n$ and a target integer t, we can write the following n-fold (ILP) on variables $y_1, \ldots, y_n, y'_1, \ldots, y'_n \in \mathbb{Z}^n$: $y_i + y'_i = 1$ for all $i = 1, \ldots, n$ and $\sum_{i=1}^n a_i y_i = t$.

Notice that in the algorithm for n-fold (ILP), the maximum number of *columns* of a block is required to be a parameter, and this is heavily exploited, setting our approach apart from the previous work.

Further, observe that the algorithm for 2-stage stochastic (ILP) applies only to the feasibility problem. We actually do not know whether this positive result can be extended to the linear or even separable-convex optimization problem as well, and we consider determining this an outstanding open problem. Also, notice that this algorithm seems to be the first one for feasibility of 2-stage stochastic (ILP) that achieves truly linear dependence of the running time on the total input size; the earlier algorithm of [19] had at least some additional polylogarithmic factors.

Finally, note that the algorithms of Theorem 10 are not strongly polynomial (i.e., the running time depends on the total bitsize of the input, rather than is counted in the number of arithmetic operations), while the previous algorithms of **[H1]** and [18, 19] for the stronger parameterization are. This is justified because no strongly polynomial algorithm is known, and none is suspected to exist, for the greatest common divisor

problem, which is an instance of (ILP) with two variables. Given integers a, b, computing $\gcd(a, b)$ is equivalent to finding integers p, q such that $ap + bq$ is positive but as small as possible. This is an (ILP) min $ap + bq$ subject to $ap+bq \geq 1, p, q \in \mathbb{Z}$, which is a special case of the problem solved by the second part of Theorem 10. Similarly, fixed-dimension (ILP) feasibility is not known to be strongly FPT, and suspected not to be, and this is a special case of the problem solved by the first part of Theorem 10.

Proof Ideas. The two parts of Theorem 10 each rely on different techniques, and, not surprisingly, they significantly depart from the by now standard approach through Graver bases. They are based on entirely new ideas, with some key Graver-based insight needed in the case of the algorithm for n-fold (ILP). In both cases, the problem is ultimately reduced to (mixed) integer programming with a bounded number of (integral) variables, which we then solve using Lenstra's algorithm [83] (or any of its newer strengthenings [68, 98]), and this allows us to cope with large entries on input.

We find this connection to fixed-dimension (ILP) fascinating, and as if we have come full circle – from the oldest tractable class of small dimension programs, through the newer and seemingly unrelated class of block-structured programs, back to programs with small dimension. Still, while the present techniques suffice for n-fold programs with *linear* objectives, they do not seem to extend to the case of *separable convex* objectives. We suspect that there the analogy ends and such programs cannot reasonably be "embedded" in fixed-dimension. We believe this is an important open problem.

The first part of Theorem 10 is based on a new structural result about integer cones, which eventually allows us to mark all but a few blocks as "irrelevant" and remove them, leaving us with a fixed-dimension (ILP) feasibility problem.

The second part of Theorem 10 uses a new insight that, for each brick $i \in [n]$, the right hand side $\mathbf{b}^i$ can be decomposed into two vectors $(\mathbf{b}^i)'$ and $(\mathbf{b}^i)''$ such that $\mathbf{b}^i = (\mathbf{b}^i)' + (\mathbf{b}^i)''$, $(\mathbf{b}^i)', (\mathbf{b}^i)''$ are in the same orthant as $\mathbf{b}^i$ and below it, and, most importantly, they have the property that every solution $\mathbf{x}^i$ satisfying $A\mathbf{x}^i = \mathbf{b}^i$ can be decomposed into $\mathbf{x}^i = (\mathbf{x}^i)' + (\mathbf{x}^i)''$ such that $A(\mathbf{x}^i)' = (\mathbf{b}^i)'$ and $A(\mathbf{x}^i)'' = (\mathbf{b}^i)''$. Iteratively applying this decomposition, we can reduce the problem to a high-multiplicity n-fold (ILP) with few brick types, since eventually each right hand side becomes small. Note that because of the large coefficients in the global constraints, this instance is not solvable by Theorem 3. Still, with a few more insights, this instance can be massaged to a form solvable by Lenstra's algorithm. An interesting subproblem we brushed over is how to efficiently decompose each brick's right hand side $\mathbf{b}^i$ into the, possibly many, smaller bricks. It turns out that the desired property can be expressed in Presburger arithmetic, and the decomposition can be found using Cooper's algorithm, whose parameterized complexity we analyzed in another work [81].

Recall the n-fold model of the makespan minimization on uniformly related machines ($Q||C_{\max}$) problem introduced in Section 1.3. This is a program with many bricks which only differ by their right hand sides. Applying the decomposition technique can then be interpreted as replacing one fast machine with two slower machines, and the result of iterating this process is that there are "few" different machines, and each has a "small" capacity. This seems to be the gist of an argument carried out by Brinkop and Jansen [12], so their result can be seen as a particular instance of the decomposition insight above. Also, the fact that $Q||C_{\max}$ is FPT parameterized by $p_{\max}$ can now either be shown by a direct application to n-fold (ILP) (as we have done in [74]), or by the argument above and then solving the resulting fixed-dimension (ILP) instance using, e.g., the algorithm of Reis and Rothvoss [98]. This second approach gives a worse complexity bound, but provides a useful new perspective.

7 CONCLUSIONS AND OPEN PROBLEMS

We have taken a solid foundation of results built by researchers primarily from the mathematical programming community between roughly 2000-2015, and connected it to the theories of parameterized complexity and graph and matroid sparsity, together with the respective communities. Many interesting open problems and research directions remain, and we list a few of those that seem the most important to us.

4-block n-fold (IP). The arguably most important open problem is the complexity of 4-block n-fold (IP). It is known to be XP [54], but it is not known whether it is W[1]-hard or FPT. In **[H8]**, we have shown that completely new techniques are necessary, but also within reach, in order to make progress on this problem. The nearest open question seems to be the complexity of 2-stage stochastic (ILP) with large coefficients in the global blocks: recall that **[H8]** resolves the complexity of the feasibility problem, but the optimization problem remains open. Intuitively, linear optimization corresponds to adding one global constraint, so it constitutes a natural first step towards 4-block n-fold (ILP), which is equivalent to 2-stage stochastic (ILP) with k additional linear constraints.

Another question related to 4-block n-fold (IP) stems from **[H7]**. What is the complexity of computing the optimal incidence treedepth $\mathrm{td}^*_I(A)$, or equivalently the matroid parameter $\mathring{\mathrm{cdd}}(M(A))$? A similar problem which is likely to be easier and is still open is the following: given a matrix A and an integer k, decide whether A is row-equivalent to a 4-block n-fold matrix A' with block sizes bounded by k. A "yes"-instance of this problem is a matrix A with $g_\infty(A) \leq n^{g(k,\|A\|_\infty)}$ for some function g, which is a stronger bound on $g_\infty(A)$ than available in general for matrices with small $\mathring{\mathrm{cdd}}(M(A))$. Still, it is not strong enough, e.g., to allow showing that $M(A)$ is isomorphic to a matroid representable over a finite field of a small enough order to efficiently run Hliněný's algorithm.

Configuration (IP). The structural results of **[H3]** are related to the seminal work of Goemans and Rothvoss [48] on the polynomiality of Bin Packing with few item types. Their technique yields FPT algorithms for many important scheduling problems, yet it is inherently limited to models with linear objectives, and thus, for example, does not lead to efficient algorithms for ℓ_2-norm minimization or minimization of sum of weighted completion times. Consequently, the complexity of problems like $P||\ell_2$ or $P||\sum w_jC_j$ parameterized by the number of job types d remain open, even in the regime when the largest processing time $p_{\max}$ is bounded by a polynomial of the input length. Because the techniques of **[H3]** do apply to the non-linear, separable convex setting, we believe that the complexity of the aforementioned scheduling problems may be tackled using some novel approach building on **[H3]**.

2-stage (MIP). In **[H4]**, we have shown that 2-stage stochastic (MILP) with integral data is FPT, and in **[H5]**, we have shown that 2-stage stochastic ($\mathrm{MILP}_{\mathrm{frac}}$) is XP and W[1]-hard. This result crucially relies on the linearity of the objective (namely the fact that an optimum is attained at a vertex of the mixed integer hull), so it is unclear whether 2-stage stochastic (MIP) is also XP, and how to show this.

Practical Sparsity: Automatic Decomposition. The original work on block-structured (IP) assumes that the block structure is given as part of the input. In **[H2]**, we point out that the treedepth-decomposition of the relevant graphs can be computed in FPT time, and the block structure can be obtained from it. This suggests an automatic way to decompose a given matrix into a block-structured one, which is a problem that has long been studied in practice [5, 7, 11, 35, 41, 72, 108, 110, 111]. It would be interesting to verify how practically useful might treedepth-based decompositional methods be in practice. This seems within reach: thanks to the PACE challenge [82], there are now efficient implementations of treedepth algorithms, and the block structure can be obtained from the decomposition in linear time.

A much more challenging problem is employing the matroid-based methods of **[H6]** and **[H7]**. No practical methods for computing the relevant matroid parameters have been suggested, much less implemented.

REFERENCES

[1] Sibel A. Alumur and Bahar Yetis Kara. Network hub location problems: The state of the art. *European Journal of Operational Research*, 190(1):1–21, 2008.

[2] Gautam Appa, Balázs Kotnyek, Konstantinos Papalamprou, and Leonidas Pitsoulis. Optimization with binet matrices. *Operations research letters*, 35(3):345–352, 2007.

[3] Stephan Artmann, Robert Weismantel, and Rico Zenklusen. A strongly polynomial algorithm for bimodular integer linear programming. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 1206–1219. ACM, 2017.

[4] Matthias Aschenbrenner and Raymond Hemmecke. Finiteness theorems in stochastic integer programming. *Foundations of Computational Mathematics*, 7(2):183–227, 2007.

[5] Cevdet Aykanat, Ali Pinar, and Ümit V. Çatalyürek. Permuting sparse rectangular matrices into block-diagonal form. *SIAM Journal on Scientific Computing*, 25(6):1860–1879, 2004.

[6] Jacques F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Comput. Manag. Sci.*, 2(1):3–19, 2005. URL: https://doi.org/10.1007/s10287-004-0020-y, doi:10.1007/S10287-004-0020-Y.

[7] Martin Bergner, Alberto Caprara, Alberto Ceselli, Fabio Furini, Marco E Lübbecke, Enrico Malaguti, and Emiliano Traversi. Automatic Dantzig–Wolfe reformulation of mixed integer programs. *Mathematical Programming*, 149(1-2):391–424, 2015.

[8] Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813–852, 2016. URL: http://www.jstor.org/stable/43818629.

[9] John R. Birge and François Louveaux. *Introduction to stochastic programming*. Springer-Verlag, New York, 1997.

[10] Robert E Bixby. Solving real-world linear programs: A decade and more of progress. *Operations research*, 50(1):3–15, 2002.

[11] Ralf Borndörfer, Carlos E. Ferreira, and Alexander Martin. Decomposing matrices into blocks. *SIAM J. Optim.*, 9(1):236–269, 1998. doi:10.1137/S1052623497318682.

[12] Hauke Brinkop and Klaus Jansen. High multiplicity scheduling on uniform machines in fpt-time. *CoRR*, abs/2203.01741, 2022. URL: https://doi.org/10.48550/arXiv.2203.01741, arXiv:2203.01741, doi:10.48550/ARXIV.2203.01741.

[13] Lin Chen and Dániel Marx. Covering a tree with rooted subtrees–parameterized and approximation algorithms. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 2801–2820. SIAM, 2018.

[14] Lin Chen, Dániel Marx, Deshi Ye, and Guochuan Zhang. Parameterized and approximation results for scheduling with a low rank processing time matrix. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 22:1–22:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. URL: https://doi.org/10.4230/LIPIcs.STACS.2017.22, doi:10.4230/LIPICS.STACS.2017.22.

[15] Sergei Chubanov. A polynomial-time descent method for separable convex optimization problems with linear constraints. *SIAM Journal on Optimization*, 26(1):856–889, 2016.

[16] Michele Conforti, Marco Di Summa, Friedrich Eisenbrand, and Laurence A. Wolsey. Network formulations of mixed-integer programs. *Math. Oper. Res.*, 34(1):194–209, 2009. doi:10.1287/moor.1080.0354.

[17] Stavros S. Cosmadakis and Christos H. Papadimitriou. The traveling salesman problem with many visits to few cities. *SIAM J. Comput.*, 13(1):99–108, 1984.

[18] Jana Cslovjecsek, Friedrich Eisenbrand, Christoph Hunkenschröder, Lars Rohwedder, and Robert Weismantel. Block-structured integer and linear programming in strongly polynomial and near linear time. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1666–1681. SIAM, 2021. doi:10.1137/1.9781611976465.101.

[19] Jana Cslovjecsek, Friedrich Eisenbrand, Michał Pilipczuk, Moritz Venzin, and Robert Weismantel. Efficient sequential and parallel algorithms for multistage stochastic integer programming using proximity. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 33:1–33:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.33.

[20] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[21] George B Dantzig. Application of the simplex method to a transportation problem. *Activity Analysis and Production and Allocation*, 1951.

[22] George B Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

[23] Jesús A. De Loera, Raymond Hemmecke, and Jon Lee. On augmentation algorithms for linear and integer-linear programming: From Edmonds–Karp to Bland and beyond. *SIAM Journal on Optimization*, 25(4):2494–2511, 2015.

[24] Jesús A. De Loera, Raymond Hemmecke, Shmuel Onn, and Robert Weismantel. N-fold integer programming. *Discrete Optimization*, 5(2):231–241, 2008.

[25] Jesús A. De Loera and Shmuel Onn. The complexity of three-way statistical tables. *SIAM J. Comput*, 33(4):819–836, 2004.

[26] Jesús A. De Loera and Shmuel Onn. All linear and integer programs are slim 3-way transportation programs. *SIAM Journal on Optimization*, 17(3):806–821, 2006.

[27] Matt DeVos, O-joung Kwon, and Sang-il Oum. Branch-depth: Generalizing tree-depth of graphs. *European Journal of Combinatorics*, 90:103186, 2020.

[28] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

[29] Pavel Dvořák, Eduard Eiben, Robert Ganian, Dušan Knop, and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP: programs with few global variables and constraints. *Artif. Intell.*, 300:103561, 2021. doi:10.1016/j.artint.2021.103561.

[30] Eduard Eiben, Robert Ganian, Dušan Knop, and Sebastian Ordyniak. Unary integer linear programming with structural restrictions. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 1284–1290. ijcai.org, 2018.

[31] Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. Reducibility bounds of objective functions over the integers. *Oper. Res. Lett.*, 51(6):595–598, 2023. URL: https://doi.org/10.1016/j.orl.2023.10.001, doi:10.1016/J.ORL.2023.10.001.

[32] Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. An algorithmic theory of integer programming, 2019. URL: https://arxiv.org/abs/1904.01361v3.

[33] Farbod Ekbatani, Bento Natura, and László A. Végh. Circuit imbalance measures and linear programming. *CoRR*, abs/2108.03616, 2021. URL: https://arxiv.org/abs/2108.03616, arXiv:2108.03616.

[34] Piotr Faliszewski, Rica Gonen, Martin Koutecký, and Nimrod Talmon. Opinion diffusion and campaigning on society graphs. *J. Log. Comput.*, 32(6):1162–1194, 2022. doi:10.1093/logcom/exac014.

[35] Michael C. Ferris and Jeffrey D. Horn. Partitioning mathematical programs for parallel solution. *Mathematical Programming*, 80(1):35–61, 1998.

[36] Stephen E Fienberg and Alessandro Rinaldo. Three centuries of categorical data analysis: Log-linear models and maximum likelihood estimation. *Journal of Statistical Planning and Inference*, 137(11):3430–3445, 2007.

[37] Samuel Fiorini, Gwenaël Joret, Stefan Weltge, and Yelena Yuditsky. Integer programs with bounded subdeterminants and two nonzeros per row. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 13–24. IEEE, 2021. doi:10.1109/FOCS52979.2021.00011.

[38] Christodoulos A. Floudas and Xiaoxia Lin. Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. *Ann. Oper. Res.*, 139(1):131–162, 2005. URL: https://doi.org/10.1007/s10479-005-3446-x, doi:10.1007/S10479-005-3446-X.

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

[40] Eugene C. Freuder. Complexity of K-tree structured constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 4–9, 1990.

[41] Gerald Gamrath and Marco E. Lübbecke. Experiments with a generic Dantzig–Wolfe decomposition for integer programs. *Experimental Algorithms*, 6049:239 – 252, 2010.

[42] Robert Ganian and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP. *Artificial Intelligence*, 2018.

[43] Robert Ganian, Sebastian Ordyniak, and M. S. Ramanujan. Going beyond primal treewidth for (M)ILP. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 815–821. AAAI Press, 2017.

[44] Tomáš Gavenčiak, Martin Koutecký, and Dušan Knop. Integer programming in parameterized complexity: Five miniatures. *Discret. Optim.*, 44(Part):100596, 2022. doi:10.1016/j.disopt.2020.100596.

[45] Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961.

[46] Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting stock problem—part ii. *Operations research*, 11(6):863–888, 1963.

[47] Fred W. Glover. Tabu search - part II. *INFORMS J. Comput.*, 2(1):4–32, 1990. URL: https://doi.org/10.1287/ijoc.2.1.4, doi:10.1287/IJOC.2.1.4.

[48] Michel X. Goemans and Thomas Rothvoss. Polynomiality for bin packing with a constant number of item types. *J. ACM*, 67(6):38:1–38:21, 2020. doi:10.1145/3421750.

[49] Dmitry V. Gribanov, Dmitry S. Malyshev, and Ivan A. Shumilov. On a simple connection between Δ-modular ILP and lp, and a new bound on the number of integer vertices. *Oper. Res. Forum*, 5(2):32, 2024. URL: https://doi.org/10.1007/s43069-024-00310-2, doi:10.1007/S43069-024-00310-2.

[50] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, second edition, 1993.

[51] Claire Hanen and Alix Munier Kordon. Fixed-parameter tractability of scheduling dependent typed tasks subject to release times and deadlines. *J. Sched.*, 27(2):119–133, 2024. URL: https://doi.org/10.1007/s10951-023-00788-4, doi:10.1007/S10951-023-00788-4.

[52] Raymond Hemmecke. *On the decomposition of test sets*. PhD thesis, Universität Duisburg, 2001.

[53] Raymond Hemmecke. On the positive sum property and the computation of graver test sets. *Math. Program.*, 96(2):247–269, 2003. URL: https://doi.org/10.1007/s10107-003-0385-7, doi:10.1007/S10107-003-0385-7.

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

[55] Raymond Hemmecke, Shmuel Onn, and Lyubov Romanchuk. N-fold integer programming in cubic time. *Mathematical Programming*, pages 1–17, 2013.

[56] Raymond Hemmecke and Rüdiger Schultz. Decomposition of test sets in stochastic integer programming. *Mathematical Programming*, 94:323–341, 2003.

[57] Julia L Higle and Suvrajeet Sen. *Stochastic decomposition: a statistical method for large scale stochastic linear programming*, volume 8. Springer Science & Business Media, 1996.

[58] Frank L Hitchcock. The distribution of a product from several sources to numerous localities. *Journal of mathematics and physics*, 20(1-4):224–230, 1941.

[59] Petr Hliněný. Branch-width, parse trees, and monadic second-order logic for matroids. *J. Comb. Theory B*, 96(3):325–351, 2006. URL: https://doi.org/10.1016/j.jctb.2005.08.005, doi:10.1016/J.JCTB.2005.08.005.

[60] Dorit S. Hochbaum and Ron Shamir. Strongly polynomial algorithms for the high multiplicity scheduling problem. *Oper. Res.*, 39(4):648–653, 1991.

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

[62] Alan J Hoffman and Joseph B Kruskal. Integral boundary points of convex polyhedra. *Linear inequalities and related systems*, pages 223–246, 1956.

[63] Serkan Hoşten and Seth Sullivant. A finiteness theorem for markov bases of hierarchical models. *J. Comb. Theory A*, 114(2):311–321, 2007. URL: https://doi.org/10.1016/j.jcta.2006.06.001, doi:10.1016/J.JCTA.2006.06.001.

[64] Bart M. P. Jansen and Stefan Kratsch. A structural approach to kernels for ILPs: Treewidth and total unimodularity. In Nikhil Bansal and Irene Finocchi, editors, *Proceedings of the 23rd Annual European Symposium, ESA 2015, Patras, Greece, September 14-16, 2015*, volume 9294 of *Lecture Notes in Computer Science*, pages 779–791. Springer, 2015.

[65] Klaus Jansen, Kim-Manuel Klein, Marten Maack, and Malin Rau. Empowering the configuration-ip: new PTAS results for scheduling with setup times. *Math. Program.*, 195(1):367–401, 2022. URL: https://doi.org/10.1007/s10107-021-01694-3, doi:10.1007/S10107-021-01694-3.

[66] Klaus Jansen, Alexandra Lassota, and Marten Maack. Approximation algorithms for scheduling with class constraints. In Christian Scheideler and Michael Spear, editors, *SPAA '20: 32nd ACM Symposium on Parallelism in Algorithms and Architectures, Virtual Event, USA, July 15-17, 2020*, pages 349–357. ACM, 2020. doi:10.1145/3350755.3400247.

[67] Peter Kall and Stein W. Wallace. *Stochastic Programming*. Wiley, Chichester etc., 1994.

[68] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, August 1987.

[69] Leonid V Kantorovich. On the translocation of masses. In *Dokl. Akad. Nauk. USSR (NS)*, volume 37, pages 199–201, 1942.

[70] František Kardoš, Daniel Kráľ, Anita Liebenau, and Lukáš Mach. First order convergence of matroids. *European Journal of Combinatorics*, 59:150–168, 2017.

[71] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger, editors, *Complexity of Computer Computations, The IBM Research Symposia Series*, pages 85–103. Springer, 1972.

[72] Taghi Khaniyev, Samir Elhedhli, and Fatih Safa Erenay. Structure detection in mixed-integer programs. *INFORMS Journal on Computing*, 30(3):570–587, 2018.

[73] Kim-Manuel Klein, Adam Polak, and Lars Rohwedder. On minimizing tardy processing time, max-min skewed convolution, and triangular structured ilps. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 2947–2960. SIAM, 2023. URL: https://doi.org/10.1137/1.9781611977554.ch112, doi:10.1137/1.9781611977554.CH112.

[74] Dušan Knop and Martin Koutecký. Scheduling meets n-fold integer programming. *J. Scheduling*, 21(5):493–503, 2018.

[75] Dušan Knop and Martin Koutecký. Scheduling kernels via configuration LP. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 73:1–73:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. URL: https://doi.org/10.4230/LIPIcs.ESA.2022.73, doi:10.4230/LIPICS.ESA.2022.73.

[76] Dušan Knop, Martin Koutecký, Asaf Levin, Matthias Mnich, and Shmuel Onn. Multitype integer monoid optimization and applications. *CoRR*, abs/1909.07326, 2019. URL: http://arxiv.org/abs/1909.07326, arXiv:1909.07326.

[77] Dušan Knop, Martin Koutecký, and Matthias Mnich. A unifying framework for manipulation problems. In Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar, editors, *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pages 256–264. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018. URL: http://dl.acm.org/citation.cfm?id=3237427.

[78] Dušan Knop, Martin Koutecký, and Matthias Mnich. Combinatorial n-fold integer programming and applications. *Math. Program.*, 184(1):1–34, 2020. doi:10.1007/s10107-019-01402-2.

[79] Dušan Knop, Martin Koutecký, and Matthias Mnich. Voting and bribing in single-exponential time. *ACM Trans. Economics and Comput.*, 8(3):12:1–12:28, 2020. doi:10.1145/3396855.

[80] Balazs Kotnyek. *A generalization of totally unimodular and network matrices*. PhD thesis, London School of Economics and Political Science (United Kingdom), 2002.

[81] Martin Koutecký and Nimrod Talmon. Multi-party campaigning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 5506–5513. AAAI Press, 2021. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16693.

[82] Lukasz Kowalik, Marcin Mucha, Wojciech Nadara, Marcin Pilipczuk, Manuel Sorge, and Piotr Wygocki. The PACE 2020 parameterized algorithms and computational experiments challenge: Treedepth. In Yixin Cao and Marcin Pilipczuk, editors, *15th International Symposium on Parameterized and Exact Computation, IPEC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 180 of *LIPIcs*, pages 37:1–37:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. URL: https://doi.org/10.4230/LIPIcs.IPEC.2020.37, doi:10.4230/LIPICS.IPEC.2020.37.

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

[84] Andrea Lodi, Silvano Martello, and Michele Monaci. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252, 2002.

[85] Diane Maclagan. Antichains of monomial ideals are finite. *Proceedings of the American Mathematical Society*, 129(6):1609–1615, 2001.

[86] Hugues Marchand, Alexander Martin, Robert Weismantel, and Laurence A. Wolsey. Cutting planes in integer and mixed integer programming. *Discret. Appl. Math.*, 123(1-3):397–446, 2002. doi:10.1016/S0166-218X(01)00348-1.

[87] Alexander Martin. *Integer programs with block structure*. PhD thesis, 1999.

[88] TS Motzkin. The multi-index transportation problem. In *Bulletin of the American Mathematical Society*, volume 58, pages 494–494, 1952.

[89] Martin Nägele, Richard Santiago, and Rico Zenklusen. Congruency-constrained TU problems beyond the bimodular case. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 2743–2790. SIAM, 2022. doi:10.1137/1.9781611977073.108.

[90] Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012.

[91] Ivo Nowak. *Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming, Second Edition*, volume 152. Birkhäuser Basel, 2005. doi:10.1007/3-7643-7374-1.

[92] John E Olson. A combinatorial problem on finite abelian groups, I. *Journal of number theory*, 1(1):8–10, 1969.

[93] Joseph Paat, Robert Weismantel, and Stefan Weltge. Distances between optimal solutions of mixed-integer programs. *Math. Program.*, 179(1):455–468, 2020. doi:10.1007/s10107-018-1323-z.

[94] Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, 1981. URL: http://doi.acm.org/10.1145/322276.322287.

[95] Marcin Pilipczuk, Michał Pilipczuk, and Sebastian Siebertz. Lecture notes from the course "Sparsity" given at the Faculty of Mathematics, Informatics, and Mechanics of the University of Warsaw, Winter semesters 2017/18 and 2019/20. Available online at https://www.mimuw.edu.pl/~mp248287/sparsity2.

[96] András Prékopa. *Stochastic programming*, volume 324 of *Mathematics and its Applications*. Kluwer Academic Publishers Group, Dordrecht, 1995.

[97] Harilaos N. Psaraftis. A dynamic programming approach for sequencing groups of identical jobs. *Oper. Res.*, 28(6):1347–1359, 1980.

[98] Victor Reis and Thomas Rothvoss. The subspace flatness conjecture and faster integer programming. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 974–988. IEEE, 2023. doi:10.1109/FOCS57990.2023.00060.

[99] Andrzej Ruszczyński. Some advances in decomposition methods for stochastic linear programming. *Ann. Oper. Res.*, 85:153–172, 1999.

[100] Francisco Santos and Bernd Sturmfels. Higher lawrence configurations. *J. Comb. Theory A*, 103(1):151–164, 2003. doi:10.1016/S0097-3165(03)00092-X.

[101] Guntram Scheithauer and Johannes Terno. The modified integer round-up property of the one-dimensional cutting stock problem. *European J. Oper. Res.*, 84(3):562–571, 1995.

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

[103] D. Antony Tarvin, R. Kevin Wood, and Alexandra M. Newman. Benders decomposition: Solving binary master problems by enumeration. *Oper. Res. Lett.*, 44(1):80–85, 2016. URL: https://doi.org/10.1016/j.orl.2015.11.009, doi:10.1016/J.ORL.2015.11.009.

[104] Paolo Toth and Daniele Vigo, editors. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[105] Janna M. van den Akker, Jan A. Hoogeveen, and Steef L. van de Velde. Parallel machine scheduling by column generation. *Oper. Res.*, 47(6):862–872, 1999.

[106] Menkes van den Briel, Thomas Vossen, and Subbarao Kambhampati. Reviving integer programming approaches for AI planning: A branch-and-cut framework. In *Automated Planning and Scheduling, ICAPS 2005. Proceedings*, pages 310–319. AAAI, 2005.

[107] François Vanderbeck and Martin W. P. Savelsbergh. A generic view of dantzig-wolfe decomposition in mixed integer programming. *Oper. Res. Lett.*, 34(3):296–306, 2006. URL: https://doi.org/10.1016/j.orl.2005.05.009, doi:10.1016/J.ORL.2005.05.009.

[108] François Vanderbeck and Laurence A Wolsey. Reformulation and decomposition of integer programs. In *50 Years of Integer Programming 1958-2008*, pages 431–502. Springer, 2010.

[109] Thomas Vossen, Michael O. Ball, Amnon Lotem, and Dana S. Nau. On the use of integer programming models in AI planning. In *International Joint Conference on Artificial Intelligence, IJCAI 99. Proceedings*, pages 304–309. Morgan Kaufmann, 1999.

[110] Jiadong Wang and Ted Ralphs. Computational experience with hypergraph-based methods for automatic decomposition in discrete optimization. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 394–402. Springer, 2013.

[111] Roman L. Weil and Paul C. Kettler. Rearranging matrices to block-angular form for decomposition (and other) algorithms. *Management Science*, 18(1):98–108, 1971.

[112] Laurence A Wolsey. *Integer programming*. John Wiley & Sons, 2020.

A Parameterized Strongly Polynomial Algorithm for Block Structured Integer Programs

Martin Koutecký[1]
Technion – Israel Institute of Technology, Haifa, Israel, and
Charles University, Prague, Czech Republic
koutecky@technion.ac.il
https://orcid.org/0000-0002-7846-0053

Asaf Levin[2]
Technion – Israel Institute of Technology, Haifa, Israel
levinas@ie.technion.ac.il

Shmuel Onn[3]
Technion – Israel Institute of Technology, Haifa, Israel
onn@ie.technion.ac.il

Abstract

The theory of n-fold integer programming has been recently emerging as an important tool in parameterized complexity. The input to an n-fold integer program (IP) consists of parameter A, dimension n, and numerical data of binary encoding length L. It was known for some time that such programs can be solved in polynomial time using $O(n^{g(A)}L)$ arithmetic operations where g is an exponential function of the parameter. In 2013 it was shown that it can be solved in fixed-parameter tractable time using $O(f(A)n^3L)$ arithmetic operations for a single-exponential function f. This, and a faster algorithm for a special case of *combinatorial* n-fold IP, have led to several very recent breakthroughs in the parameterized complexity of scheduling, stringology, and computational social choice. In 2015 it was shown that it can be solved in strongly polynomial time using $O(n^{g(A)})$ arithmetic operations.

Here we establish a result which subsumes all three of the above results by showing that n-fold IP can be solved in strongly polynomial fixed-parameter tractable time using $O(f(A)n^6 \log n)$ arithmetic operations. In fact, our results are much more general, briefly outlined as follows.

- There is a strongly polynomial algorithm for integer linear programming (ILP) whenever a so-called Graver-best oracle is realizable for it.
- Graver-best oracles for the large classes of multi-stage stochastic and tree-fold ILPs can be realized in fixed-parameter tractable time. Together with the previous oracle algorithm, this newly shows two large classes of ILP to be strongly polynomial; in contrast, only few classes of ILP were previously known to be strongly polynomial.
- We show that ILP is fixed-parameter tractable parameterized by the largest coefficient $\|A\|_\infty$ and the primal or dual treedepth of A, and that this parameterization cannot be relaxed, signifying substantial progress in understanding the parameterized complexity of ILP.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases integer programming, parameterized complexity, Graver basis, n-fold integer programming

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.85

[1] Supported by a Technion postdoctoral fellowship and the project 17-09142S of GA ČR.
[2] Supported by a grant from the GIF, the German-Israeli Foundation for Scientific Research and Development (grant number I-1366-407.6/2016).
[3] Supported by the Dresner chair.

45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).
Editors: Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella;
Article No. 85; pp. 85:1–85:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Related Version A full version of the paper is available at `https://arxiv.org/abs/1802.05859`.

1 Introduction

In this article we consider the general linear integer programming (ILP) problem in standard form,

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

with A an integer $m \times n$ matrix, $\mathbf{b} \in \mathbb{Z}^m$, $\mathbf{w} \in \mathbb{Z}^n$, $\mathbf{l}, \mathbf{u} \in (\mathbb{Z} \cup \{\pm\infty\})^n$. It is well known to be strongly NP-hard, which motivates the search for tractable special cases.

The first important special case is ILP in fixed dimension. In the '80s it was shown by Lenstra and Kannan [17, 20] that (ILP) can be solved in time $n^{O(n)}L$, where L is the length of the binary encoding of the input. Secondly, it is known that if the matrix A is totally unimodular (all subdeterminants between -1 and 1), all vertices of the feasible region are integral and thus applying any polynomial algorithm for linear programming (LP) suffices. Later, Veselov and Chirkov [25] have shown that the more general class of bimodular ILP is also polynomial-time solvable. Other results exploit certain structural properties of A. These include the large classes of n-fold [13], tree-fold [4], 2-stage and multi-stage stochastic [3], and 4-block n-fold [12] ILPs, as well as algorithms for ILPs with bounded treewidth [11], treedepth [10] and fracture number [7] of certain graphs related to the matrix A.

A fundamental question regarding problems involving large numbers is whether there exists an algorithm whose number of arithmetic operations does not depend on the length of the numbers involved; if this number is polynomial, this is a *strongly polynomial algorithm* [24]. For example, the ellipsoid method or the interior-point method which solve LP take time which does depend on the encoding length, and the existence of a strongly polynomial algorithm for LP remains a major open problem. So far, the only strongly polynomial ILP algorithms we are aware of exist for totally unimodular ILP [14], bimodular ILP [2], so-called binet ILP [1], and n-fold IP with constant block dimensions [6]. All remaining results, such as Lenstra's famous algorithm or the fixed-parameter tractable algorithm for n-fold IP which has recently led to several breakthroughs [4, 16, 18, 19], are not strongly polynomial.

1.1 Our Contributions

To clearly state our results 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*. A *strongly fixed-parameter tractable* (FPT) *algorithm* for the problem is one that solves it using $f(\alpha)\text{poly}(\beta)$ arithmetic operations and $g(\alpha)\text{poly}(\beta, \gamma)$ time, where f, g are some computable functions. If such an algorithm exists, we say that the problem is *strongly fixed-parameter tractable* (FPT) *parameterized by* α. Thus, such an algorithm both demonstrates that the problem is FPT *parameterized by* α because it runs in FPT time $g(\alpha)\text{poly}(\beta, \gamma)$, and provides a strongly polynomial algorithm for each fixed α. Having multiple parameters $\alpha_1, \ldots, \alpha_k$ simultaneously is understood as taking the *aggregate parameter* $\alpha = \alpha_1 + \cdots + \alpha_k$. If the algorithm involves oracles then the oracle queries are also counted as arithmetic operations and the answers to oracle queries should be polynomial in (β, γ). Each part of the input may have several entities, which may be presented in unary or binary, where $\langle e \rangle$ denotes the encoding length of an entity e presented in binary. For the parametric input the distinction between unary and binary is irrelevant.

ILP abounds in natural parameters: the dimension n, number of inequalities m, largest coefficient $\|A\|_\infty$, largest right-hand side $\|\mathbf{b}\|_\infty$, various structural parameters of A, etc. Here, we are interested in algorithms which are both strongly polynomial *and* FPT.

Recently it was shown that, if we have access to the so-called *Graver basis* of A, the problem (ILP) is polynomial time solvable even for various nonlinear objective functions [5, 21]. We show that all of these results can be extended to be strongly polynomial with only $\langle A \rangle$ as the arithmetic input.

▶ **Theorem 1.** *The problem* (ILP) *with arithmetic input* $\langle A \rangle$ *and numeric input* $\langle \mathbf{w}, \mathbf{b}, \mathbf{l}, \mathbf{u} \rangle$*, endowed with a Graver-best oracle for* A*, is solvable by a strongly polynomial oracle algorithm.*

The existence of Graver-best oracles is thus of prime interest. We show such oracles for the wide classes of multi-stage stochastic and tree-fold ILPs; for precise definitions of these classes cf. Section 3.1.2. See Table 1 for a summary of improvements over the current state of the art.

▶ **Theorem 2.** *Multi-stage stochastic ILP with blocks* $B_1, \ldots, B_\tau$*,* $B_i \in \mathbb{Z}^{l \times n_i}$*, is strongly* FPT *parameterized by* $l + n_1 + \cdots + n_\tau$ *and* $\|A\|_\infty$*.*

▶ **Theorem 3.** *Tree-fold ILP with blocks* $A_1, \ldots, A_\tau$*,* $A_i \in \mathbb{Z}^{r_i \times t}$*, is strongly* FPT *parameterized by* $r_1 + \cdots + r_\tau$ *and* $\|A\|_\infty$*.*

This improves on the algorithm for tree-fold ILP [4] not only by making it strongly FPT, but also by leaving the block length t out of the parameter. Similarly, the following algorithm for the special case of n-fold ILP greatly improves both on the previous results of Hemmecke et al. [13] and Knop et al. [18] and is the currently fastest algorithm for this problem:

▶ **Theorem 4.** *n-fold ILP with blocks* $A_1 \in \mathbb{Z}^{r \times t}$ *and* $A_2 \in \mathbb{Z}^{s \times t}$ *can be solved in time* $a^{O(r^2 s + r s^2)}(nt)^6 \log(nt) + \mathcal{L}(\langle A \rangle)$*, where* $\mathcal{L}(\langle A \rangle)$ *is the runtime of a strongly polynomial LP algorithm.*

Next, we turn our attention to structural parameters of the constraint matrix A. We focus on two graphs which can be associated with A:

- the *primal graph* $G_P(A)$, which has a vertex for each column and two vertices are connected if there exists a row such that both columns are non-zero, and,
- the *dual graph* $G_D(A) = G_P(A^\intercal)$, which is the above with rows and columns swapped.

Two standard parameters of structural sparsity are the *treewidth* (measuring the "tree-likeness" of a graph) and the more restrictive *treedepth* (measuring its "star-likeness"). We denote the treewidth of $G_P(A)$ and $G_D(A)$ by $\mathrm{tw}_P(A)$ and $\mathrm{tw}_D(A)$; for treedepth we have $\mathrm{td}_P(A)$ and $\mathrm{td}_D(A)$. Note that bounded treedepth implies bounded treewidth but not vice versa.

We show that ILP parameterized by $\mathrm{td}_P(A) + \|A\|_\infty$ and $\mathrm{td}_D(A) + \|A\|_\infty$ can be reduced to the previously mentioned classes, respectively, implying (ILP) with these parameters is strongly FPT.

▶ **Theorem 5.** (ILP) *is strongly* FPT *parameterized by* $\mathrm{td}_P(A)$ *and* $\|A\|_\infty$*.*

This improves in two ways upon the result of Ganian and Ordyniak [10] who show that (ILP) with $\mathbf{w} \equiv \mathbf{0}$ (i.e. deciding the feasibility) is FPT parameterized by $\mathrm{td}_P(A) + \|A, \mathbf{b}\|_\infty$ [10]. First, we use the smaller parameter $\|A\|_\infty$ instead of $\|A, \mathbf{b}\|_\infty$, and second, we solve not only the feasibility but also the optimization problem. An analogous result holds for the parameter $\mathrm{td}_D(A)$, for which previously nothing was known at all.

▶ **Theorem 6.** (ILP) *is strongly* FPT *parameterized by* $\mathrm{td}_D(A)$ *and* $\|A\|_\infty$*.*

Table 1 Run time improvements implied by this paper. We denote by L the binary length of the numeric input $\mathbf{b}, \mathbf{l}, \mathbf{u}, \mathbf{w}$, i.e., $L = \langle \mathbf{b}, \mathbf{l}, \mathbf{u}, \mathbf{w} \rangle$, and consider $\langle A \rangle$ to be part of the arithmetic input. We denote by $a = \max\{2, \|A\|_\infty\}$, by r, s, t the relevant block dimensions (cf. Section 3.1), and by $\mathcal{L}(\langle A \rangle)$ the runtime of a strongly polynomial LP algorithm [24].

Type of instance	Previous best run time	Our result
n-fold ILP	$a^{O(rst+st^2)}n^3L$ [13]	$a^{O(r^2s+sr^2)}(nt)^6 \log(nt) + \mathcal{L}(\langle A \rangle)$ Thm 4
n-fold ILP	$n^{f_1(a,r,s,t)}$ [6]	
n-fold ILP	$t^{O(r)}(ar)^{r^2}n^3L$ if $A_2 = (1\ 1\ \cdots\ 1)$ [18]	$a^{O(r^2)}(nt)^6 \log(nt) + \mathcal{L}(\langle A \rangle)$ Thm 4
tree-fold ILP	$f_{\text{tf}'}(a, n_1, \ldots, n_\tau, t)n^3L$ [4]	$f_{\text{tf}}(a, n_1, \ldots, n_\tau)(nt)^3 + \mathcal{L}(\langle A \rangle)$ Thm 3
Multi-stage stochastic ILP	$f_{\text{mss}}(a, n_1, \ldots, n_\tau, l)n^3L$ [3]	$f_{\text{mss}}(a, n_1, \ldots, n_\tau, l)n^3 + \mathcal{L}(\langle A \rangle)$ Thm 2
Bounded dual treedepth	Open whether fixed-parameter tractable	$f_D(a, \text{td}_D(A)(nt)^3 + \mathcal{L}(\langle A \rangle)$ Thm 6
Bounded primal treedepth	$f_{P'}(a, \|\mathbf{b}\|_\infty, \text{td}_P(A))nL$ [10]	$f_P(a, \text{td}_P(A))n^3 + \mathcal{L}(\langle A \rangle)$ Thm 5

We emphasize that the parameterizations cannot be relaxed neither from treedepth to treewidth, nor by removing the parameter $\|A\|_\infty$: (ILP) is NP-hard already on instances with $\text{tw}_P(A) = 3$ and $\|A\|_\infty = 2$ [10, Thm 12], and it is strongly W[1]-hard parameterized by $\text{td}_P(A)$ alone [10, Thm 11]; the fact that a problem is W[1]-hard is strong evidence that it is not FPT. Similarly, deciding feasibility is NP-hard on instances with $\text{tw}_D(A) = 3$ and $\|A\|_\infty = 2$ (Lemma 18) and strongly W[1]-hard parameterized by $\text{td}_D(A)$ alone [19, Thm 5].

1.2 Interpretation of Results

We believe our approach also leads to several novel insights. First, we make it clear that the central question is finding Graver-best oracles; provided these oracles, Theorem 1 shows that tasks such as optimization and finding initial solutions can be handled under very mild assumptions. Even though we show these tasks are routine, they have been reimplemented repeatedly [4, 12, 13, 18].

Second, we show that the special classes of highly uniform block structured ILPs, namely multi-stage stochastic and tree-fold ILPs, are in some sense *universal* for all ILPs of bounded primal or dual treedepth, respectively. Specifically, we show that any ILP with bounded primal or dual treedepth can be embedded in an equivalent multi-stage stochastic or tree-fold ILP, respectively (Lemmas 25 and 26).

Third, we show that, besides bounded primal or dual treedepth, the crucial property for efficiency is the existence of augmenting steps with bounded ℓ_∞- or ℓ_1-norms, respectively (Lemmas 19 and 21). This suggests that for ILPs whose primal or dual graph is somehow "sparse" and "shallow", finding augmenting steps of bounded ℓ_∞- or ℓ_1-norm might be both sufficient for reaching the optimum and computationally efficient.

1.3 Related Work

We have already covered all relevant work regarding strongly polynomial algorithms for ILP.

Let us focus on structural parameterizations. It follows from Freuder's algorithm [9] and was reproven by Jansen and Kratsch [15] that (ILP) is FPT parameterized by $\text{tw}_P(A)$ and the largest domain $\|\mathbf{u} - \mathbf{l}\|_\infty$. Regarding the dual graph $G_D(A)$, the parameters $\text{td}_D(A)$ and $\text{tw}_D(A)$ were only recently considered by Ganian et al. [11]. They show that even deciding feasibility of (ILP) is NP-hard on instances with $\text{tw}_I(A) = 3$ ($\text{tw}_I(A)$ denotes the treewidth of the *incidence graph*; $\text{tw}_I(A) \leq \text{tw}_D(A) + 1$ always holds) and $\|A\|_\infty = 2$ [11, Theorem 12].

Furthermore, they show that (ILP) is FPT parameterized by $\mathrm{tw}_I(A)$ and parameter Γ, which is an upper bound on any prefix sum of $A\mathbf{x}$ for any feasible solution $\mathbf{x}$.

Dvořák et al [7] introduce the parameter fracture number; having a bounded *variable fracture number* $\mathfrak{p}^V(A)$ implies that deleting a few columns of A breaks it into independent blocks of small size; similarly for *constraint fracture number* $\mathfrak{p}^C(A)$ and deleting a few rows. Because bounded $\mathfrak{p}^V(A)$ implies bounded $\mathrm{td}_P(A)$ and bounded $\mathfrak{p}^C(A)$ implies bounded $\mathrm{td}_D(A)$, our results generalize theirs. The remaining case of *mixed fracture number* $\mathfrak{p}(A)$, where deleting both rows and columns is allowed, reduces to the 4-block n-fold ILP problem, which is not known to be either FPT or W[1]-hard. Because bounded $\mathfrak{p}(A)$ implies bounded $\mathrm{td}_I(A)$, ILP parameterized by $\mathrm{td}_I(A) + \|A\|_\infty$ is at least as hard as 4-block n-fold ILP, highlighting its status as an important open problem.

Organization. The paper contains three main parts. In Section 2, we provide the proof of Theorem 1, showing the existence of a strongly polynomial algorithm whenever a Graver-best oracle is provided. Then, in Section 3, we provide Graver-best oracles for multi-stage stochastic and tree-fold ILPs and discuss n-fold ILP, and prove Theorems 2, 3 and 4. Finally, in Section 4 we show how to embed any instance of bounded primal or dual treedepth into a multi-stage stochastic or tree-fold ILP without increasing the relevant parameters, proving Theorems 5 and 6. Due to space restrictions the proofs of our technical statement and other supplementary material are moved to the full version available at `https://arxiv.org/abs/1802.05859`; the statements whose proofs are presented there are marked with (*) .

2 The Graver-best Oracle Algorithm

2.1 Preliminaries

For positive integers m, n, $m \leq n$, we set $[m, n] = \{m, \ldots, n\}$ and $[n] = [1, n]$. 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). If A is a matrix, A_r denotes its r-th column. For an integer $a \in \mathbb{Z}$, we denote by $\langle a \rangle = 1 + \log_2 a$ 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 $\langle A \rangle = \sum_{i,j} \langle a_{ij} \rangle$. For a graph G we denote by $V(G)$ its set of vertices.

Graver bases and augmentation. Let us now 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 $x_i y_i \geq 0$ (that is, $\mathbf{x}$ and $\mathbf{y}$ lie in the same orthant) and $|x_i| \leq |y_i|$ for $i \in [n]$. It is well known that every subset of $\mathbb{Z}^n$ has finitely many $\sqsubseteq$-minimal elements.

▶ **Definition 7** (Graver basis)**.** 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 $\{\mathbf{x} \in \mathbb{Z}^n : A\mathbf{x} = 0,\ \mathbf{x} \neq 0\}$.

We say that $\mathbf{x}$ is *feasible* for (ILP) if $A\mathbf{x} = \mathbf{b}$ and $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$. Let $\mathbf{x}$ be a feasible solution for (ILP). We call $\mathbf{g}$ a *feasible step* if $\mathbf{x} + \mathbf{g}$ is feasible for (ILP). Further, call a feasible step $\mathbf{g}$ *augmenting* if $\mathbf{w}(\mathbf{x} + \mathbf{g}) < \mathbf{w}(\mathbf{x})$. An augmenting step $\mathbf{g}$ and a *step length* $\alpha \in \mathbb{Z}$ form an $\mathbf{x}$-*feasible step pair* with respect to a feasible solution $\mathbf{x}$ if $\mathbf{l} \leq \mathbf{x} + \alpha\mathbf{g} \leq \mathbf{u}$. An augmenting step $\mathbf{h}$ is a *Graver-best step* for $\mathbf{x}$ if $\mathbf{w}(\mathbf{x} + \mathbf{h}) \leq \mathbf{w}(\mathbf{x} + \lambda\mathbf{g})$ for all $\mathbf{x}$-feasible step pairs $(\mathbf{g}, \lambda) \in \mathcal{G}(A) \times \mathbb{Z}$. The *Graver-best augmentation procedure* for (ILP) with a given feasible solution $\mathbf{x}_0$ works as follows:

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

▶ **Proposition 8** ([21, Lemma 3.10]). *Given a feasible solution* $\mathbf{x}_0$ *for* (ILP)*, the Graver-best augmentation procedure finds an optimum of* (ILP) *in at most* $(2n-2)\log F$ *steps, where* $F = \mathbf{w}\mathbf{x}_0 - \mathbf{w}\mathbf{x}^*$ *and* $\mathbf{x}^*$ *is any minimizer of* $\mathbf{w}\mathbf{x}$*.*

▶ **Definition 9** (Graver-best oracle). A *Graver-best oracle* for an integer matrix A is one that, queried on $\mathbf{w}, \mathbf{b}, \mathbf{l}, \mathbf{u}$ and $\mathbf{x}$ feasible to (ILP), returns a Graver-best step $\mathbf{h}$ for $\mathbf{x}$.

2.2 The Algorithm

It follows from Proposition 8 that given a Graver-best oracle, problem (ILP) can be solved in time which is polynomial in the binary encoding length $\langle A, \mathbf{w}, \mathbf{b}, \mathbf{l}, \mathbf{u}\rangle$ of the input. We now show that, in fact, given such an oracle, the problem admits a *strongly* polynomial algorithm. In the next theorem the input has only arithmetic and numeric parts and no parametric part.

▶ **Theorem 1.** *The problem* (ILP) *with arithmetic input* $\langle A\rangle$ *and numeric input* $\langle \mathbf{w}, \mathbf{b}, \mathbf{l}, \mathbf{u}\rangle$*, endowed with a Graver-best oracle for* A*, is solvable by a strongly polynomial oracle algorithm.*

▶ Remark. 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, 24].

Proof. The algorithm which demonstrates the theorem consists of several steps as follows.

Step 1: Reducing $\mathbf{b}, \mathbf{l}, \mathbf{u}$. Apply the strongly polynomial algorithm of Tardos [24] to the linear programming relaxation $\min\{\mathbf{w}\mathbf{y} \mid \mathbf{y} \in \mathbb{R}^n,\ A\mathbf{y} = \mathbf{b},\ \mathbf{l} \le \mathbf{y} \le \mathbf{u}\}$; the algorithm performs $\mathcal{L}(\langle A\rangle) = \mathrm{poly}(\langle A\rangle)$ 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 $\lfloor \mathbf{y}^* \rfloor \in \mathbb{Z}^n$. Let $a := \max\{2, \|A\|_\infty\}$. Let $\mathcal{C}(A) \subseteq \mathcal{G}(A)$ be the set of *circuits* of A, which are those $\mathbf{c} \in \mathcal{G}(A)$ with support which is a circuit of the linear matroid of A. Let $c_\infty := \max_{\mathbf{c}\in\mathcal{C}(A)} \|\mathbf{c}\|_\infty$. We have $c_\infty \le n^{\frac{n}{2}} a^n$ [21, Lemma 3.18].

We now use the proximity results of [12, 14] which assert that either (ILP) is infeasible or it has an optimal solution $\mathbf{x}^*$ with $\|\mathbf{x}^* - \mathbf{y}^*\|_\infty \le n c_\infty$ and hence $\|\mathbf{x}^* - \lfloor \mathbf{y}^* \rfloor\|_\infty \le n^{\frac{n}{2}+1} a^n + 1$. Thus, making the variable transformation $\mathbf{x} = \mathbf{z} + \lfloor \mathbf{y}^* \rfloor$, problem (ILP) reduces to following,

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

which is equivalent to the program

$$\min\left\{\mathbf{w}\mathbf{z} \mid \mathbf{z} \in \mathbb{Z}^n\,,\ A\mathbf{z} = \bar{\mathbf{b}}\,,\ \bar{\mathbf{l}} \le \mathbf{z} \le \bar{\mathbf{u}}\right\} \tag{1}$$

where

$$\bar{\mathbf{b}} := \mathbf{b} - A\lfloor \mathbf{y}^* \rfloor,\ \ \bar{l}_i := \max\{l_i - \lfloor y_i^* \rfloor, -(n^{\frac{n}{2}+1} a^n + 1)\},\ \ \bar{u}_i := \min\{u_i - \lfloor y_i^* \rfloor, n^{\frac{n}{2}+1} a^n + 1\}\quad .$$

If some $\bar{l}_i > \bar{u}_i$ then (1) is infeasible and hence so is (ILP), so we may assume that

$$-(n^{\frac{n}{2}+1} a^n + 1) \le \bar{l}_i \le \bar{u}_i \le n^{\frac{n}{2}+1} a^n + 1, \quad \text{for all } i\ .$$

This implies that if $\mathbf{z}$ is any feasible point in (1) then $\|A\mathbf{z}\|_\infty \le na(n^{\frac{n}{2}+1} a^n + 1)$ and so we may assume that $\|\bar{\mathbf{b}}\|_\infty \le na(n^{\frac{n}{2}+1} a^n + 1)$ else there is no feasible solution. So we have

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

Step 2: Solving the system of equations. We first search for an integer solution to the system of equations $A\mathbf{z} = \bar{\mathbf{b}}$. This can be done by computing the Hermite normal form of A, see [23], using a number of arithmetic operations polynomial in $\langle A\rangle$ and time polynomial in $\langle A, \bar{\mathbf{b}}\rangle$ which is polynomial in $\langle A\rangle$, and hence strongly polynomially in our original input. Then either we conclude that there is no integer solution to $A\mathbf{z} = \bar{\mathbf{b}}$ and hence (1) is infeasible, or we find a solution $\mathbf{z} \in \mathbb{Z}^n$ with $\langle \mathbf{z}\rangle$ polynomially bounded in $\langle A, \bar{\mathbf{b}}\rangle$ and hence also in $\langle A\rangle$.

Step 3: Finding a feasible point. Define relaxed bounds by

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

Now for $i \in [n]$ iterate the following. If $\bar{l}_i \le z_i \le \bar{u}_i$ then simply increment i and repeat. If $z_i < \bar{l}_i$ (and hence $\hat{l}_i = z_i$ and $\hat{u}_i = \bar{u}_i$) then consider the following auxiliary integer program,

$$\max\left\{x_i \mid \mathbf{x} \in \mathbb{Z}^n \,,\ A\mathbf{x} = \bar{\mathbf{b}} \,,\ \hat{\mathbf{l}} \le \mathbf{x} \le \hat{\mathbf{u}}\right\}. \tag{2}$$

Starting from the point $\mathbf{z}$ feasible in (2), and using the Graver-best oracle for A, we can solve program (2) using Proposition 8 in polynomial time and in a number of arithmetic operations and oracle queries which is polynomial in n and $\log F$ (recall $F = z_i^* - z_i$ for some minimizer z_i^*), which is bounded by $\log(\hat{u}_i - \hat{l}_i) = \log(\bar{u}_i - z_i)$, thus polynomial in $\langle A\rangle$.

Let $\mathbf{x}$ be an optimal solution of (2). If $x_i < \bar{l}_i$ then (1) is infeasible and we are done. Otherwise (in which case $\bar{l}_i \le x_i \le \bar{u}_i$) we update $\hat{l}_i := \bar{l}_i$ and $\mathbf{z} := \mathbf{x}$, increment i and repeat. The last case $z_i > \bar{u}_i$ is treated similarly where in (2) we minimize rather than maximize x_i.

Thus, strongly polynomially we either conclude at some iteration i that program (1) is infeasible or complete all iterations and obtain $\hat{\mathbf{l}} = \bar{\mathbf{l}}$, $\hat{\mathbf{u}} = \bar{\mathbf{u}}$, and a point $\mathbf{z}$ feasible in (1).

Step 4: Reducing $\mathbf{w}$. Let $N := 2n(n^{\frac{n}{2}+1}a^n + 1) + 1$. 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 $\|\bar{\mathbf{w}}\|_\infty \le 2^{O(n^3)}N^{O(n^2)}$ such that $\operatorname{sign}(\mathbf{wx}) = \operatorname{sign}(\bar{\mathbf{w}}\mathbf{x})$ for all $\mathbf{x} \in \mathbb{Z}^n$ with $\|\mathbf{x}\|_1 < N$. Since $\langle N\rangle = O(\log N) = O(n\log n + n\log a)$ 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 (1) we have $\|\mathbf{x} - \mathbf{z}\|_1 < 2n(n^{\frac{n}{2}+1}a^n + 1) + 1 = N$, so that for any two such points we have $\mathbf{wx} \le \mathbf{wz}$ if and only if $\bar{\mathbf{w}}\mathbf{x} \le \bar{\mathbf{w}}\mathbf{z}$, and therefore we can replace (1) by the equivalent program

$$\min\left\{\bar{\mathbf{w}}\mathbf{z} \ :\ \mathbf{z} \in \mathbb{Z}^n \,,\ A\mathbf{z} = \bar{\mathbf{b}} \,,\ \bar{\mathbf{l}} \le \mathbf{z} \le \bar{\mathbf{u}}\right\}, \tag{3}$$

where

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

Step 5: Finding an optimal solution. Starting from the point $\mathbf{z}$ which is feasible in (3), and using the Graver-best oracle for A, we can solve program (3) using again Proposition 8 in polynomial time and in a number of arithmetic operations and oracle queries which is polynomial in n and in $\log F$, 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. ◀

▶ Remark. In fact, the reduced objective $\bar{\mathbf{w}}$ in step 4 need not be constructed: already its *existence* implies that (1) is solved in the same number of iterations as (3).

3 Multi-stage Stochastic and Tree-fold ILP

In this section we prove Theorems 2 and 3. We first formalize a common construction for a Graver-best oracle: one constructs a set of relevant step lenghts Λ and then for each $\lambda \in \Lambda$ finds a λ-Graver-best step. A step with the best improvement among these is then guaranteed to be a Graver-best step. Thus, we reduce our task to constructing a Λ-Graver-best oracle.

Both algorithms for multi-stage stochastic ILP and tree-fold ILP follow the same pattern:

1. show that all elements of $\mathcal{G}(A)$ have bounded norms (ℓ_∞ and ℓ_1, respectively),
2. show that A has bounded treewidth (primal and dual, respectively),
3. apply existing algorithms for (ILP) which are FPT parameterized by $\|A\|_\infty$, $\max \|\mathbf{x}\|_\infty$ and $\max \|\mathbf{x}\|_1$, and $\mathrm{tw}_P(A)$ and $\mathrm{tw}_D(A)$, respectively.

3.1 Preliminaries

3.1.1 Relevant Step Lengths

We say that $\mathbf{h} \in \{\mathbf{x} \in \mathbb{Z}^n \mid A\mathbf{x} = \mathbf{0}\}$ is a *λ-Graver-best step* if $\lambda\mathbf{h}$ is a feasible step and $\lambda\mathbf{w}\mathbf{h} \leq \lambda\mathbf{w}\mathbf{g}$ for any $\mathbf{g} \in \mathcal{G}(A)$ such that $\lambda\mathbf{g}$ is a feasible step. We denote by $g_1(A) = \max_{\mathbf{g}\in\mathcal{G}(A)} \|\mathbf{g}\|_1$ and $g_\infty(A) = \max_{\mathbf{g}\in\mathcal{G}(A)} \|\mathbf{g}\|_\infty$. The following lemma states that provided a bound on $g_\infty(A)$, in order to find a Graver-best step, it is sufficient to find a λ-Graver-best step for all $\lambda \in \Lambda$ for some not too large set Λ.

▶ **Definition 10** (Graver-best step-lengths)**.** Let $\mathbf{x}$ be a feasible solution to (ILP). We say that $\lambda \in \mathbb{N}$ is a *Graver-best step-length for* $\mathbf{x}$ if there exists $\mathbf{g} \in \mathcal{G}(A)$ with $\mathbf{x} + \lambda\mathbf{g}$ feasible, such that $\forall \lambda' \in \mathbb{N}$ and $\forall \mathbf{g}' \in \mathcal{G}(A)$, $\mathbf{x} + \lambda'\mathbf{g}'$ is either infeasible or $\mathbf{w}(\mathbf{x} + \lambda\mathbf{g}) \leq \mathbf{w}(\mathbf{x} + \lambda'\mathbf{g}')$. We denote by $\Lambda(\mathbf{x}) \subseteq \mathbb{N}$ the set of Graver-best step-lengths for $\mathbf{x}$.

▶ **Lemma 11** (Polynomial $\Lambda \supseteq \Lambda(\mathbf{x})$)**. (*)** *Let $\mathbf{x}$ be a feasible solution to* (ILP)*, let $M \in \mathbb{N}$ be such that $g_\infty(A) \leq M$. Then it is possible to construct in time $O(Mn)$ a set $\Lambda \subseteq \mathbb{N}$ of size at most $2Mn$ such that $\Lambda(\mathbf{x}) \subseteq \Lambda$.*

With this Λ at hand, in order to realize a Graver-best oracle, it suffices to realize an oracle which finds a λ-Graver-best step for a given λ:

▶ **Definition 12** (Λ-Graver-best oracle)**.** A *Λ-Graver-best oracle* for an integer matrix A is one that, queried on $\mathbf{w}, \mathbf{b}, \mathbf{l}, \mathbf{u}$, $\mathbf{x}$ feasible to (ILP), and an integer $\lambda \in \mathbb{N}$, returns a λ-Graver-best step $\mathbf{h}$ for $\mathbf{x}$.

▶ **Lemma 13** (Λ-Graver-best oracle $\Rightarrow$ Graver-best oracle)**. (*)** *Let A be an integer matrix and let $M \in \mathbb{N}$ satisfy $g_\infty(A) \leq M$. Then a Graver-best oracle for A can be realized with $2Mn$ calls to a Λ-Graver-best oracle for A.*

3.1.2 Multi-stage Stochastic and Tree-fold Matrices

Let the *height* of a rooted tree or forest be the maximum root-to-leaf distance in it (i.e., the number of edges along the root-to-leaf path). In the following we let T be a rooted tree of height $\tau - 1 \in \mathbb{N}$ whose all leaves are at *depth* $\tau - 1$, that is, the length of every root-leaf path is exactly $\tau - 1$. For a vertex $v \in T$, let T_v be the subtree of T rooted in v and let $\ell(v)$ denote the number of leaves of T contained in T_v. Let $B_1, B_2, \ldots, B_\tau$ be a sequence of integer matrices with each B_s having $l \in \mathbb{N}$ rows and n_s columns, where $n_s \in \mathbb{N}$, $n_s \geq 1$. We shall define a *multi-stage stochastic* matrix $T^P(B_1, \ldots, B_\tau)$ inductively; the superscript P

refers to the fact that $T^P(B_1,\dots,B_\tau)$ has bounded *primal* treedepth td_P, as we will later see.

For a leaf $v \in T$, $T^P_v(B_\tau) := B_\tau$. Let $d \in \mathbb{N}$, $0 \le d \le \tau - 2$, and assume that for all vertices $v \in T$ at depth $d+1$, matrices $T^P_v(B_{\tau-d},\dots,B_\tau)$ have been defined. For $s \in \mathbb{N}$, $1 \le s \le \tau$, we set $T^P_v(B_{[s:\tau]}) = T^P_v(B_s,\dots,B_\tau)$. Let $v \in T$ be a vertex at depth d with δ children $v_1,\dots,v_\delta$. We set

$$T^P_v(B_{[\tau-d-1:\tau]}) := \begin{pmatrix} B_{\tau-d-1,\ell(v_1)} & T^P_{v_1}(B_{[\tau-d:\tau]}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ B_{\tau-d-1,\ell(v_\delta)} & 0 & \cdots & T^P_{v_\delta}(B_{[\tau-d:\tau]}) \end{pmatrix}$$

where, for $N \in \mathbb{N}$, $B_{s,N} = \begin{pmatrix} B_s \\ \vdots \\ B_s \end{pmatrix}$ consists of N copies of the matrix B_s.

The structure of a multi-stage stochastic matrix makes it natural to partition any solution of a multi-stage stochastic ILP into *bricks*. Bricks are defined inductively: for $T^P_v(B_\tau)$ there is only one brick consisting of all coordinates; for $T^P_v(B_{[s:\tau]})$ the set of bricks is composed of all bricks for all descendants of v, plus the first n_s coordinates form an additional brick.

▶ **Example 14.** For $\tau = 3$ and T with root r of degree 2 and its children u and v of degree 2 and 3, we have $T^P_u(B_2,B_3) = \left(\begin{smallmatrix} B_2 & B_3 \\ B_2 & B_3 \end{smallmatrix}\right)$, $T^P_v(B_2,B_3) = \left(\begin{smallmatrix} B_2 & B_3 & \\ B_2 & & B_3 \\ B_2 & & B_3 \end{smallmatrix}\right)$, and $T^P(B_1,B_2,B_2) = T^P_r(B_1,B_2,B_2) = \left(\begin{smallmatrix} B_1 & B_2 & B_3 & & \\ B_1 & B_2 & & B_3 & \\ B_1 & & B_2 & B_3 & \\ B_1 & & B_2 & & B_3 \end{smallmatrix}\right)$, with a total of 8 bricks.

Tree-fold matrices are essentially transposes of multi-stage stochastic ILP matrices. Let T be as before and $A_1,\dots,A_\tau$ be a sequence of integer matrices with each $A_s \in \mathbb{Z}^{r_s \times t}$, where $t \in \mathbb{N}$, $r_s \in \mathbb{N}$, $r_s \ge 1$. We shall define $T^D(A_1,\dots,A_\tau)$ inductively; the superscript D refers to the fact that $T^D(A_1,\dots,A_\tau)$ has bounded *dual* treedepth. The inductive definition is the same as before except that, for a vertex $v \in T$ at depth d with δ children $v_1,\dots,v_\delta$, we set

$$T^D_v(A_{[\tau-d-1:\tau]}) := \begin{pmatrix} A_{\tau-d-1,\ell(v_1)} & A_{\tau-d-1,\ell(v_2)} & \cdots & A_{\tau-d-1,\ell(v_\delta)} \\ T^D_{v_1}(A_{[\tau-d:\tau]}) & 0 & \cdots & 0 \\ 0 & T^D_{v_2}(A_{[\tau-d:\tau]}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & T^D_{v_\delta}(A_{[\tau-d:\tau]}) \end{pmatrix}$$

where, for $N \in \mathbb{N}$, $A_{s,N} = (A_s \cdots A_s)$ consists of N copies of the matrix A_s. A solution $\mathbf{x}$ of a tree-fold ILP is partitioned into bricks $(\mathbf{x}^1,\dots,\mathbf{x}^n)$ where n is the number of leaves of T, and each $\mathbf{x}^i$ is a t-dimensional vector.

3.1.3 Structural Parameters

We consider two graph parameters, namely *treewidth* $\mathrm{tw}(G)$ and *treedepth* $\mathrm{td}(G)$. We postpone the definition of treewidth to the full version as it is not central for us.

▶ **Definition 15** (Treedepth)**.** The *closure* $\mathrm{cl}(F)$ of a rooted forest F is the graph obtained from F by making every vertex adjacent to all of its ancestors. The *treedepth* $\mathrm{td}(G)$ of a graph G is one more than the minimum height of a forest F such that $G \subseteq \mathrm{cl}(F)$.

It is known that $\mathrm{tw}(G) \le \mathrm{td}(G)$. The treedepth $\mathrm{td}(G)$ of a graph G with a witness forest F can be computed in time $f_{\mathrm{td}}(\mathrm{td}(G)) \cdot |V(G)|$ for some computable function f_{td} [22].

▶ **Definition 16** (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 $E = \{\{i, j\} \in \binom{[n]}{2} \mid \exists k \in [m] : A_{k,i}, A_{k,j} \neq 0\}$. 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.

▶ **Definition 17** (Matrix treewidth)**.** Given a matrix A, its *primal treewidth* $\mathrm{tw}_P(A)$ is defined as the treewidth of its primal graph, i.e., $\mathrm{tw}(G_P(A))$, and its *dual treewidth* $\mathrm{tw}_D(A)$ is $\mathrm{tw}(G_D(A))$. Similarly, we define the *primal* and *dual treedepth* as $\mathrm{td}_P(A) = \mathrm{td}(G_P(A))$ and $\mathrm{td}_D(A) = \mathrm{td}(G_D(A))$, respectively.

Using a proof of Ganian et al. [11, Theorem 12] we show that we cannot hope to relax the parameter $\mathrm{td}_D(A)$ to $\mathrm{tw}_D(A)$, even if $\|A\|_\infty$ was a constant.

▶ **Lemma 18. (*)** (ILP) *is* NP*-hard already when* $\mathrm{tw}_D(A) = 3$*,* $\|A\|_\infty = 2$*, and* $\mathbf{w} = \mathbf{0}$*.*

3.2 Multi-stage Stochastic ILP is strongly FPT

To prove Theorem 2, we need two ingredients: a bound on $g_\infty(A)$, and an algorithm for (ILP) with bounded $\mathrm{tw}_P(A)$ and $\max \|\mathbf{x}\|_\infty$.

▶ **Lemma 19** (Multi-stage stochastic ⇒ bounded $g_\infty(A)$)**. (*)** *Let* $A = T^P(B_1, \ldots, B_\tau)$*. Then* $g_\infty(A) \leq f_{mss\text{-}norm}(a, n_1, \ldots, n_\tau, l)$ *for some computable function* $f_{mss\text{-}norm}$*.*

▶ **Lemma 20. (*)** *Let* $X \in \mathbb{N}$*. Problem* (ILP) *with the additional constraint* $\|\mathbf{x}\|_\infty \leq X$ *can be solved in time* $(X + 1)^{O(\mathrm{tw}_P(A))} \cdot (n + m)$*.*

Proof of Theorem 2. Let $A = T^P(B_1, \ldots, B_\tau)$ be a multi-stage stochastic matrix. By Lemma 19, $g_\infty(A)$ is bounded by $M = f_{\text{mss-norm}}(a, n_1, \ldots, n_\tau, l)$. We show how to construct a Λ-Graver-best oracle. Given an integer $\lambda \in \mathbb{N}$, use Lemma 20 to solve

$$\min\{\lambda\mathbf{wh} \mid A\mathbf{h} = 0,\ \mathbf{l} \leq \mathbf{x} + \lambda\mathbf{h} \leq \mathbf{u},\ \|\mathbf{h}\|_\infty \leq M\}\ .$$

This returns a λ-Graver-best step, because any optimal solution satisfies $\lambda\mathbf{h} \leq \lambda\mathbf{g}$ for all $\mathbf{g} \in \mathcal{G}(A)$. Using a simple induction and the inductive construction of A, one gets that A has $\mathrm{tw}_P(A) \leq \mathrm{td}_P(A) \leq n_1 + \cdots + n_\tau + 1$ and thus the oracle is realized in FPT time. Lemma 13 then yields a Graver-best oracle, which, combined with Theorem 1, finishes the proof. ◀

3.3 Tree-fold ILP is strongly FPT

As before, to prove Theorem 3, we need two ingredients: a bound on $g_1(A)$, and an algorithm for (ILP) with bounded $\mathrm{tw}_D(A)$ and $\max \|\mathbf{x}\|_1$.

▶ **Lemma 21** (Tree-fold ⇒ bounded $g_1(A)$)**. (*)** *Let* $A_i \in \mathbb{Z}^{r_i\times t}$ *for* $i \in [\tau]$ *with* $a = \max\{2, \max_{i\in[\tau]} \|A_i\|_\infty\}$*,* $r = \sum_{i=1}^{\tau} r_i$*. Let* $A = T^D(A_1, \ldots, A_\tau)$*. There exists a computable function* $f_{tf\text{-}norm}(a, r_1, \ldots, r_\tau)$ *such that* $g_1(A) \leq f_{tf\text{-}norm}(a, r_1, \ldots, r_\tau)$*.*

Proof sketch. Chen and Marx [4] prove a similar result under the assumption that t is also a parameter; thus, the remaining problem are essentially duplicitous columns. However, De Loera et al. [5] show that repeating columns of any matrix A' does not increase $g_1(A')$, and thus we can take A, delete duplicitous columns, apply the result of Chen and Marx, and our Lemma follows.

▶ **Lemma 22. (*)** *Let* $X \in \mathbb{N}$*. Problem* (ILP) *with the additional constraint* $\|\mathbf{x}\|_1 \leq X$ *can be solved in time* $(aX)^{O(\mathrm{tw}_D(A))} \cdot n$*, where* $a = \max\{2, \|A\|_\infty\}$*.*

Proof sketch. Lemma 22 is proved by reformulating the nonlinear constraint $\|\mathbf{x}\|_1 \leq X$ by "splitting" each variable x_i into two non-negative variables $x_i = x_i^+ - x_i^-$, imposing the constraint $\sum_{i=1}^n (x_i^+ + x_i^-) \leq X$, and showing that this does not increase $\mathrm{tw}_D(A)$ much; then, a recent dynamic programming algorithm of Ganian et al. [11, Theorem 6] does the job.

Proof of Theorem 3. Let $A = T^D(A_1, \ldots, A_\tau)$ be a tree-fold matrix. By Lemma 21 we have that $g_1(A) \leq f_{\text{tf-norm}}(a, r_1, \ldots, r_\tau) =: M$ We show how to construct a Λ-Graver-best oracle. Given an integer $\lambda \in \mathbb{N}$, solve $\min\{\lambda \mathbf{wh} \mid A\mathbf{h} = 0, \mathbf{l} \leq \mathbf{x} + \lambda\mathbf{h} \leq \mathbf{u}, \|\mathbf{h}\|_1 \leq M\}$ using Lemma 22; clearly the result is a λ-Graver-best step. Using a simple induction and the inductive construction of A, one gets that A has $\mathrm{tw}_D(A) \leq \mathrm{td}_D(A) \leq r_1 + \cdots + r_\tau + 1$ and thus the oracle is realized in FPT time. Lemma 13 then yields a Graver-best oracle, which, combined with Theorem 1, finishes the proof. ◀

***n*-fold ILP.** A special case of tree-fold ILP is n-fold ILP, obtained by taking T to be the star with n leaves and $A = T^D(A_1, A_2)$, where $A_1 \in \mathbb{Z}^{r \times t}$ and $A_2 \in \mathbb{Z}^{s \times t}$.

Proof of Theorem 4. Before we apply Lemma 22, we need to bound $g_1(A)$. It follows from the proof of [13, Lemma 6.1] that there is a number $g(A) = \max_{\mathbf{v} \in \mathcal{G}(A_1\mathcal{G}(A_2))} \|\mathbf{v}\|_1$ such that $g_1(A) \leq g(A) \cdot g_1(A_2)$.

Let $d_2 \leq (2a+1)^s$ be the number of distinct columns of A_2. De Loera et al. [5] give a bound on $g_1(A)$ in terms of the number of distinct columns of a matrix A:

▶ **Lemma 23** ([5, Corollary 3.7.4]). *Let $A \in \mathbb{Z}^{m \times n}$ be a matrix of rank r, let d be the number of different columns in A, and let $a = \max\{2, \|A\|_\infty\}$. Then $g_1(A) \leq (d-r)(r+1)(\sqrt{ma})^m$.*

Thus, $g_1(A_2) \leq (d_2 - s)(s+1)(\sqrt{s}a)^s \leq (as)^{O(s)}$. Let G_2 be a matrix whose columns are elements of $\mathcal{G}(A_2)$. We have that $\|A_1G_2\|_\infty \leq a \cdot (as)^{O(s)} \leq (as)^{O(s)}$. Moreover, since A_1G_2 has r rows, it has at most $d_1 = \left((as)^{O(s)}\right)^r = (as)^{O(rs)}$ distinct columns. Again, by Lemma 23 we have that $g(A) = g_1(A_1G_2) \leq (d_1 - r)(r+1)(\sqrt{r}as)^{O(s)})^r \leq (ars)^{O(rs)}$. Combining, we get $g_1(A) \leq (ars)^{O(rs)} \cdot (as)^{O(s)} \leq (ars)^{O(rs)} =: M$.

We have $\mathrm{tw}_D(A) \leq r + s + 1$ and thus running the algorithm of Lemma 22 once takes time $\left((ars)^{O(rs)}\right)^{r+s} nt \leq (ars)^{O(r^2s+rs^2)}nt$ and finds the λ-Graver-best step. Lemma 13 then yields a Graver-best oracle. The reduced objective function $\bar{\mathbf{w}}$ in Step 4 of the proof of Theorem 1 satisfies $\|\bar{\mathbf{w}}\|_\infty \leq (ant)^{O((nt)^3)}$ and thus the number of calls to the Graver-best oracle is bounded by $(nt)^3 \log(nt)$, concluding the proof. ◀

4 Primal and Dual Treedepth

We prove Theorems 5 and 6 by showing that an ILP with bounded primal (dual) treedepth can be embedded into a multi-stage stochastic (tree-fold) ILP without increasing the parameters too much. The precise notion of how one ILP is embedded in another is captured as follows.

▶ **Definition 24** (Extended formulation). Let $n' \geq n$, $m' \in \mathbb{N}$, $A \in \mathbb{Z}^{m \times n}$, $\mathbf{b} \in \mathbb{Z}^m$, $\mathbf{l}, \mathbf{u} \in (\mathbb{Z} \cup \{\pm\infty\}^n$ and $A' \in \mathbb{Z}^{m' \times n'}$, $\mathbf{b}' \in \mathbb{Z}^{m'}$, $\mathbf{l}', \mathbf{u}' \in (\mathbb{Z} \cup \{\pm\infty\})^{n'}$. We say that $A'(\mathbf{x}, \mathbf{y}) = \mathbf{b}', \mathbf{l}' \leq (\mathbf{x}, \mathbf{y}) \leq \mathbf{u}'$ is an *extended formulation* of $A\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$ if $\{\mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\} = \{\mathbf{x} \mid \exists \mathbf{y} : A'(\mathbf{x}, \mathbf{y}) = \mathbf{b}', \mathbf{l}' \leq (\mathbf{x}, \mathbf{y}) \leq \mathbf{u}'\}$.

We note that from here on we always assume that if $\mathrm{td}_P(A) = k$ or $\mathrm{td}_D(A) = k$, then there is a *tree* (not a forest) F of height $k-1$ such that $G_P(A) \subseteq \mathrm{cl}(F)$ or $G_D(A) \subseteq \mathrm{cl}(F)$, respectively. Otherwise $G_P(A)$ is not connected and each component corresponds to a subset of variables which defines an ILP that can be solved independently; similarly for $G_D(A)$.

4.1 Primal Treedepth

▶ **Lemma 25** (Bounded primal treedepth ⇒ multi-stage stochastic). *Let $A, \mathbf{b}, \mathbf{l}$ and $\mathbf{u}$ as in* (ILP) *be given, let $a = \max\{2, \|A\|_\infty\}$ and $\tau + 1 = \mathrm{td}_P(A)$. Then there exists $C \in \mathbb{Z}^{m' \times n'}$, $\mathbf{b}' \in \mathbb{Z}^{m'}$ and $\mathbf{l}', \mathbf{u}' \in (\mathbb{Z} \cup \{\pm\infty\})^{n'}$, $n' \leq n\tau$, $m' \leq (2a+1)^{\tau^2}$, which define an integer program $C(\mathbf{x}, \mathbf{y}) = \mathbf{b}', \mathbf{l}' \leq (\mathbf{x}, \mathbf{y}) \leq \mathbf{u}'$, which is an extended formulation of $A\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$. Moreover, there exist matrices $B_1, \ldots, B_{\tau-1} \in \mathbb{Z}^{(2a+1)^{\tau^2} \times \tau}$ and $B_\tau \in \mathbb{Z}^{(2a+1)^{\tau^2} \times \left(\tau + (2a+1)^{\tau^2}\right)}$ and a tree T such that $C = T^P(B_1, \ldots, B_\tau)$ is a multi-stage stochastic constraint matrix, and all can be computed in time $f_{P\text{-}embed}(a, \mathrm{td}_P(A)) \cdot n^2$ for some computable function $f_{P\text{-}embed}$.*

Proof. Let F be a rooted tree of height τ such that $G_P(A) \subseteq \mathrm{cl}(F)$ (recall that it can be computed in time $f_{\mathrm{td}}(\mathrm{td}_P(A)) \cdot |V(G_P(A))|$).

Step 1: Dummy columns. We make F structured by adding dummy columns. Observe that every root-leaf path is of length at most τ and thus contains at most τ branching vertices. Unless F is a path, obtain a matrix A' from A by inserting zero columns into A in order to make the path between any two branching vertices of length τ; a special case is the root which we force to be in distance $\tau - 1$ from the closest branching vertex. Set lower and upper bounds on the corresponding new variables to 0. A zero column is an isolated vertex in the primal graph and thus can be inserted to an arbitrary path of the tree F. Moreover, if any leaf is at depth less than $\tau^2 - 1$, insert zero columns in the same way to make it be at depth exacty $\tau^2 - 1$. Now there exists a rooted tree F' of height $\tau^2 - 1$ such that $G_P(A') \subseteq \mathrm{cl}(F')$, all branching vertices are in distances $0, \tau - 1, 2\tau - 1, \ldots, (\tau - 1)\tau - 1$ from the root, and all leaves are at depth exactly $\tau^2 - 1$. We call all vertices at depth 0 or $i\tau - 1$, for $i \in [\tau]$, *frets*, including the root and leaves.

Step 2: Multi-stage stochastic extended formulation. Consider a root-leaf path P in F': its vertex set $V(P)$ corresponds to a certain subset of the columns of A', with $|V(P)| \leq \tau^2$. Furthermore, any row $\mathbf{a}$ of A' with $\mathrm{supp}(\mathbf{a}) \subseteq V(P)$ can be written as a vector $(\mathbf{a}^1, \ldots, \mathbf{a}^\tau) \in \mathbb{Z}^{\tau^2}$ with $\mathbf{a}^i \in \mathbb{Z}^\tau$ for each $i \in [\tau]$, and with $\|\mathbf{a}\|_\infty \leq a$. The *bricks* $\mathbf{a}^i$ correspond to segments between frets (including the end fret, i.e., the fret farthest from the root; segments adjacent to the root also contain the root). Also, for any row $\mathbf{a}$ of A', there exists some root-leaf path P such that $\mathrm{supp}(\mathbf{a}) \subseteq V(P)$.

This inspires the following construction: let $B \in \mathbb{Z}^{(2a+1)^{\tau^2} \times \tau^2}$ be the matrix whose columns are all the possible vectors $\mathbf{a} \in \mathbb{Z}^{\tau^2}$ with $\|\mathbf{a}\|_\infty \leq \|A\|_\infty$. Let $B_i \in \mathbb{Z}^{(2a+1)^{\tau^2} \times \tau}$, for $i \in [\tau]$, be the submatrix of B formed by rows $(i-1)\tau + 1, \ldots, i\tau$ and modify the last such submatrix B_τ by putting $B_\tau := (B_\tau \mid I)$ where $I \in \mathbb{Z}^{(2a+1)^{\tau^2} \times (2a+1)^{\tau^2}}$ is the identity matrix; the variables corresponding to columns of I will play the role of slack variables. Let T be the tree of height τ obtained from F' by contracting all paths between frets.

Now, let $C = T^P(B_1, \ldots, B_\tau)$. Obtain $\tilde{F}$ from F' by appending a leaf to every leaf, and observe that $G_P(T^P(B_1, \ldots, B_\tau)) \subseteq \mathrm{cl}(\tilde{F})$; the new leaves correspond to the slack variables in B_τ. Our goal now is to construct a right hand side vector $\mathbf{b}'$ and lower and upper bounds $\mathbf{l}', \mathbf{u}'$ to enforce exactly the constraints present in $A\mathbf{x} = \mathbf{b}$. For every root-leaf path P in F' there is a corresponding root-leaf path $\tilde{P}$ in $\tilde{F}$ such that $\tilde{P}$ is P with an additional leaf. Fix a root-leaf path P in F'. For every row $\mathbf{a}$ of A' with $\mathrm{supp}(\mathbf{a}) \subseteq V(P)$ and right hand side β, there exists a unique row $\mathbf{c}$ of C with $\mathrm{supp}(\mathbf{c}) \subseteq V(\tilde{P})$ such that $\mathbf{c} = (\mathbf{a}, 1)$, and we set the right hand side of row $\mathbf{c}$ to β.

For every row $\mathbf{c}$ of C which was not considered in the previous paragraph, set the right hand side to 0 and for the slack variable of this row set the lower bound to $-\infty$ and the

upper bound to ∞. Let us remark in passing that we are not limited by using the standard equality form of ILP: transforming an instance $A\mathbf{x} \leq \mathbf{b}$ into the standard equality form by adding slack variables only possibly increases the treedepth by 1. ◀

4.2 Dual Treedepth

▶ **Lemma 26** (Bounded dual treedepth ⇒ tree-fold)**. (*)** *Let $A, \mathbf{b}, \mathbf{l}$ and $\mathbf{u}$ be as in* (ILP)*, $a = \max\{2, \|A\|_\infty\}$ and $\tau + 1 = \mathrm{td}_D(A)$. Then there exists $D \in \mathbb{Z}^{m' \times n'}$, $\mathbf{b}' \in \mathbb{Z}^{m'}$ and $\mathbf{l}', \mathbf{u}' \in (\mathbb{Z} \cup \{\pm\infty\})^{n'}$, $n' \leq nt$, $t \leq n$, $m' \leq m \cdot \tau$, which define an extended formulation of $A\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$. Moreover, there exist matrices $A_1, \ldots, A_\tau \in \mathbb{Z}^{\tau \times t}$ and a tree T such that $D = T^D(A_1, \ldots, A_\tau)$ is a tree-fold constraint matrix, and all can be computed in time $f_{D\text{-}embed}(a, \mathrm{td}_D(A)) \cdot n^2$ for some computable function $f_{D\text{-}embed}$.*

References

1. Gautam Appa, Balázs Kotnyek, Konstantinos Papalamprou, and Leonidas Pitsoulis. Optimization with binet matrices. *Operations research letters*, 35(3):345–352, 2007.
2. Stephan Artmann, Robert Weismantel, and Rico Zenklusen. A strongly polynomial algorithm for bimodular integer linear programming. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 1206–1219. ACM, 2017.
3. Matthias Aschenbrenner and Raymond Hemmecke. Finiteness theorems in stochastic integer programming. *Foundations of Computational Mathematics*, 7(2):183–227, 2007.
4. Lin Chen and Dániel Marx. Covering a tree with rooted subtrees–parameterized and approximation algorithms. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 2801–2820. SIAM, 2018.
5. Jesús A. De Loera, Raymond Hemmecke, and Matthias Köppe. *Algebraic and Geometric Ideas in the Theory of Discrete Optimization*, volume 14 of *MOS-SIAM Series on Optimization*. SIAM, 2013.
6. Jesús A. De Loera, Raymond Hemmecke, and Jon Lee. On augmentation algorithms for linear and integer-linear programming: From Edmonds–Karp to Bland and beyond. *SIAM Journal on Optimization*, 25(4):2494–2511, 2015.
7. Pavel Dvořák, Eduard Eiben, Robert Ganian, Dušan Knop, and Sebastian Ordyniak. Solving integer linear programs with a small number of global variables and constraints. *arXiv preprint arXiv:1706.06084*, 2017.
8. András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.
9. Eugene C. Freuder. Complexity of K-tree structured constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 4–9, 1990.
10. Robert Ganian and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP. *Artificial Intelligence*, 2018.
11. Robert Ganian, Sebastian Ordyniak, and M. S. Ramanujan. Going beyond primal treewidth for (M)ILP. In Satinder P. Singh and Shaul Markovitch, editors, *AAAI*, pages 815–821. AAAI Press, 2017.
12. Raymond Hemmecke, Matthias Köppe, and Robert Weismantel. Graver basis and proximity techniques for block-structured separable convex integer minimization problems. *Mathematical Programming*, 145(1-2, Ser. A):1–18, 2014.
13. Raymond Hemmecke, Shmuel Onn, and Lyubov Romanchuk. N-fold integer programming in cubic time. *Mathematical Programming*, pages 1–17, 2013.
14. Dorit S. Hochbaum and J. George Shanthikumar. Convex separable optimization is not much harder than linear optimization. *Journal of the ACM*, 37(4):843–862, 1990.

15 Bart M. P. Jansen and Stefan Kratsch. A structural approach to kernels for ILPs: Treewidth and total unimodularity. In Nikhil Bansal and Irene Finocchi, editors, *ESA*, volume 9294 of *Lecture Notes in Computer Science*, pages 779–791. Springer, 2015.

16 Klaus Jansen, Kim-Manuel Klein, Marten Maack, and Malin Rau. Empowering the configuration-IP – new PTAS results for scheduling with setups times. *arXiv preprint arXiv:1801.06460*, 2018.

17 Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, 1987.

18 Dušan Knop, Martin Koutecký, and Matthias Mnich. Combinatorial n-fold integer programming and applications. In *ESA*, volume 87 of *LIPIcs*, pages 54:1–54:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. Extended version available at `https://arxiv.org/abs/1705.08657`.

19 Dušan Knop, Martin Koutecký, and Matthias Mnich. Voting and bribing in single-exponential time. In *STACS*, volume 66 of *LIPIcs*, pages 46:1–46:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

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

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

22 Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. A faster parameterized algorithm for treedepth. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP (1)*, volume 8572 of *Lecture Notes in Computer Science*, pages 931–942. Springer, 2014.

23 Alexander Shrijver. *Theory of linear and integer programming.* Interscience Series in Discrete Mathematics and Optimization. Wiley, 1986.

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

25 Sergey I. Veselov and Aleksandr J. Chirkov. Integer program with bimodular matrix. *Discrete Optimization*, 6(2):220–222, 2009.

https://pubsonline.informs.org/journal/moor

MATHEMATICS OF OPERATIONS RESEARCH
Articles in Advance, pp. 1–16
ISSN 0364-765X (print), ISSN 1526-5471 (online)

Sparse Integer Programming Is Fixed-Parameter Tractable

Friedrich Eisenbrand,[a] Christoph Hunkenschröder,[b] Kim-Manuel Klein,[c] Martin Koutecký,[d,*] Asaf Levin,[e] Shmuel Onn[e]

[a] EPFL SB MATH DISOPT, 1015 Lausanne, Switzerland; [b] Institut für Mathematik, Technische Universität Berlin, D-10623 Berlin, Germany; [c] Institut für Theoretische Informatik, Universität zu Lübeck, 23538 Lübeck, Germany; [d] Computer Science Institute, Faculty of Mathematics and Physics, Charles University, 118 00 Praha 1, Czech Republic; [e] Faculty of Data and Decision Sciences, Technion – Israel Institute of Technology, Technion City, Haifa 3200003, Israel

*Corresponding author

Contact: friedrich.eisenbrand@epfl.ch, https://orcid.org/0000-0001-7928-1076 (FE); chr.hunkenschroeder@gmail.com, https://orcid.org/0000-0001-5580-3677 (CH); kimmanuel.klein@uni-luebeck.de, https://orcid.org/0000-0002-0188-9492 (K-MK); koutecky@iuuk.mff.cuni.cz, https://orcid.org/0000-0002-7846-0053 (MK); levinas@technion.ac.il, https://orcid.org/0000-0001-7935-6218 (AL); onn@technion.ac.il, https://orcid.org/0000-0002-4526-5534 (SO)

Received: May 31, 2023
Revised: January 30, 2024
Accepted: June 28, 2024
Published Online in Articles in Advance: August 19, 2024

MSC2020 Subject Classifications: 90C10, 68Q27

https://doi.org/10.1287/moor.2023.0162

Abstract. We study the general integer programming problem where the number of variables n is a variable part of the input. We consider two natural parameters of the constraint matrix A: its *numeric measure* a and its *sparsity measure* d. We present an algorithm for solving integer programming in time $g(a,d)\text{poly}(n,L)$, where g is some computable function of the parameters a and d, and L is the binary encoding length of the input. In particular, integer programming is fixed-parameter tractable parameterized by a and d, and is solvable in polynomial time for every fixed a and d. Our results also extend to nonlinear separable convex objective functions.

Funding: F. Eisenbrand, C. Hunkenschröder, and K.-M. Klein were supported by the Swiss National Science Foundation (SNSF) within the project "Convexity, geometry of numbers, and the complexity of integer programming" [Grant 163071]. A. Levin and S. Onn are partially supported by the Israel Science Foundation [Grant 308/18]. A. Levin is also partially supported by the Israel Science Foundation [Grant 1467/22]. S. Onn is also partially supported by the Dresner Chair at the Technion. M. Koutecký is partially supported by Charles University project UNCE 24/SCI/008, and by the project 22-22997S of the Grantová Agentura České Republiky (GA ČR).

Keywords: integer programming • parameterized complexity • Graver basis • treedepth • *n*-fold • tree-fold • 2-stage stochastic • multistage stochastic

1. Introduction

Our focus is on the integer (linear) programming (I(L)P) 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 $\mathbf{l}, \mathbf{u} \in \mathbb{Z}^n$. Note that in this work, we assume that the bounds $\mathbf{l}, \mathbf{u}$ are finite.

(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 $\mathbf{w} \in \mathbb{Z}^n$. In spite of that, in this paper we identify broad natural and useful conditions under which (IP) can be solved in polynomial time, even when the number of variables n is a variable part of the input. Parameterized complexity (Cygan et al. [16], Downey and Fellows [18]) offers a framework for studying such efficient algorithms for hard problems. Specifically, a *fixed-parameter tractable algorithm* for problem P with input length n and parameter k is one that runs in time $g(k)\text{poly}(n)$, where g is some computable function and $\text{poly}(n)$ is a polynomial of degree independent of k (as opposed to a much less desirable running time of the form $n^{g(k)}$). A problem P admitting a fixed-parameter algorithm is said to be *fixed-parameter tractable (FPT) parameterized by k*.

Here, we consider two natural parameters of the constraint matrix A: its numeric measure a and its sparsity measure d defined as follows. The numeric measure depends only on the values of the entries of the matrix A and is essentially the largest absolute value of any coefficient: specifically, we define a to be $a := \max\{2, \|A\|_\infty\}$. On the other hand, the sparsity measure d depends only on the structure of nonzeroes of A, and we use the notion of primal and dual treedepth to capture it. These are defined as follows. Let $G_P(A)$ denote the *primal graph of A*, which has $\{1,\dots,n\}$ as its vertex set, and an edge between vertices i and j exists if A contains a row which is nonzero in

coordinates i and j. The *dual graph of* A is $G_D(A) := G_P(A^\top)$. (These notions are the exact analogues of the variable and constraint graphs, respectively, from areas like constraint satisfaction, artificial intelligence (AI), SAT (satisfiability) solving, etc. (Rossi et al. [52]).) The *treedepth* of a graph denoted $\mathrm{td}(G)$ is the smallest height of a rooted forest F such that each edge of G is between vertices which are in a descendant-ancestor relationship in F (Nešetřil and Ossona de Mendez [46] and Pilipszuk et al. [50]). For example, the treedepth of any star is two, of the path on n vertices is $\lceil \log_2(n+1) \rceil$, and of the n-vertex complete graph K_n is n. The *primal treedepth of* A is $\mathrm{td}_P(A) := \mathrm{td}(G_P(A))$, and analogously the *dual treedepth of* A is $\mathrm{td}_D(A) := \mathrm{td}(G_D(A))$. Then, the sparsity measure d is defined as $d := \min\{\mathrm{td}_P(A), \mathrm{td}_D(A)\}$.

Denote by $\langle A, f, \mathbf{b}, \mathbf{l}, \mathbf{u} \rangle$ the binary encoding length of an (IP) instance. Here, we define the encoding length of f to be the length of f_{gap}, which is the difference between the maximum and minimum values of f on the domain. We assume that $f(\mathbf{x}) \in \mathbb{Z}$ whenever $\mathbf{x} \in \mathbb{Z}^n$. The function f is given by a comparison oracle, and we assume one comparison has a unit cost.

Our main result is stated as follows.

Theorem 1. *There exists a computable function g such that problem (IP) can be solved in time*

$$g(a,d)\mathrm{poly}(n,L), \qquad \text{where } d := \min\{\mathrm{td}_P(A), \mathrm{td}_D(A)\} \text{ and } L := \langle A, f, \mathbf{b}, \mathbf{l}, \mathbf{u} \rangle.$$

In other words, (IP) is FPT parameterized by a and d. Note that for our algorithm to be fast, it suffices if at least one of $\mathrm{td}_P(A)$ and $\mathrm{td}_D(A)$ is small. We note that our results cannot be improved in multiple senses (see Eisenbrand et al. [23] for details); for example, we focus on *separable* convex objective functions because both nonseparable convex and separable concave functions lead to NP-hardness, already when A is empty or just a row of 1's, respectively (Eisenbrand et al. [23, proposition 107]). Observe that already for $a = 1$ or $d = 1$ (ILP) is NP-hard (Eisenbrand et al. [23]). Moreover, arguably the two most important tractable classes of (IP) are formed by instances whose constraint matrix either is totally unimodular or has a small number n of columns, yet our results are incomparable with either: the class of totally unimodular matrices might have large d, but has $a = 1$, and the matrices considered here have variable n.

Our results also hold for (IP), whose constraints are given in the inequality form $A\mathbf{x} \leq \mathbf{b}$: introducing slack variables leads to (IP) in standard form with a constraint matrix $A_I := (A\ I)$, with $\min\{\mathrm{td}_P(A_I), \mathrm{td}_D(A_I)\} \leq \min\{\mathrm{td}_P(A) + 1, \mathrm{td}_D(A)\}$.

1.1. Related Work

In the literature there are some FPT algorithms for special cases of our settings. However, they do not capture all cases in which a, d are relatively small. We distinguish work prior to 2018 when the conference papers (Eisenbrand et al. [22] and Koutecký et al. [43]) forming the basis of our work were published, and subsequent results. We wish to highlight that a major contribution of this paper, besides the main result itself, is its self-contained, unified, and streamlined presentation.

1.2. Work Prior to 2018

In the 1980s it was shown by Lenstra [45] and Kannan [38] that (ILP) can be solved in time $n^{\mathcal{O}(n)}L$, where L is the length of the binary encoding of the input. Other large tractable classes are n-fold (Hemmecke et al. [34]), tree-fold (Chen and Marx [8]), and two-stage and multistage stochastic (Aschenbrenner and Hemmecke [1]), as well as algorithms for ILPs with bounded treewidth (Ganian et al. [27]), treedepth (Ganian and Ordyniak [26]) and fracture number (Dvořák et al. [19]) of graphs related to the matrix A. The class of 4-block n-fold IPs has an algorithm with time complexity $n^{g(k)}$ (Hemmecke et al. [33]) where k is the maximum of the largest absolute value of a coefficient and the largest dimension, and is not known to be solvable in time $g(k)\mathrm{poly}(n)$. (IP) is FPT parameterized by the primal treewidth $\mathrm{tw}_P(A)$ together with the largest domain $\|\mathbf{u} - \mathbf{l}\|_\infty$ (Freuder [25] and Jansen and Kratsch [36]). The parameters $\mathrm{td}_D(A)$ and $\mathrm{tw}_D(A)$ were only recently considered by Ganian et al. [27]. They show that (IP) is FPT parameterized by $\mathrm{tw}_I(A)$, the *incidence treewidth*, together with the parameter Γ, which is an upper bound on any prefix sum of $A\mathbf{x}$ for any feasible solution $\mathbf{x}$.

1.3. Subsequent Work

Knop et al. [41] gave lower bounds for (ILP) with few rows and also (ILP) parameterized by $\mathrm{td}_D(A)$. On the side of hardness, Eiben et al. [20] have shown that (ILP) is NP-hard already when the more permissive *incidence treedepth* $\mathrm{td}_I(A)$ is five and $\|A\|_\infty = 2$. Chen et al. [9] study the complexity of 4-block n-fold IPs with blocks of constant size, but possibly containing large coefficients. Hunkenschröder et al. [35] show near-optimal lower bounds for the

parameters $a, \mathrm{td}_P(A), \mathrm{td}_D(A)$ studied here. They show that the complexity grows from single- to double-exponential as one increases the number of levels in a treefold IP, and similarly from double- to triple-exponential for multistage IPs.

Based on the parametric search framework and using new formulations and proximity theorems, strongly polynomial near-linear algorithms for (ILP) parameterized by a and d have been devised by Cslovjecsek et al. [13] and Cslovjecsek [14]. Chen et al. [11] show a faster algorithm for 4-block n-fold IPs. Chen et al. [10] show an FPT algorithm for a subclass of 4-block n-fold IPs which can be used to model certain scheduling problems. Knop et al. [42] give an algorithm for a high-multiplicity variant of the n-fold IP problem. Brand et al. [3] show that the mixed ILP problem where some variables are allowed to be continuous is still FPT in the same regime as the (purely) integer problem (ILP). However, the linearity of the objective and integrality of bounds are crucial, because removing either quickly leads to hardness, as shown in Brand et al. [4]. Chan et al. [7] and Briański et al. [5] consider generalizations of the parameter d: for a matrix A to be viewed as sparse, it suffices that its column matroid is sparse in some well-defined sense, because then one can formulate an equivalent auxiliary problem with a matrix A' which has small d and a and use, for example, the algorithms described here. Klein and Reuter [40] improve the function $g(k)$ in the case of primal treedepth $\mathrm{td}_P(A)$. Cslovjecsek et al. [15] study the complexity of n-fold and two-stage stochastic IPs which contain large coefficients, but only in the horizontal or vertical blocks, respectively, and show FPT algorithms. Notably, to accomplish their results, they develop a new approach which does not use either iterative augmentation or proximity techniques. Eisenbrand et al. [24] consider when a (linear or separable convex) objective function can be replaced by a simpler one, and show lower and upper bounds.

1.4. Organization

In Section 2, we set the notation, terminology, and definitions. In Section 3, we show how the general problem (IP) can be reduced to a polynomial number of instances of a simpler subproblem corresponding to finding augmenting steps. In Section 4, we characterize the block structure of matrices of small treedepth. Finally in Section 5, we first prove bounds on the norms of augmenting steps, and then show how the problem of finding them can be efficiently solved for the matrices of interest, concluding the proof of Theorem 1.

2. Preliminaries

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 $a \in \mathbb{Z}$, $\langle a \rangle := 1 + \lceil \log(|a| + 1) \rceil$ denotes 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 $\langle A \rangle = \sum_{i,j} \langle A_{i,j} \rangle$. Let $\mathrm{nnz}(A)$ be the number of nonzeroes of A.

For a function $f : \mathbb{Z}^n \to \mathbb{Z}$ and two vectors $\mathbf{l}, \mathbf{u} \in \mathbb{Z}^n$, we define $f_{\mathrm{gap}}^{[\mathbf{l},\mathbf{u}]} := \max_{\mathbf{x},\mathbf{x}' \in [\mathbf{l},\mathbf{u}]} |f(\mathbf{x}) - f(\mathbf{x}')|$; if $[\mathbf{l}, \mathbf{u}]$ is clear from the context, we omit it and write just f_{gap}. This quantity provides an upper bound on the *optimality gap* of our iterative algorithm, that is, on the difference between the value of the current iterate and the optimum.

We assume that $f : \mathbb{R}^n \to \mathbb{R}$ is a separable convex function; that is, 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 $\mathbf{x} \in \mathbb{Z}^n$, $f(\mathbf{x}) \in \mathbb{Z}$. Because we do not want to assume that the values of f are given as a list, we use the standard assumption of f being given as a comparison oracle: that is, an oracle which for any two points $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$ answers the query "$f(\mathbf{x}) < f(\mathbf{y})$?".

We use ω to denote the smallest number such that matrix multiplication of $n \times n$ matrices can be performed in time $\mathcal{O}(n^\omega)$. The current best bound is $\omega \le 2.371552$ (Williams et al. [56]); for more context, see the references therein. 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 $m \le n$ throughout the paper.

Proposition 1 (Purification (Grötschel et al. [32, Theorem 1.4.8])). *Given $A \in \mathbb{Z}^{m \times n}$ and $\mathbf{b} \in \mathbb{Z}^m$ in time $\mathcal{O}(\min\{n,m\}nm)$, one can either declare $A\mathbf{x} = \mathbf{b}$ infeasible or output a pure equivalent subsystem $A'\mathbf{x} = \mathbf{b}'$; that is, $A'\mathbf{x} = \mathbf{b}'$ is pure, its rows are a subset of $A\mathbf{x} = \mathbf{b}$, and $\forall \mathbf{x} : A\mathbf{x} = \mathbf{b} \Leftrightarrow A'\mathbf{x} = \mathbf{b}'$.*

For a graph G we denote by $V(G)$ its set of vertices.

2.1. 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 $i \in [n]$, $x_i y_i \ge 0$ and $|x_i| \le |y_i|$. 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 (Gordan's lemma [30]; for a modern proof see Bruns and Gubeladze [6]). Using these notions, we define the Graver basis as follows.

Definition 1 (Graver Basis [31]). 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 $\ker_{\mathbb{Z}}(A) \setminus \{\mathbf{0}\}$.

The Graver basis $\mathcal{G}(A)$ can be also equivalently defined as the union of the Hilbert bases of the orthant cones intersected with $\ker(A)$; see De Loera et al. [17, lemma 3.2.2]. One important property of $\mathcal{G}(A)$ is the following "positive sum property."

Proposition 2 (Positive Sum Property (Onn [48, Lemma 3.4])). *Let $A \in \mathbb{Z}^{m\times n}$. For any $\mathbf{x} \in \ker_{\mathbb{Z}}(A)$, there exists an $n' \leq 2n-2$ 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 $\mathbf{g}_j \sqsubseteq \mathbf{x}$.*

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 2 is that if *any* augmenting step exists, then there exists one in $\mathcal{G}(A)$ (De Loera et al. [17, lemma 3.3.2]).

Example 1 (Simple Iterative Procedure). Consider the task of minimizing a linear function $\mathbf{w}$ over the integers of a box $[\mathbf{l}, \mathbf{u}]$. In this case, the constraint matrix A is empty; thus, $\mathcal{G}(A)$ is the set of n unit vectors. Let $\mathbf{e}_j \in \mathbb{Z}^n$ be the j-th unit vector. Starting from the (feasible) solution $\mathbf{x}_0 := \mathbf{l}$, the smallest point in the box, a simple iterative procedure would be to, in step $i = 0, 1, \ldots$, choose any $\mathbf{e}_j$ with $\mathbf{w}\mathbf{e}_j < 0$ and with $(\mathbf{x}_i + \mathbf{e}_j) \in [\mathbf{l}, \mathbf{u}]$ (i.e., feasible), and set $\mathbf{x}_{i+1} := \mathbf{x}_i + \mathbf{e}_j$. When no such $\mathbf{e}_j$ exists, $\mathbf{x}_i$ is optimal.

A clear weakness of this procedure is that it can make as many as $\|\mathbf{u} - \mathbf{l}\|_1$ steps to converge. An easy fix is to take "long steps": whenever $\mathbf{w}\mathbf{e}_j < 0$, apply $\mathbf{e}_j$ exhaustively; that is, find the largest $\lambda \in \mathbb{N}$ such that $\mathbf{x}_i + \lambda\mathbf{e}_j$ is feasible, and set $\mathbf{x}_{i+1} := \mathbf{x}_i + \lambda\mathbf{e}_j$. (Notice that in this example we will always have $\lambda = u_j - l_j$, and $(\mathbf{x}_{i+1})_j = u_j$.) This will inform our next development in Section 3.

2.2. Standard Properties of the Graphs of *A*

Recall that $G_P(A)$ denotes the *primal graph of A*, which has $V(G_P(A)) = [n]$ and $E(G_P(A)) = \{ij \mid \exists r \in [m] \text{ s.t. } A_{r,i} \neq 0 \wedge A_{r,j} \neq 0\}$, and the *dual graph of A* is $G_D(A) := G_P(A^\top)$. 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 2 (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 $G \subseteq \mathrm{cl}(F)$; namely, every edge of G belongs to the edge set of $\mathrm{cl}(F)$ where the vertex set of F is the same as the one of G. 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 3 (Reidl et al. [51]). *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 $\mathrm{td}_D(A) := \mathrm{td}(G_D(A))$. For an example, see Figure 1.

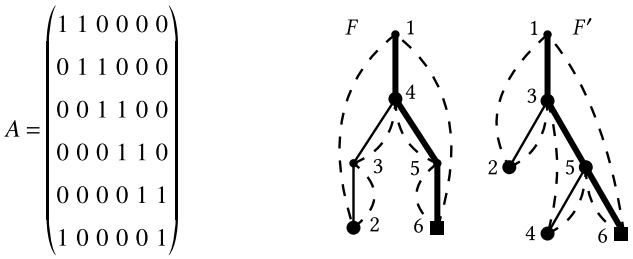
We often assume that an optimal td-decomposition is given because the time required to find it is dominated by other terms. Moreover, in many applications a small td-decomposition of $G_P(A)$ or $G_D(A)$ is clear from the way A was constructed.

By definition, 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 $|V(G)|$ leaves. Thus, when assuming that $\mathrm{td}_P(A)$ or $\mathrm{td}_D(A)$ is small, as we do here, 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 $G_D(A)$ is given.

3. Refined Augmentation Procedures

Our algorithm is based on the approach of augmentation procedures. These are iterative algorithms that start with a feasible solution, and at each step the algorithm tries to improve the current solution until it converges to an optimal solution.

Figure 1. The primal graph $G_P(A)$ of the matrix A on the left is the cycle of length six. On the right, F and F' are two different td-decompositions of $G_P(A)$. To see that they are indeed td-decompositions of $G_P(A)$, it suffices to verify that the edges of $G_P(A)$ (dashed) only occur between vertices which are in an ancestor-descendant relationship. The height of a decomposition is the length of the longest root-leaf path. In both F and F', this is three, as witnessed by the path from the root (1) to a furthest leaf (6, marked by a square).



Example 1 demonstrates this in a very simple setting; however, in the full generality of solving (IP), several challenges arise. First, the directions that are easily considered (e.g., axis parallel directions) may lead to infeasible solutions; thus, we need to consider directions in the Graver basis. However, the set $\mathcal{G}(A)$ may be too large to enumerate directly. Second, an optimal step-length may be difficult to determine, particularly when f is not linear but separable convex. Third, in our example it was trivial to obtain an initial feasible solution, but in general this may be much harder. In the following, we will gradually show how to reduce the general task of solving (IP) to polynomially many instances of a related but easier subproblem.

An augmenting step $\mathbf{g}$ and a *step-length* $\lambda \in \mathbb{N}$ form an $\mathbf{x}$-*feasible step pair* if $\mathbf{l} \le \mathbf{x} + \lambda\mathbf{g} \le \mathbf{u}$. An augmenting step $\mathbf{h}$ is a *Graver-best step* for $\mathbf{x}$ if $f(\mathbf{x}+\mathbf{h}) \le f(\mathbf{x}+\lambda\mathbf{g})$ 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 that we introduce here for the first time is a halfling: an augmenting step $\mathbf{h}$ is a *halfling* for $\mathbf{x}$ if $f(\mathbf{x}) - f(\mathbf{x}+\mathbf{h}) \ge \frac{1}{2}(f(\mathbf{x}) - f(\mathbf{x}+\lambda\mathbf{g}))$ 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 step 1.

Say that the algorithm has *made i steps* if x_{i-1} is returned as optimal in step 1.

Lemma 1 (Halfling Convergence). *Given a feasible solution $\mathbf{x}_0$ for (IP), the halfling augmentation procedure finds an optimum of (IP) in at most two steps if $f(\mathbf{x}_0) - f(\mathbf{x}^*) \le 1$, and in at most $3n\log(f(\mathbf{x}_0) - f(\mathbf{x}^*)) \le 3n\log(f_{\mathrm{gap}}^{[\mathbf{l},\mathbf{u}]})$ steps otherwise.*

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

Proposition 4 (Separable Convex Superadditivity (De Loera et al. [17, 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 $\mathbf{g}_1, \dots, \mathbf{g}_k \in \mathbb{R}^n$ be vectors with the same sign pattern from $\{\le 0, \ge 0\}^n$; that is, they belong to a common orthant of $\mathbb{R}^n$. Then*

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

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

Proof of Lemma 1. Let $\mathbf{x}^*$ be an optimal solution of (IP). If $\mathbf{x}_0$ is optimal, the halfling augmentation procedure terminates with $\mathbf{x}_0$, that is, in one step. If $f(\mathbf{x}_0) - f(\mathbf{x}^*) = 1$, then the procedure will terminate after two steps because any augmenting step (in particular, any halfling) improves the objective by one. Thus, assume that $f(\mathbf{x}_0) - f(\mathbf{x}^*) \ge 2$. By Proposition 2 we may write $\mathbf{x}^* - \mathbf{x}_0 = \sum_{j=1}^{n'} \lambda_j \mathbf{g}_j$ such that $\mathbf{g}_j \sqsubseteq \mathbf{x}^* - \mathbf{x}_0$, $\mathbf{g}_j \in \mathcal{G}(A)$, and $\lambda_j \in \mathbb{N}$ for all $j \in [n']$, and $n' \le 2n-2$. We apply Proposition 4 to $\mathbf{x}_0$ and the n' vectors $\lambda_j \mathbf{g}_j$ with $\alpha_j := 1$, so by (1) we have

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

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

$$f(\mathbf{x}_0) - f(\mathbf{x}_0 + \lambda_\ell \mathbf{g}_\ell) \ge \frac{1}{n'}(f(\mathbf{x}_0) - f(\mathbf{x}^*)). \tag{2}$$

Consider a halfling $\mathbf{h}$ for $\mathbf{x}_0$: by definition, it satisfies $f(\mathbf{x}_0) - f(\mathbf{x}_0 + \mathbf{h}) \geq \frac{1}{2}(f(\mathbf{x}_0) - f(\mathbf{x}_0 + \lambda_\ell \mathbf{g}_\ell)) \geq \frac{1}{2n'}(f(\mathbf{x}_0) - f(\mathbf{x}^*))$. Say that the halfling augmentation procedure required s iterations, and recall that $n' \leq 2n-2$. We apply the same reasoning as developed above for $\mathbf{x}^* - \mathbf{x}_0$ to each $\mathbf{x}^* - \mathbf{x}_i$, $i \in [s-1]$, as we have applied to $\mathbf{x}^* - \mathbf{x}_0$ (yielding different decompositions each time) to show that

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

and, by repeated application of this inequality, $f(\mathbf{x}_i) - f(\mathbf{x}^*) \leq \left(\frac{4n-5}{4n-4}\right)^i (f(\mathbf{x}_0) - f(\mathbf{x}^*))$. Because 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 an upper bound on i. We start with $1 \leq \left(\frac{t-1}{t}\right)^i (f(\mathbf{x}_0) - f(\mathbf{x}^*))$. Taking the natural logarithm gives $0 \leq i \ln\left(\frac{t-1}{t}\right) + \ln(f(\mathbf{x}_0) - f(\mathbf{x}^*))$ and moving terms around then gives $-i \ln\left(\frac{t-1}{t}\right) = i \ln\left(\frac{t}{t-1}\right) \leq \ln(f(\mathbf{x}_0) - f(\mathbf{x}^*))$. Dividing by $\ln\left(\frac{t-1}{t}\right)$, we obtain $i \leq \left(\ln\left(\frac{t}{t-1}\right)\right)^{-1} \ln(f(\mathbf{x}_0) - f(\mathbf{x}^*))$. Now Taylor expansion gives for $t \geq 3$ that $\ln\left(1 + \frac{1}{t-1}\right) \geq \frac{1}{t-1} - \frac{1}{2(t-1)^2}$. From this it follows for all $t \geq 3$ that $\left(\ln\left(1 + \frac{1}{t-1}\right)\right)^{-1} \leq t$. Plugging back $t := 4n-4$, we get that for all $n \geq 2$ we have $t \geq 3$ and hence

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

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

Clearly, we would like to find halflings quickly using the following lemma.

Lemma 2 (Powers of Two). *Let* $\Gamma_2 = \{1, 2, 4, 8, \ldots\}$ *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 $1/2 < \gamma \leq 1$ in such a way that $\lambda = \gamma\lambda^*$. Convexity of f yields

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

Thus, $\lambda\mathbf{g}^*$ is a halfling, and by the definition of $\mathbf{h}$, $f(\mathbf{x}+\mathbf{h}) \leq f(\mathbf{x}+\lambda\mathbf{g}^*)$ and so $\mathbf{h}$ is also a halfling. □

Therefore, 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 use the notion of a *best* solution:

Definition 3 (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 we want (S-best) to be an easier problem if S is somehow "simpler" than P. The following is a central notion in our development.

Definition 4 (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 2 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 3 (((AugIP) and $\mathbf{x}_0$) $\Rightarrow$ (IP)). *Given an initial feasible solution* $\mathbf{x}_0$ *to* (IP), (IP) *can be solved by solving*

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

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 nonzero $\mathbf{x}$-feasible step pair. Thus, by Lemma 2, 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 1, $3n \log(f(\mathbf{x}_0) - f(\mathbf{x}^*)) \leq 3n \log(f_{\text{gap}})$ halfling augmentations suffice and the claim follows. □

3.1. Feasibility

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

Lemma 4 ((AugIP) $\Rightarrow \mathbf{x}_0$). *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* $\mathcal{O}(n \log(\|A, \mathbf{b}, \mathbf{l}, \mathbf{u}, n\|_\infty)^2)$ *many* (AugIP) *instances, plus* $\mathcal{O}(n^\omega)$ *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 $\mathcal{O}(n^{\omega-1}m) \leq \mathcal{O}(n^\omega)$ (Storjohann and Labahn [55]) (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 $\hat{l}_i := \min\{l_i, z_i\}$ and $\hat{u}_i := \max\{u_i, z_i\}$ for all $i \in [n]$, 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 zero 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 1, an optimum $\mathbf{x}_0$ of $\min\{\hat{f}(\mathbf{x}) \mid A\mathbf{x} = \mathbf{b}, \hat{\mathbf{l}} \leq \mathbf{x} \leq \hat{\mathbf{u}}, \mathbf{x} \in \mathbb{Z}^n\}$ can be computed by solving $3n(\log\|\hat{\mathbf{u}} - \hat{\mathbf{l}}\| + 1)\log(\hat{f}_{\text{gap}}^{[\hat{\mathbf{l}}, \hat{\mathbf{u}}]})$ instances of (AugIP). Because $\|\hat{\mathbf{l}}, \hat{\mathbf{u}}\|_\infty$ is polynomially bounded in $\|A, \mathbf{b}, n\|_\infty$ and $\|\mathbf{l}, \mathbf{u}\|_\infty$ and, by definition of $\hat{f}$, $\hat{f}_{\text{gap}}^{[\hat{\mathbf{l}}, \hat{\mathbf{u}}]}$ 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 $\mathcal{O}(n \log(\|A, \mathbf{b}, \mathbf{l}, \mathbf{u}, n\|_\infty)^2)$. 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 4 and 1, we immediately obtain that a polynomial (AugIP) algorithm is sufficient for solving (IP) in polynomial time:

Corollary 1 ((AugIP) $\Rightarrow$ (IP)). *Problem* (IP) *can be solved by solving* $\mathcal{O}(nL^2)$ *instances of* (AugIP), *where* $L := \log(\|A, f_{\text{gap}}, \mathbf{b}, \mathbf{l}, \mathbf{u}, n\|_\infty)$, *plus time* $\mathcal{O}(n^\omega + \min\{n, m\}nm)$.

4. Block-Structured Matrices

To facilitate our proofs and to provide more refined complexity bounds, we introduce for the first time a parameter called *topological height*. This notion is useful in our analysis and proofs, and furthermore it plays a crucial role in complexity estimates of (IP) (Eisenbrand et al. [23]).

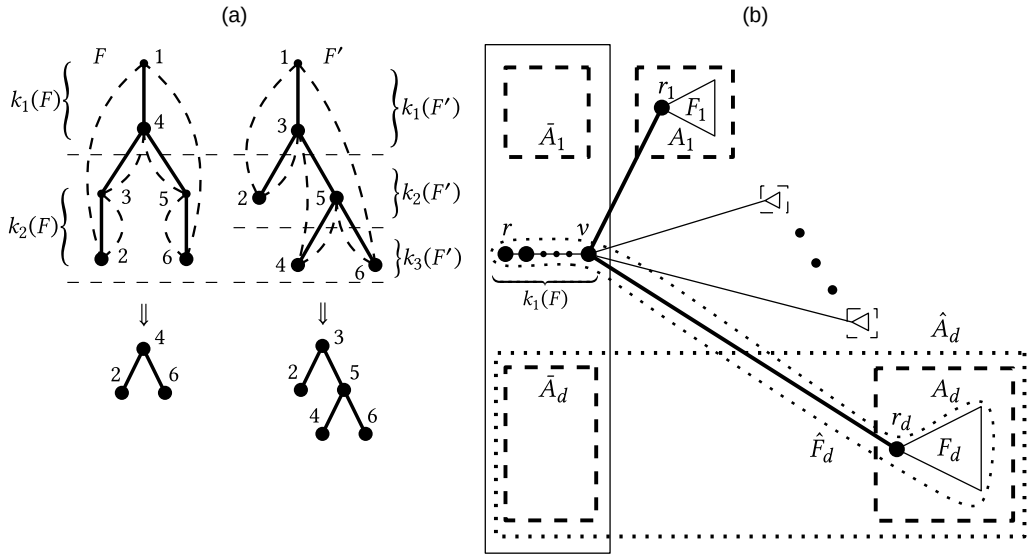
Definition 5 (Topological Height). A vertex of a rooted tree F is *degenerate* if it has exactly one child, and *nondegenerate* otherwise (i.e., if it is a leaf or has at least two children). The *topological height of* F, denoted $\text{th}(F)$, is the maximum number of nondegenerate vertices on any root-leaf path in F. Equivalently, $\text{th}(F)$ is the height of F after contracting each edge from a degenerate vertex to its unique child. Clearly, $\text{th}(F) \leq \text{height}(F)$. Note that when F is an optimal td-decomposition of $G_P(A)$ or $G_D(A)$, then $\text{height}(F) = \text{td}_P(A)$ or $\text{height}(F) = \text{td}_D(A)$, respectively. For a graph G, the value $\min_{F:\text{height}(F)=\text{td}(G)} \text{th}(F)$ can be intuitively thought of as a notion of complexity of G among other graphs of the same treedepth. For example, a graph has *vertex integrity* (Barefoot et al. [2], Gima and Otachi [28], Gima et al. [29]) k precisely when it admits a td-decomposition of depth k and topological height 2.

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

Definition 6 (Block-Structured Matrix). 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 either if $\text{th}(F) = 1$, or if $\text{th}(F) > 1$ and the following holds. Let v be the first nondegenerate vertex in F on a path from the root, $r_1, \dots, r_d$ be the children of v, F_i be the subtree of F rooted in r_i, and $n_i := |V(F_i)|$, for $i \in [d]$, and

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

Figure 2. Illustration of Definitions 2 and 5 (a) and Lemmas 5 and 6 (b). (a) Two optimal td-decompositions F and F' of the cycle on six vertices (in dashed edges). Nondegenerate 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 5: a td-decomposition F of $G_P(A)$ pictured in the matrix A, the decomposition into smaller blocks $\overline{A}_1, \ldots, \overline{A}_d, A_1, \ldots, A_d$ 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 6).



where, for $i \in [d]$, $\overline{A}_i \in \mathbb{Z}^{m_i \times k_1(F)}$ 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. To attain the complexity stated below, we assume A is represented as a column-indexed array of lists of values, that is, $A[i]$ is a list of (row, value) pairs encoding the nonzeroes of column i. We also assume there are no zero columns in A. Because the vertices of F are the column indices of A, we interchangeably speak about vertices of F and columns of A. For example, for $v \in V(F)$, the nonzero rows of v are those rows of A which are nonzero in the column whose index is v, or for a $S \subseteq V(F)$, we speak of the columns S, and mean the columns of A whose indices are S.

Lemma 5 (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 $\mathcal{O}(\mathrm{nnz}(A))$ returns a permutation $\pi_{\mathrm{rows}} : [m] \to [m]$ of the rows and a permutation $\pi_{\mathrm{cols}} : [n] \to [n]$ of the columns of A such that the resulting matrix A' is block-structured along F.*

Proof. The algorithm is recursive, proceeds in two passes, and uses three global arrays and two global integers to construct the permutations. Specifically, in the beginning initialize π_{cols} to be an empty array of length n, and $\pi_{\mathrm{rows}}, \pi_{\mathrm{rows}}^{-1}$ to be two empty arrays of length m. We will maintain that for every $i, j \in [m]$, we have $\pi_{\mathrm{rows}}[i] = j \iff \pi_{\mathrm{rows}}^{-1}[j] = i$. Also initialize r and c to be global integers equaling zero. The output is a labeling of all nondegenerate vertices of F with information encoding the blocks as follows. For a nondegenerate vertex v of F, say that its *boss*, denoted $\mathrm{boss}(v)$, is either the root of F if no ancestor of v is nondegenerate, or the unique vertex u which is a child of the nondegenerate vertex closest to but distinct from v on the path from v to the root of F. The inverse mapping boss^{-1} is defined for each vertex u which is either a degenerate root or a child of a nondegenerate vertex, and assigns to it a vertex $\mathrm{boss}^{-1}(u)$ which is the first nondegenerate vertex on a path from u to any leaf which is a descendant of u; clearly then $\mathrm{boss}(\mathrm{boss}^{-1}(u)) = u$. Let F_v be the subtree of F rooted in a vertex v. Then the algorithm assigns to each nondegenerate vertex v a label $(r^v_{\mathrm{start}}, r^v_{\mathrm{end}})$ such that columns $V(F_{\mathrm{boss}(v)})$ and rows $[r^v_{\mathrm{start}}, r^v_{\mathrm{end}}]$ form a block in the matrix A permuted according to $\pi_{\mathrm{rows}}, \pi_{\mathrm{cols}}$.

First Pass. Let us define a function BlockStructure which takes as an argument a vertex $v \in V(F)$. This function will perform the first pass. The second pass is much simpler and only serves to fix up possible inconsistencies which may occur because of rows which are nonzero in the $\overline{A}_i$ block but zero in the A_i blocks.

BlockStructure(v) starts by walking from v (which, at the top of the recursion, is the root of F) to the first nondegenerate vertex u. Let P be the path from v to u. As the algorithm visits the vertices of P, it incrementally numbers the columns, using the global index c, in this way constructing the permutation π_{cols}. It also constructs a set P_{rows} of rows which are nonzero in the columns of the path P, as follows. In v, it initializes P_{rows} to be an empty set (implemented as a hash table), and in each visited vertex w, it adds each row which is nonzero in column w to P_{rows}. The complexity of this step depends on the number of nonzeroes of each particular column, but because each nonzero is encountered exactly once, in total this step takes time $\mathcal{O}(\text{nnz}(A))$.

After reaching u, we distinguish two cases. If u is a leaf, then the columns of P correspond to a block, and P_{rows} is the set of nonzero rows of this block. Thus, we incrementally number them using the global index r, and we label the leaf u with the starting and ending indices of its rows $(r_{\text{start}}, r_{\text{end}})$.

On the other hand, if u is a branching vertex, we remember the current value of r as r_{start}, and we make a recursive call BlockStructure(w) on each child w of u. After finishing these calls, we note the value of $r-1$ (the last used index) as r_{end}, obtaining an interval $[r_{\text{start}}, r_{\text{end}}]$ of rows which are nonzero in the blocks corresponding to the children of u. However, we cannot yet label u with $(r_{\text{start}}, r_{\text{end}})$ because the columns of P may contain some additional nonzero rows. To account for these, we go over the rows in P_{rows} and for each check, using the array π^{-1}_{rows}, whether it has already received an index, and if not, we use the global index r to index it. Because this step, over the run of the algorithm, handles each nonzero of A exactly once, it takes in total $\mathcal{O}(\text{nnz}(A))$ time. Now we are ready to label v with $(r_{\text{start}}, r-1)$.

Second Pass: Fixing Inconsistencies. Let x be the root of F. We call BlockStructure(x) to perform a first pass over F and assign a certain row range to every nondegenerate vertex. The reason we are not done yet is the following. Consider a nondegenerate vertex v with d children $u_1, \dots, u_d$, let $v_i = \text{boss}^{-1}(u_i)$ for each $i = 1, \dots, d$, and let $(r^u_{\text{start}}, r^u_{\text{end}})$ for $u \in \{v, v_1, \dots, v_d\}$ be the label assigned to u. The problem occurs when the columns on the path from boss(v) to v contain some nonzero rows which are zero in the blocks of all the children. We will (correctly) have $r^v_{\text{start}} = r^{v_1}_{\text{start}}$ and the intervals $[r^{v_i}_{\text{start}}, r^{v_i}_{\text{end}}]$ are consecutive (i.e., $r^{v_i}_{\text{end}} + 1 = r^{v_{i+1}}_{\text{start}}$ for $i = 1, \dots d-1$), but (incorrectly) $r^{v_d}_{\text{end}} < r^v_{\text{end}}$. This is incorrect because in Definition 6 the number of rows of $\overline{A}_d$ and A_d needs to be the same. However, we also cannot simply fix this problem locally by setting $r^{v_d}_{\text{end}}$ to r^v_{end}, because v_d potentially has its own children who would in turn become misaligned. What we need is another pass.

The second pass is performed by a function BlockStructure2, which takes one optional integer argument r_{end}. In the main call of this function, this argument is set to the number of rows m. The second pass traverses F in the same order as the first pass; when it encounters a nondegenerate vertex v, it replaces its original label $(r^v_{\text{start}}, r^v_{\text{end}})$ with $(r^v_{\text{start}}, r_{\text{end}})$ if the argument r_{end} was defined, and keeps the original label otherwise. Assume d is the number of children of v. The algorithm then calls itself with no argument on the first $d-1$ of its children, and calls itself with argument r^v_{end} on the last child. Clearly this accomplishes that for every nondegenerate vertex v with children $u_1, \dots, u_d$, $r^v_{\text{end}} = r^{\text{boss}^{-1}(u_d)}_{\text{end}}$ as desired.

Complexity. The algorithm visits each node of F $\mathcal{O}(1)$ times, and for any column i spends $\mathcal{O}(\#\text{nonzero rows of column } i)$ time processing it. Because $\text{nnz}(A) \geq n$ because there are no all-zero columns, the algorithm runs in time $\mathcal{O}(\text{nnz}(A))$.

Correctness. As in the beginning of the proof, let u be the first nondegenerate vertex on any root-leaf path, and $u_1, \dots, u_d$ its children, and let A' be the matrix A with rows and columns permuted according to π_{rows} and π_{cols}, respectively. What is left is to argue is that A' has the form (block-structure), in particular, that there is no overlap between the blocks $A_1, \dots, A_d$. Note that A_i has columns $V(F_{u_i})$ and rows $[r^{\text{boss}(u_i)}_{\text{start}}, r^{\text{boss}(u_i)}_{\text{end}}]$. The lack of overlap between A_i, A_j for $i \neq j$ follows simply from the fact that by the definition of treedepth there are no edges between any two $w_i \in F_{u_i}$, $w_j \in F_{u_j}$ for $i \neq j$, and thus, by definition of $G_P(A)$, there is no row containing a nonzero at both indices w_i and w_j; see Figure 2(b). □

Note that given an (IP), the primal decomposition naturally partitions the right-hand side $\mathbf{b} = (\mathbf{b}^1, \dots, \mathbf{b}^d)$ according to the rows of $A_1, \dots, A_d$, 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 $\overline{A}_1, A_1, \dots, A_d$ with indices $0, 1, \dots, d$ (written as superscripts). For example, we write $\mathbf{x} = (\mathbf{x}^0, \mathbf{x}^1, \dots, \mathbf{x}^d)$.

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

Definition 7 (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 either if $\text{th}(F) = 1$, or if $\text{th}(F) > 1$ and the following holds. Let v be the first nondegenerate vertex in F on a path from the root, $r_1, \dots, r_d$ 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

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

where $d \in \mathbb{N}$, and for all $i \in [d]$, $\overline{A}_i \in \mathbb{Z}^{k_1(F) \times n_i}$, 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 $\text{th}(F_i) \leq \text{th}(F) - 1$, $\text{height}(F_i) \leq \text{height}(F) - k_1(F)$, for $i \in [d]$.

Corollary 2 (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 $\mathcal{O}(\text{nnz}(A))$ returns a permutation $\pi_{\text{rows}} : [m] \to [m]$ of the rows and a permutation $\pi_{\text{cols}} : [n] \to [n]$ of the columns of A such that the resulting matrix A' is 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 $\overline{A}_1, A_1, \ldots, A_d$, and each object of length n into d objects according to the columns of $A_1, \ldots, A_d$.

Lemma 6. *Let $A \in \mathbb{Z}^{n \times m}$, a* td*-decomposition F of $G_P(A)$ (or $G_D(A)$), and $\overline{A}_i, A_i, F_i$, for all $i \in [d]$, be as in Definition 6 (or 7). Let $\hat{A}_i := (\overline{A}_i\ A_i)$ $\left(\text{or } \hat{A}_i := \begin{pmatrix} \overline{A}_i \\ A_i \end{pmatrix}, \text{respectively}\right)$ 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 $\hat{A}_i$, $\text{th}(\hat{F}_i) < \text{th}(F)$, and $\text{height}(\hat{F}_i) \leq \text{height}(F)$.*

Proof. Consider Figure 2(b). 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 $\text{th}(\hat{F}_i) < \text{th}(F)$ because v was nondegenerate in F but is degenerate in $\hat{F}_i$. The dual case follows by transposition. □

5. An FPT Algorithm for (IP)

Our goal now is to show that (AugIP) can be solved quickly when the largest absolute value of a coefficient in A, $\|A\|_\infty$, and the primal or dual treedepth $\text{td}_P(A)$ or $\text{td}_D(A)$, respectively, are small. Together with Corollary 1, this implies an efficient algorithm. To that end, we need two key ingredients. The first are algorithms solving (AugIP) quickly when $\|A\|_\infty$ and $\text{td}_P(A)$ or $\text{td}_D(A)$ are small and when restricted to solutions of small ℓ_∞- or ℓ_1-norm, respectively. The second are theorems showing that this is in fact sufficient because the elements of $\mathcal{G}(A)$ have bounded ℓ_∞- and ℓ_1-norms, respectively. We start with the second ingredient.

5.1. Norm Bounds

For a matrix $A \in \mathbb{Z}^{m \times n}$, let

$$g_1(A) := \max_{\mathbf{g} \in \mathcal{G}(A)} \|\mathbf{g}\|_1 \qquad \text{and} \qquad g_\infty(A) := \max_{\mathbf{g} \in \mathcal{G}(A)} \|\mathbf{g}\|_\infty.$$

Note that previously we have used "g_i" to refer to the i-th coordinate of a vector $\mathbf{g}$; however, no confusion should arise with $g_1(A)$ because this quantity always has a matrix as an argument. We begin by using the Steinitz Lemma to obtain a basic bound on $g_1(A)$.

The first application of the Steinitz Lemma to show structural results about integer programs comes from Eisenbrand and Weismantel [21], who give several applications, such as improving the dynamic program of Papadimitriou [49] or tightening the proximity bound of Cook et al. [12]. Many other applications and extensions have been devised since then (Chen et al. [10], Jansen and Rohwedder [37], Klein [39], Klein and Reuter [40], Lee et al. [44], Oertel et al. [47]). It is worth noting that there are two different formal and corresponding intuitive phrasings of the Steinitz Lemma. The one below says that each vector walk which begins and ends in $\mathbf{0}$ can be rearranged to stay in a small box. The alternative one (see, e.g., Eisenbrand and Weismantel [21, equation (5)]) states that any vector walk from $\mathbf{0}$ to $\mathbf{b}$ can be rearranged to stay within a narrow "tube" around the $\mathbf{0} - \mathbf{b}$ line.

Proposition 5 (Sevastjanov and Banaszczyk [53], Steinitz [54]). *Let $\|\cdot\|$ be any norm, and let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ be such that $\|\mathbf{x}_i\| \leq 1$ for $i \in [n]$ and $\sum_{i=1}^n \mathbf{x}_i = \mathbf{0}$. Then there exists a permutation π such that for each $k \in [n]$, $\|\sum_{i=1}^k \mathbf{x}_{\pi(i)}\| \leq d$.*

Lemma 7 (Base Bound). *Let $A \in \mathbb{Z}^{m \times n}$. Then $g_1(A) \leq (2m\|A\|_\infty + 1)^m$.*

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, and 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 $\mathbf{p}_k := \sum_{j=1}^{k} \mathbf{u}^j$ is upper bounded by $m\|A\|_\infty$ in the l_∞-norm. Clearly

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

Next, assume for contradiction that $\|\mathbf{g}\|_1 > (2m\|A\|_\infty + 1)^m$. Then, by the pigeonhole principle, 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 $g'_i := 0$ for each $i \in [n]$, and for every $\mathbf{u}^\ell$ in the sequence, set

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

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 nonzero 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. □

5.1.1. Norm of Primal Treedepth

Theorem 2 (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*

$$g_\infty(A) \leq 2^{2^{\cdot^{\cdot^{\cdot^{2^{(2\|A\|_\infty)^{2^{\mathrm{th}(F)} \cdot \alpha \cdot \mathrm{td}_P(A)^2}}}}}}} \quad \text{(th}(F)\text{ twos)}$$

We will use the following result due to Klein [39].

Proposition 6 (Klein [39]). *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 nonempty 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

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

Before proceeding with the proof of the theorem, let us give a high-level proof idea of Klein's result above. The key fact is that a vector path in d-dimensional space can be seen as a linear combination of "simpler" paths. We say a path is "simple" if it can be generated by an integer cone having a basis with small entries. Using this linear combination, without loss of generality, we can restrict ourselves to a simpler path. Having those simpler vector paths at hand, an intersecting point can be determined by examining the intersection of the corresponding integer cones. The proposition is proven by considering the generating set of such integer cones.

Proof of Theorem 2. 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 $A\mathbf{x} = \mathbf{b}$ to be pure. By Lemma 7 we then have that

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

In the inductive step, we assume without loss of generality that A is block-structured along F. Let $\hat{A}_i = (\overline{A}_i \; A_i) \in \mathbb{Z}^{m_i \times k' + n_i}$ and $\hat{F}_i$ be this block structure, and let $\hat{g}_\infty := \max_{i \in [d]} g_\infty(\hat{A}_i)$. Note that $\mathrm{td}_P(\hat{A}_i) \leq \mathrm{td}_P(A)$. Because $\hat{F}_i$ is a

td-decomposition of $G_P(\hat{A}_i)$ and $\text{th}(\hat{F}_i) < \text{th}(F)$, we may apply induction on $\hat{A}_i$, showing

$$\hat{g}_\infty \le 2^{2^{\iddots^{2^{(2\|A\|_\infty)^{2^{\text{th}(F)}\cdot\alpha\cdot \text{td}_P(A)^2}}}}}\Bigg\}\text{th}(F)-2 \quad . \tag{3}$$

Consider $\mathbf{g} = (\mathbf{g}^0, \mathbf{g}^1, \dots, \mathbf{g}^d) \in \mathcal{G}(A)$. For each $i \in [d]$, decompose $(\mathbf{g}^0, \mathbf{g}^i) = \sum_{j=1}^{N_i} (\mathbf{h}_j^0, \mathbf{h}_j^i)$ with $(\mathbf{h}_j^0, \mathbf{h}_j^i) \in \mathcal{G}(\hat{A}_i)$ by the positive sum property (Proposition 2). Let $T_i := \{\mathbf{h}_j^0 \mid j \in [N_i]\}$ 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 6 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 6 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 (3) for $\hat{g}_\infty$ then gives

$$\zeta = 2\alpha k_1^2 + \log \hat{g}_\infty + \log\log \hat{g}_\infty \le 2\alpha k_1^2 + 2\log \hat{g}_\infty \le$$

$$2\alpha k_1^2 + 2 \cdot 2^{\iddots^{2^{(2\|A\|_\infty)^{2^{\text{th}(F)-3}\cdot\alpha\cdot \text{td}_P(A)^2}}}}}\Bigg\}\text{th}(F)-3 \le 2^{\iddots^{2^{(2\|A\|_\infty)^{2^{\text{th}(F)}\cdot\alpha\cdot \text{td}_P(A)^2}}}}}\Bigg\}\text{th}(F)-3 \quad , \quad \text{and thus,}$$

$$2^{2^\zeta} \le 2^{2^{\iddots^{2^{(2\|A\|_\infty)^{2^{\text{th}(F)}\cdot\alpha\cdot \text{td}_P(A)^2}}}}}\Bigg\}\text{th}(F)-3} \le g_\infty(A) \le 2^{\iddots^{2^{(2\|A\|_\infty)^{2^{\text{th}(F)}\cdot\alpha\cdot \text{td}_P(A)^2}}}}}\Bigg\}\text{th}(F)-1} \quad . \quad \square$$

5.2. Norm of Dual Treedepth

The dual case is quite different than the primal case above, and there is no prior work on which we could base our approach. Essentially, the proof is a recursive extension of the proof of the Base bound (Lemma 7).

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

Proof. The proof will proceed by induction over $\text{th}(F)$. In the base case we have $\text{th}(F) = 1$ 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 7 to get that $g_1(A) \le (2\|A\|_\infty \text{height}(F) + 1)^{\text{height}(F)}$, which is at most $(3\|A\|_\infty K)^{K-1}$, where $K = \text{height}(F) + 1 = k_1(F) + 1$. (Note that $k_1(F) = k_1(P)$ for all root-leaf paths P in F because all paths share an identical segment from the root to the first nondegenerate vertex.)

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

Applying the Steinitz Lemma (Proposition 5) 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 at most $k_1(F)\|A\|_\infty \hat{g}_1$. As in the proof of Lemma 7, 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 $\|\mathbf{g}\|_1 \le N\hat{g}_1 \le \hat{g}_1 (2k_1(F)\|A\|_\infty \hat{g}_1 + 1)^{k_1(F)}$. Plugging in $\hat{g}_1 = (3\|A\|_\infty K')^{K'-1} \le (3\|A\|_\infty K)^{K'-1}$ and simplifying yields

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

Assume to the contrary that some two prefix sums $\mathbf{p}_\alpha$ and $\mathbf{p}_\beta$, for $\alpha < \beta$, are identical. Then the sequence $\mathbf{u}^{\alpha+1}, \dots, \mathbf{u}^\beta$ 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 $\bar{\mathbf{g}}^i := \bar{\mathbf{g}}^i + \mathbf{g}_j^i$. $\square$

Remark 1. Our definition of K allows us to recover the currently best-known upper bounds on $g_1(A)$ from Theorem 3. Specifically, Knop et al. [41, lemma 10] show that $g_1(A) \le (2\|A\|_\infty + 1)^{2^{\text{td}_D(A)}-1}$. This pertains to the worst case when $\text{th}(F) = \text{height}(F) = \text{td}_D(A)$. Then, we have $K = \prod_{i=1}^{\text{th}(F)} (k_i(P)+1) = 2^{\text{td}_D(A)}$ 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 treefold matrices discussed in Eisenbrand et al. [23].

5.3. Solving (AugIP) Quickly When $\|A\|_\infty$ and $\mathrm{td}_P(A)$ or $\mathrm{td}_D(A)$ Are Small and When Restricted to Solutions of Small Norms

We would like to define more specifically our next goal. Denote by $B_\infty(\rho)$ and $B_1(\rho)$ the ℓ_∞- and ℓ_1-norm balls, respectively, of appropriate dimension and of radius ρ, centered at the origin, that is, $B_p(\rho) = \{\mathbf{x} \mid \|\mathbf{x}\|_p \le \rho\}$ for $p \in \{1, +\infty\}$. Observe that if $S \subseteq S'$, then an S'-best solution is certainly also an S-best solution. Because $\mathcal{G}(A) \subseteq B_\infty(g_\infty(A))$ and $\mathcal{G}(A) \subseteq B_1(g_1(A))$, it follows that a $B_\infty(g_\infty(A))$-best solution or a $B_1(g_1(A))$-best solution of (AugIP) is also a $\mathcal{G}(A)$-best solution. This implies that solving (AugIP) roughly amounts to solving an instance of (IP) with an additional norm bound. We first consider the primal treedepth case and later the dual treedepth case.

Lemma 8 (Primal Lemma). *Problem* (AugIP) *can be solved in time* $\mathrm{td}_P(A)^2(2\rho+1)^{\mathrm{td}_P(A)}n$ *for any* $\rho \ge g_\infty(A)$.

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

Claim 1. *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*

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

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 $\rho := g_\infty(A)$, $f(\mathbf{g}) := f(\mathbf{x} + \lambda\mathbf{g})$, $\mathbf{b} := \mathbf{0}$, $\mathbf{l} := \lceil \frac{\mathbf{l}-\mathbf{x}}{\lambda} \rceil$, and $\mathbf{u} := \lfloor \frac{\mathbf{u}-\mathbf{x}}{\lambda} \rfloor$. 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 $\mathcal{G}(A) \subseteq B_\infty(g_\infty(A))$. Thus, it remains to prove the claim.

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. Because 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 5); hence, we have matrices $\overline{A}_1, \dots, \overline{A}_d, A_1, \dots, A_d$ for some d and td-decompositions $F_1, \dots, F_d$ for $G_P(A_1), \dots, G_P(A_d)$, respectively, with, for each $i \in [d]$, $\overline{A}_i$ having $k_1(F)$ columns, F_i having $\mathrm{th}(F_i) < \mathrm{th}(F)$, and $\mathrm{td}_P(A_i) \le \mathrm{td}_P(A) - k_1(F)$. 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

$$B_\infty(\rho)\text{-best}\{f(\mathbf{g}^i) \mid A_i\mathbf{g}^i = -\overline{A}_i\mathbf{g}^0 + \mathbf{b}^i,\ \mathbf{l}^i \le \mathbf{g}^i \le \mathbf{u}^i,\ \mathbf{g}^i \in \mathbb{Z}^{n_i}\}. \tag{4}$$

Finally return the vector $(\mathbf{g}^0, \dots, \mathbf{g}^d)$ which minimizes $\sum_{i=0}^d f(\mathbf{g}^i)$. If $\mathbf{g}^i$ is undefined for some $i \in [d]$ because the subproblem (4) 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 $(\mathbf{g}^1, \dots, \mathbf{g}^d)$ 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 $-\overline{A}_i\mathbf{g}^0$, which is possible in time $k_1(F) \cdot \mathrm{td}_P(A)$ because the number of rows of $\overline{A}_i$ is at most $\mathrm{td}_P(A)$. The total time needed is thus $(2\rho+1)^{k_1(F)} \cdot (\mathrm{td}_P(A) \cdot k_1(F) + k^2(2\rho+1)^k)n \le \mathrm{td}_P(A)^2(2\rho+1)^{\mathrm{td}_P(A)}n$. □

Lemma 9 (Dual Lemma). *Problem* (AugIP) *can be solved in time* $(2\|A\|_\infty\rho+1)^{\mathcal{O}(\mathrm{td}_D(A))}n$ *for any* $\rho \ge g_1(A)$.

Proof. We solve an auxiliary problem analogous to the one in Lemma 8: given $\rho \in \mathbb{N}$ and a separable convex function f, solve

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

The lemma then follows by the same substitution described at the beginning of the proof of Lemma 8. We assume that $\|\mathbf{b}\|_\infty \le \rho\|A\|_\infty$ because 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 $\mathrm{th}(F) \ge 2$, we assume A is dual block-structured along F (otherwise apply Corollary 2) and we have, for every $i \in [d]$, matrices $A_i, \overline{A}_i, \hat{A}_i$ and a tree $\hat{F}_i$ (see Lemma 6) 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 $\mathbf{b}^1, \dots, \mathbf{b}^n$ 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 $\|\mathbf{g}\|_1 \le \rho$ and each $i \in [d]$, both $\overline{A}_i\mathbf{g}^i$ and $\sum_{j=1}^{i} \overline{A}_j\mathbf{g}^j$ belong to $R := [-\rho\|A\|_\infty, \rho\|A\|_\infty]^{k_1(F)}$. For every $i \in [d]$ and every $\mathbf{r} \in R$, solve

$$B_1(\rho)\text{-best}\{f^i(\mathbf{g}^i) \mid \hat{A}_i\mathbf{g}^i = (\mathbf{r}, \mathbf{b}^i),\ \mathbf{l}^i \le \mathbf{g}^i \le \mathbf{u}^i,\ \mathbf{g}^i \in \mathbb{Z}^{n_i}\}. \tag{5}$$

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 $\text{th}(F) > 1$, then we use recursion to solve (5). The recursive call is well defined, because, for all $i \in [d]$, $\text{th}(\hat{F}_i) < \text{th}(F)$ 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 (5); 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 $\mathbf{r}_1, \dots, \mathbf{r}_d \in R$ 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. For us it suffices to say that this problem can be easily solved using dynamic programming (DP) 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 $\mathbf{g} = (\mathbf{g}^1, \dots, \mathbf{g}^d)$ 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 $(\mathbf{r}, \mathbf{b}^1, \dots, \mathbf{b}^d)$ where $\mathbf{r} \in R$ and $\mathbf{b}^1, \dots, \mathbf{b}^d$ 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 $\text{th}(F)$ levels, which we number $1, \dots, \text{th}(F)$, with level 1 being the base of the recursion. Let us compute the time required at each level. In the base case $\text{th}(F) = 1$, recall that the matrix $\hat{A}_i$ in subproblem (5) is a single column with height(F) rows, and solving (5) amounts to trying at most $2\rho + 1$ feasible valuations of $\mathbf{g}^i$ (which is a scalar variable) satisfying $\mathbf{l}^i \le \mathbf{g}^i \le \mathbf{u}^i$ and returning the best feasible one. Because there are n columns in total, computing the solutions of (5) 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 $\alpha_1, \dots, \alpha_{N_1}$ stages, where $\sum_{i=1}^{N_1} \alpha_i = n$. This takes time $\sum_{i=1}^{N_1} |R|^2 \cdot \alpha_i \le (2\|A\|_\infty\rho + 1)^{\text{td}_D(A)} \cdot n$, because a td-decomposition of each column is a path on $\text{td}_D(A)$ vertices. In total, computing the first level of recursion takes time $(2\|A\|_\infty\rho + 1)^{\text{td}_D(A)} n$.

Consider a recursion level $\ell \in [2, \text{th}(F)]$ and subproblem (5). The crucial observation is that when the algorithm asks for the answer to (5) 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 $|R'| \le (2\|A\|_\infty\rho + 1)^{k_1(F')}$, with $k_1(F') \le \text{td}_D(A)$. 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 $|R|^2 \cdot N_\ell \le (2\|A\|_\infty\rho + 1)^{\text{td}_D(A)} N_\ell$. Adding up across all levels, we get that the total complexity is at most $(n + \sum_{\ell=2}^{\text{th}(F)} N_\ell) \cdot (2\|A\|_\infty\rho + 1)^{\text{td}_D(A)}$ where $\sum_{\ell=2}^{\text{th}(F)} N_\ell < n$ because F has n leaves and each level corresponds to a vertex with degree at least two. The lemma follows. □

5.4. The Algorithm

We are now ready to conclude our main result.

Theorem 1 (Repeated). *There is an algorithm solving* (IP) *in time* $g(a, d)\text{poly}(n, L)$, *for some computable function* g; *that is,* (IP) *is fixed-parameter tractable parameterized by* a *and* d.

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

First, run the algorithm of Proposition 3 on $G(A)$ to obtain its optimal td-decomposition. By Lemma 7 there is a computable function g' such that the maximum ℓ_p-norm of elements of $\mathcal{G}(A)$ is bounded by $g'(\|A\|_\infty, \mathrm{td}(A))$. By Lemmas 8 and 9, there is a computable function g'' such that (AugIP) is solvable in time $g''(g'(\mathrm{td}(A), \|A\|_p), \|A\|_p, \mathrm{td}(A))n$ and thus in time $g(\|A\|_p, \mathrm{td}(A))n$ for some computable function g. Then, solve (IP) using the algorithm of Corollary 1 in the claimed time. □

Acknowledgments

Preliminary versions of some of the results appearing here have appeared in ICALP 2018 (Eisenbrand et al. [22], Koutecký et al. [43]).

References

[1] Aschenbrenner M, Hemmecke R (2007) Finiteness theorems in stochastic integer programming. *Found. Comput. Math.* 7(2):183–227.
[2] Barefoot CA, Entringer R, Swart HC (1987) Vulnerability in graphs—A comparative survey. *J. Combin. Math. Combin. Comput.* 1(38):13–22.
[3] Brand C, Koutecký M, Ordyniak S (2021) Parameterized algorithms for MILPs with small treedepth. *Proc. AAAI Conf. Artificial Intelligence* 35(14):12249–12257.
[4] Brand C, Koutecký M, Lassota A, Ordyniak S (2024) Separable convex mixed-integer optimization: Improved algorithms and lower bounds. Chan T, Fischer J, Iacono J, eds. *32nd Annual Eur. Sympos. Algorithms (ESA 2024)*, LIPIcs, vol. 308 (Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Wadern, Germany), 32:1–32:18.
[5] Briański M, Koutecký M, Král' D, Pekárková K, Schröder F (2024) Characterization of matrices with bounded Graver bases and depth parameters and applications to integer programming. *Math. Programming*. https://doi.org/10.1007/s10107-023-02048-x.
[6] Bruns W, Gubeladze J (2009) *Polytopes, Rings, and K-Theory* (Springer Science & Business Media, Berlin).
[7] Chan TFN, Cooper JW, Koutecký M, Král' D, Pekárková K (2022) Matrices of optimal tree-depth and a row-invariant parameterized algorithm for integer programming. *SIAM J. Comput.* 51(3):664–700.
[8] Chen L, Marx D (2018) Covering a tree with rooted subtrees–Parameterized and approximation algorithms. Czumaj A, ed. *Proc. Twenty-Ninth Annual ACM-SIAM Sympos. Discrete Algorithms (SODA 2018)* (SIAM, Philadelphia), 2801–2820.
[9] Chen H, Chen L, Zhang G (2022) Block-structured integer programming: Can we parameterize without the largest coefficient? *Discrete Optim.* 46:100743.
[10] Chen H, Chen L, Zhang G (2024) FPT algorithms for a special block-structured integer program with applications in scheduling. *Math. Programming*. https://doi.org/10.1007/s10107-023-02046-z.
[11] Chen L, Koutecký M, Xu L, Shi W (2020) New bounds on augmenting steps of block-structured integer programs. Grandoni F, Herman G, Sanders P, eds. *28th Annual Eur. Sympos. Algorithms (ESA 2020)*, vol. 173 (Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Wadern, Germany), 33:1–33:19.
[12] Cook WJ, Gerards AMH, Schrijver A, Tardos É (1986) Sensitivity theorems in integer linear programming. *Math. Programming* 34(3):251–264.
[13] Cslovjecsek J, Eisenbrand F, Hunkenschröder C, Rohwedder L, Weismantel R (2021) Block-structured integer and linear programming in strongly polynomial and near linear time. Marx D, ed. *Proc. 2021 ACM-SIAM Sympos. Discrete Algorithms (SODA 2021)* (SIAM, Philadelphia), 1666–1681.
[14] Cslovjecsek J, Eisenbrand F, Pilipczuk M, Venzin M, Weismantel R (2021) Efficient sequential and parallel algorithms for multistage stochastic integer programming using proximity. Mutzel P, Pagh R, Herman G, eds. *29th Annual Eur. Sympos. Algorithms (ESA 2021)*, LIPIcs, vol. 204 (Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Wadern, Germany), 33:1–33:14.
[15] Cslovjecsek J, Koutecký M, Lassota A, Pilipczuk M, Polak A (2024) Parameterized algorithms for block-structured integer programs with large entries. Woodruff DP, ed. *Proc. 2024 Annual ACM-SIAM Sympos. Discrete Algorithms (SODA)* (SIAM, Philadelphia), 740–751.
[16] Cygan M, Fomin FV, Kowalik L, Lokshtanov D, Marx D, Pilipczuk M, Pilipczuk M, Saurabh S (2015) *Parameterized Algorithms* (Springer, Berlin).
[17] De Loera JA, Hemmecke R, Köppe M (2013) *Algebraic and Geometric Ideas in the Theory of Discrete Optimization*, MOS-SIAM Series on Optimization, vol. 14 (SIAM, Philadelphia).
[18] Downey RG, Fellows MR (2013) *Fundamentals of Parameterized Complexity* (Springer, Berlin).
[19] Dvořák P, Eiben E, Ganian R, Knop D, Ordyniak S (2021) The complexity landscape of decompositional parameters for ILP: Programs with few global variables and constraints. *Artificial Intelligence* 300:103561.
[20] Eiben E, Ganian R, Knop D, Ordyniak S, Pilipczuk M, Wrochna M (2019) Integer programming and incidence treedepth. Lodi A, Nagarajan V, eds. *Integer Programming Combin. Optim. 20th Internat. Conf. (IPCO 2019)*, Lecture Notes in Computer Science, vol. 11480 (Springer, Berlin), 194–204.
[21] Eisenbrand F, Weismantel R (2020) Proximity results and faster algorithms for integer programming using the Steinitz lemma. *ACM Trans. Algorithms* 16(1):1–14.
[22] Eisenbrand F, Hunkenschröder C, Klein K-M (2018) Faster algorithms for integer programs with block structure. Chatzigiannakis I, Kaklamanis C, Marx D, Sannella D, eds. *45th Internat. Colloquium Automata Languages Programming (ICALP 2018)*, LIPIcs, vol. 107 (Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Wadern, Germany), 49:1–49:13.
[23] Eisenbrand F, Hunkenschröder C, Klein K-M, Koutecký M, Levin A, Onn S (2019) An algorithmic theory of integer programming. CoRR abs/1904.01361. Preprint, submitted April 2, http://arxiv.org/abs/1904.01361.
[24] Eisenbrand F, Hunkenschröder C, Klein K-M, Koutecký M, Levin A, Onn S (2023) Reducibility bounds of objective functions over the integers. *Oper. Res. Lett.* 51(6):595–598.
[25] Freuder EC (1990) Complexity of *K*-tree structured constraint satisfaction problems. Shrobe HE, Dietterich TG, Swartout WR, eds. *Proc. 8th Natl. Conf. Artificial Intelligence*, 2 volumes (AAAI Press/The MIT Press, Palo Alto, CA/Cambridge, MA), 4–9.
[26] Ganian R, Ordyniak S (2018) The complexity landscape of decompositional parameters for ILP. *Artificial Intelligence* 257:61–71.

[27] Ganian R, Ordyniak S, Ramanujan MS (2017) Going beyond primal treewidth for (M)ILP. Singh SP, Markovitch S, eds. *Proc. Thirty-First AAAI Conf. Artificial Intelligence* (AAAI Press, Palo Alto, CA), 815–821.
[28] Gima T, Otachi Y (2024) Extended MSO model checking via small vertex integrity. *Algorithmica* 86(1):147–170.
[29] Gima T, Hanaka T, Kobayashi Y, Murai R, Ono H, Otachi Y (2024) Structural parameterizations of vertex integrity. Uehara R, Yamanaka K, Yen H-C, eds. *WALCOM Algorithms Comput. 18th Internat. Conf. Workshops Algorithms Comput. (WALCOM 2024)*, Lecture Notes in Computer Science, vol. 14549 (Springer, Berlin), 406–420.
[30] Gordan P (1873) Ueber die Auflösung linearer Gleichungen mit reellen Coefficienten. *Math. Ann.* 6(1):23–28.
[31] Graver JE (1975) On the foundations of linear and integer linear programming I. *Math. Programming* 9(1):207–226.
[32] Grötschel M, Lovász L, Schrijver A (1988) *Geometric Algorithms and Combinatorial Optimization,* Algorithms and Combinatorics, vol. 2 (Springer, Berlin).
[33] Hemmecke R, Köppe M, Weismantel R (2014) Graver basis and proximity techniques for block-structured separable convex integer minimization problems. *Math. Programming* 145(1–2):1–18.
[34] Hemmecke R, Onn S, Romanchuk L (2013) *n*-Fold integer programming in cubic time. *Math. Programming* 137(1–2):325–341.
[35] Hunkenschröder C, Klein K-M, Koutecký M, Lassota A, Levin A (2024) Tight lower bounds for block-structured integer programs. Vygen J, Byrka J, eds. *Integer Programming Combin. Optim. 25th Internat. Conf. (IPCO 2024)*, Lecture Notes in Computer Science, vol. 14679 (Springer, Berlin), 224–237.
[36] Jansen BMP, Kratsch S (2015) A structural approach to kernels for ILPs: Treewidth and total unimodularity. Bansal N, Finocchi I, eds. *Proc. 23rd Annual Eur. Sympos. (ESA 2015)*, Lecture Notes in Computer Science, vol. 9294 (Springer, Berlin), 779–791.
[37] Jansen K, Rohwedder L (2023) On integer programming, discrepancy, and convolution. *Math. Oper. Res.* 48(3):1481–1495.
[38] Kannan R (1987) Minkowski's convex body theorem and integer programming. *Math. Oper. Res.* 12(3):415–440.
[39] Klein K-M (2022) About the complexity of two-stage stochastic IPs. *Math. Programming* 192(1):319–337.
[40] Klein K-M, Reuter J (2024) Collapsing the tower - On the complexity of multistage stochastic IPs. *ACM Trans. Algorithms* 20(3):22.
[41] Knop D, Pilipczuk M, Wrochna M (2020) Tight complexity lower bounds for integer linear programming with few constraints. *ACM Trans. Comput. Theory* 12(3):19:1–19:19.
[42] Knop D, Koutecký M, Levin A, Mnich M, Onn S (2023) High-multiplicity N-fold IP via configuration LP. *Math. Programming* 200(1):199–227.
[43] Koutecký M, Levin A, Onn S (2018) A parameterized strongly polynomial algorithm for block structured integer programs. Chatzigiannakis I, Kaklamanis C, Marx D, Sannella D, eds. *45th Internat. Colloquium Automata Languages Programming (ICALP 2018)*, LIPIcs, vol. 107 (Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Wadern, Germany), 85:1–85:14.
[44] Lee J, Paat J, Stallknecht I, Xu L (2020) Improving proximity bounds using sparsity. Baïou M, Gendron B, Günlük O, Mahjoub AR, eds. *Combin. Optim. 6th Internat. Sympos. (ISCO 2020)*, Lecture Notes in Computer Science, vol. 12176 (Springer, Berlin), 115–127.
[45] Lenstra HW Jr (1983) Integer programming with a fixed number of variables. *Math. Oper. Res.* 8(4):538–548.
[46] Nešetřil J, Ossona de Mendez P (2012) *Sparsity—Graphs, Structures, and Algorithms,* Algorithms and Combinatorics, vol. 28 (Springer, Berlin).
[47] Oertel T, Paat J, Weismantel R (2023) A colorful Steinitz Lemma with application to block-structured integer programs. *Math. Programming* 204(1–2):677–702.
[48] Onn S (2010) *Nonlinear Discrete Optimization: An Algorithmic Theory,* Zurich Lectures in Advanced Mathematics (European Mathematical Society, Zurich).
[49] Papadimitriou CH (1981) On the complexity of integer programming. *J. ACM* 28(4):765–768.
[50] Pilipczuk M, Pilipczuk M, Siebertz S (2020) Winter semesters 2017/18 and 2019/20. Lecture notes from the course "Sparsity" given at the Faculty of Mathematics, Informatics, and Mechanics of the University of Warsaw, https://www.mimuw.edu.pl/~mp248287/sparsity2/.
[51] Reidl F, Rossmanith P, Villaamil FS, Sikdar S (2014) A faster parameterized algorithm for treedepth. Esparza J, Fraigniaud P, Husfeldt T, Koutsoupias E, eds. *Automata Languages Programming 41st Internat. Colloquium (ICALP 2014) Proc.*, Part I, Lecture Notes in Computer Science, vol. 8572 (Springer, Berlin), 931–942.
[52] Rossi F, van Beek P, Walsh T (2008) Constraint programming. van Harmelen F, Lifschitz V, Porter BW, eds. *Handbook of Knowledge Representation,* Foundations of Artificial Intelligence, vol. 3 (Elsevier, Amsterdam), 181–211.
[53] Sevast'janov SV, Banaszczyk W (1997) To the Steinitz lemma in coordinate form. *Discrete Math.* 169(1–3):145–152.
[54] Steinitz E (1916) Bedingt konvergente Reihen und konvexe Systeme. *J. Reine Angew. Math.* 146:1–52.
[55] Storjohann A, Labahn G (1996) Asymptotically fast computation of Hermite normal forms of integer matrices. Engeler E, Caviness BF, Lakshman YN, eds. *Proc. 1996 Internat. Sympos. Symbolic Algebraic Comput. (ISSAC '96)* (ACM, New York), 259–266.
[56] Williams VV, Xu Y, Xu Z, Zhou R (2024) New bounds for matrix multiplication: From alpha to omega. Woodruff DP, ed. *Proc. 2024 ACM-SIAM Sympos. Discrete Algorithms (SODA 2024)* (SIAM, Philadelphia), 3792–3835.

Mathematical Programming (2023) 200:199–227
https://doi.org/10.1007/s10107-022-01882-9

FULL LENGTH PAPER

Series A

High-multiplicity *N*-fold IP via configuration LP

Dušan Knop[1] · Martin Koutecký[2] · Asaf Levin[3] · Matthias Mnich[4] · Shmuel Onn[3]

Received: 6 July 2021 / Accepted: 9 August 2022 / Published online: 21 September 2022

Abstract

N-fold integer programs (IPs) form an important class of block-structured IPs for which increasingly fast algorithms have recently been developed and successfully applied. We study *high-multiplicity N*-fold IPs, which encode IPs succinctly by presenting a description of each block *type* and a vector of block multiplicities. Our goal is to design algorithms which solve *N*-fold IPs in time polynomial in the size of the succinct encoding, which may be significantly smaller than the size of the explicit (non-succinct) instance. We present the first fixed-parameter algorithm for high-multiplicity *N*-fold IPs, which even works for convex objectives. Our key contribution is a novel proximity theorem which relates fractional and integer optima of the Configuration LP, a fundamental notion by Gilmore and Gomory [Oper. Res., 1961] which we generalize. Our algorithm for *N*-fold IP is faster than previous algorithms whenever the number of blocks is much larger than the number of block types, such as in *N*-fold IP models for various scheduling problems.

✉ Matthias Mnich
matthias.mnich@tuhh.de

Dušan Knop
dusan.knop@fit.cvut.cz

Martin Koutecký
koutecky@iuuk.mff.cuni.cz

Asaf Levin
levinas@ie.technion.ac.il

Shmuel Onn
onn@ie.technion.ac.il

1 Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic

2 Computer Science Institute, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

3 Technion – Israel Institute of Technology, Haifa, Israel

4 Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany

Keywords Integer programming · Configuration IP · Fixed-parameter algorithms · Scheduling

Mathematics Subject Classification 90C10 · 90C27 · 49M27

1 Introduction

The fundamental Integer Programming (IP) problem is to solve:

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

where $f : \mathbb{R}^n \to \mathbb{R}$, $A \in \mathbb{Z}^{m \times n}$, $\mathbf{b} \in \mathbb{Z}^m$, and $\mathbf{l}, \mathbf{u} \in (\mathbb{Z} \cup \{\pm\infty\})^n$. Any IP instance with infinite bounds $\mathbf{l}, \mathbf{u}$ can be reduced to an instance with finite bounds using standard techniques (solving the continuous relaxation and using proximity bounds to restrict the relevant region), so that from now on we will assume finite bounds $\mathbf{l}, \mathbf{u} \in \mathbb{Z}^n$. We denote $f_{\max} = \max_{\substack{\mathbf{x} \in \mathbb{Z}^n : \\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}}} |f(\mathbf{x})|$.

Integer Programming is a fundamental problem with vast importance both in theory and practice. Because it is NP-hard already with a single row (by reduction from Subset Sum) or with A a 0/1-matrix (by reduction from Vertex Cover), there is high interest in identifying tractable subclasses of IP. One such tractable subclass is N-fold IPs, whose constraint matrix A is defined as

$$A := E^{(N)} := \begin{pmatrix} E_1^1 & E_1^2 & \cdots & E_1^N \\ E_2^1 & 0 & \cdots & 0 \\ 0 & E_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & E_2^N \end{pmatrix}. \tag{1}$$

Here, $r, s, t, N \in \mathbb{N}$, $E^{(N)}$ is an $(r + Ns) \times Nt$-matrix, $E_1^i \in \mathbb{Z}^{r \times t}$ and $E_2^i \in \mathbb{Z}^{s \times t}$, $i \in [N]$, are integer matrices. We define $E := \begin{pmatrix} E_1^1 & E_1^2 & \cdots & E_1^N \\ E_2^1 & E_2^2 & \cdots & E_2^N \end{pmatrix}$, and call $E^{(N)}$ the *N-fold product of E*. The structure of $E^{(N)}$ allows us to divide any Nt-dimensional object, such as the variables of $\mathbf{x}$, bounds $\mathbf{l}$, $\mathbf{u}$, or the objective f, into N *bricks* of size t, e.g. $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^N)$. We use subscripts to index within a brick and superscripts to denote the index of the brick, i.e., x_j^i is the j-th variable of the i-th brick with $j \in [t]$ and $i \in [N]$. Problem (IP) with $A = E^{(N)}$ is known as *N-fold integer programming* (N-fold IP).

Such block-structured matrices have been the subject of extensive research stretching back to the '70s [3–5, 15, 16, 28, 42, 44, 45], as this special structure allows applying methods like the Dantzig-Wolfe decomposition and others, leading to significant speed-ups in practice. On the theoretical side, the term "N-fold IP" has been coined by De Loera et al. [9], and since then increasingly efficient algorithms have been developed and applied to various problems relating to N-fold IPs [2, 6, 25, 26, 29,

32]. This line of research culminated with an algorithm by Eisenbrand et al. [13] which solves N-fold IPs in time $(\|E\|_\infty rs)^{\mathcal{O}(r^2s+rs^2)} \cdot N \log N \cdot \log \|\mathbf{u}-\mathbf{l}\|_\infty \cdot \log f_{\max}$ for all separable convex objectives f (i.e., when $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i)$ and each $f_i : \mathbb{R} \to \mathbb{R}$ is convex).

1.1 Our contribution

Previous algorithms for N-fold IP have focused on reducing the run-time dependency on N down to almost linear. Instead, our interest here is on N-fold IPs which model applications where many bricks are of the same *type*, that is, they share the same bounds, right-hand side, and objective function. For those applications, it is natural to encode an N-fold IP instance *succinctly* by describing each brick type by its constraint matrix, bounds, right-hand side, and objective function, and giving a vector of brick multiplicities. When the number of brick types τ is much smaller than the number N of bricks, e.g., if $N \approx 2^\tau$, this succinct instance is (much) smaller than the previously studied encoding of N-fold IP, and an algorithm running in time polynomial in the size of the succinct instance may be (much) faster than current algorithms. We call the N-fold IP where the instance is given succinctly the *huge N-fold IP* problem, and we present a fast algorithm for it:

Theorem 1 *Huge N-fold IP with any separable convex objective can be solved in time*

$$(\|E\|_\infty rs)^{\mathcal{O}(r^2s+rs^2)} \operatorname{poly}(\tau, t, \log \|\mathbf{l}, \mathbf{u}, \mathbf{b}, N, f_{\max}\|_\infty) \ .$$

A natural application of Theorem 1 are scheduling problems. In many scheduling problems, the number n of jobs that must be assigned to machines, as well as the number m of machines, are very large, whereas the number of types of jobs and the number of kinds of machines are relatively small. An instance of such a scheduling problem can thus be compactly encoded by simply stating, for each job type and machine kind, the number of jobs with that type and machines with that kind together with their characteristics (like processing time, weight, release time, due date, etc.), respectively. This key observation was made by several researchers [7, 37], until Hochbaum and Shamir [20] coined the term *high-multiplicity scheduling problem*. Clearly, many efficient algorithms for scheduling problems, where all jobs are assumed to be distinct, become exponential-time algorithms for the corresponding high-multiplicity problem.

Let us shortly demonstrate how Theorem 1 allows designing algorithms which are efficient for the succinct high-multiplicity encoding of the input. In modern computational clusters, it is common to have several kinds of machines differing by processing unit type (high single- or multi-core performance CPUs, GPUs), storage type (HDD, SSD, etc.), network connectivity, etc. However, the number of machine kinds τ is still much smaller (perhaps 10) than the number of machines, which may be in the order of tens of thousands or more. Many scheduling problems have N-fold IP models [31] where τ is the number of machine kinds and N is the number of machines. On these models, Theorem 1 would likely outperform the currently fastest N-fold IP algorithms.
Proof ideas. To solve a high-multiplicity problem, one needs a succinct way to argue about solutions. In 1961, Gilmore and Gomory [17] introduced the fundamental and

widely influential notion of Configuration IP (ConfIP) which describes a solution (e.g., a schedule) by a list of pairs "(machine schedule s, multiplicity μ of machines with schedule s)". The linear relaxation of ConfIP, called the Configuration LP (ConfLP), can often be solved efficiently, and is known to provide solutions of strikingly high quality in practice [41]; for example, the optimum of the ConfLP for Bin Packing is conjectured to have value x such that an optimal integer packing uses $\leq \lceil x \rceil + 1$ bins [38]. However, surprisingly little is known *in general* about the structure of solutions of ConfIP and ConfLP, and how they relate to each other.

We define the Configuration IP and LP of an N-fold IP instance, and show how to solve the ConfLP quickly using the property that the ConfLP and ConfIP have polynomial encoding length even for huge N-fold IP. Our main technical contribution is a novel proximity theorem about N-fold IP, showing that a solution of its relaxation corresponding to the ConfLP optimum is very close to the integer optimum. Thus, the algorithm of Theorem 1 proceeds in three steps: (**1**) it solves the ConfLP, (**2**) it uses the proximity theorem to create a "residual" N'-fold instance with N' upperbounded by $(\|E\|_\infty rs)^{\mathcal{O}(rs)}$, and (**3**) it solves the residual instance by an existing N-fold IP algorithm.

1.2 Related work

Besides the references mentioned already, we point out that solving ConfLP is commonly used as subprocedure in approximation algorithms, e.g. [1, 14, 22, 27]. Jansen and Solis-Oba use a mixed ConfLP to give a parameterized $OPT + 1$ algorithm for bin packing [24]; Onn [36] gave a weaker form of Theorem 1 which only applies to the setting where $E_1^i = I$ and E_2^i is totally unimodular, for all i. Jansen et al. [25] extend the ConfIP to multiple "levels" of configurations. An extended version [31] of this paper shows how to model many scheduling problems as high multiplicity N-fold IPs, so that an application of Theorem 1 yields new parameterized algorithms for these problems. Knop and Koutecký [30] use our new proximity theorem to show efficient preprocessing algorithms (kernels) for scheduling problems.

There are currently several "fastest" algorithms for N-fold IP with standard (non-succinct) encoding. First, we have already mentioned the algorithm of Eisenbrand et al. [13]. Second, the algorithm of Jansen et al. [26] has a better parameter dependency of $(\|E\|_\infty rs)^{\mathcal{O}(r^2s+s^2)}$ (as compared with $(\|E\|_\infty rs)^{\mathcal{O}(r^2s+rs^2)}$ of the previous algorithm), but has a slightly worse dependence on N of $N \log^5 N$, and only works for linear objectives. Third, a recent algorithm of Cslovjecsek et al. [8] again only works for linear objectives and runs in time $(\|E\|_\infty s)^{\mathcal{O}(s^2)} \operatorname{poly}(r) N \log^2(Nt) \log^2(\|\mathbf{l}, \mathbf{u}, \mathbf{b}, f_{\max}\|_\infty) + (\|E\|_\infty rs)^{\mathcal{O}(r^2s+s^2)} Nt$. While the authors claim that this constitutes the currently fastest algorithm, it seems that it is only potentially faster than prior work in a narrow parameter regime.

The third paper, by Cslovjecsek et al. [8], is the closest to ours in its approach: it solves a strong relaxation of N-fold IP which coincides with the ConfLP if each brick is of a distinct type, and which is generalized by the ConfLP (in our work) otherwise. The authors show that this relaxation can be solved in near-linear time, and then develop a proximity theorem similar to ours (but using different techniques)

and a dynamic program, which allows them to construct and solve a residual instance in linear time. An earlier version of our paper [31] stated a worse proximity bound than that of Cslovjecsek et al. [8], but our bound applies to separable convex objective whereas theirs [8] does not. Presently, we adapt one of their lemmas ([8, Lemma 3]) (Lemma 5) and a modeling idea (Sect. 3.4) to obtain the same proximity bound as they have [8], but which also works for separable convex objectives. It is likely that the complexity of our algorithm to solve the ConfLP could be improved along the lines of their work [8]. Despite these similarities, we highlight that only our algorithm solves the high-multiplicity version of N-fold IP.

2 Preliminaries

For positive integers m, n with $m \leq n$ we set $[m, n] = \{m, m+1, \ldots, n\}$ and $[n] = [1, n]$. 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 or $x(i)$). For $\alpha \in \mathbb{R}$, $\lfloor \alpha \rfloor$ is the floor of α, $\lceil \alpha \rceil$ is the ceiling of α, and we define $\{\alpha\} = \alpha - \lfloor \alpha \rfloor$, similarly for vectors where these operators are defined component-wise.

We call a brick of $\mathbf{x}$ *integral* if all of its coordinates are integral, and *fractional* otherwise.

Huge N-fold IP. The *huge N-fold IP* problem is an extension of N-fold IP to the high-multiplicity scenario, where there are potentially *exponentially* many bricks. This requires a succinct representation of the input and output. The input to a huge N-fold IP problem with τ *brick types* is defined by matrices $E_1^i \in \mathbb{Z}^{r \times t}$ and $E_2^i \in \mathbb{Z}^{s \times t}$, $i \in [\tau]$, vectors $\mathbf{l}^1, \ldots, \mathbf{l}^\tau, \mathbf{u}^1, \ldots, \mathbf{u}^\tau \in \mathbb{Z}^t$, $\mathbf{b}^0 \in \mathbb{Z}^r$, $\mathbf{b}^1, \ldots, \mathbf{b}^\tau \in \mathbb{Z}^s$, functions $f^1, \ldots, f^\tau : \mathbb{R}^t \to \mathbb{R}$ satisfying $\forall i \in [\tau]$, $\forall \mathbf{x} \in \mathbb{Z}^t : f^i(\mathbf{x}) \in \mathbb{Z}$ and given by evaluation oracles, and integers $\mu^1, \ldots, \mu^\tau \in \mathbb{N}$ such that $\sum_{i=1}^{\tau} \mu^i = N$. We say that a brick is *of type i* if its lower and upper bounds are $\mathbf{l}^i$ and $\mathbf{u}^i$, its right hand side is $\mathbf{b}^i$, its objective is f^i, and the matrices appearing at the corresponding coordinates are E_1^i and E_2^i. The task is to solve (IP) with a matrix $E^{(N)}$ which has μ^i bricks of type i for each i. Onn [35] shows that for any solution, there exists a solution which is at least as good and has only few (at most $\tau \cdot 2^t$) distinct bricks. In Sect. 3 we show new bounds which do not depend exponentially on t.

2.1 Graver bases and the Steinitz lemma

Let $\mathbf{x}, \mathbf{y}$ be n-dimensional vectors. We call $\mathbf{x}, \mathbf{y}$ *sign-compatible* if they lie in the same orthant, that is, for each $i \in [n]$, $x_i \cdot y_i \geq 0$. We call $\sum_i \mathbf{g}^i$ a *sign-compatible sum* if all $\mathbf{g}^i$ are pair-wise sign-compatible. Moreover, we write $\mathbf{y} \sqsubseteq \mathbf{x}$ if $\mathbf{x}$ and $\mathbf{y}$ are sign-compatible and $|y_i| \leq |x_i|$ for each $i \in [n]$. Clearly, $\sqsubseteq$ imposes a partial order, called "conformal order", on n-dimensional vectors. For an integer matrix $A \in \mathbb{Z}^{m \times n}$, its *Graver basis* $\mathcal{G}(A)$ is the set of $\sqsubseteq$-minimal non-zero elements of the *lattice* of A, $\ker_{\mathbb{Z}}(A) = \{\mathbf{z} \in \mathbb{Z}^n \mid A\mathbf{z} = \mathbf{0}\}$. A *circuit* of A is an element $\mathbf{g} \in \ker_{\mathbb{Z}}(A)$ whose support $\mathrm{supp}(\mathbf{g})$ (i.e., the set of its non-zero entries) is minimal under inclusion and whose entries are coprime. We denote the set of *circuits of* A by $\mathcal{C}(A)$. It is known

that $\mathcal{C}(A) \subseteq \mathcal{G}(A)$ [34, Definition 3.1 and remarks]. We make use of the following two propositions:

Proposition 1 (Positive Sum Property [34, Lemma 3.4]) *Let $A \in \mathbb{Z}^{m\times n}$ be an integer matrix. For any integer vector $\mathbf{x} \in \ker_{\mathbb{Z}}(A)$, there exists an $n' \leq 2n-2$ and a decomposition $\mathbf{x} = \sum_{j=1}^{n'} \alpha_j \mathbf{g}_j$ with $\alpha_j \in \mathbb{N}$ for each $j \in [n']$, into a sum of $\mathbf{g}_j \in \mathcal{G}(A)$. For any fractional vector $\mathbf{x} \in \ker(A)$ (that is, $A\mathbf{x} = \mathbf{0}$), there exists a decomposition $\mathbf{x} = \sum_{j=1}^{n} \alpha_j \mathbf{g}_j$ into $\mathbf{g}_j \in \mathcal{C}(A)$, where $\alpha_j \geq 0$ for each $j \in [n]$.*

Proposition 2 (Separable convex superadditivity [10, 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 with the same sign-pattern from $\{\leq 0, \geq 0\}^n$, that is, belonging to the same orthant of $\mathbb{R}^n$. Then*

$$f\left(\mathbf{x} + \sum_{j=1}^{k} \alpha_j \mathbf{g}_j\right) - f(\mathbf{x}) \geq \sum_{j=1}^{k} \alpha_j \left(f(\mathbf{x}+\mathbf{g}_j) - f(\mathbf{x})\right) \tag{2}$$

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

Our proximity theorem relies on the Steinitz Lemma, which has recently received renewed attention [11, 12, 23].

Lemma 1 (Steinitz [40], Sevastjanov, Banaszczyk [39]) *Let $\|\cdot\|$ denote any norm, and let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ be such that $\|\mathbf{x}_i\| \leq 1$ for $i \in [n]$ and $\sum_{i=1}^{n} \mathbf{x}_i = 0$. Then there exists a permutation $\pi \in S_n$ such that for all $k = 1, \ldots, n$, the prefix sum satisfies $\left\|\sum_{i=1}^{k} \mathbf{x}_{\pi(i)}\right\| \leq d$.*

For an integer matrix A, we define $g_1(A) = \max_{\mathbf{g}\in\mathcal{G}(A)} \|\mathbf{g}\|_1$. When it could make a difference, we will state our bounds both in terms of $\|E\|_\infty$ (worst-case, when we have no other information) and in terms of $g_1(E_2) := \max_i g_1(E_2^i)$, e.g. in Lemma 10 and Theorem 2.

3 Proof of Theorem 1

We first give a relatively high-level description of the proof, before we present all its details.

3.1 Proof overview and ideas

3.1.1 Configuration LP and IP

Given an input to the huge N-fold IP, we first reformulate it as another IP, which we refer to as the Configuration IP. We then consider its fractional relaxation, the so-called Configuration LP. Our approach is to (efficiently) solve the Configuration LP, and bound the distance of its LP optimum to the integer optimum (of the Configuration

IP). We use this bound to reduce the input to the huge N-fold IP from a high-multiplicity input to an input of a standard N-fold IP which is small both in terms of the number of bricks and size of the bounding box. This small input we then solve using an existing N-fold IP algorithm. On this way, there are several non-trivial obstacles that we need to overcome.

We will refer to huge N-fold IP as HugeIP, its corresponding fractional relaxation as HugeCP (this is a convex program if the objective f is convex), the Configuration LP of the HugeIP as ConfLP, and to its integer version as ConfIP. We define a mapping φ from the solutions of ConfLP to the solutions of HugeCP which, for every variable $y_{\mathbf{c}}$ of the ConfLP introduces $\lfloor y_{\mathbf{c}} \rfloor$ bricks with configuration $\mathbf{c}$, and then introduces $\sum_{\mathbf{c}} \{y_{\mathbf{c}}\}$ bricks with configuration $\frac{1}{\sum_{\mathbf{c}} \{y_{\mathbf{c}}\}} \sum_{\mathbf{c}} \{y_{\mathbf{c}}\} \cdot \mathbf{c}$ (i.e., an "average" configuration). We call a solution $\mathbf{x}^*$ of HugeCP "conf-optimal" if it is the image $\varphi(\mathbf{y}^*)$ of some ConfLP optimum $\mathbf{y}^*$. One would hope that then the objective value of a conf-optimal solution $\mathbf{x}^*$ in HugeCP and of $\mathbf{y}^*$ in ConfLP were identical. While this is true for any linear objective f, it need not be true for a convex objective f. To overcome this impediment, we introduce an auxiliary objective $\hat{f}$ which preserves the values of optima of ConfLP and conf-optimal solutions of HugeCP.

3.1.2 Proximity theorem

The bulk of our work is showing that for each conf-optimal solution $\mathbf{x}^*$ of the HugeLP, there is an optimum $\mathbf{z}^*$ of the HugeIP whose ℓ_1-distance from $\mathbf{x}^*$ is bounded by $P := (\|E\|_\infty rs)^{\mathcal{O}(rs)}$. We will show that we can obtain a ConfLP optimum $\mathbf{y}$ with support of size at most $r + \tau$, and by the definition of φ (recall that $\mathbf{x}^* = \varphi(\mathbf{y})$), this means that $\mathbf{x}^*$ has at most $r + \tau + 1$ distinct bricks (the $+1$ is due to φ creating an additional "average configuration" brick type). This, in turn, means that our bound on the ℓ_1-distance between $\mathbf{z}^*$ and $\mathbf{x}^*$ says something about ConfLP and ConfIP: for any ConfLP optimum $\mathbf{y}$ there is a ConfIP optimum $\mathbf{y}^*$ in ℓ_1-distance at most P where any configuration $\mathbf{c}$ in the support of $\mathbf{y}^*$ is at most P far from some configuration $\mathbf{c}'$ in the support of $\mathbf{y}$. As far as we know, this is a unique result about the Configuration LP.

A way of bounding the distance between some types of optima in an integer program has been introduced by Hochbaum and Shanthikumar [21] and adapted to the setting of N-fold IP by Hemmecke at al. [19]. A somewhat different approach was later developed by Eisenbrand and Weismantel [11] in the setting of IPs with few rows, and was adapted to the setting of N-fold IPs soon after [12, 13]. The idea is as follows. Let $\mathbf{x}^*$ be a HugeCP optimum, and $\mathbf{z}^*$ be a HugeIP optimum, We call a non-zero integral vector $\mathbf{p} \sqsubseteq \mathbf{x}^* - \mathbf{z}^*$, i.e., which is sign-compatible (i.e., has the same sign-pattern) with $\mathbf{x}^* - \mathbf{z}^*$ and which is smaller in absolute value than $\mathbf{x}^* - \mathbf{z}^*$ in each coordinate, a *cycle* of $\mathbf{x}^* - \mathbf{z}^*$. If $\mathbf{z}^*$ minimizes $\|\mathbf{x}^* - \mathbf{z}^*\|_1$, it can be shown that no cycle of $\mathbf{x}^* - \mathbf{z}^*$ exists. Moreover, if a cycle exists, then a cycle of ℓ_1-norm at most B exists, which implies $\|\mathbf{x}^* - \mathbf{z}^*\|_1 \leq B$.

Notice that the previous argument assumes $\mathbf{x}^*$ to be a HugeCP optimum: this cannot be replaced with a conf-optimal solution for the following reason. The existence of a cycle $\mathbf{p}$ leads to a contradiction because either $\mathbf{z}^* + \mathbf{p}$ is also a HugeIP optimum (but closer to $\mathbf{x}^*$) or $\mathbf{x}^* - \mathbf{p}$ is also a HugeCP optimum (but closer to $\mathbf{z}^*$). But if $\mathbf{x}^*$ is a conf-

optimal solution, we have no guarantee that $\mathbf{x}^* - \mathbf{p}$ is again a configurable solution, and the argument breaks down. This means that we need to restrict our attention to cycles with the property that if $\mathbf{x}^*$ is a configurable solution, then $\mathbf{x}^* - \mathbf{p}$ is also configurable.

We call such a $\mathbf{p}$ a *configurable cycle*. The next task is an analogy of the argument above: if $\mathbf{x}^*$ is conf-optimal and $\mathbf{z}^*$ is a HugeIP optimum, then the existence of a configurable cycle $\mathbf{p}$ of $\mathbf{x}^* - \mathbf{z}^*$ leads to a contradiction. For that, we need the separability and convexity of the objective f and a careful use of the configurability of $\mathbf{p}$. With this argument at hand, we have reduced our task to bounding the norm of any configurable cycle (Lemma 7).

However, the main existing tool for showing proximity is by ruling out cycles. To overcome this, we develop new tools to deal with configurable cycles.

3.1.3 The algorithm

It remains to use our proximity bound P. As already hinted at, if two solutions differ in ℓ_1-norm by at most P, then they may differ in at most P bricks. This means that we may fix all but P bricks for each configuration appearing in the ConfLP optimum. Since the size of the support of the ConfLP optimum is small $(r + \tau)$, the total number of bricks to be determined is also small, and can be done using a standard N-fold IP algorithm in the required time complexity (Proof of Theorem 1)

To recap, the algorithm works in the following steps.

1. We solve the ConfLP and obtain its optimum $\mathbf{y}$ by solving its Dual LP using a separation oracle. The separation oracle is implemented using a fixed-parameter algorithm for IP with small coefficients.
2. We use the ConfLP optimum $\mathbf{y}$ to fix the solution on all but $(r + \tau)P$ bricks.
3. The remaining instance can be encoded as an N-fold IP with at most $(r + \tau)P$ bricks and solved using an existing algorithm.

Let us now go back to a detailed proof of Theorem 1.

3.2 Configurations of huge *N*-fold IP

Fix a huge N-fold IP instance with τ types. Recall that μ^i denotes the number of bricks of type i, and $\boldsymbol{\mu} = (\mu^1, \dots, \mu^\tau)$. We define for each $i \in [\tau]$ the set of configurations of type i as

$$\mathcal{C}^i = \left\{ \mathbf{c} \in \mathbb{Z}^t \mid E_2^i \mathbf{c} = \mathbf{b}^i,\ \mathbf{l}^i \le \mathbf{c} \le \mathbf{u}^i \right\} .$$

Here we are interested in four instances of convex programming (CP) and convex integer programming (IP) related to huge N-fold IP. First, we have the *Huge IP*

$$\min f(\mathbf{x}) :\ E^{(N)}\mathbf{x} = \mathbf{b},\ \mathbf{l} \le \mathbf{x} \le \mathbf{u},\ \mathbf{x} \in \mathbb{Z}^{Nt}, \tag{HugeIP}$$

and the *Huge CP*, which is a relaxation of (HugeIP),

$$\min \hat{f}(\mathbf{x}) :\ E^{(N)}\mathbf{x} = \mathbf{b},\ \mathbf{l} \le \mathbf{x} \le \mathbf{u},\ \mathbf{x} \in \mathbb{R}^{Nt} . \tag{HugeCP}$$

We shall define the objective function $\hat{f}$ later, for now it suffices to say that for all integral feasible $\mathbf{x} \in \mathbb{Z}^{Nt}$ we have $f(\mathbf{x}) = \hat{f}(\mathbf{x})$ so that indeed the optimum of (HugeCP) lower bounds the optimum of (HugeIP) and that $\hat{f}$ is convex. Then, there is the *Configuration LP* of (HugeIP), that is, the following linear program:

$$\min \mathbf{vy} = \min \sum_{i=1}^{\tau} \sum_{\mathbf{c} \in \mathcal{C}^i} f^i(\mathbf{c}) \cdot y(i, \mathbf{c}) \tag{3}$$

$$\sum_{i=1}^{\tau} E_1^i \sum_{\mathbf{c} \in \mathcal{C}^i} \mathbf{c} y(i, \mathbf{c}) = \mathbf{b}^0,$$

$$\sum_{\mathbf{c} \in \mathcal{C}^i} y(i, \mathbf{c}) = \mu^i \qquad \forall i \in [\tau],$$

$$\mathbf{y} \geq \mathbf{0} \ . \tag{4}$$

Letting B be its constraint matrix and $\mathbf{d} = \begin{pmatrix} \mathbf{b}^0 \\ \boldsymbol{\mu}^\intercal \end{pmatrix}$ be the right hand side, we can shorten (3)–(4) as

$$\min \mathbf{vy} : \ B\mathbf{y} = \mathbf{d}, \ \mathbf{y} \geq \mathbf{0} \ . \tag{ConfLP}$$

Finally, by observing that $B\mathbf{y} = \mathbf{d}$ implies $y(i, \mathbf{c}) \leq \|\boldsymbol{\mu}\|_\infty$ for all $i \in [\tau], \mathbf{c} \in \mathcal{C}^i$, defining $C = \sum_{i \in [\tau]} |\mathcal{C}^i|$, leads to the *Configuration ILP*,

$$\min \mathbf{vy} : \ B\mathbf{y} = \mathbf{d}, \ \mathbf{0} \leq \mathbf{y} \leq (\|\boldsymbol{\mu}\|_\infty, \ldots, \|\boldsymbol{\mu}\|_\infty)^\intercal, \ \mathbf{y} \in \mathbb{N}^C \ . \tag{ConfILP}$$

A solution $\mathbf{x}$ of (HugeCP) is *configurable* if, for every $i \in [\tau]$, each brick $\mathbf{x}^j$ of type i is a convex combination of $\mathcal{C}^i$, i.e., $\mathbf{x}^j \in \operatorname{conv}(\mathcal{C}^i)$. We shall define a mapping from solutions of (ConfLP) to configurable solutions of (HugeCP) as follows. For every solution $\mathbf{y}$ of (ConfLP) we define a solution $\mathbf{x} = \varphi(\mathbf{y})$ of (HugeCP) to have $\lfloor y(i, \mathbf{c}) \rfloor$ bricks of type i with configuration $\mathbf{c}$ and, for each $i \in [\tau]$, let $\mathfrak{f}^i = \sum_{\mathbf{c} \in \mathcal{C}^i} \{y(i, \mathbf{c})\}$ and let $\mathbf{x}$ have $\mathfrak{f}^i$ bricks with value $\hat{\mathbf{c}}_i = \frac{1}{\mathfrak{f}^i} \sum_{\mathbf{c} \in \mathcal{C}^i} \{y(i, \mathbf{c})\} \mathbf{c}$. (Because $\sum_{\mathbf{c} \in \mathcal{C}^i} y(i, \mathbf{c}) = \mu^i$ and $\sum_{\mathbf{c} \in \mathcal{C}^i} \lfloor y(i, \mathbf{c}) \rfloor$ is clearly integral, $\mathfrak{f}^i = \mu^i - \sum_{\mathbf{c} \in \mathcal{C}^i} \lfloor y(i, \mathbf{c}) \rfloor$ is also integral.) Note that $\varphi(\mathbf{y})$ has at most as many fractional bricks as $\mathbf{y}$ has fractional entries since each $\mathfrak{f}^i < 1$ and the number of non-zero $\mathfrak{f}^i$ is at most the number of fractional entries of $\mathbf{y}$. Call a solution $\mathbf{x}$ of (HugeCP) *conf-optimal* if there is an optimal solution $\mathbf{y}$ of (ConfLP) such that $\mathbf{x} = \varphi(\mathbf{y})$.

We are going to introduce an auxiliary objective function $\hat{f}$, but we first want to discuss our motivation in doing so. The reader might already see that for any integer solution $\mathbf{y} \in \mathbb{Z}^C$ of (ConfILP), $\mathbf{vy} = f(\varphi(\mathbf{y}))$ holds, as we shall prove in Lemma 4. Our natural hope would be that for a fractional optimum $\mathbf{y}^*$ of (ConfLP) we would have $\mathbf{vy}^* = f(\varphi(\mathbf{y}^*))$. However, by convexity of f and the construction of $\hat{\mathbf{c}}_i$ it only follows that $\mathbf{vy}^* \geq f(\varphi(\mathbf{y}^*))$. Even worse, there may be two conf-optimal solutions $\mathbf{x}$ and $\mathbf{x}'$ with $f(\mathbf{x}) < f(\mathbf{x}')$. To overcome this, we define an auxiliary objective function $\hat{f}$ with the property that for any conf-optimal solution $\mathbf{x}^*$ of (HugeCP) and any optimal solution $\mathbf{y}^*$ of (ConfLP), $\mathbf{vy}^* = \hat{f}(\mathbf{x}^*)$.

Fix a brick $\mathbf{x}^j$ of type i. We say that a multiset $\Gamma^j \subseteq (\mathcal{C}^i \times \mathbb{R}_{\geq 0})$ is a *decomposition of* $\mathbf{x}^j$ and write $\mathbf{x}^j = \sum \Gamma^j$ if $\mathbf{x}^j = \sum_{(\mathbf{c},\lambda_\mathbf{c})\in\Gamma^j} \lambda_\mathbf{c}\mathbf{c}$ and $\sum_{(\mathbf{c},\lambda_\mathbf{c})\in\Gamma^j} \lambda_\mathbf{c} = 1$. We define the objective $\hat{f}(\mathbf{x})$ for all configurable solutions as $\hat{f}(\mathbf{x}) = \sum_{j=1}^{N} \hat{f}^i(\mathbf{x}^j)$, where

$$\hat{f}^i(\mathbf{x}^j) = \min_{\Gamma^j : \sum \Gamma^j = \mathbf{x}^j} \sum_{(\mathbf{c},\lambda_\mathbf{c})\in\Gamma^j} \lambda_\mathbf{c} \cdot f^i(\mathbf{c}) \ . \tag{5}$$

In a sense, $\hat{f}(\mathbf{x})$ is the value of the minimum (w.r.t. f) interpretation of $\mathbf{x}$ as a convex combination of feasible integer solutions. Correspondingly, we call a decomposition Γ^j of $\mathbf{x}^j$ $\hat{f}$*-optimal* if it is a minimizer of (5). Formally, we let $\hat{f}^i(\mathbf{x}^j) = f^i(\mathbf{x}^j)$ for a non-configurable $\mathbf{x}^j$ in order to make the definition of (HugeCP) valid; however, we are never interested in the value of $\hat{f}$ for non-configurable bricks in the following.

Lemma 2 *Let* $\mathbf{x}$ *be a configurable solution of* (HugeCP)*, and* $\mathbf{x}^j$ *be a brick of type* i*. Then* $f^i(\mathbf{x}^j) \leq \hat{f}^i(\mathbf{x}^j)$*. If* $\mathbf{x}^j$ *is integral, then* $f^i(\mathbf{x}^j) = \hat{f}^i(\mathbf{x}^j)$*.*

Proof By convexity of f^i we have

$$f^i(\mathbf{x}^j) = f^i\left(\sum_{(\mathbf{c},\lambda_\mathbf{c})\in\Gamma^j} \lambda_\mathbf{c}\mathbf{c}\right) \leq \sum_{(\mathbf{c},\lambda_\mathbf{c})\in\Gamma^j} \lambda_\mathbf{c} f^i(\mathbf{c}),$$

for any decomposition Γ^j of $\mathbf{x}^j$. If $\mathbf{x}^j$ is integral, then $\Gamma^j = \{(\mathbf{x}^j, 1)\}$ is its optimal decomposition (not necessarily unique[1]), concluding the proof. □

Moreover, for each $\mathbf{x}^j$ there is an $\hat{f}$-optimal decomposition Γ^j with $|\Gamma^j| \leq t + 1$ since $\hat{f}$-optimal decompositions correspond to optima of a linear program with $t + 1$ equality constraints, namely

$$\min \sum_{\mathbf{c}\in\mathcal{C}^i} \lambda_\mathbf{c} f^i(\mathbf{c}) \quad \text{s.t.} \quad \sum_{\mathbf{c}\in\mathcal{C}^i} \lambda_\mathbf{c}\mathbf{c} = \mathbf{x}^j,\ \|\boldsymbol{\lambda}\|_1 = 1,\ \boldsymbol{\lambda} \geq \mathbf{0} \ . \tag{6}$$

Let us describe the relationship of the objective values of the various formulations.

Lemma 3 *For any feasible solution* $\tilde{\mathbf{y}}$ *of* (ConfLP)*,*

$$\mathbf{v}\tilde{\mathbf{y}} \geq \hat{f}(\varphi(\tilde{\mathbf{y}})) \ . \tag{7}$$

Proof Let $\tilde{\mathbf{x}} = \varphi(\tilde{\mathbf{y}})$. We can decompose $\hat{f}(\varphi(\tilde{\mathbf{y}})) = U_1 + U_2$, where U_1 is the cost of integer bricks of $\varphi(\tilde{\mathbf{y}})$ and U_2 is the cost of its fractional bricks. It is easy to see that $U_1 = \mathbf{v}\lfloor\tilde{\mathbf{y}}\rfloor$ by the equality of f^i and $\hat{f}^i$, for all $i \in [\tau]$, over integer vectors. We shall further decompose the value U_2 into costs of fractional bricks of each type. For each $i \in [\tau]$, the cost of each fractional brick of type i is at most $\frac{1}{f^i} \sum_{\mathbf{c}\in\mathcal{C}^i} \{\tilde{y}(i,\mathbf{c})\} f^i(\mathbf{c})$ because

[1] e.g., potentially $\mathbf{x}^j = \frac{1}{2}((\mathbf{x}^j + \mathbf{c}) + (\mathbf{x}^j - \mathbf{c}))$ for some $\mathbf{c}$, and Γ^j is optimal for any linear objective.

the decomposition $\left\{\left(\mathbf{c}, \frac{1}{\mathfrak{f}^i}\{\tilde{y}^i_{\mathbf{c}}\}\right) \middle| \mathbf{c} \in \mathcal{C}^i\right\}$ of $\hat{\mathbf{c}}_i$ (recall that $\hat{\mathbf{c}}_i = \frac{1}{\mathfrak{f}^i}\sum_{\mathbf{c}\in\mathcal{C}^i}\{\tilde{y}(i,\mathbf{c})\}\mathbf{c}$) is merely a feasible (not necessarily optimal) solution of (6) Summing this estimate up over all $\mathfrak{f}^i$ fractional bricks of type i gives $\mathfrak{f}^i \cdot \frac{1}{\mathfrak{f}^i}\sum_{\mathbf{c}\in\mathcal{C}^i}\{\tilde{y}(i,\mathbf{c})\}f^i(\mathbf{c}) = \mathbf{v}^i\{\tilde{\mathbf{y}}^i\}$, concluding the proof. □

Lemma 4 *Let* $\hat{\mathbf{y}}$ *be an optimum of* (ConfILP)*,* $\mathbf{z}^*$ *be an optimum of* (HugeIP)*,* $\mathbf{y}^*$ *be an optimum of* (ConfLP)*,* $\tilde{\mathbf{x}} = \varphi(\mathbf{y}^*)$*, and* $\mathbf{x}^*$ *be a configurable optimum of* (HugeCP)*. Then*

$$\hat{f}(\mathbf{z}^*) = f(\mathbf{z}^*) = f(\varphi(\hat{\mathbf{y}})) = \mathbf{v}\hat{\mathbf{y}} \geq \mathbf{v}\mathbf{y}^* = \hat{f}(\tilde{\mathbf{x}}) = \hat{f}(\mathbf{x}^*) \ .$$

Proof We have $\hat{f}(\mathbf{z}^*) = f(\mathbf{z}^*)$ by equality of $\hat{f}$ and f on integer solutions (Lemma 2), and $f(\mathbf{z}^*) = f(\varphi(\hat{\mathbf{y}})) = \mathbf{v}\hat{\mathbf{y}}$ by the definition of φ and the fact that $\hat{\mathbf{y}}$ is an integer optimum. Clearly, $\mathbf{v}\hat{\mathbf{y}} \geq \mathbf{v}\mathbf{y}^*$, because (ConfLP) is a relaxation of (ConfILP) and thus the former lower bounds the latter.

Let us construct a mapping ϕ for any configurable solution $\mathbf{x}$ of (HugeCP). Start with $\phi(\mathbf{x}) = \mathbf{y} = \mathbf{0}$. For each brick $\mathbf{x}^j$ of type i let Γ^j be a $\hat{f}$-optimal decomposition of $\mathbf{x}^j$ and update $y^i_{\mathbf{c}} := y^i_{\mathbf{c}} + \lambda_{\mathbf{c}}$ for each $(\mathbf{c}, \lambda_{\mathbf{c}}) \in \Gamma^j$. Now it is easy to see that

$$\mathbf{v}\phi(\mathbf{x}^*) = \hat{f}(\mathbf{x}^*) \ . \tag{8}$$

Our goal is to argue that $\mathbf{v}\mathbf{y}^* = \hat{f}(\tilde{\mathbf{x}}) = \hat{f}(\mathbf{x}^*)$. We have $\hat{f}(\tilde{\mathbf{x}}) = \hat{f}(\varphi(\mathbf{y}^*)) \leq \mathbf{v}\mathbf{y}^*$ by (7), but by optimality of $\mathbf{y}^*$ and (8) it must be that $\mathbf{v}\phi(\tilde{\mathbf{x}}) = \hat{f}(\tilde{\mathbf{x}}) \geq \mathbf{v}\mathbf{y}^*$ and hence $\mathbf{v}\mathbf{y}^* = \hat{f}(\tilde{\mathbf{x}})$. Similarly,

$$\hat{f}(\mathbf{x}^*) = \mathbf{v}\phi(\mathbf{x}^*) \geq \mathbf{v}\mathbf{y}^* \geq \hat{f}(\varphi(\mathbf{y}^*))$$

with the "=" by (8), the first "≥" by optimality of $\mathbf{y}^*$, and the second "≥" by (7). However, since $\hat{f}(\varphi(\mathbf{y}^*)) \geq \hat{f}(\mathbf{x}^*)$ by optimality of $\mathbf{x}^*$, all inequalities are in fact equalities and thus $\mathbf{v}\mathbf{y}^* = \hat{f}(\mathbf{x}^*)$. □

Remark 1 We only need the properties of $\hat{f}$ that we have proved so far. To gain a little bit more intuition, consider the dual of the LP (6). Notice that the set of right hand sides $\mathbf{x}^j$ whose optimum is attained by a particular set of configurations $\mathrm{supp}(\boldsymbol{\lambda})$ is a polyhedron. Call such a set a cell. This means that $\hat{f}$ is a convex function which is linear in each cell. Another observation is that $\hat{f}$ is non-separable.

We do not have a more intuitive explanation of $\hat{f}$. It would be tempting to think that $\hat{f}$ is the piece-wise linear approximation of f in which, for every $i \in [Nt]$, we replace each segment of f_i between two adjacent integers $k, k+1$ by the affine function going through the points $(k, f_i(k))$ and $(k+1, f_i(k+1))$. However, this turns out to be incorrect: for example, say that $f_1(x_1) = |x_1 - 1|$ (thus $f_1(0) = f_1(2) = 1$ and $f_1(1) = 0$) and that we set $x_1 = 2x_2$ for a new integer variable x_2. This constraint ensures that x_1 only takes on even values. Thus, x_1 never attains the value 1 and $\hat{f}_1(1) \geq 1$ even though the piece-wise linear approximation of f_1 has value 0 at 1.

Bounding the number of fractional coordinates.

Lemma 5 (Adaptation of [8, Lemma 4.1]) *An optimal vertex solution* $\mathbf{y}^*$ *of* (ConfLP) *has at most* $2r$ *fractional coordinates.*

Proof Notice that if a brick $(\mathbf{y}^*)^i$ is a vertex of the set $Q^i := \operatorname{conv}\{\mathbf{y}^i \in \mathbb{R}^{\mathcal{C}^i} \mid \mathbf{1}\mathbf{y}^i = \mu^i, \mathbf{y}^i \geq \mathbf{0}\}$, then it is integral. Thus, any brick of $\mathbf{y}^*$ which is fractional cannot be a vertex of Q^i and hence there exists a direction $\mathbf{e}^i \in \operatorname{Ker}_{\mathbb{Z}}(\mathbf{1})$ and a length $\lambda^i > 0$ such that $(\mathbf{y}^*)^i \pm \lambda^i \mathbf{e}^i \in Q^i$. For the sake of contradiction, assume there are $r+1$ bricks of $\mathbf{y}^*$ which contain a fractional coordinate and I is the index set of such bricks. Hence we have $\mathbf{e}^i, \lambda^i$ as above for each $i \in I$. We abuse the notation and treat $\mathcal{C}^i$ as a matrix whose columns are the configurations. Consider the vectors $E_1^i \mathcal{C}^i \lambda^i \mathbf{e}^i \in \mathbb{R}^r$: because there are $r+1$ of them, they are linearly dependent, and, by rescaling, there must be coefficients $\bar{\boldsymbol{\lambda}}$ such that $|\bar{\lambda}^i| \leq \lambda^i$ for each $i \in I$ and $\sum_{i \in I} E_1^i \mathcal{C}^i \bar{\lambda}^i \mathbf{e}^i = \mathbf{0}$. Define $\mathbf{e} \in \mathbb{R}^C$ (recall that C is the total number of configurations) such that its i-th brick is equal to $\bar{\lambda}^i \mathbf{e}^i$ if $i \in I$, and is $\mathbf{0}$ otherwise. Then $\mathbf{y}^* \pm \mathbf{e}$ are both feasible solutions of (ConfLP), and thus $\mathbf{y}^*$ is not a vertex solution—a contradiction.

So far, we have shown there are at most r fractional bricks of $\mathbf{y}^*$. Notice that all we needed for that was $r+1$ linearly dependent vectors which can be added to some brick in both directions while preserving feasibility. Because $\mathbf{e}^i \in \operatorname{Ker}_{\mathbb{Z}}(\mathbf{1})$ for each $i \in I$, we can decompose $\mathbf{e}^i$ into elements of $\mathcal{G}(\mathbf{1})$, which are exactly vectors with one 1 and one -1. Hence, to avoid the contradiction above, there can be at most r vectors $\mathbf{e}^i$, and, additionally, all of them must belong to $\mathcal{G}(\mathbf{1})$. Thus, the resulting vector $\mathbf{e}$ has support of size at most $2r$, and $\mathbf{y}^*$ has at most $2r$ fractional coordinates. □

Finding a conf-optimal solution with small number of fractional bricks.

Our goal is to show that the proximity of any conf-optimal solution $\mathbf{x}^*$ of (HugeCP) from an integer optimum $\mathbf{z}^*$ of (HugeIP) depends on the number of fractional bricks. This number, by definition of φ, depends on the number of fractional coordinates of the corresponding solution $\mathbf{y}$ of (ConfLP). The following lemma shows how to produce optima of (ConfLP) with small support. We emphasize that our proximity theorem does not require that the fractional solution be optimal but rather conf-optimal.

Lemma 6 *There is an algorithm that finds an optimal vertex solution* $\mathbf{y}^*$ *of* (ConfLP) *with* $|supp(\mathbf{y}^*)| \leq r + \tau$ *and at most* $2r$ *fractional coordinates, and a conf-optimal solution* $\mathbf{x}^* = \varphi(\mathbf{y}^*)$ *of* (HugeCP) *with at most* $2r$ *fractional bricks, in time* $g_1(E_2)^{\mathcal{O}(s)} \operatorname{poly}(rt\tau \log \|f_{\max}, \mathbf{l}, \mathbf{u}, \mathbf{b}, \boldsymbol{\mu}, E\|_\infty)$.

Proof The proof has three parts. First, we describe how to find an optimal basic solution of the dual of (ConfLP). Next, we identify $r + \tau$ inequalities of this dual which fully determine the optimal dual LP solution. Finally, we show how to use this information to solve (ConfLP) itself.

Recall that τ is the number of brick types in the huge N-fold instance. Since (ConfLP) has exponentially many variables, we take the standard approach and solve the dual LP of (ConfLP) by the ellipsoid method and the equivalence of optimization

and separation. The Dual LP of (ConfLP) in variables $\boldsymbol{\alpha} \in \mathbb{R}^r$, $\boldsymbol{\beta} \in \mathbb{R}^\tau$ is:

$$\begin{aligned} \max \quad & \mathbf{b}^0\boldsymbol{\alpha} + \sum_{i=1}^{\tau} \mu^i \beta^i \\ \text{s.t.} \quad & (\boldsymbol{\alpha} E_1^i)\mathbf{c} - f^i(\mathbf{c}) \le -\beta^i \qquad \forall i \in [\tau],\ \forall \mathbf{c} \in \mathcal{C}^i. \end{aligned} \tag{9}$$

To verify feasibility of $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, we need, for each $i \in [\tau]$, to maximize the left-hand side of (9) over all $\mathbf{c} \in \mathcal{C}^i$ and check if it is at most $-\beta^i$. This corresponds to finding integer variables $\mathbf{c}$ which for given $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ solve

$$\min\left(f^i(\mathbf{c}) - (\boldsymbol{\alpha} E_1^i)\mathbf{c}\right) = -\max\left((\boldsymbol{\alpha} E_1^i)\mathbf{c} - f^i(\mathbf{c})\right) \ :\ E_2^i\mathbf{c} = \mathbf{b}^i,\ \mathbf{l}^i \le \mathbf{c} \le \mathbf{u}^i, \mathbf{c} \in \mathbb{Z}^t.$$

This program can be solved in time $T''' \le g_1(E_2)^{\mathcal{O}(s)} t^3 \cdot \text{poly}(\log \|\mathbf{b}^i, \mathbf{l}^i, \mathbf{u}^i, \|E\|_\infty\|_\infty)$ [33, Theorem 4].

Grötschel et al. [18, Theorem 6.4.9] show that an optimal solution of an LP (even one which is a vertex [18, Remark 6.5.2]) can be found in a number of calls to a separation oracle which is polynomial in the dimension and the encoding length of the inequalities returned by a separation oracle. Clearly the inequalities (9) have encoding length bounded by $\log \|f_{\max}, \mathbf{l}, \mathbf{u}, \mathbf{b}, \boldsymbol{\mu}\|_\infty$ and thus $T = \text{poly}(rt\tau \log \|f_{\max}, \mathbf{l}, \mathbf{u}, \mathbf{b}, \boldsymbol{\mu}, E\|_\infty)$ calls to a separation oracle are sufficient to find an optimal vertex solution, which amounts to $T \cdot T'''$ arithmetic operations.

Next, we will identify $r + \tau$ inequalities determining the previously found optimal vertex solution of the dual of (ConfLP). Observe that the dimension of the dual LP is the number of rows of the primal LP, which is $r + \tau$. Since each point in $(r + \tau)$-dimensional space is fully determined by $r + \tau$ linearly independent inequalities, there must exist a subset I of $r + \tau$ inequalities among the T inequalities considered by the ellipsoid method which fully determines the dual optimum. We can find them as follows.

We initialize I to be the empty set. Taking the T considered inequalities one by one, we process the inequality if it is satisfied as equality by the given optimal basic solution for the dual LP, and we discard other inequalities. If we process the current inequality and either some inequality of I or the present inequality is dominated[2] by an inequality that can be obtained as a non-negative linear combination of the others, discard it; otherwise, include it in I and continue. Testing whether an inequality $\mathbf{dz} \le e'$ is dominated by a non-negative combination of a system of inequalities $D\mathbf{z} \le \mathbf{e}$ can be decided by solving

$$\min \boldsymbol{\alpha}\mathbf{e} \quad \text{s.t.} \quad \boldsymbol{\alpha}^\intercal D = \mathbf{d},\ \boldsymbol{\alpha} \ge \mathbf{0}, \tag{10}$$

and checking whether the optimal value is at most e'. If it is, then the solution $\boldsymbol{\alpha}$ encodes a non-negative linear combination of the inequalities $D\mathbf{z} \le \mathbf{e}$ which yields an inequality dominating $\mathbf{dz} \le e'$, and if it is not, then such a combination does

[2] An inequality $\mathbf{ax} \le b$ is dominated by $\mathbf{cx} \le d$ if for every $\mathbf{x}$ such that $\mathbf{cx} \le d$ we also have $\mathbf{ax} \le b$.

not exists. Thus, when a new inequality is considered, we solve (10) for at most $r + \tau$ inequalities (the new one and all less than $r + \tau$ already selected ones), and there are at most T inequalities considered. The time needed to solve (10) is $\mathrm{poly}(r + \tau, \log \|\mathbf{l}, \mathbf{u}, \mathbf{b}, f_{\max}, E\|_\infty)$ because its dimension is at most $r + \tau$ and its encoding length is at most $\log \|\mathbf{l}, \mathbf{u}, \mathbf{b}, f_{\max}, E\|_\infty$. Altogether, we need time

$$\begin{aligned} &T \cdot (r + \tau) \cdot \mathrm{poly}(r + \tau, \log \|\mathbf{l}, \mathbf{u}, \mathbf{b}, f_{\max}, E\|_\infty) \\ &\leq \mathrm{poly}(rt\tau \log \| f_{\max}, \mathbf{l}, \mathbf{u}, \mathbf{b}, \boldsymbol{\mu}, E\|_\infty) =: T'. \end{aligned}$$

Finally, let the *restricted* (ConfLP) be the (ConfLP) restricted to the variables corresponding to the inequalities in I. We claim that an optimal solution to the restricted (ConfLP) is also an optimal solution to (ConfLP). To see that, use LP duality: the optimal objective value of the dual LP restricted to inequalities in I is the same as one of the dual optima, and thus an optimal solution of the restricted (ConfLP) must be an optimal solution of (ConfLP). We solve the restricted (ConfLP) using any polynomial LP algorithm in time $T'' \leq \mathrm{poly}((r + \tau), \log \| f_{\max}, \mathbf{l}, \mathbf{u}, \boldsymbol{\mu}, \mathbf{b}^0, E\|_\infty)$. The resulting total time complexity is thus $T \cdot T''' + T'$ to construct the restricted (ConfLP) instance and time T'' to solve it, $T \cdot T''' + T' + T''$ total, which is upper bounded by $g_1(E_2)^{\mathcal{O}(s)} \mathrm{poly}(rt\tau \log \| f_{\max}, \mathbf{l}, \mathbf{u}, \mathbf{b}, \boldsymbol{\mu}, E\|_\infty)$, as claimed.

Let $\mathbf{y}^*$ be an optimum of (ConfLP) we have thus obtained. Since $|I| \leq r + \tau$, the support of $\mathbf{y}^*$ is of size at most $r + \tau$. By Lemma 5, $\mathbf{y}^*$ has at most $2r$ fractional coordinates. Now setting $\mathbf{x}^* = \varphi(\mathbf{y}^*)$ is enough, since we have already argued (see definition of φ) that $\mathbf{x}^*$ has at most as many fractional bricks as $\mathbf{y}^*$ has fractional coordinates and $\mathbf{x}^*$ can be computed from $\mathbf{y}^*$ in $\mathcal{O}(r + \tau)$ time. □

3.3 Proximity theorem

Let us give a plan for the next subsection. We wish to prove that for every conf-optimal solution $\mathbf{x}^*$ of (HugeCP) there is an integer solution $\mathbf{z}^*$ of (HugeIP) nearby. In the following, let $\mathbf{x}^*$ be a conf-optimal solution of (HugeCP) and $\mathbf{z}^*$ be an optimal solution of (HugeIP) minimizing $\|\mathbf{x}^* - \mathbf{z}^*\|_1$. A technique for proving proximity theorems which was introduced by Eisenbrand and Weismantel [11] works as follows. A vector $\mathbf{h} \in \mathbb{Z}^{Nt}$ is called a *cycle* of $\mathbf{x}^* - \mathbf{z}^*$ if $\mathbf{h} \neq \mathbf{0}$, $E^{(N)}\mathbf{h} = \mathbf{0}$, and $\mathbf{h} \sqsubseteq \mathbf{x}^* - \mathbf{z}^*$. It is not too difficult to see that if $\mathbf{x}'$ is an optimal (*not* necessarily conf-optimal) solution of (HugeCP) with the objective f, then there cannot exist a cycle of $\mathbf{x}' - \mathbf{z}^*$ (cf. proof of Lemma 9). Based on a certain decomposition of $\mathbf{x}' - \mathbf{z}^*$ into integer and fractional smaller dimensional vectors and by an application of the Steinitz Lemma, the existence of a cycle is proven unless $\|\mathbf{x}' - \mathbf{z}^*\|_1$ is roughly bounded by the number of fractional bricks of $\mathbf{x}'$. However, we cannot apply this technique directly as an optimal solution $\mathbf{x}'$ of (HugeCP) might have many fractional bricks. At the same time, an existence of a cycle $\mathbf{h}$ of $\mathbf{x}^* - \mathbf{z}^*$ does not necessarily contradict that $\|\mathbf{x}^* - \mathbf{z}^*\|_1$ is minimal, because $\mathbf{x}^* + \mathbf{h}$ might not be a configurable solution, which is an essential part of the argument.

All of this leads us to introduce a stronger notion of a cycle. We say that $\mathbf{h} \in \mathbb{Z}^{Nt}$ is a *configurable cycle* of $\mathbf{x}^* - \mathbf{z}^*$ (with respect to $\mathbf{x}^*$) if (**1**) $\mathbf{h}$ is a cycle of $\mathbf{x}^* - \mathbf{z}^*$, (**2**) for each brick $j \in [N]$ of type $i \in [\tau]$ there exists an $\hat{f}$-optimal decomposition Γ^j of

$(\mathbf{x}^*)^j$ such that we may write $\mathbf{h}^j = \sum_{(\mathbf{c},\lambda_\mathbf{c})\in\Gamma^j} \lambda_\mathbf{c}\mathbf{h}_\mathbf{c}$, and (**3**) for each $(\mathbf{c},\lambda_\mathbf{c}) \in \Gamma^j$ we have $\mathbf{h}_\mathbf{c} \sqsubseteq \mathbf{c} - (\mathbf{z}^*)^j$ and $\mathbf{h}_\mathbf{c} \in \mathrm{Ker}_\mathbb{Z}(E_2^i)$. Soon we will show that if $\|\mathbf{x}^* - \mathbf{z}^*\|_1$ is minimal, $\mathbf{x}^* - \mathbf{z}^*$ does not have a configurable cycle. The next task becomes to show how large must $\|\mathbf{x}^* - \mathbf{z}^*\|_1$ be in order for a configurable cycle to exist. Recall that the technique of Eisenbrand and Weismantel [11] can be used to rule out an existence of a (regular) cycle, not a configurable cycle. To overcome this, we "lift" both $\mathbf{x}^*$ and $\mathbf{z}^*$ to a higher-dimensional space and show that a cycle in this space corresponds to a configurable cycle in the original space. Only then are we ready to prove a proximity bound using the aforementioned technique.

Lemma 7 *If* $\mathbf{h}$ *is a configurable cycle of* $\mathbf{x}^* - \mathbf{z}^*$*, then* $\mathbf{x}^* - \mathbf{h}$ *is configurable.*

Proof Fix $j \in [N]$. Let $\mathbf{p}$ be the brick $(\mathbf{x}^* - \mathbf{h})^j$ and let $i \in [\tau]$ be its type. Now $\mathbf{p}$ can be written as $\mathbf{p} = \sum_{(\mathbf{c},\lambda_\mathbf{c})\in\Gamma^j} \lambda_\mathbf{c}(\mathbf{c} - \mathbf{h}_\mathbf{c})$. Furthermore, we have $E_2^i(\mathbf{c} - \mathbf{h}_\mathbf{c}) = E_2^i\mathbf{c} = \mathbf{b}^j$, and, by $\mathbf{h} \sqsubseteq \mathbf{x}^* - \mathbf{z}^*$, we also have $\mathbf{l} \le \mathbf{x}^* - \mathbf{h} \le \mathbf{u}$. □

We now need a technical lemma:

Lemma 8 *Let* $\mathbf{x}^*$ *be a conf-optimal solution of* (HugeCP)*, let* $\mathbf{z}^*$ *be an optimum of* (HugeIP)*, and let* $\mathbf{h}^*$ *be a configurable cycle of* $\mathbf{x}^* - \mathbf{z}^*$*. Then*

$$\hat{f}(\mathbf{z}^* + \mathbf{h}^*) + \hat{f}(\mathbf{x}^* - \mathbf{h}^*) \le \hat{f}(\mathbf{z}^*) + \hat{f}(\mathbf{x}^*) \ . \tag{11}$$

Proof We begin by a simple observation: let $g : \mathbb{R} \to \mathbb{R}$ be a convex function, $x \in \mathbb{R}$, $z \in \mathbb{Z}$, and $r \in \mathbb{Z}$ be such that $r \sqsubseteq x - z$ (that is, there is some ρ, $0 \le \rho \le 1$, such that $r = \rho \cdot (x - z)$). By convexity of g we have that

$$g(z + r) + g(x - r) \le g(z) + g(x) \ . \tag{12}$$

Fix $j \in [N]$ and $\mathbf{z} = (\mathbf{z}^*)^j$, $\mathbf{x} = (\mathbf{x}^*)^j$, $\mathbf{h} = (\mathbf{h}^*)^j$, and let i be the type of brick j. Since $\mathbf{h}^*$ is a configurable cycle there exists an $\hat{f}$-optimal decomposition Γ of $\mathbf{x}$ such that, for each $(\mathbf{c},\lambda_\mathbf{c}) \in \Gamma$, there exists a $\mathbf{h}_\mathbf{c} \sqsubseteq \mathbf{c} - \mathbf{z}$, $\mathbf{h}_\mathbf{c} \in \mathrm{Ker}_\mathbb{Z}(E_2^i)$, and $\mathbf{h} = \sum_{(\mathbf{c},\lambda_\mathbf{c})\in\Gamma} \lambda_\mathbf{c}\mathbf{h}_\mathbf{c}$. Due to separability of f we may apply (12) independently to each coordinate, obtaining for each $\mathbf{c}$

$$f^i(\mathbf{z} + \mathbf{h}_\mathbf{c}) + f^i(\mathbf{c} - \mathbf{h}_\mathbf{c}) \le f^i(\mathbf{z}) + f^i(\mathbf{c}) \ .$$

Since all arguments of f^i are integral, we immediately get

$$\hat{f}^i(\mathbf{z} + \mathbf{h}_\mathbf{c}) + \hat{f}^i(\mathbf{c} - \mathbf{h}_\mathbf{c}) \le \hat{f}^i(\mathbf{z}) + \hat{f}^i(\mathbf{c}) \ .$$

Aggregating according to Γ, we get (recall that we have $\sum_{(\mathbf{c},\lambda_\mathbf{c})\in\Gamma} \lambda_\mathbf{c} = 1$)

$$\begin{aligned}\sum_{(\mathbf{c},\lambda_\mathbf{c})\in\Gamma} \lambda_\mathbf{c}\left(\hat{f}^i(\mathbf{z} + \mathbf{h}_\mathbf{c}) + \hat{f}^i(\mathbf{c} - \mathbf{h}_\mathbf{c})\right) &\le \sum_{(\mathbf{c},\lambda_\mathbf{c})\in\Gamma} \lambda_\mathbf{c}\left(\hat{f}^i(\mathbf{z}) + \hat{f}^i(\mathbf{c})\right) \\ &= \hat{f}^i(\mathbf{z}) + \sum_{(\mathbf{c},\lambda_\mathbf{c})\in\Gamma} \lambda_\mathbf{c}\hat{f}^i(\mathbf{c}),\end{aligned}$$

where by $\hat{f}$-optimality of Γ the right-hand side is equal to $\hat{f}^i(\mathbf{z}) + \hat{f}^i(\mathbf{x})$. As for the left-hand side, observe that decompositions $\Gamma' = \{(\mathbf{z} + \mathbf{h}_\mathbf{c}, \lambda_\mathbf{c}) \mid (\mathbf{c}, \lambda_\mathbf{c}) \in \Gamma\}$ and $\Gamma'' = \{(\mathbf{c} - \mathbf{h}_\mathbf{c}, \lambda_\mathbf{c}) \mid (\mathbf{c}, \lambda_\mathbf{c}) \in \Gamma\}$ satisfy $\sum \Gamma' = \mathbf{z} + \mathbf{h}$ and $\sum \Gamma'' = \mathbf{x} - \mathbf{h}$ but are only feasible (not necessarily optimal) solutions of (6). Thus, we have

$$\hat{f}^i(\mathbf{z} + \mathbf{h}) + \hat{f}^i(\mathbf{x} - \mathbf{h}) \leq \sum_{(\mathbf{c}, \lambda_\mathbf{c}) \in \Gamma} \lambda_\mathbf{c} \left(\hat{f}^i(\mathbf{z} + \mathbf{h}_\mathbf{c}) + \hat{f}^i(\mathbf{c} - \mathbf{h}_\mathbf{c}) \right) .$$

Combining over Γ then yields

$$\hat{f}^i(\mathbf{z} + \mathbf{h}) + \hat{f}^i(\mathbf{x} - \mathbf{h}) \leq \hat{f}^i(\mathbf{z}) + \hat{f}^i(\mathbf{x}),$$

and since we have proven this claim for every brick j, aggregation over bricks concludes the proof of the main claim (11). □

Let us show that if $\mathbf{x}^*$ and $\mathbf{z}^*$ are as stated, then there is no configurable cycle of $\mathbf{x}^* - \mathbf{z}^*$.

Lemma 9 *Let $\mathbf{x}^*$ be a conf-optimal solution of* (HugeCP) *and let $\mathbf{z}^*$ be an optimal solution of* (HugeIP) *such that $\|\mathbf{x}^* - \mathbf{z}^*\|_1$ is minimal. Then there is no configurable cycle of $\mathbf{x}^* - \mathbf{z}^*$.*

Proof For the sake of contradiction, suppose that there exists a configurable cycle $\mathbf{h}^*$ of $\mathbf{x}^* - \mathbf{z}^*$. By Lemma 8, one of two cases must occur:

Case 1: $\hat{f}(\mathbf{z}^* + \mathbf{h}^*) \leq \hat{f}(\mathbf{z}^*)$. Then $\mathbf{z}^* + \mathbf{h}^*$ is an optimal integer solution (by $\mathbf{h} \sqsubseteq \mathbf{x}^* - \mathbf{z}^*$ we have $\mathbf{l} \leq \mathbf{z}^* + \mathbf{h} \leq \mathbf{u}$ and by $\mathbf{h}^* \in \ker_\mathbb{Z}\left(E^{(N)}\right)$ we have $E^{(N)}(\mathbf{z}^* + \mathbf{h}) = \mathbf{b}$) which is closer to $\mathbf{x}^*$, a contradiction to minimality of $\|\mathbf{x}^* - \mathbf{z}^*\|_1$.

Case 2: $\hat{f}(\mathbf{x}^* - \mathbf{h}^*) < \hat{f}(\mathbf{x}^*)$. Since $\mathbf{h}^*$ is a configurable cycle, Lemma 7 states that $\mathbf{x}^* - \mathbf{h}^*$ is configurable, so we have a contradiction with conf-optimality of $\mathbf{x}^*$. □

Overview of the remainder of the proof

In order to use existing proximity arguments to bound the norm of a cycle, our plan is to move into an extended (higher-dimensional) space which corresponds to decomposing each brick $\mathbf{x}^i$ of $\mathbf{x}^*$ into configurations as $\mathbf{x}^i = \sum_\mathbf{c} \lambda_\mathbf{c} \mathbf{c}$ – each summand becomes a new brick in the extended space.

We denote this new higher-dimensional representation of $\mathbf{x}^*$ with respect to Γ as $\uparrow\mathbf{x}^*$ and call it the *rise of* $\mathbf{x}^*$, and define similarly the rise of $\mathbf{z}^*$ (with respect to a given decomposition of each brick of $\mathbf{x}^*$). The situation gets very delicate at this point.

First, we require that each decomposition of a brick of $\mathbf{x}^*$ is optimal with respect to the auxiliary objective $\hat{f}$ so that we can use the argument about non-existence of a cycle. Second, because the proximity bound depends on the number of fractional bricks of $\uparrow\mathbf{x}^*$, we require that the decomposition of each brick is small, i.e., into only few elements. Third, we require that each coefficient $\lambda_\mathbf{c}$ is of the form $1/q_\mathbf{c}$ for an integer $q_\mathbf{c}$, because we need to ensure that, for a corresponding cycle brick $\mathbf{h}_\mathbf{c}$, $\lambda_\mathbf{c}^{-1}\mathbf{h}_\mathbf{c}$

is an integer vector, so $\lambda_{\mathbf{c}}^{-1}$ has to be an integer. To ensure the second and third condition simultaneously, we first show that there is a decomposition of each brick of size at most $t+1$ and with each coefficient bounded by P, and then show that each fraction p/q can be written as an *Egyptian fraction* $p/q = 1/a_1 + 1/a_2 + \cdots 1/a_{\mathfrak{c}}$ with $\mathfrak{c} \leq 2\log_2 q$ (Lemmas 10–12). (Bounds on the length of Egyptian fractions have been studied in the past and our bound is not the best possible, but in order to use our proximity theorem, we need exact and not merely asymptotic bounds, so we prove this worse but exact bound of $2\log_2 q$.) We call a decomposition of a brick satisfying all three criteria given above a *small scalable decomposition*.

Fix a small scalable decomposition for each brick of $\mathbf{x}^*$, and let $\uparrow\mathbf{x}^*$ be the rise of $\mathbf{x}^*$ with respect to this decomposition. Since this decomposition is small, $\uparrow\mathbf{x}^*$ has at most $\text{poly}(\|E\|_\infty, r, s)$ fractional bricks. Moreover, the other properties above allow us to say the following: if $\mathbf{r}$ is a cycle of $\uparrow\mathbf{x}^* - \uparrow\mathbf{z}^*$, then the compression of $\mathbf{r}$ back to the original space is a configurable cycle of $\mathbf{x}^* - \mathbf{z}^*$ (Lemma 14). So in order to bound $\|\mathbf{x}^* - \mathbf{z}^*\|_1$, it suffices (by triangle inequality) to bound $\|\uparrow\mathbf{x}^* - \uparrow\mathbf{z}^*\|_1$. We do this by adapting the approach of Eisenbrand and Weismantel [11] to bound the length of any cycle $\mathbf{r}$ of $\uparrow\mathbf{x}^* - \uparrow\mathbf{z}^*$.

The remainder of the proof

We say that $|\Gamma|$ is the *size* of the decomposition. Let us show that for each brick, there exists an $\hat{f}$-optimal decomposition whose coefficients have small encoding length, and its size is small. For any matrix A, define $g_\infty(A) = \max_{\mathbf{g}\in\mathcal{G}(A)} \|\mathbf{g}\|_\infty$.

Lemma 10 *Each brick of* $\mathbf{x}^*$ *of type* i *has an* $\hat{f}$*-optimal decomposition* Γ

1. *of size at most* $t+1$, *and*
2. $\max_{(\mathbf{c},\lambda_{\mathbf{c}}=p_{\mathbf{c}}/q_{\mathbf{c}})\in\Gamma}\{p_{\mathbf{c}}, q_{\mathbf{c}}\} \leq (t+1)!((2t-2)g_\infty(E_2^i))^{t+1} \leq (t+1)^{(t+1)}(g_1(E_2))^{(t+2)} \leq (t+1)^{(t+1)}(s\|E_2^i\|_\infty+1)^{(s+1)(t+2)}$.

Proof An $\hat{f}$-optimal decomposition corresponds to a solution of the LP (6). We will argue that there is a solution whose support is composed of columns which do not differ by much, which corresponds to a solution of an LP with small coefficients, and the claimed bound can then be obtained by Cramer's rule.

Specifically, we claim that there exists an $\hat{f}$-optimal decomposition Γ which corresponds to an optimal solution $\boldsymbol{\lambda}$ of (6) such that there exists a point $\boldsymbol{\zeta} \in \mathbb{Z}^t$ and if $\mathbf{c} \in \text{supp}(\boldsymbol{\lambda})$, then $\|\mathbf{c} - \boldsymbol{\zeta}\|_\infty \leq (t-1)g_\infty(E_2^i)$. For a solution $\boldsymbol{\lambda}$ of (6), define $R' := \max_{\mathbf{c},\mathbf{c}'\in\text{supp}(\boldsymbol{\lambda})} \|\mathbf{c} - \mathbf{c}'\|_\infty$ to be the longest side of the bounding box of all $\mathbf{c} \in \text{supp}(\boldsymbol{\lambda})$. For a point $\boldsymbol{\zeta} \in \mathbb{Z}^t$, say, for $\mathbf{c} \in \text{supp}(\boldsymbol{\lambda})$, that a coordinate $j \in [t]$ is *tight* if $c_j = \zeta_j - \lceil \frac{R'}{2} \rceil$ or $c_j = \zeta_j + \lceil \frac{R'}{2} \rceil$, and define $S = \sum_{\mathbf{c}\in\text{supp}(\boldsymbol{\lambda})} \sum_{j=1}^t \lambda_{\mathbf{c}}[j \text{is tight in } \mathbf{c}]$ (where "$[X]$" is an indicator of the statement X) to be the weighted number of tight coordinates. Now let $\boldsymbol{\zeta} \in \mathbb{Z}^t$ be any point which is an integer center of the bounding box (i.e., $\|\mathbf{c} - \boldsymbol{\zeta}\|_\infty \leq \lceil \frac{R'}{2} \rceil$ for all $\mathbf{c} \in \text{supp}(\boldsymbol{\lambda})$) and which minimizes S. For contradiction assume that $\boldsymbol{\lambda}$ is an optimal solution of (6) which minimizes R' and S (lexicographically in this order) and $R' > (2t-2)g_\infty(E_2^i)$. Assuming Γ is a decomposition of a brick of type i, we have $\mathbf{c}, \mathbf{c}' \in \mathcal{C}^i = \{\tilde{\mathbf{c}} \in \mathbb{Z}^t \mid E_2^i\tilde{\mathbf{c}} = \mathbf{b}^i,\ \mathbf{l}^i \leq \tilde{\mathbf{c}} \leq \mathbf{u}^i\}$

and thus $\mathbf{c}-\mathbf{c}' \in \mathrm{Ker}_{\mathbb{Z}}(E_2^i)$. By Proposition 1 we may write $\mathbf{c}-\mathbf{c}' = \sum_{j=1}^{2t-2} \gamma_j \mathbf{g}_j$ with $\mathbf{g}_j \in \mathcal{G}(E_2^i)$ and $\mathbf{g}_j \sqsubseteq \mathbf{c}-\mathbf{c}'$ for all $j \in [2t-2]$. Note that because $\|\mathbf{c}-\mathbf{c}'\|_\infty > R := (2t-2)g_\infty(E_2^i)$, we have that there exists $j \in [2t-2]$ such that $\gamma_j > 1$. Hence $\mathbf{g} := \sum_{j=1}^{2t-2} \lfloor \frac{\gamma_j}{2} \rfloor \mathbf{g}_j$ satisfies $\mathbf{g} \neq \mathbf{0}$. Let $\bar{\mathbf{c}} := \mathbf{c}-\mathbf{g}$, and $\bar{\mathbf{c}}' := \mathbf{c}'+\mathbf{g}$.

First, because $\bar{\mathbf{c}}-\bar{\mathbf{c}}' = (\mathbf{c}-\mathbf{c}') - 2\mathbf{g} = \sum_{j=1}^{2t-2}(\gamma_j - 2\lfloor \frac{\gamma_j}{2} \rfloor)\mathbf{g}_i$, we may bound $\|\bar{\mathbf{c}}-\bar{\mathbf{c}}'\|_\infty \leq (2t-2)g_\infty(E_2^i) = R$. Second, by the conformality of the decomposition, $\bar{\mathbf{c}}, \bar{\mathbf{c}}' \in \mathcal{C}^i$. Third, by separable convex superadditivity (Proposition 2), we have that $f(\mathbf{c}) + f(\mathbf{c}') \geq f(\bar{\mathbf{c}}) + f(\bar{\mathbf{c}}')$. Fourth, there exist a coordinate $j \in [t]$ such that $|c_j - c'_j| = R'$ but, since $\|\bar{\mathbf{c}}-\bar{\mathbf{c}}'\|_\infty \leq R$, $|\bar{c}_j - \bar{c}'_j| \leq R < R'$ and thus j is no longer a tight coordinate for either $\bar{\mathbf{c}}$ or $\bar{\mathbf{c}}'$ (or both), and no new tight coordinates can be introduced because $R < R'$. Without loss of generality, let $\lambda_{\mathbf{c}} \leq \lambda_{\mathbf{c}'}$. Now initialize $\boldsymbol{\lambda}' := \boldsymbol{\lambda}$ and modify it by setting $\lambda'_{\bar{\mathbf{c}}}, \lambda'_{\bar{\mathbf{c}}'} := \lambda_{\mathbf{c}}, \lambda'_{\mathbf{c}} := 0, \lambda'_{\mathbf{c}'} := \lambda_{\mathbf{c}'} - \lambda_{\mathbf{c}}$. By our arguments above, $\boldsymbol{\lambda}'$ is another optimal solution of (6) but the weighted number S of tight coordinates has decreased by the fourth point, a contradiction.

Thus, there exists a point $\boldsymbol{\zeta} \in \mathbb{Z}^t$ and an optimal solution $\boldsymbol{\lambda}$ of (6) such that for each $\mathbf{c} \in \mathrm{supp}(\boldsymbol{\lambda})$, it holds that $\|\mathbf{c}-\boldsymbol{\zeta}\|_\infty \leq R/2 = (t-1)g_\infty(E_2^i)$. We obtain the following reduced LP from (6) by deleting all columns $\mathbf{c}$ with $\|\mathbf{c}-\boldsymbol{\zeta}\|_\infty > R/2$, and denote the remaining set of columns by $\bar{\mathcal{C}}^i$:

$$\min \sum_{\mathbf{c} \in \bar{\mathcal{C}}^i} \lambda_{\mathbf{c}} f^i(\mathbf{c}) \quad \text{s.t.} \quad \sum_{\mathbf{c} \in \bar{\mathcal{C}}^i} \lambda_{\mathbf{c}} \mathbf{c} = \mathbf{x}^j,\ \|\boldsymbol{\lambda}\|_1 = 1,\ \boldsymbol{\lambda} \geq \mathbf{0}\ . \tag{13}$$

This LP is equivalent to one obtained by subtracting $\boldsymbol{\zeta}$ from all columns and the right hand side:

$$\min \sum_{\mathbf{c} \in \bar{\mathcal{C}}^i} \lambda_{\mathbf{c}} f^i(\mathbf{c}) \quad \text{s.t.} \quad \sum_{\mathbf{c} \in \bar{\mathcal{C}}^i} \lambda_{\mathbf{c}} (\mathbf{c}-\boldsymbol{\zeta}) = (\mathbf{x}^j - \boldsymbol{\zeta}),\ \|\boldsymbol{\lambda}\|_1 = 1,\ \boldsymbol{\lambda} \geq \mathbf{0}\ . \tag{14}$$

Now, this LP has $t+1$ rows and its columns have the largest coefficient bounded by $R/2$ in absolute value. A basic solution $\boldsymbol{\lambda}$ has $|\mathrm{supp}(\boldsymbol{\lambda})| \leq t+1$ and, by Cramer's rule, the denominator of each $\lambda_{\mathbf{c}}$ is bounded by $(t+1)!$ times the largest coefficient to the power of $t+1$, thus bounded by

$$(t+1)!(R/2)^{(t+1)} \leq (t+1)!((t-1)g_\infty(E_2^i))^{(t+1)} \leq (t+1)^{(t+1)}(g_1(E_2))^{(t+2)}.$$

In the worst case, we can bound this as

$$(t+1)^{(t+1)}(s\|E_2^i\|_\infty + 1)^{s(t+2)},$$

where we use

$$g_\infty(E_2^i) \leq \|E_2^i\|_\infty (2s\|E_2^i\|_\infty + 1)^s$$

[12, Lemma 2]. □

Next, we will need the notion of an Egyptian fraction. For a rational number p/q, $p, q \in \mathbb{N}$, its *Egyptian fraction* is a finite sum of distinct unit fractions such that

$$\frac{p}{q} = \frac{1}{q_1} + \frac{1}{q_2} + \cdots + \frac{1}{q_k},$$

for $q_1, \ldots, q_k \in \mathbb{N}$ distinct. Call the number of terms k the *length* of the Egyptian fraction. Vose [43] has proven that any p/q has an Egyptian fraction of length $\mathcal{O}(\sqrt{\log q})$. Since our algorithm requires an exact bound, we present the following weaker yet exact result:

Lemma 11 (Egyptian Fractions) *Let $p, q \in \mathbb{N}$, $1 \leq p < q$. Then p/q has an Egyptian fraction of length at most $2(\log_2 q) + 1$ and all denominators are at most q^2.*

Proof Let $a = 2^k$ be largest such that $a < q$, so $k = \lceil (\log_2 q) - 1 \rceil < \log_2 q$. Write $ap = bq + r$, $0 \leq r < q$. Note that $p < q \implies b < a$ and $q \leq 2a \implies r < 2a$. Now let $(b_{k-1}, \ldots, b_1, b_0)$ be the binary representation of $b < a$ so $b = \sum_{i=0}^{k-1} 2^i b_i$ and $e(r_k, \ldots, r_1, r_0)$ be that of $r < 2a$ so $r = \sum_{i=0}^{k} r_i 2^i$. Then we have

$$\frac{p}{q} = \frac{ap}{aq} = \frac{bq + r}{aq} = \frac{b}{a} + \frac{1}{q}\frac{r}{a} = \sum_{i=0}^{k-1} \frac{b_i}{2^{k-i}} + \sum_{i=0}^{k} \frac{r_i}{q \cdot 2^{k-i}},$$

where $b_i, r_i \in \{0, 1\}$, so a sum of at most $2k + 1 \leq 2(\log_2 q) + 1$ terms with all denominators $d_i \leq q2^k = qa \leq q^2$. Moreover, all denominators in the first sum are distinct and at most 2^k, and all in the second sum are distinct and at least $q > 2^k$, hence all distinct, so this is an Egyptian fraction of p/q of length $2(\log_2 q) + 1$ and denominators are at most q^2. □

Recall that our goal is to obtain a configurable cycle. However, for that we also need a special form of a decomposition. Say that Γ is a *scalable decomposition* of a brick $(\mathbf{x}^*)^j$ of type i if it is a $\hat{f}$-optimal decomposition, and for each $(\mathbf{c}_\gamma, \lambda_\gamma) \in \Gamma$, λ_γ is of the form $1/q_\gamma$ for some $q_\gamma \in \mathbb{N}$. We note that in what follows we do not need an algorithm computing a scalable decomposition, only the following existence statement.

Lemma 12 *Each brick of $\mathbf{x}^*$ has a scalable decomposition of size at most $\kappa_1 \cdot t^3 \log(t\|E_2\|_\infty)$, where $\kappa_1 = 52$.*

Proof Fix $j \in [N]$. Let $\mathbf{x} = (\mathbf{x}^*)^j$ be a brick of $\mathbf{x}^*$ of type i. By Lemma 10, there exists an $\hat{f}$-optimal decomposition of $\mathbf{x}$ of size $t + 1$ where each coefficient $\lambda_\mathbf{c} = p_\mathbf{c}/q_\mathbf{c}$ satisfies $p_\mathbf{c}, q_\mathbf{c} \leq (t+1)^{(t+1)}(s\|E_2^i\|_\infty + 1)^{(s+1)(t+2)}$. For each $\mathbf{c}$ in the decomposition now express $\lambda_\mathbf{c}$ as an Egyptian fraction:

$$\lambda_\mathbf{c} = \frac{p_\mathbf{c}}{q_\mathbf{c}} = \frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_\mathfrak{e}}\ .$$

By Lemma 11,

$$\begin{aligned}\mathfrak{e} &\le 2(\log_2 q_{\mathbf{c}}) + 1 = 2\left(\log\left((t+1)^{(t+1)}(s\|E_2^i\|_\infty + 1)^{(s+1)(t+2)}\right)\right) + 1\\ &\le 25st\log(st\|E_2^i\|_\infty)\ .\end{aligned}$$

Thus, the resulting decomposition is of size at most $(t+1)25st\log(st\|E_2^i\|_\infty) \le 2\cdot 26t^3\log(t\|E_2^i\|_\infty)$ (by $s \le t$ this justifies the deletion of s in the log() at the cost of a factor of 2, so the last bound holds) and is scalable, since each coefficient is of the form $1/q_\gamma$ for some $q_\gamma \in \mathbb{N}$. □

We will now show that we are guaranteed a configurable cycle of $\mathbf{x}^* - \mathbf{z}^*$ if there exists an analogue of a regular cycle of a certain "lifting" of $\mathbf{x}^*$ and $\mathbf{z}^*$.

Fix for each brick of $\mathbf{x}^*$ a scalable decomposition Γ^j. Let $\uparrow\mathbf{x}^*$ be the *rise of* $\mathbf{x}^*$ defined as a vector obtained from $\mathbf{x}^*$ by keeping every integer brick $(\mathbf{x}^*)^j$, and replacing every fractional brick $(\mathbf{x}^*)^j$ with $|\Gamma^j|$ terms $\lambda_\gamma\mathbf{c}_\gamma$, one for each $(\mathbf{c}_\gamma,\lambda_\gamma) \in \Gamma^j$. Observe that each brick of $\uparrow\mathbf{x}^*$ is of the form $\lambda_{\mathbf{c}}\mathbf{c}$ for some configuration $\mathbf{c}$ and some coefficient $0 \le \lambda_{\mathbf{c}} \le 1$. Thus, for a brick $\lambda_{\mathbf{c}}\mathbf{c}$ we say that $\mathbf{c}$ is its configuration, $\lambda_{\mathbf{c}}$ is its coefficient, and its type is identical to the type of brick it originated from; in particular, bricks which originated from an integer brick $\mathbf{p} = (\mathbf{x}^*)^j$ are of the form $\lambda_{\mathbf{p}}\mathbf{p}$ with $\lambda_{\mathbf{p}} = 1$. Let N' be the number of bricks of $\uparrow\mathbf{x}^*$ and define a mapping $\nu : [N'] \to [N]$ such that if a brick $j \in [N']$ of $\uparrow\mathbf{x}^*$ was defined from brick $\ell \in [N]$ of $\mathbf{x}^*$, then $\nu(j) = \ell$. The natural inverse ν^{-1} is defined such that, for $\ell \in [N]$, $\nu^{-1}(\ell)$ is the set of bricks of $\uparrow\mathbf{x}^*$ which originated from $(\mathbf{x}^*)^\ell$.

Lemma 13 *The vector* $\uparrow\mathbf{x}^*$ *has at most* $\kappa_2\cdot r\cdot t^3\log(t\|E_2^1,\ldots,E_2^\tau\|_\infty)$ *fractional bricks, where* $\kappa_2 = 2\kappa_1$.

Proof By Lemma 6 there is a conf-optimal $\mathbf{x}^*$ with at most $2r$ fractional bricks. By Lemma 12 for each fractional brick of $\mathbf{x}^*$ of type i there is a scalable decomposition of size at most $\kappa_1\cdot t^3\log(t\|E_2^i\|_\infty) \le \kappa_1\cdot t^3\log(t\|E_2^1,\ldots,E_2^\tau\|_\infty)$. Thus, $\uparrow\mathbf{x}^*$ has at most $\kappa_1\cdot t^3\log(t\|E_2^1,\ldots,E_2^\tau\|_\infty)$ fractional bricks for each fractional brick of $\mathbf{x}^*$, of which there are at most $2r$, totaling $2\kappa_1\cdot r\cdot t^3\log(t\|E_2^1,\ldots,E_2^\tau\|_\infty)$ fractional bricks. □

Denote by $\uparrow\mathbf{z}^* \in \mathbb{R}^{N't}$ the *rise of* $\mathbf{z}^*$ (with respect to $\mathbf{x}^*$) defined as follows. Let $j \in [N']$, $\ell = \nu(j)$, and λ be the coefficient of the j-th brick of $\uparrow\mathbf{x}^*$. Then the j-th brick of $\uparrow\mathbf{z}^*$ is $(\uparrow\mathbf{z}^*)^j := \lambda(\mathbf{z}^*)^\ell$. Observe that $\|\uparrow\mathbf{x}^* - \uparrow\mathbf{z}^*\|_1 \ge \|\mathbf{x}^* - \mathbf{z}^*\|_1$ by applying triangle inequality to each brick and its decomposition individually and aggregating.

For any vector $\mathbf{x} \in \mathbb{R}^{N't}$, define the *fall of* $\mathbf{x}$ as a vector $\downarrow\mathbf{x} \in \mathbb{R}^{Nt}$ such that for $\ell \in [N]$, $(\downarrow\mathbf{x})^\ell = \sum_{j\in\nu^{-1}(\ell)}\mathbf{x}^j$. We see that $\downarrow(\uparrow\mathbf{x}^*) = \mathbf{x}^*$ and $\downarrow(\uparrow\mathbf{z}^*) = \mathbf{z}^*$. Say that $\mathbf{r}$ is a *cycle of* $\uparrow\mathbf{x}^* - \uparrow\mathbf{z}^*$ if $\mathbf{r} \sqsubseteq \uparrow\mathbf{x}^* - \uparrow\mathbf{z}^*$ and $\mathbf{r} \in \mathrm{Ker}_{\mathbb{Z}}(E^{(N')})$.[3]

Lemma 14 *If* $\mathbf{r}$ *is a cycle of* $\uparrow\mathbf{x}^* - \uparrow\mathbf{z}^*$, *then* $\downarrow\mathbf{r}$ *is a configurable cycle of* $\mathbf{x}^* - \mathbf{z}^*$.

[3] Recall that $E^{(N')}$ is the N'-fold matrix formed from blocks E, see Eq. (1).

Proof To show that $\downarrow\mathbf{r}$ is a configurable cycle, we need to show that **(1)** $\downarrow\mathbf{r} \in \mathrm{Ker}_{\mathbb{Z}}(E^{(N)})$ and, **(2)** for each brick $\mathbf{x} = (\mathbf{x}^*)^j$ of $\mathbf{x}^*$, there is an $\hat{f}$-optimal decomposition of $\mathbf{x}$ such that $\mathbf{h} = (\downarrow\mathbf{r})^j$ decomposes accordingly. For the first part, $\downarrow\mathbf{r}$ is integral because it is obtained by summing bricks of $\mathbf{r}$, which is integral. Denote by $i(j)$ the type of a brick j (we abuse this notation; note that $i(j)$ for $j \in [N]$ may differ from $i(j)$ for $j \in [N']$, but context always makes clear what we mean). By the fact that $\mathbf{r} \in \mathrm{Ker}_{\mathbb{Z}}(E^{(N')})$ and the definition of $\downarrow\mathbf{r}$, we have $\mathbf{0} = \sum_{j=1}^{N'} E_1^{i(j)}\mathbf{r}^j = \sum_{j=1}^{N} E_1^{i(j)}(\downarrow\mathbf{r})^j$, and, for each $\ell \in [N]$, $\mathbf{0} = \sum_{j\in\nu^{-1}(\ell)} E_2^{i(j)}\mathbf{r}^j = E_2^{i(\ell)}(\downarrow\mathbf{r})^\ell$, thus $\downarrow\mathbf{r} \in \mathrm{Ker}_{\mathbb{Z}}(E^{(N)})$.

To see the second part, fix a brick $j \in [N]$ of type i and let $\mathbf{x} = (\mathbf{x}^*)^j$, $\mathbf{z} = (\mathbf{z}^*)^j$ and $\mathbf{h} = (\downarrow\mathbf{r})^j$. We need to show that $\mathbf{h} = \sum_{\gamma\in\nu^{-1}(j)} \mathbf{h}_\gamma$ can be written as $\sum_{\mathbf{c}\in\mathcal{C}^i} \lambda_{\mathbf{c}}\mathbf{h}_{\mathbf{c}}$ with $\mathbf{h}_{\mathbf{c}} \sqsubseteq \mathbf{c} - \mathbf{z}$ and $\mathbf{h}_{\mathbf{c}} \in \mathrm{Ker}_{\mathbb{Z}}(E_2^i)$. By definition of $\uparrow\mathbf{x}^*$ and $\mathbf{r}$, there is a scalable decomposition Γ of $\mathbf{x}$ (namely the one used to define $\uparrow\mathbf{x}^*$) such that for each $\gamma \in \nu^{-1}(j)$, $\mathbf{h}_\gamma \sqsubseteq \lambda_\gamma(\mathbf{c}_\gamma - \mathbf{z})$ and $\mathbf{h}_\gamma \in \mathrm{Ker}_{\mathbb{Z}}(E_2^i)$. Thus we may write $\mathbf{h} = \sum_{\gamma\in\nu^{-1}(j)} \lambda_\gamma \cdot (\lambda_\gamma^{-1}\mathbf{h}_\gamma)$ with $\lambda_\gamma^{-1}\mathbf{h}_\gamma \sqsubseteq \mathbf{c}_\gamma - \mathbf{z}$ and $\lambda_\gamma^{-1}\mathbf{h}_\gamma$ integral by the fact that $\lambda_\gamma = 1/q_\gamma$ with $q_\gamma \in \mathbb{N}$, concluding the proof. □

We are finally ready to use the Steinitz Lemma to derive a bound on $\|\mathbf{x}^* - \mathbf{z}^*\|_1$.

Theorem 2 *Let $\mathbf{x}^*$ be a conf-optimal solution of* (HugeCP) *with at most $2r$ fractional bricks. Then there exists an optimal solution $\mathbf{z}^*$ of* (HugeIP) *such that*

$$\begin{aligned}\|\mathbf{z}^* - \mathbf{x}^*\|_1 &\le \left(\kappa_2 t^4 \log(t\|E_2^1, \ldots, E_2^\tau\|_\infty)\right)(2r\|E_1\|_\infty g_1(E_2)))^{r+2}\\ &\le \left(\kappa_2 t^4 \log(t\|E_2^1, \ldots, E_2^\tau\|_\infty)\right)(2r)^{r+2}(\|E\|_\infty s)^{3rs}\ .\end{aligned}$$

Proof Denote by $\bar{E}_1$ the first r rows of the matrix $E^{(N)}$. Let $\mathbf{z}^*$ be an optimal integer solution such that $\|\mathbf{z}^* - \mathbf{x}^*\|_1$ is minimal, let $\uparrow\mathbf{x}^*$ be the rise of $\mathbf{x}^*$ with at most $\kappa_2 \cdot r \cdot t^3 \log(t\|E_2^1, \ldots, E_2^\tau\|_\infty)$ fractional bricks (see Lemma 13), let $\uparrow\mathbf{z}^*$ be the rise of $\mathbf{z}^*$ with respect to $\mathbf{x}^*$, and let $\mathbf{q} = \uparrow\mathbf{x}^* - \uparrow\mathbf{z}^*$.

We want to get into the setting of the Steinitz Lemma, that is, to obtain a sequence of vectors with small ℓ_1-norm and summing up to zero. To this end, we shall decompose $\bar{E}_1\mathbf{q}$ in the following way; we stress that we have $\bar{E}_1\mathbf{q} = \mathbf{0}$. For every integral brick $\mathbf{q}^i$ of type $\ell \in [\tau]$ we have its decomposition $\mathbf{q}^i = \sum_j \mathbf{g}_j^i$ into elements of $\mathcal{G}(E_2^\ell)$ by the Positive Sum Property (Proposition 1); for each $\mathbf{g}_j^i$ append $E_1^\ell\mathbf{g}_j^i$ into the sequence. For every fractional brick $\mathbf{q}^i$ of type $\ell \in [\tau]$ we have its decomposition $\mathbf{q}^i = \sum_{j=1}^t \alpha_j\mathbf{g}_j^i$, $\alpha_j \ge 0$ for each j, into elements of $\mathcal{C}(E_2^\ell)$; for each $\mathbf{g}_j^i$ append $\lfloor\alpha_j\rfloor$ copies of $E_1^\ell\mathbf{g}_j^i$ into the sequence, and finally append $E_1^\ell\{\alpha_j\}\mathbf{g}_j^i$. Observe that since $\uparrow\mathbf{x}^*$ has at most $\kappa_2 \cdot r \cdot t^3 \log(t\|E_2^1, \ldots, E_2^\tau\|_\infty)$ fractional bricks (Lemma 13), so does $\mathbf{q}$, and thus we have appended $\mathfrak{f} \le t \cdot \kappa_2 \cdot r \cdot t^3 \log(t\|E_2^1, \ldots, E_2^\tau\|_\infty) \le \kappa_2 \cdot r \cdot t^4 \log(t\|E_2^1, \ldots, E_2^\tau\|_\infty)$ fractional vectors into the sequence. Now we have a sequence

$$\mathbf{o}_1, \ldots, \mathbf{o}_m, \mathbf{p}_{m+1}, \ldots, \mathbf{p}_{m+\mathfrak{f}} \tag{15}$$

with m integer vectors $\mathbf{o}_1, \ldots, \mathbf{o}_m$ and $\mathfrak{f}$ fractional vectors $\mathbf{p}_{m+1}, \ldots, \mathbf{p}_{m+\mathfrak{f}}$. Moreover, since, for each $i \in [\tau]$, $\mathcal{C}(E_2^i) \subseteq \mathcal{G}(E_2^i)$,

each vector has ℓ_∞-norm of $\|E_1^1, \ldots, E_1^\tau\|_\infty \cdot g_1(E_2)$ and they sum up to $\mathbf{0}$. Observe that $(m+\mathfrak{f}) \cdot g_1(E_2) \geq \|\mathbf{q}\|_1 = \| \uparrow \mathbf{x}^* - \uparrow \mathbf{z}^* \|_1 \geq \|\mathbf{x}^* - \mathbf{z}^*\|_1$. We now focus on bounding $m + \mathfrak{f}$. The Steinitz Lemma (Lemma 1) implies that there exists a permutation π such that the sequence (15) can be re-arranged as

$$\mathbf{v}_1, \ldots, \mathbf{v}_{m+\mathfrak{f}}, \tag{16}$$

where $\mathbf{v}_i$ is $\mathbf{o}_{\pi^{-1}(i)}$ if $i \in [1, m]$ and $\mathbf{p}_{\pi^{-1}(i)}$ if $i \in [m+1, m+\mathfrak{f}]$, respectively, and for each $1 \leq k \leq m + \mathfrak{f}$ the prefix sum $\mathbf{t}_k := \sum_{i=1}^k \mathbf{v}_i$ satisfies

$$\|\mathbf{t}_k\|_\infty \leq r\|E_1\|_\infty g_1(E_2) \ .$$

We will now argue that there cannot be indices $1 \leq k_1 < \cdots < k_{\mathfrak{f}+2} \leq \mathfrak{f} + m$ with

$$\mathbf{t}_{k_1} = \cdots = \mathbf{t}_{k_{\mathfrak{f}+2}}, \tag{17}$$

which implies that $\mathfrak{f} + m$ is bounded by $\mathfrak{f} + 1$ times the number of integer points of norm at most $r\|E_1\|_\infty g_1(E_2)$ and therefore,

$$\begin{aligned}
\|\mathbf{x}^* - \mathbf{z}^*\|_1 &\leq \| \uparrow \mathbf{x}^* - \uparrow \mathbf{z}^* \|_1 \leq (\mathfrak{f}+1)\,(2r\|E_1\|_\infty g_1(E_2) + 1)^r \cdot g_1(E_2) \\
&\leq \kappa_2 \cdot t^4 \log(t\|E_2^1, \ldots, E_2^\tau\|_\infty) \cdot r\,(2r\|E_1\|_\infty g_1(E_2) + 1)^r \cdot g_1(E_2) \\
&\leq \left(\kappa_2 t^4 \log(t\|E_2^1, \ldots, E_2^\tau\|_\infty)\right)(2r\|E_1\|_\infty g_1(E_2)))^{r+2} \ .
\end{aligned}$$

Assume for contradiction that there exist $\mathfrak{f}+2$ indices $1 \leq k_1 < \cdots < k_{\mathfrak{f}+2} \leq \mathfrak{f}+m$ satisfying (17). By the pigeonhole principle, there is an index k_ℓ such that all the vectors $\mathbf{v}_{k_\ell+1}, \ldots, \mathbf{v}_{k_{\ell+1}}$ from the rearrangement (16) correspond to integer vectors $\mathbf{o}_{\pi^{-1}(p)}$ for $p \in [k_\ell + 1, k_{\ell+1}]$. We will show that this collection of vectors corresponds to a cycle $\mathbf{h}$ of $\uparrow \mathbf{x}^* - \uparrow \mathbf{z}^*$ which by the minimality of $\|\mathbf{x}^* - \mathbf{z}^*\|_1$ and Lemmas 9 and 14 is impossible. To obtain the cycle, for each $p \in [k_\ell + 1, k_{\ell+1}]$, let $i(p)$, $j(p)$, and $\ell(p)$ be such that $\mathbf{o}_{\pi^{-1}(p)} = E_1^{\ell(p)} \mathbf{g}_{j(p)}^{i(p)}$. Initialize $\mathbf{h} := \mathbf{0} \in \mathbb{Z}^{N't}$ and, for each $p \in [k_\ell + 1, k_{\ell+1}]$, let $\mathbf{h}^{i(p)} := \mathbf{h}^{i(p)} + g_{j(p)}^{i(p)}$. Now we check that $\mathbf{h}$ is, in fact, a cycle. First, to see that $E^{(N')}\mathbf{h} = \mathbf{0}$, we have $E_2^\ell \mathbf{h}^i = \mathbf{0}$ for every brick $i \in [N']$ of type ℓ by the fact that $\mathbf{h}^i$ is a sum of $\mathbf{g}_j^i \in \mathcal{G}(E_2^\ell) \subseteq \mathrm{Ker}_\mathbb{Z}(E_2^\ell)$, and we have $\bar{E}_1 \mathbf{h} = \mathbf{0}$ by the fact that $\mathbf{t}_{k_\ell} = \mathbf{t}_{k_{\ell+1}}$ and thus $\sum_{p \in [m+\mathfrak{f}]} E_1^{\ell(p)} \mathbf{g}_{j(p)}^{i(p)} = \mathbf{0}$. Second, $\mathbf{h} \sqsubseteq \mathbf{q}$ because, for every brick $i \in [N']$, $\mathbf{h}^i$ is a sign-compatible sum of elements $\mathbf{g}_j^i \sqsubseteq \mathbf{q}^i$. □

3.4 Improving the proximity theorem when *I* has identical columns

In this section we will show how to construct a huge n-fold instance I' from any input instance I such that the number of columns of I' per brick is at most $(2\|E\|_\infty + 1)^{r+s}$,

and in some sense I and I' are equivalent. Specifically, we will show a mapping between the solutions of I and I' which maps integer or configurable optima of I to integer or configurable optima of I' and vice versa, respectively, and such that proximity bounds from I' can be transferred to I. This will eventually allow us to show that even if I has very large t, we can bound the distance between a configurable optimum and some integer optimum of I by a function independent of t.

Construction of I'.

Note that $(2\|E\|_\infty + 1)^{r+s}$ is the number of distinct $(r + s)$-dimensional integer vectors with entries bounded by $\|E\|_\infty$ in absolute value, hence the number of possible distinct columns per brick. We will show how to "join" variables corresponding to identical columns. Consider any IP with a separable convex objective where columns corresponding to variables x_1 and x_2 are identical. Let f_1 and f_2 be the objective functions corresponding to x_1 and x_2, and l_1, l_2 and u_1, u_2 be their lower and upper bounds, respectively. Let x_{12} be a new variable which replaces x_1, x_2 in I'. Set the lower bound of x_{12} to be $l_{12} = l_1 + l_2$, upper bound $u_{12} = u_1 + u_2$, and define its objective function as the (min, +)-convolution of f_1 and f_2:

$$f_{12}(x_{12}) = \min_{\substack{x_1,x_2\in\mathbb{Z},\, x_{12}=x_1+x_2 \\ (l_1,l_2)\le(x_1,x_2)\le(u_1,u_2)}} f_1(x_1) + f_2(x_2)\ . \tag{18}$$

Note that if f_1 and f_2 are convex, then f_{12} is also convex. Extend f_{12} to fractional values as a linear interpolation, that is, for $x_{12} = \lfloor x_{12}\rfloor + \{x_{12}\}$ fractional, let $f_{12}(x_{12})$ be $f_{12}(\lfloor x_{12}\rfloor) + \{x_{12}\}(f_{12}(\lceil x_{12}\rceil) - f_{12}(\lfloor x_{12}\rfloor))$. The value $f_{12}(x_{12})$ can be obtained by binary search on x_1 (which determines $x_2 = x_{12} - x_1$) in $\mathcal{O}(\log(u_{12} - l_{12}))$ calls to evaluation oracles for f_1 and f_2. When merging a set S of more than 2 variables, one would compute $f_S(x_S)$ as the solution of the corresponding integer program whose objective is $\sum_{i\in S} f_i(x_i)$ and its constraints are $\sum_{i\in S} x_i = x_S$ and appropriate lower and upper bounds; by [13], this is solvable in time $\mathrm{poly}(|S|)\log(f_{\max}, u_S - l_S)$. However, our goal here is to strengthen our proximity result for I by studying I', without actually attempting to solve I'.

For a solution $\mathbf{x}$ of I (not necessarily integral), we define $\sigma(\mathbf{x})$ to be a solution of I' where x_1 and x_2 are replaced by $x_{12} = x_1 + x_2$. Clearly, for integer $\mathbf{x}$, the value of $\sigma(\mathbf{x})$ under the objective of I' is at most the value of $\mathbf{x}$ under f, and if $\mathbf{x}$ is an integer optimum of I, then $\sigma(\mathbf{x})$ will be an integer optimum of I' because we then have $f_{12}(x_{12}) = f_1(x_1) + f_2(x_2)$. We abuse the notation and for an integer $\mathbf{x}'$ define $\sigma^{-1}(\mathbf{x}')$ to be some integral member $\mathbf{x}$ of the set $\sigma^{-1}(\mathbf{x}')$ which satisfies $f_1(x_1) + f_2(x_2) = f_{12}(x'_{12})$. For a configurable solution $\mathbf{x}'$ we define $\sigma^{-1}(\mathbf{x}')$ by taking an $\hat{f}$-optimal decomposition Γ' of the brick of $\mathbf{x}'$ containing x_{12} and applying σ^{-1} to the configurations in Γ'; this defines a decomposition Γ and thus a brick $\sum\Gamma$ of a solution $\mathbf{x}$ of I. The next lemma shows that this construction preserves the value of the solution.

Lemma 15 *If* $\mathbf{x}$ *is an integer optimum of* I*, then* $\sigma(\mathbf{x})$ *is an integer optimum of* I'*, respectively. Similarly, if* $\mathbf{x}$ *is a configurable optimum of* I*, then* $\sigma(\mathbf{x})$ *is a configurable*

optimum of I'. Analogously, if $\mathbf{x}'$ an integer optimum of I', then $\sigma^{-1}(\mathbf{x}')$ is an integer optimum of I, and if $\mathbf{x}'$ is a configurable optimum of I', then $\sigma^{-1}(\mathbf{x}')$ is a configurable optimum of I.

Proof It follows from the definition of f_{12} that for any integer solution of I we get an integer solution of I' which is at least as good, and for any integer solution of I' we get an integer solution of I with the same value. For configurable solutions we apply the observation above to each configuration in some $\hat{f}$-optimal decomposition and use the fact that $\hat{f}$ is defined via f_{12}. □

This approach generalizes readily to any number of variables. For the sake of simplicity we continue with the example of "joining" two variables whose columns in $E^{(N)}$ are identical.

We are left to argue about proximity. While we believe that it holds in general that any proximity bound between integer and configurable optima of I' transfers to I, we only need this for our specific bound, so we take a less general route.

Lemma 16 *Let $\mathbf{x}$ be a configurable optimum of I with at most $2r$ fractional bricks, $\mathbf{x}' = \sigma(\mathbf{x})$ a configurable optimum of I', $\mathbf{z}'$ an ℓ_1-closest integer optimum of I', and $\mathbf{z} = \sigma^{-1}(\mathbf{z}')$ an integer optimum of I. Let P be the bound of Theorem 2 on $\|\mathbf{x}' - \mathbf{z}'\|_1$. Then $\|\mathbf{x} - \mathbf{z}\|_1 \leq P$.*

Proof Consider the proof of Theorem 2. In it, we create a sequence of vectors $v_1, \ldots, v_{m+\mathfrak{f}}$. Each of these vectors corresponds to some $E_1^{\ell}\lambda_j g_j^i$. The crucial observation is that the sequence $(\mathbf{v}_i)_i$ obtained from $\mathbf{x}$, $\mathbf{z}$ is identical to the sequence obtained from $\mathbf{x}'$, $\mathbf{z}'$, so if $\|\mathbf{x}' - \mathbf{z}'\|_1 \leq P$, then also $\|\mathbf{x} - \mathbf{z}\|_1 \leq P$. □

The next corollary is now immediate:

Corollary 1 *Let $\mathbf{x}^*$ be a conf-optimal solution of* (HugeCP) *with at most $2r$ fractional bricks. Then there is an optimal solution $\mathbf{z}^*$ of* (HugeIP) *such that*

$$\begin{aligned}\|\mathbf{z}^* - \mathbf{x}^*\|_1 &\leq \left(\kappa_2(r+s)(2\|E\|_\infty + 1)^{4s}\right)(2r\|E_1\|_\infty g_1(E_2)))^{6r} \\ &\leq (2\|E\|_\infty + 1)^{\mathcal{O}(s)}(2r\|E\|_\infty g_1(E_2))^{\mathcal{O}(r)} \leq (\|E\|_\infty rs)^{\mathcal{O}(rs)} .\end{aligned}$$

3.5 Algorithm

Recall the statement of the theorem we are proving: **Theorem 1.** Huge N-fold IP with any separable convex objective can be solved in time

$$(\|E\|_\infty rs)^{\mathcal{O}(r^2s+rs^2)} \operatorname{poly}(\tau, t, \log \|\mathbf{l}, \mathbf{u}, \mathbf{b}, N, f_{\max}\|_\infty) .$$

Proof We first give a description of the algorithm which solves huge N-fold IP, then show its correctness, and finally give a time complexity analysis.
Description of the algorithm. First, obtain an optimal solution $\mathbf{y}$ of (ConfLP) and from it a conf-optimal solution $\mathbf{x}^* = \varphi(\mathbf{y})$ with at most $2r$ fractional bricks by Lemma 6.

Applying Corollary 1 to $\mathbf{x}^*$ guarantees the existence of an integer optimum $\mathbf{z}^*$ satisfying

$$\|\mathbf{x}^* - \mathbf{z}^*\|_1 \leq P := \left((r+s)(2\|E\|_\infty + 1)^{4s}\right)(2r\|E_1\|_\infty g_1(E_2)))^{6r} \ . \quad (19)$$

Together with the fact that there are at most $2r$ fractional bricks, this implies that $\mathbf{z}^*$ differs from $\mathbf{x}^*$ in at most $P' = P + 2r$ bricks. The idea of the algorithm is to "fix" the value of the solution on "almost all" bricks and compute the rest using an auxiliary $\bar{N}$-fold IP problem with a polynomial $\bar{N}$.

Formally, our goal is to compute an optimal solution $\mathbf{z}$ of (HugeIP) represented succinctly by multiplicities of configurations, or in other words, as a solution $\boldsymbol{\zeta}$ of (ConfILP). Denote by $\mathbf{y}_{-P'}$ the vector whose coordinates are defined by setting, for every type $i \in [\tau]$ and every configuration $\mathbf{c} \in \mathcal{C}^i$, $\mathbf{y}_{-P'}(i, \mathbf{c}) = \max\{0, \lfloor y(i, \mathbf{c})\rfloor - P'\}$ This leaves us with $\|\mathbf{y}\|_1 - \|\mathbf{y}_{-P'}\|_1 \leq |\mathrm{supp}(\mathbf{y})|P' \leq (r+\tau)P' =: \bar{P}$ bricks to determine. Let $\bar{\boldsymbol{\zeta}} = \mathbf{y} - \mathbf{y}_{-P'}$, define $\bar{\boldsymbol{\mu}}$ by setting, for each $i \in [\tau]$, $\bar{\boldsymbol{\mu}}_i := \sum_{\mathbf{c}\in\mathcal{C}^i} \bar{\zeta}(i, \mathbf{c})$, let $\bar{\mathbf{x}} = \varphi(\bar{\boldsymbol{\zeta}})$, and let $\bar{N} = \|\bar{\boldsymbol{\zeta}}\|_1 = \|\bar{\boldsymbol{\mu}}\|_1 \leq \bar{P}$. Construct an auxiliary $\bar{N}$-fold IP instance with the same blocks E_1^i, E_2^i, $i \in [\tau]$, by, for each brick $\bar{\mathbf{x}}^j$ of type i, setting

$$- \ \bar{f}^j = f^i, \qquad - \ \bar{\mathbf{b}}^j = \mathbf{b}^i, \qquad - \ \bar{\mathbf{l}}^j = \mathbf{l}^i, \qquad - \ \bar{\mathbf{u}}^j = \mathbf{u}^i.$$

We say that such a brick was *derived from type* i. Lastly, let $\bar{\mathbf{b}}^0 = \mathbf{b}^0 - \sum_{i=1}^{\tau}\sum_{\mathbf{c}\in\mathcal{C}^i} \zeta(i, \mathbf{c})E_1^i\mathbf{c}$.

After obtaining an optimal solution $\bar{\mathbf{z}}$ of this instance we update $\boldsymbol{\zeta}$ as follows. For each brick $\bar{\mathbf{z}}^j$ derived from type i, increment $\zeta(i, \bar{\mathbf{z}}^j)$ by one.

Correctness. By (19), it is correct to assume that there exists a solution $\boldsymbol{\zeta}$ of (ConfILP) which has $\zeta(i, \mathbf{c}) \geq \max\{0, \lfloor y(i, \mathbf{c})\rfloor - P'\}$ for each $i \in [\tau]$ and $\mathbf{c} \in \mathcal{C}^i$. Thus we may do a variable transformation of (ConfILP) $\boldsymbol{\zeta} = \bar{\boldsymbol{\zeta}} + \mathbf{y}_{-P'}$, obtaining an auxiliary (ConfILP) instance

$$\min \mathbf{v}(\bar{\boldsymbol{\zeta}} + \mathbf{y}_{-P'}) \ : \ B(\bar{\boldsymbol{\zeta}} + \mathbf{y}_{-P'}) = \mathbf{d}, \ \mathbf{0} \leq \bar{\boldsymbol{\zeta}} \ .$$

The auxiliary huge $\bar{N}$-fold instance is simply the instance corresponding to the above, and the final construction of $\boldsymbol{\zeta}$ corresponds to the described variable transformation.

Complexity. Since $\|\bar{\boldsymbol{\zeta}}\|_1 \leq \bar{P}$, we can obtain an optimal solution $\bar{\mathbf{z}}$ of the auxiliary instance in time $(\|E\|_\infty rs)^{\mathcal{O}(r^2s+rs^2)}(t\bar{P})\log(t\bar{P})\log\|f_{\max}, \bar{\mathbf{b}}, \bar{\mathbf{l}}, \bar{\mathbf{u}}\|_\infty^2$ [13, Corollary 91]. Let us now compute the time needed altogether. To solve (ConfLP), we need time (Lemma 6)

$$\|E\|_\infty^{\mathcal{O}(s^2)} \mathrm{poly}(rt\tau \log\|f_{\max}, \mathbf{l}, \mathbf{u}, \mathbf{b}, \boldsymbol{\mu}\|_\infty) \ .$$

To solve the auxiliary instance above, we need time

$$(\|E\|_\infty rs)^{\mathcal{O}(r^2s+rs^2)}(t\bar{P})\log(t\bar{P})\log\|f_{\max}, \bar{\mathbf{b}}, \bar{\mathbf{l}}, \bar{\mathbf{u}}\|_\infty^2,$$

$$\text{where } \bar{P} = (r+\tau)P' \leq \tau(2\|E\|_\infty + 1)^{\mathcal{O}(s)}(2r\|E\|_\infty g_1(E_2))^{\mathcal{O}(r)} \ .$$

Hence we can solve huge N-fold IP in time at most

$$(\|E\|_\infty rs)^{\mathcal{O}(r^2s+rs^2)} \operatorname{poly}(t\tau \log \|f_{\max}, \mathbf{l}, \mathbf{u}, \mathbf{b}, \boldsymbol{\mu}\|_\infty) \ .$$

□

4 Concluding remarks

At this point one may wonder why bother with the ConfLP rather than solving HugeCP and showing that its optima are close to those of HugeIP. The reason is that even though handling optima of HugeCP is much easier than handling conf-optimal solutions, and even though solving HugeCP is easier than solving ConfLP,[4] a HugeCP optimum can be very far from a HugeIP optimum [8, Proposition 1]. In other words, ConfLP is a stronger relaxation than HugeCP: consider a brick $\mathbf{p}$ of a HugeCP optimum and a brick $\mathbf{q}$ of a conf-optimal solution; then

$$\mathbf{q} \in \operatorname{conv}\{\mathbf{c} \in \mathbb{Z}^d \mid E_2\mathbf{c} = \mathbf{0}, \mathbf{l}^i \leq \mathbf{c} \leq \mathbf{u}^i\} \subset \{\mathbf{p}' \in \mathbb{R}^d \mid E_2\mathbf{p}' = \mathbf{0}, \mathbf{l}^i \leq \mathbf{p} \leq \mathbf{u}^i\} \ .$$

In plain language, while $\mathbf{q}$ lies in the integer hull of all configurations, $\mathbf{p}$ only lies in the fractional relaxation of this hull.

Another obstacle is that even though Configuration LP is a standard tool, it is typical that the separation problem is merely approximated rather than solved exactly, leading to approximate solutions of ConfLP. But, we require an exact solution, and so we use a parameterized exact algorithm for IP to solve the separation problem. It is an interesting question when a k-approximate solution of ConfLP, i.e., a solution whose value is at most $k \cdot OPT$, may be used to obtain an $h(k)$-accurate configurable solution of HugeCP, i.e., a configurable solution which is at ℓ_1-distance at most $h(k)$ from a configurable optimum. An approximate solution of ConfLP might be much easier to obtain, and yet it may be almost as good as an exact solution for our purposes here.

Another interesting question is a tight complexity bound for the algorithm of Lemma 6. It seems likely that the recent approach of Cslovjecsek et al. [8] could also apply in our high-multiplicity setting, which would yield a near-linear fixed-parameter algorithm. Notice that the iterative augmentation algorithms for standard

[4] We only outline the reason. We claim that an optimal solution of HugeCP can be obtained in the following way: construct an auxiliary τ-fold CP with bricks $\bar{\mathbf{x}}^1, \ldots, \bar{\mathbf{x}}^\tau$ where the i-th brick $\bar{\mathbf{x}}^i$, $i \in [\tau]$, represents the μ^i bricks of type i in the original instance. This is achieved by setting the bounds to $\mu^i\mathbf{l}^i$ and $\mu^i\mathbf{u}^i$, the right hand side to $\mu^i\mathbf{b}^i$, and the objective to $\mu^i f^i(\bar{\mathbf{x}}^i/\mu^i)$. Given an optimum $\bar{\mathbf{x}}$ of this τ-fold CP, we set each brick of type $\mathbf{x}^i$ to the value $\bar{\mathbf{x}}^i/\mu^i$, and claim that $\mathbf{x}$ is an optimum of HugeCP. Thus, while HugeCP can be solved in polynomial time, the separation subproblem needed to solve ConfLP can be NP-hard, in this sense making ConfLP harder.

N-fold IP have a strong combinatorial flavor and use no "black boxes". Could the ellipsoid method behind Lemma 6 be replaced by a (more) combinatorial algorithm, at least for some important problems which have huge N-fold IP models, such as the scheduling problems studied by Knop et al. [31]?

Funding Open Access funding enabled and organized by Projekt DEAL. D.K. partially supported by OP VVV MEYS project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics". M.K. partially supported by Charles University project UNCE/SCI/004, and by project 19-27871X of Grantová agentura České republiky (GA ČR). A.L. partially supported by Israel Science Foundation grant 308/18. M.M. supported by Deutsche Forschungsgemeinschaft (DFG) grant MN 59/4-1. S.O. partially supported by the Dresner chair and Israel Science Foundation grant 308/18.

Availability of data and material (data transparency) Not applicable.

Code Availability Not applicable.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Alon, N., Azar, Y., Woeginger, G.J., Yadid, T.: Approximation schemes for scheduling on parallel machines. J. Sched. **1**(1), 55–66 (1998)
2. Altmanová, K., Knop, D., Koutecký, M.: Evaluating and tuning n-fold integer programming. ACM J. Exp. Algorithmics **24**(1), 1–22 (2019)
3. Aykanat, C., Pinar, A., Çatalyürek, Ü.V.: Permuting sparse rectangular matrices into block-diagonal form. SIAM J. Scientific Comput. **25**(6), 1860–1879 (2004)
4. Bergner, M., Caprara, A., Ceselli, A., Furini, F., Lübbecke, M.E., Malaguti, E., Traversi, E.: Automatic Dantzig-Wolfe reformulation of mixed integer programs. Math. Prog. **149**(1–2), 391–424 (2015)
5. Borndörfer, R., Ferreira, C.E., Martin, A.: Decomposing matrices into blocks. SIAM J. Optim. **9**(1), 236–269 (1998)
6. Chen, L., Marx, D.: Covering a tree with rooted subtrees—parameterized and approximation algorithms. In Proc. SODA **2018**, 2801–2820 (2018)
7. Cosmadakis, S.S., Papadimitriou, C.H.: The traveling salesman problem with many visits to few cities. SIAM J. Comput. **13**(1), 99–108 (1984)
8. Cslovjecsek, J., Eisenbrand, F., Hunkenschröder, C., Rohwedder, L., Weismantel, R.: Block-structured integer and linear programming in strongly polynomial and near linear time. In: Proc. SODA 2021, pp. 1666–1681 (2021)
9. De Loera, J.A., Hemmecke, R., Onn, S., Weismantel, R.: n-fold integer programming. Discrete Optim. **5**(2), 231–241 (2008)
10. De Loera, J. A., Hemmecke, R., Köppe, M.: Algebraic and geometric ideas in the theory of discrete optimization, volume 14 of MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA; Mathematical Optimization Society, Philadelphia, PA (2013)

11. Eisenbrand, F., Weismantel, R.: Proximity results and faster algorithms for integer programming using the Steinitz lemma. ACM Trans. Algorithms **16**(1), 5:1-5:14 (2020)
12. Eisenbrand, F., Hunkenschröder, C., Klein, K.-M.: Faster algorithms for integer programs with block structure. In Proc. ICALP 2018, volume 107 of Leibniz Int. Proc. Informatics, pp. 49:1–49:13 (2018)
13. Eisenbrand, F., Hunkenschröder, C., Klein, K., Koutecký, M., Levin, A., Onn, S.: An algorithmic theory of integer programming. Technical report (2019). http://arXiv.org/abs/1904.01361
14. Fernandez de la Vega, W., Lueker, G.S.: Bin packing can be solved within $1 + \varepsilon$ in linear time. Combinatorica **1**(4), 349–355 (1981)
15. Ferris, M.C., Horn, J.D.: Partitioning mathematical programs for parallel solution. Math. Prog. **80**(1), 35–61 (1998)
16. Gamrath, G., Lübbecke, M. E.: Experiments with a generic Dantzig–Wolfe decomposition for integer programs. In: Proc. SEA 2010, Lecture Notes in Computer Science, vol. 6049, pp. 239–252. Springer (2010)
17. Gilmore, P.C., Gomory, R.E.: A linear programming approach to the cutting-stock problem. Oper. Res. **9**, 849–859 (1961)
18. Grötschel, M., Lovász, L., Schrijver, A.: Geometric algorithms and combinatorial optimization, vol. 2 of Algorithms and Combinatorics. Springer-Verlag, Berlin, second edition (1993)
19. Hemmecke, R., Köppe, M., Weismantel, R.: Graver basis and proximity techniques for block-structured separable convex integer minimization problems. Math. Prog. **145**(1–2, Ser. A), 1–18 (2014)
20. Hochbaum, D.S., Shamir, R.: Strongly polynomial algorithms for the high multiplicity scheduling problem. Oper. Res. **39**(4), 648–653 (1991)
21. Hochbaum, D.S., Shantikumar, J.G.: Convex separable optimization is not much harder than linear optimization. J. ACM **37**(4), 843–862 (1990)
22. Hochbaum, D.S., Shmoys, D.B.: Using dual approximation algorithms for scheduling problems: theoretical and practical results. J. Assoc. Comput. Mach. **34**(1), 144–162 (1987)
23. Jansen, K., Rohwedder, L.: On integer programming and convolution. In Proc. ITCS 2019, vol. 124 of Leibniz Int. Proc. Informatics, pp. 43:1–43:17 (2019)
24. Jansen, K., Solis-Oba, R.: A polynomial time $OPT + 1$ algorithm for the cutting stock problem with a constant number of object lengths. Math. Oper. Res. **36**(4), 743–753 (2011)
25. Jansen, K., Klein, K., Maack, M., Rau, M.: Empowering the Configuration-IP — new PTAS results for scheduling with setups times. In Proc. ITCS 2019, volume 124 of Leibniz Int. Proc. Informatics, pp. 44:1–44:19 (2019)
26. Jansen, K., Lassota, A., Rohwedder, L.: Near-linear time algorithm for n-fold ilps via color coding. SIAM J. Discret. Math. **34**(4), 2282–2299 (2020)
27. Karmarkar, N., Karp, R.M.: An efficient approximation scheme for the one-dimensional bin-packing problem. In Proc. FOCS 1982, pp. 312–320 (1982)
28. Khaniyev, T., Elhedhli, S., Erenay, F.S.: Structure detection in mixed-integer programs. INFORMS J. Comput. **30**(3), 570–587 (2018)
29. Knop, D., Koutecký, M.: Scheduling meets n-fold integer programming. J. Sched. **21**(5), 493–503 (2018)
30. Knop, D., Koutecký, M.: Scheduling kernels via configuration LP. Technical report, (2020). arXiv:2003.02187
31. Knop, D., Koutecký, M., Levin, A., Mnich, M., Onn, S.: Multitype integer monoid optimization and applications. Technical report (2019) arXiv:1909.07326
32. Knop, D., Koutecký, M., Mnich, M.: Combinatorial n-fold integer programming and applications. Math. Program. **184**(1), 1–34 (2020)
33. Koutecký, M., Levin, A., Onn, S.: A parameterized strongly polynomial algorithm for block structured integer programs. In Proc. ICALP 2018, vol. 107 of Leibniz Int. Proc. Informatics, pp. 85:1–85:14 (2018)
34. Onn, S.: Nonlinear discrete optimization—an algorithmic theory. Zurich Lectures in Advanced Mathematics. European Mathematical Society (EMS), Zürich (2010)
35. Onn, S.: Huge multiway table problems. Discrete Optim. **14**, 72–77 (2014)
36. Onn, S.: Huge tables and multicommodity flows are fixed-parameter tractable via unimodular integer Carathéodory. J. Comput. Syst. Sci. **83**(1), 207–214 (2017)
37. Psaraftis, H.N.: A dynamic programming approach for sequencing groups of identical jobs. Oper. Res. **28**(6), 1347–1359 (1980)

38. Scheithauer, G., Terno, J.: The modified integer round-up property of the one-dimensional cutting stock problem. European J. Oper. Res. **84**(3), 562–571 (1995)
39. Sevast'janov, S., Banaszczyk, W.: To the Steinitz lemma in coordinate form. Discrete Math. **169**(1–3), 145–152 (1997)
40. Steinitz, E.: Bedingt konvergente Reihen und konvexe Systeme. J. Reine Angew. Math. **146**, 1–52 (1916)
41. van den Akker, J.M., Hoogeveen, J.A., van de Velde, S.L.: Parallel machine scheduling by column generation. Oper. Res. **47**(6), 862–872 (1999)
42. Vanderbeck, F., Wolsey, L. A.: Reformulation and decomposition of integer programs. In: 50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art, pp. 431–502. Springer (2010)
43. Vose, M.D.: Egyptian fractions. Bull. London Math. Soc. **17**(1), 21–24 (1985)
44. Wang, J., Ralphs, T.: Computational experience with hypergraph-based methods for automatic decomposition in discrete optimization. In: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, pp. 394–402. Springer (2013)
45. Weil, R.L., Kettler, P.C.: Rearranging matrices to block-angular form for decomposition (and other) algorithms. Mgmt. Sci. **18**(1), 98–108 (1971)

Parameterized Algorithms for MILPs with Small Treedepth

Cornelius Brand,[1] **Martin Koutecký,**[1] **Sebastian Ordyniak,**[2]
[1] Computer Science Institute, Charles University, Prague, Czech Republic,
[2] School of Computing, University of Leeds, United Kingdom,
{cbrand, koutecky}@iuuk.mff.cuni.cz, s.ordyniak@leeds.ac.uk

Abstract

Solving (mixed) integer (linear) programs, (M)I(L)Ps for short, is a fundamental optimisation task with a wide range of applications in artificial intelligence and computer science in general. While hard in general, recent years have brought about vast progress for solving structurally restricted, (non-mixed) ILPs: n-fold, tree-fold, 2-stage stochastic and multi-stage stochastic programs admit efficient algorithms, and all of these special cases are subsumed by the class of ILPs of small treedepth.

In this paper, we extend this line of work to the mixed case, by showing an algorithm solving MILP in time $f(a, d)\,\mathrm{poly}(n)$, where a is the largest coefficient of the constraint matrix, d is its treedepth, and n is the number of variables.

This is enabled by proving bounds on the denominators (fractionality) of the vertices of bounded-treedepth (non-integer) linear programs. We do so by carefully analysing the inverses of invertible sub-matrices of the constraint matrix. This allows us to afford scaling up the mixed program to the integer grid, and applying the known methods for integer programs.

We then trace the limiting boundary of our "bounded fractionality" approach both in terms of going beyond MILP (by allowing non-linear objectives) as well as its usefulness for generalising other important known tractable classes of ILP. On the positive side, we show that our result can be generalised from MILP to MIP with piece-wise linear separable convex objectives with integer breakpoints. On the negative side, we show that going even slightly beyond such objectives or considering other natural related tractable classes of ILP leads to unbounded fractionality.

Finally, we show that restricting the structure of only the integral variables in the constraint matrix does not yield tractable special cases.

Introduction

Integer Linear Programming (ILP) is a fundamental hard problem as well as a widely used and highly successful framework for solving difficult computational problems in AI, e.g., problems related to planning (van den Briel, Vossen, and Kambhampati 2005; Vossen et al. 1999), vehicle routing (Toth and Vigo 2001), process scheduling (Floudas and Lin 2005), packing (Lodi, Martello, and Monaci 2002), and network hub location (Alumur and Kara 2008) that can often be solved efficiently using a translation to ILP. This naturally motivates the search for tractable classes for ILP. In the '80s, Lenstra and Kannan (Kannan 1987; Lenstra 1983) and Papadimitriou (Papadimitriou 1981) have shown that the classes of ILPs with few variables or few constraints and small coefficients, respectively, are polynomially solvable. A line of research going back almost 20 years (Hemmecke, Onn, and Romanchuk 2013; Chen and Marx 2018; Eisenbrand, Hunkenschröder, and Klein 2018; Aschenbrenner and Hemmecke 2007; Hemmecke, Köppe, and Weismantel 2014; Ganian, Ordyniak, and Ramanujan 2017; Ganian and Ordyniak 2018; Dvorák et al. 2017) has recently culminated with the discovery of another tractable class of ILPs (Eisenbrand et al. 2019; Koutecký, Levin, and Onn 2018), namely ILPs with small treedepth and coefficients. The obtained results already found various algorithmic applications in areas such as scheduling (Knop and Koutecký 2018; Chen et al. 2017; Jansen et al. 2018), stringology and social choice (Knop, Koutecký, and Mnich 2017a,b), and the travelling salesman problem (Chen and Marx 2018).

The language of "special tractable cases" has been developed in the theory of parameterized complexity (Cygan et al. 2015). We say that a problem is *fixed-parameter tractable* (FPT) parameterized by k if it has an algorithm solving every instance I in time $f(k)\,\mathrm{poly}(|I|)$ for some computable function f, and we call this an FPT *algorithm*. Say that the height of a rooted forest is its largest root-leaf distance. A graph $G = (V, E)$ has treedepth d if d is the smallest height of a rooted forest $F = (V, E')$ in which each edge of G is between an ancestor-descendant pair in F, and we write $\mathrm{td}(G) = d$. The *primal graph* $G_P(A)$ of a matrix $A \in \mathbb{R}^{m\times n}$ has a vertex for each column of A, and two vertices are connected if an index $k \in [m] = \{1, \ldots, m\}$ exists such that both columns are non-zero in row k. The *dual graph* $G_D(A)$ is defined as $G_D(A) := G_P(A^\intercal)$. Define the *primal treedepth of* A to be $\mathrm{td}_P(A) = \mathrm{td}(G_P(A))$, and analogously $\mathrm{td}_D(A) = \mathrm{td}(G_D(A))$. The recent results state that there is an algorithm solving ILP in time $f(\|A\|_\infty, \min\{\mathrm{td}_P(A), \mathrm{td}_D(A)\})\,\mathrm{poly}(n)$, hence ILP is FPT parameterized by $\|A\|_\infty$ and $\min\{\mathrm{td}_P(A), \mathrm{td}_D(A)\}$. Besides this class, other parameterizations of ILP have been successfully employed to show tractability results, such as bounding the treewidth of the primal graph and the largest variable domain (Jansen and Kratsch 2015), the treewidth of

the incidence graph and the largest solution prefix sum (Ganian, Ordyniak, and Ramanujan 2017), or the signed clique-width of the incidence graph (Eiben et al. 2018).

It is therefore natural to ask whether these tractability results can be generalised to more general settings than ILP. In this paper we ask this question for Mixed ILP (MILP), where both integer and non-integer variables are allowed:

$$\min \{\mathbf{cx} \mid A\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{Z}^z \times \mathbb{Q}^q\}, \quad (1)$$

with $A \in \mathbb{Z}^{m \times z+q}$, $\mathbf{l}, \mathbf{u}, \mathbf{c} \in \mathbb{Z}^{z+q}$ and $\mathbf{b} \in \mathbb{Z}^m$.

MILP is a prominent modelling tool widely used in practice. For example, Bixby (Bixby 2002) says in his famous analysis of LP solver speed-ups, *"[I]nteger programming, and most particularly the mixed-integer variant, is the dominant application of linear programming in practice."* Already Lenstra has shown that MILP with few integer variables is polynomially solvable, naturally extending his result on ILPs with few variables. Analogously, we seek to extend the recent tractability results from ILP to MILP, most importantly for the parameterization by treedepth and largest coefficient. Our main result is as follows:

Theorem 1. *MILP is* FPT *parameterized by* $\|A\|_\infty$ *and* $\min\{\mathrm{td}_P(A), \mathrm{td}_D(A)\}$.

We note that our result also extends to the inequality form of MILP with constraints of the form $A\mathbf{x} \leq \mathbf{b}$ by the fact that introducing slack variables does not increase treedepth too much (Eisenbrand et al. 2019, Lemma 56).

The proof goes by reducing an MILP instance to an ILP instance whose parameters do not increase too much, and then applying the existing algorithms for ILP. A key technical result concerns the *fractionality* of an MILP instance, which is the minimum of the maxima of the denominators in optimal solutions. For example, it is well-known that the natural LP for the VERTEX COVER problem has half-integral optima, that is, there exists an optimum with all values in $\{0, \frac{1}{2}, 1\}$.

The usual way to go about proving fractionality bounds is via Cramer's rule and a sufficiently good bound on the determinant. As witnessed by any proper integer multiple of the identity, determinants can grow large even for matrices of very benign structure. Instead, we need to analyse much more carefully the structure of the inverse of the appearing invertible sub-matrices, allowing us to show:

Theorem 2. *A MILP instance with a constraint matrix A has an optimal solution $\mathbf{x}$ whose largest denominator is bounded by $(\|A\|_\infty)^{d!}(d!)^{d!/2}$, where $d = \min\{\mathrm{td}_P(A), \mathrm{td}_D(A)\}$.*

We are not aware of any prior work which lifts a positive result for ILP to a result for MILP in this way.

We also explore the limits of approaching the problem by bounding the fractionality of inverses: Other ILP classes with parameterized algorithms involve constraint matrices with small primal treewidth (Jansen and Kratsch 2015), small incidence treewidth (Ganian, Ordyniak, and Ramanujan 2017), small signed clique-width (Eiben et al. 2018) and 4-block n-fold matrices (Hemmecke, Köppe, and Weismantel 2014). Here, we obtain a negative answer: For each of these parameters, there exist families of MILP-instances with constant parameters, but unbounded fractionality. This is detailed in Lemma 18 below. The produced families also show that Theorem 2 is almost optimal:

Corollary 3. *There is a MILP instance with $\mathrm{td}_P(A), \mathrm{td}_D(A) = d$, $\|A\|_\infty = 2$, and fractionality 2^{2^d}.*

Compare this with our upper bound $2^{2^{d+\log d+\log\log d}}$. Next, we consider extending the positive result of Theorem 1 to separable convex functions, which is the regime considered in (Eisenbrand et al. 2019). We show that merely bounding the fractionality will unfortunately not suffice, which is detailed in Lemma 20 below. However, we show that for one important class of separable convex objectives, the fractionality does *not* increase, specifically: piece-wise linear functions with integer breakpoints. Let f be any separable convex function, and define f' to agree with f on integer points, and to be linear between them. In a sense, f' is an approximation of f which has a simpler structure. Using f' as a proxy for f is thus common in practice (Bazaraa, Sherali, and Shetty 2013; Lin et al. 2013). Moreover, functions of this form appear in applications of IPs with small treedepth (Knop, Koutecký, and Mnich 2017a; Bredereck et al. 2020).

Theorem 4. *MIP is* FPT *parameterized by* $\|A\|_\infty$ *and* $\min\{\mathrm{td}_P(A), \mathrm{td}_D(A)\}$ *if the objective function is piece-wise linear separable convex with integer breakpoints.*

By appropriate scaling, the integrality of breakpoints in the preceding theorem can be relaxed to requiring only breakpoints with fractionality bounded in the parameters.

Finally, we consider a different way to extend tractable ILP classes to MILP. Divide the constraint matrix A of an MILP instance in two parts corresponding to the integer and continuous variables as $A = (A_\mathbb{Z}\ A_\mathbb{Q})$. What structural restrictions have to be placed on $A_\mathbb{Z}$ and $A_\mathbb{Q}$ in order to obtain tractability of MILP? We show a general hardness result in this direction, which is made precise in Lemma 21. Note that the main reason for intractability is that we allow arbitrary interactions between the integer and the non-integer variables of the instance. Thus, Lemma 21 implies that this interaction between integral and fractional variables has to be restricted in some way in order to obtain a tractable fragment of MILP.

Related Work

We have already mentioned related work on structural parameterizations of ILP. The closest work to ours was done by Hemmecke (Hemmecke 2003) in 2003 when he studied a mixed-integer test set related to the Graver basis, which is the engine behind all recent progress on ILPs of small treedepth. It is unclear how to apply his approach, however, because it requires bounding the norm of elements of the mixed-integer test set, where the bound obtained by (a strengthening of) (Hemmecke 2003, Lemma 19),(Hemmecke 2001, Lemma 2.7.2), is polynomial in n, too much to obtain an FPT algorithm. Kotnyek (Kotnyek 2002) characterised k-integral matrices, i.e., matrices whose solutions have fractionality bounded by k, however it is unclear how his characterisation could be used to show Theorem 2, so we take a different route. Lenstra (Lenstra 1983) showed how to solve MILPs with few integer variables using the fact that a projection

of a polytope is again a polytope; applying this approach to our case would require us to show that if P is a polytope described by inequalities with small treedepth, then a projection of P also has an inequality description of small treedepth. This is unclear. In a vein somewhat similar to our bounded-fractionality approach, ideas related to half-integrality have recently led to improved FPT algorithms (Iwata, Wahlstrom, and Yoshida 2016; Iwata, Yamaguchi, and Yoshida 2018; Guillemot 2011), some of which have been experimentally evaluated (Pilipczuk and Ziobro 2018). More fundamentally, half-integrality of two-commodity flow (Hu 1963; Karzanov 1998) and VERTEX COVER (Nemhauser and Trotter 1974) has been known and made use of for half a century.

Preliminaries

We consider zero a natural number, i.e., $0 \in \mathbb{N}$. 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 $m \leq n$ we set $[m, n] := \{m, \dots, n\}$ and $[n] := [1, n]$. The following Proposition is well-known, but crucial.

Proposition 5. *Let $A \in \mathbb{Z}^{n \times n}$ be a full rank square matrix. Then,* $\operatorname{frac}(A^{-1}) \leq (\|A\|_\infty)^n n^{n/2}$.

Proof. Let $\operatorname{adj}(A)$ be the adjugate matrix of A (which has certain subdeterminants of A as entries.) Cramer's rule states that $A^{-1} = \frac{1}{\det(A)} \cdot \operatorname{adj}(A)$ holds. Hadamard's bound on $|\det(A)|$ can be derived from the fact that $|\det(A)$ is the volume of the parallelepiped spanned by the columns A_i of A. This in turn has the product of their lengths as an upper bound, attained precisely on pairwise orthogonal columns. Therefore, $|\det(A)| \leq \prod_{i=1}^{n} ||A_i||_2$, whence the Proposition easily follows. □

The proof makes it clear what makes it necessary to go beyond Cramer's rule: $|\det(A)|$ might be prohibitively large, while cancellations in $\operatorname{adj}(A)/\det(A)$ may lead to low fractionality in A^{-1} nonetheless.

Reducing MILP to ILP

Assume that an MILP instance is given and that some optimum $\mathbf{x} = (\mathbf{x}_\mathbb{Z}, \mathbf{x}_\mathbb{Q})$ exists whose set of denominators is D, and we know $M = \max D$. Recall $\operatorname{lcm}(D)$ is the least common multiple of the elements of D, and $\operatorname{lcm}(D) \leq M! =: \tilde{M}$. Then $\operatorname{lcm}(D)\mathbf{x}_\mathbb{Q}$ is an integral vector. Our idea here is to restrict our search among all optima of (1) to search among those optima with small fractionality, that is, with small denominators. Consider the *integralized MILP* instance:

$$\min\{(\tilde{M}\mathbf{c}_\mathbb{Z}\ \mathbf{c}_\mathbb{Q})\mathbf{z} \colon \mathbf{z} \in \mathbb{Z}^{z+q}, (\tilde{M} \cdot A_\mathbb{Z}\ A_\mathbb{Q})\mathbf{z} = \tilde{M} \cdot \mathbf{b}, (\mathbf{l}_\mathbb{Z}, \tilde{M}\mathbf{l}_\mathbb{Q}) \leq (\mathbf{z}_\mathbb{Z}, \mathbf{z}_\mathbb{Q}) \leq (\mathbf{u}_\mathbb{Z}\tilde{M}\mathbf{u}_\mathbb{Q})\} \tag{2}$$

We claim that the optimum of (1) can be recovered from the optimum of (2):

Lemma 6. *Let M be the fractionality of* (1) *and $(\mathbf{z}_\mathbb{Z}\ \mathbf{z}_\mathbb{Q}) \in \mathbb{Z}^{z+q}$ be an optimum of* (2)*. Then $\mathbf{x} = (\mathbf{z}_\mathbb{Z}\ \frac{1}{M}\mathbf{z}_\mathbb{Q})$ is an optimum of* (1)*.*

Proof. It is clear that there is a bijection between solutions $\mathbf{x}$ of (1) where $\mathbf{x}_\mathbb{Q}$ has all entries with a denominator $\tilde{M}$ and solutions $\mathbf{z}$ of (2). The optimality of $\mathbf{x}$ then follows from M being the fractionality of (1) and $M!$ always being divisible by $\operatorname{lcm}(D)$. □

The Graphs of A and Treedepth

We assume that $G_P(A)$ and $G_D(A)$ are connected, otherwise A has (up to row and column permutations) a block diagonal structure and solving (1) amounts to solving smaller (1) instances (for each block) independently.

Definition 7 (Treedepth)**.** *The* closure $\operatorname{cl}(F)$ *of a rooted tree F is the graph obtained from F by making every vertex adjacent to all of its ancestors. The* height *of a tree F denoted* $\operatorname{ht}(F)$ *is the maximum number of vertices on any root-leaf path. We denote by* $\operatorname{dt}_F(v)$ *the* depth *of vertex v in F, i.e., the number of vertices on the path from v to the root of F. A* td-decomposition of G *is a tree F such that $G \subseteq \operatorname{cl}(F)$. The* treedepth $\operatorname{td}(G)$ *of a connected graph G is the minimum height of its* td*-decompositions.*

To facilitate the analysis of our results we use two parameters called topological height (introduced by Eisenbrand et al. (Eisenbrand et al. 2019)) and topological length:

Definition 8 (Topological height and Topological length)**.** *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* $\operatorname{th}(F)$*, is the maximum number of non-degenerate vertices on any root-leaf path in F. The* topological length of F*, denoted* $\operatorname{tl}(F)$*, is the maximum number of consecutive degenerate vertices on any root-leaf path in F. Clearly,* $\operatorname{th}(F), \operatorname{tl}(F) \leq \operatorname{ht}(F)$.

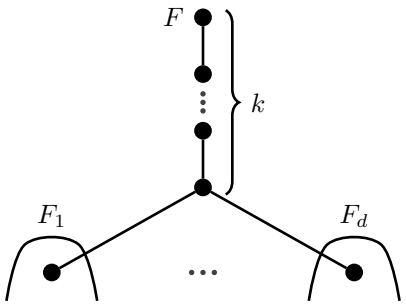


Figure 1: The treedepth decomposition F of $G_P(A)$ for the situation in Lemma 9.

We also need a lemma from (Eisenbrand et al. 2019); refer also to Figure 1 for an illustration.

Lemma 9 (Primal Decomposition (Eisenbrand et al. 2019, Lemma 19))**.** *Let $A \in \mathbb{Z}^{m \times n}$, $G_P(A)$, and a* td*-decomposition F of $G_P(A)$ be given, where $n, m \geq 1$. Then there exists an algorithm computing in time $\mathcal{O}(n)$ a decom-*

position of A

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

and td*-decompositions* $F_1, \ldots, F_d$ *of* $G_P(A_1), \ldots, G_P(A_d)$*, respectively, where* $d \in \mathbb{N}$*,* $\bar{A}_i \in \mathbb{Z}^{m_i \times k}$*,* $A_i \in \mathbb{Z}^{m_i \times n_i}$*,* $\mathrm{th}(F_i) \leq \mathrm{th}(F) - 1$*,* $\mathrm{ht}(F_i) \leq \mathrm{ht}(F) - k$*,* $k \leq \mathrm{tl}(F)$*, for* $i \in [d]$*,* $n_1, \ldots, n_d, m_1, \ldots, m_d \in \mathbb{N}$*.*

Fractionality of Bounded-Treedepth Matrices

This section is devoted to a proof of our main tractability result stated in Theorem 1, i.e., showing that MILP (like ILP) is fixed-parameter tractable parameterized by $\|A\|_\infty$ and $d = \min\{\mathrm{td}_P(A), \mathrm{td}_D(A)\}$. The main ingredient for the proof is Theorem 2 providing a bound on the fractionality of an optimal solution for MILP:

Theorem 2. *A MILP instance with a constraint matrix A has an optimal solution $\mathbf{x}$ whose largest denominator (fractionality) is bounded by $(\|A\|_\infty)^{d!}(d!)^{d!/2}$.*

We start by observing that the fractionality of an optimal solution of a MILP instance can be obtained from the fractionality of the inverse of some full rank square sub-matrix of the non-integer part of the constraint matrix A. Consider any optimal solution $(\mathbf{x}^*_\mathbb{Z}, \mathbf{x}^*_\mathbb{Q})$ of (1). The fractional part $\mathbf{x}^*_\mathbb{Q}$ is necessarily an optimal solution of the *linear* program

$$\begin{aligned} \min\{\mathbf{c}\mathbf{x}_\mathbb{Q} :\ & A_\mathbb{Q}\mathbf{x}_\mathbb{Q} = \mathbf{b} - A_\mathbb{Z}\mathbf{x}^*_\mathbb{Z}, \\ & \mathbf{l}_\mathbb{Q} \leq \mathbf{x}_\mathbb{Q} \leq \mathbf{u}_\mathbb{Q}, \mathbf{x}_\mathbb{Q} \in \mathbb{Q}^q\} \ . \end{aligned} \tag{3}$$

To bound the fractionality of (1), it therefore suffices to consider the fractionality of (3), and we shall hence assume that $A = A_\mathbb{Q}$.

Let us now recall some basic facts about vertices of polytopes adapted to the specifics of our situation. Consider a vertex of the polytope described by the solutions of the system of

$$A\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\,, \tag{4}$$

with $A, \mathbf{b}, \mathbf{x}, \mathbf{l}, \mathbf{u}$ as usual. Let $\mathbf{x}$ be any solution of (4). Being a vertex means satisfying n linearly independent constraints with equality. Without loss of generality (Eisenbrand et al. 2019, Proposition 4), A has full rank.

Since these first m equations necessarily hold for any solution $\mathbf{x}$, we have m linearly independent constraints satisfied, and there remain $n - m$ of the in total $2n$ upper and lower bounds to be satisfied. Without loss of generality, we may assume that it is indeed the first $n - m$ lower bound constraints that are met with equality, that is, $x_1 = l_1, \ldots, x_{n-m} = l_{n-m}$ holds. Let

$$\begin{aligned} \mathbf{x}_N &= (x_1, \ldots, x_{n-m}) \in \mathbb{Q}^{n-m}, \\ \mathbf{x}_B &= (x_{n-m+1}, \ldots, x_n) \in \mathbb{Q}^m\,, \end{aligned}$$

and partition accordingly the n columns of A as $A = (A_N\ A_B)$. Letting $\mathbf{b}' = \mathbf{b} - A_N\mathbf{x}_N$, the solution $\mathbf{x} = (\mathbf{x}_N, \mathbf{x}_B)$ satisfies

$$A_B\mathbf{x}_B = \mathbf{b}'. \tag{5}$$

Observe that $A_B \in \mathbb{Z}^{m\times m}$ is a square matrix with trivial kernel (that is, $A\mathbf{x} = \mathbf{0}$ only for $\mathbf{x} = \mathbf{0}$), thus invertible. Therefore, $\mathbf{x}_B = A_B^{-1}\mathbf{b}'$. (Otherwise, there is a direction $\mathbf{y}$ in the kernel such that both $\mathbf{x} + \epsilon\mathbf{y}$ and $\mathbf{x} - \epsilon\mathbf{y}$ are feasible, hence $\mathbf{x}$ was not a vertex.) Hence, in order to bound the fractionality of the vertex $\mathbf{x}$, it is enough to bound the fractionalities of the entries of A_B^{-1}. Therefore, to bound the fractionality of (1), it is sufficient to bound the fractionality of the inverse of any full rank square sub-matrix of the constraint matrix A. We will denote with $\mathrm{frac}(A)$ the *fractionality* of A, meaning the maximum denominator appearing over all entries, represented as fractions in lowest terms, of A. We will start by showing Theorem 2 for the case of primal treedepth, i.e., taking into account the discussion thus far (together with the fact that the treedepth of any sub-matrix of A is bounded by the treedepth of A) it is sufficient to show that:

Lemma 10. *Let A be a square matrix with full rank having a* td*-decomposition F of $G_P(A)$. Then, $\mathrm{frac}(A^{-1})$ is at most $(\|A\|_\infty)^b b^{b/2}$, where $b = \min\{\mathrm{tl}(F)^{\mathrm{th}(F)+1}(\mathrm{th}(F)!), \mathrm{ht}(F)!\}$.*

Remark 11. *Note that to show the bound stated in Theorem 2, it is sufficient to show the lemma for $b = (\mathrm{ht}(F)!)$. However, the bound given in Lemma 10 allows us to obtain better bounds for important special cases. For instance, for the case of 2-stage stochastic and n-fold ILP, we obtain that $\mathrm{frac}(A^{-1}) \leq (\|A\|_\infty)^{2t^3}(2t^3)^{t^3}$ since $\mathrm{th}(F) = 2$ and $t = \mathrm{tl}(F)$ is the block size.*

The main idea for the proof of Lemma 10 is to show that the matrix A contains a small sub-matrix A' with at most b columns and rows such that the fractionality of A^{-1} is at most the fractionality of $(A')^{-1}$, which can be bounded using Proposition 5. Towards showing this, we will employ a pruning procedure that works along the td-decomposition F of $G_P(A)$ in a bottom-up manner. The crucial ingredient of this procedure is given in Lemma 13 that in essence allows us to remove all but at most $\mathrm{dt}_F(v)$ many children (together with the columns and rows induced by the variables contained in the sub-trees below those children) of any non-degenerate vertex v of F. The following lemma shows a general property for the fractionality of the inverse of a matrix that makes this pruning step possible.

Lemma 12. *Let $A \in \mathbb{Z}^{n\times n}$ be a square matrix with full rank of the form $\begin{pmatrix} B & 0 \\ R & A_D \end{pmatrix}$, where A_D is a block diagonal matrix. Then, there is a block A_B in A_D such that $\mathrm{frac}(A^{-1}) \leq \mathrm{frac}(A_R^{-1})$, where A_R is obtained from A after removing all columns and rows from A that are in A_D but not in A_B.*

Proof. Note that both B and A_D are full rank square matrices because A_D is a square matrix and A is a full rank square

matrix. By elementary matrix calculus, the inverse of A is given by $\begin{pmatrix} B^{-1} & 0 \\ R' & A_D^{-1} \end{pmatrix}$, where $R' = -A_D^{-1} \cdot R \cdot B^{-1}$.

Let e be an entry of A^{-1} with the maximum fractionality (among all entries in A^{-1}). If e is contained in B^{-1}, then setting A_B to an arbitrary block of A_D satisfies the claim of the lemma. If e is in A_D^{-1}, then setting A_B to be the block in A_D containing e satisfies the lemma. This is because A_D is block diagonal, and therefore the inverse of A_D is the block diagonal matrix of the inverses of the blocks. Finally, if e is contained in R', then because $R' = -A_D^{-1} \cdot R \cdot B^{-1}$, the entry e of R' is obtained by multiplying a row r of A_D^{-1} with a column of $R \cdot B^{-1}$. Therefore and because R has only integer entries, setting A_B to be the block of A_D having a non-zero entry at row r satisfies the claim of the lemma. □

For a set of variables V, the sub-matrix of A *induced* on V contains all columns that correspond to a variable in V projected onto all rows of A that have a non-zero entry in at least one column in V.

Lemma 13. *Let $A \in \mathbb{Z}^{n \times n}$ be a square matrix with full rank having a* td*-decomposition F of $G_P(A)$, let v be a non-degenerate vertex of F and let C_v be the set of all children of v in F. Then there is a set C of at most $\mathrm{dt}_F(v)$ children of v in F such that the sub-matrix A_P of A obtained after removing all rows and columns in the sub-matrix of A induced on the set of all variables occurring in any sub-tree of F rooted at a child in $C_v \setminus C$, satisfies:*

- *A_P is a square matrix with full rank,*
- $\mathrm{frac}(A^{-1}) \leq \mathrm{frac}(A_P^{-1})$.

Proof. Let A_v be the sub-matrix of A induced on all variables occurring in the sub-tree of F rooted at v. Then A is of the form $\begin{pmatrix} B & 0 \\ R & A_v \end{pmatrix}$, since all rows not in A_v only have zero entries at all columns in A_v. Let r be the number of non-zero columns in R. Note that $r \leq \mathrm{dt}_F(v) - 1$ and because of Lemma 9, we obtain that A_v is of the form (block-structure), with $d = |C|$, and where $\bar{A}_j$ only contains the column corresponding to the variable v. Consider a block A_j with dimensions $m_j \times n_j$. Since A has full rank, $m_j \geq n_j$. Otherwise, the columns of A_j would not be linearly independent in A. Because A has full rank, we also obtain that $r + 1 + \sum_{j=1}^{|C|} n_j = \sum_{j=1}^{|C|} m_j$. Therefore, the number r' of different values for j such that $m_j > n_j$ is at most $r + 1$. W.l.o.g., we can assume that the first r' inequalities are strict and consequently A_v has the form $\begin{pmatrix} B' & 0 \\ R' & A_D \end{pmatrix}$, where A_D is a block diagonal square matrix (consisting of the blocks $A_{r'+1}, \ldots, A_{|C|}$) and B' consists only of the blocks $A_1, \ldots, A_{r'}$. Note that A now has the form $\begin{pmatrix} B & 0 \\ R & A_D \end{pmatrix}$ and satisfies the conditions in Lemma 12. Let A_k be the block of A_D, whose existence is ensured by Lemma 12. We claim that setting C to the children corresponding to the blocks $A_1, \ldots, A_{r'}, A_k$ satisfies the statement of the lemma. Indeed, $|C| \leq r' + 1 \leq \mathrm{dt}_F(v)$. Moreover, A_P is a square matrix with full rank because so is A and the removed blocks A_j are squares. Finally, $\mathrm{frac}(A^{-1}) \leq \mathrm{frac}(A_P^{-1})$ by Lemma 12. □

The following lemma now shows how to apply the reduction given in Lemma 13 along the td-decomposition F, to obtain a sub-matrix of A with at most b columns and rows.

Lemma 14. *Let $A \in \mathbb{Z}^{n \times n}$ be a square matrix with full rank having a* td*-decomposition F of $G_P(A)$. Then there exists a sub-matrix A_P of A having at most $b = \min\{\mathrm{tl}(F)^{\mathrm{th}(F)+1}(\mathrm{th}(F)!), \mathrm{ht}(F)!\}$ columns and rows such that $\mathrm{frac}(A^{-1}) \leq \mathrm{frac}(A_P^{-1})$.*

Proof. Note that Lemma 13 allows us to reduce the size of A while not decreasing the fractionality of its inverse as long as F contains a non-degenerate vertex v with more than $\mathrm{dt}_F(v)$ children. To see this let A_P be the sub-matrix of A obtained after applying the lemma for some non-degenerate vertex v of F. Then A_P together with the td-decomposition obtained from F after removing the sub-trees rooted by a child in $C_v \setminus C$ again satisfy the conditions in the statement of the lemma and moreover $\mathrm{frac}(A^{-1}) \leq \mathrm{frac}(A_P^{-1})$. Let A_P be the sub-matrix obtained from A after applying the reduction rule given by Lemma 13 exhaustively and let F_P be the td-decomposition of $G_P(A_P)$. Then $\mathrm{frac}(A^{-1}) \leq \mathrm{frac}(A_P^{-1})$ and moreover every vertex v in F_P has at most $\mathrm{dt}_F(v)$ children, which implies that F_P has at most $b = \min\{\mathrm{tl}(F)^{\mathrm{th}(F)+1}(\mathrm{th}(F)!), \mathrm{ht}(F)!\}$ vertices. Therefore, A_P has at most b columns (and rows) and satisfies the statement of the lemma. □

We are now ready to show Lemma 10.

Proof of Lemma 10. Let A_P be the sub-matrix of A, whose existence is ensured by Lemma 14. Because $\mathrm{frac}(A^{-1}) \leq \mathrm{frac}(A_P^{-1})$, it suffices to provide the bound for $\mathrm{frac}(A_P^{-1})$. Recall that A_P has at most b columns and rows. Therefore, by Proposition 5, the fractionality of the inverse of A_P is at most $(\|A\|_\infty)^b b^{b/2}$, as required. □

The following corollary shows that the fractionality can be bounded in the same manner in terms of the treedepth of the dual graph.

Corollary 15. *Let A be a square matrix with full rank having a* td*-decomposition F of $G_D(A)$. Then, $\mathrm{frac}(A^{-1})$ is at most $(\|A\|_\infty)^b b^{b/2}$, where $b = \min\{\mathrm{tl}(F)^{\mathrm{th}(F)+1}(\mathrm{th}(F)!), \mathrm{ht}(F)!\}$.*

Proof. Because $G_P(A^\intercal) = G_D(A)$, we obtain that F is a td-decomposition of $G_P(A^\intercal)$. Therefore, Lemma 10 implies that $\mathrm{frac}((A^\intercal)^{-1})$ is at most $(\|A\|_\infty)^b b^{b/2}$. The corollary now follows because $(A^{-1})^\intercal = (A^\intercal)^{-1}$. □

Theorem 2 now follows immediately from Lemma 10 and Corollary 15, which allows us to conclude with the proof of our main tractability result of this section.

Proof of Theorem 1. Theorem 2 gives us an exact bound M' on the largest coefficient of the (2) instance, and it is clear that the structure of non-zeroes (hence the primal and dual

graphs) of the constraint matrix of (2) is identical to that of A.

Hence, by Lemma 6, (1) can be solved by solving (2), which can be done (by the results of (Eisenbrand et al. 2019)) in FPT time parameterized by $\|A\|_\infty$ and $\min\{\mathrm{td}_P(A), \mathrm{td}_D(A)\}$). (To be precise, we need to solve (2) for every $1 \leq \tilde{M} \leq M'$.) □

Piece-wise Linear Separable Convex Objectives

A generalisation of (1) to non-linear objectives is

$$\min\{f(\mathbf{x}) \mid A\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{Z}^z \times \mathbb{Q}^q\}, \quad (6)$$

and here we focus on the case when f is separable convex, meaning $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i)$ with $f_i : \mathbb{R} \to \mathbb{R}$ univariate convex for each $i \in [n]$. Moreover, we assume that f is piece-wise linear with breakpoints at integer points, i.e., for every $a \in \mathbb{R}$ and $i \in [n]$, $f_i(a) = \{a\}f_i(\lfloor a \rfloor) + (1 - \{a\})f_i(\lceil a \rceil)$, where $\{a\} = a - \lfloor a \rfloor$.

We adapt a variable transformation of Hochbaum and Shantikumar (Hochbaum and Shantikumar 1990) to show that (6) admits a linearization that retains the fractionality of the original linear instance. This transformation was originally used to show that integer separable convex minimisation can be reduced to integer linear minimisation when A has small sub-determinants, but our use differs in three aspects: our variables are mixed integer, the matrix A may have large sub-determinants, and most importantly, we only use it to obtain a fractionality bound; we never need to solve the newly constructed instance. Specifically, we will transform an input (6) into a (1) whose parameters we define next:

$$\min\left\{\mathbf{c}\mathbf{y} \mid \hat{A}\mathbf{y} = \hat{\mathbf{b}}, \mathbf{0} \leq \mathbf{y} \leq \mathbf{1}, \mathbf{y} \in \mathbb{Z}^{\hat{z}} \times \mathbb{Q}^{\hat{q}}\right\} \quad (7)$$

The hatted data are obtained as follows: For each $i \in [z+q]$, replace the variable x_i with $u_i - l_i$ variables y_i^j, $j \in [u_i - l_i]$. Hence, in (7), and the number of integer variables is $\hat{z} = \sum_{i=1}^{z} u_i - l_i$, the number of continuous variables is $\hat{q} = \sum_{i=z+1}^{z+q} u_i - l_i$. We define the column of $\hat{A}$ corresponding to the variable y_i^j.The lower and upper bound for all variables is 0 and 1, respectively. Let the right-hand side be $\hat{\mathbf{b}} = \mathbf{b} - A\mathbf{l} = \sum_{i=1}^{z+q} A_i l_i$. Finally, the coefficient in the objective function $\mathbf{c}$ for variable y_i^j is the slope of f_i between points $l_i + (j-1)$ and $l_i + j$. Specifically, $c_i^j = f_i(l_i + j) - f_i(l_i + (j-1))$. Define a mapping $\varphi : \mathbb{Z}^z \times \mathbb{Q}^q \to \mathbb{Z}^{\hat{z}} \times \mathbb{Q}^{\hat{q}}$, as follows: given $\mathbf{x} \in \mathbb{Z}^z \times \mathbb{Q}^q$, $\varphi(\mathbf{x}) = \mathbf{y}$, where for each $i \in [z+q]$, $j \in [u_i - l_i]$, $y_i^j = \max\{0, \min\{1, x_i - l_i - (j-1)\}\}$.

Lemma 16. *1. A vector* $\mathbf{x}$ *is feasible in* (6) *iff* $\varphi(\mathbf{x})$ *is feasible in* (7)*.*

2. *If* $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$*, then* $f(\mathbf{x}) = \mathbf{c}\varphi(\mathbf{x}) + \sum_{i=1}^{z+q} f_i(l_i)$*.*
3. *Let* $\mathbf{x}^*$ *be an optimum of* (6)*. Then* $\varphi(\mathbf{x}^*)$ *is an optimum of* (7)*.*

The proof amounts to careful checking of the construction, and is deferred to the full version.

Lemma 17. *Every square sub-matrix* A' *of* $\hat{A}$ *of full rank has* $\mathrm{td}_P(A') \leq \mathrm{td}_P(A)$ *and* $\mathrm{td}_D(A') \leq \mathrm{td}_D(A)$*.*

Proof. For A' to have full rank, it cannot contain duplicate columns. Hence, A' is also a square sub-matrix of A, a case in which we have already shown the claim to hold. □

Proof of Theorem 4. By this lemma, the fractionality M of (7) is bounded by $\mathrm{frac}(A)$, and by Lemma 16 and the definition of φ, $\mathrm{frac}(A)$ is also a fractionality bound on (1) when f is separable convex piece-wise linear with integer breakpoints. Let $\hat{f}$ be defined component-wise from f as follows: for $i \in [1, z]$, let $\hat{f}_i = f_i$, and for $i \in [z+1, z+q]$, let $\hat{f}_i(x_i) = f_i(x_i/M)$. Then, to solve (1) in this regime, it is enough to optimise $\hat{f}$ over (2). □

Limits of the "Bounded Fractionality" Approach

In this section, we show the limits of our "bounded fractionality" approach. We start by showing its limits for various important known tractable classes of ILP, i.e., the class of small primal treewidth and domain (Jansen and Kratsch 2015), small incidence treewidth and largest solution prefix sum (Ganian, Ordyniak, and Ramanujan 2017), small signed clique-width of the incidence graph (Eiben et al. 2018), and the class of 4-block n-fold matrices (Hemmecke, Köppe, and Weismantel 2014). We show that all these classes exhibit unbounded fractionality.

Lemma 18. *For every* $n \in \mathbb{N}$*, there are MILP instances* I_1 *and* I_2 *with constraint matrices* A_1 *and* A_2*, such that* A_1 *has constant primal, dual, and incidence treewidth and signed incidence clique-width and* $\|A_1\|_\infty = 2$*, and* A_2 *is 4-block* n*-fold with all blocks being just* (1)*, and the fractionality is* $2^{\Omega(n)}$ *for* I_1 *and* $\Omega(n)$ *for* I_2*.*

Proof. Consider the $n \times n$ matrix

$$A_1 = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ 0 & 2 & -1 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & & 2 \end{pmatrix},$$

It is easy to verify that the matrix B with $B_{ij} = 2^{i-j-1}$ for $i \leq j$ and $B_{ij} = 0$ otherwise is the inverse of A_1. Moreover, the primal, dual, incidence treewidth of A_1 is at most 1, the signed incidence clique-width of A_1 is at most 2, and $\|A_1\|_\infty = 2$.

It is again easy to verify that below are A_2 and its inverse, both $n \times n$, with $n' = n - 2$, and A_2 is a 4-block n-fold matrix with all blocks of size 1:

$$A_2 = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \\ 1 & 0 & 0 & \cdots & 1 \end{pmatrix},$$

$$A_2^{-1} = \begin{pmatrix} -\frac{1}{n'} & \frac{1}{n'} & \frac{1}{n'} & \cdots & \frac{1}{n'} \\ \frac{1}{n'} & \frac{n'-1}{n} & -\frac{1}{n'} & \cdots & -\frac{1}{n'} \\ \frac{1}{n'} & -\frac{1}{n'} & \frac{n'-1}{n} & \cdots & -\frac{1}{n'} \\ \vdots & & & \ddots & \\ \frac{1}{n'} & -\frac{1}{n'} & -\frac{1}{n'} & \cdots & \frac{n'-1}{n} \end{pmatrix}.$$

Because for each vertex $\mathbf{x}$ of a polyhedron there exists an objective vector $\mathbf{c}$ such that (1) is uniquely optimal in $\mathbf{x}$, and the fact that we have demonstrated inverses with high fractionality, there must exist vertices of high fractionality and corresponding objectives, which give the desired instances I_1 and I_2. □

Remark 19. *The $\Omega(n)$ fractionality lower bound in part 2 of Lemma 18 may be seen as mild given that for 4-block n-fold we would seek an algorithm running in time $n^{f(k)}$, for f some function and k largest block size, and that (the more permissive) n-fold IP problem has such an algorithm even when its entries are polynomial in n. However, this is not true for the 2-stage stochastic IP problem, which is* NP*-hard with polynomially bounded coefficients already with constant-size blocks (Dvorák et al. 2017). Because 4-block n-fold IP is at least as hard as 2-stage stochastic IP, the bounded fractionality approach cannot work for giving an $n^{f(k)}$ algorithm for 4-block n-fold MILP.*

We now show the limits of our approach for generalising our results from MILP to MIP for certain types of separable convex functions.

Lemma 20. *There are MIP instances with the following properties:*

1. *$A = (1 \cdots 1)$, $b = 1$, $f(\mathbf{x}) = \sum_i (x_i)^2$, $\mathrm{td}_D(A) = 1$, fractionality n,*
2. *dimension 1, no constraints, $f(x) = (x - \frac{1}{k})^2$, fractionality k,*
3. *dimension 1, no equality constraints, $0 \leq x \leq 1$, $f(x) = x^3 + 2x^2 - x$ univariate cubic convex, unbounded fractionality (minimum is $\frac{\sqrt{7}}{3} - \frac{2}{3}$).*

Proof. All instances have unique optima, and it is straightforward to verify that in part 1 of the Lemma, it is the point $\mathbf{x} = (\frac{1}{n}, \ldots, \frac{1}{n})$, in part 2 it is $x = \frac{1}{k}$, for any k, and in part 3, the minimum is irrational $x = \frac{\sqrt{7}}{3} - \frac{2}{3}$, hence fractionality is unbounded. The objective $f(x) = x^3 + 2x^2 - x$ is not convex on $\mathbb{R}$, but it is between 0 and 1. □

The Limits of Tractability for Structured MILPs

It is well-known that MILP is fixed-parameter tractable parameterized by the number of integer variables. It is therefore natural to ask, whether for our Theorem 1 it could be sufficient to only put restrictions on the integer part of the instance. Here, we show that this is not the case. We show hardness for the feasibility version of MILP, which is deciding the non-emptiness of the set $\{\mathbf{x} \in \mathbb{Z}^z \times \mathbb{Q}^q \mid A\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$.

Lemma 21. *Let $\mathcal{C}$ be a class of ILP instances for which the feasibility problem is* NP*-hard. Then there exists a class of MILP instances $\mathcal{C}'$ whose feasibility problem is* NP*-hard and whose constraint matrix is $A = \begin{pmatrix} \mathbf{0} & A_{\mathbb{Q}} \\ I & -I \end{pmatrix}$, where I is the identity matrix and $A_{\mathbb{Q}}$ is a constraint matrix of an instance from $\mathcal{C}$.*

The proof is deferred to the full version of the paper.

Remark 22. *It is an interesting question for future work whether we can generalise our results for MILP if we put additional restrictions on the interactions between integer and non-integer variables. A similar approach has recently been explored for generalising the tractability result for ILP based on primal treedepth to MILP (Ganian, Ordyniak, and Ramanujan 2017) using a hybrid decompositional parameter called torso-width.*

Open Problems

We close with three open problems motivating future research. First, what is the complexity of general MIP for matrices with bounded primal and dual treedepth? Our Lemma 20 shows that a different approach is needed. Second, is 4-block n-fold MILP in XP? At first sight, it may seem that to get an XP algorithm, it should suffice to bound the fractionality by $\mathrm{poly}(n)$ (and nothing better is possible by Lemma 18). However, the current XP algorithm for the pure integer case depends *exponentially* on the largest coefficient of the constraint matrix, so solving (2) would be too slow. Third, Lemma 21 suggests that new tractable fragments of MILP may be characterized by having bounded interaction between the integer and continuous variables. Hence, we ask: what is the complexity of MILP where $A_{\mathbb{Z}}$ comes from an ILP tractable fragment, $A_{\mathbb{Q}}$ is arbitrary, and the number of rows which are nonzero in both the integer and continuous variables is small? If this is hard, what restraints need to be placed on $A_{\mathbb{Q}}$ to obtain a tractable fragment?

Acknowledgements

Cornelius Brand was supported by OP RDE project No. CZ.02.2.69/0.0/0.0/18_053/0016976 International mobility of research, technical and administrative staff at the Charles University.

Martin Koutecký was partially supported by Charles University project UNCE/SCI/004 and by the project 19-27871X of GA ČR. Sebastian Ordyniak acknowledges support from the Engineering and Physical Sciences Research Council (EPSRC, project EP/V00252X/1).

References

Alumur, S. A.; and Kara, B. Y. 2008. Network hub location problems: The state of the art. *European Journal of Operational Research* 190(1): 1–21.

Aschenbrenner, M.; and Hemmecke, R. 2007. Finiteness theorems in stochastic integer programming. *Foundations of Computational Mathematics* 7(2): 183–227.

Bazaraa, M. S.; Sherali, H. D.; and Shetty, C. M. 2013. *Nonlinear programming: theory and algorithms*. John Wiley & Sons.

Bixby, R. E. 2002. Solving real-world linear programs: A decade and more of progress. *Operations research* 50(1): 3–15.

Bredereck, R.; Faliszewski, P.; Niedermeier, R.; Skowron, P.; and Talmon, N. 2020. Mixed integer programming with convex/concave constraints: Fixed-parameter tractability and applications to multicovering and voting. *Theor. Comput. Sci.* 814: 86–105. doi:10.1016/j.tcs.2020.01.017. URL https://doi.org/10.1016/j.tcs.2020.01.017.

Chen, L.; and Marx, D. 2018. Covering a tree with rooted subtrees - parameterized and approximation algorithms. In Czumaj, A., ed., *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, 2801–2820. SIAM.

Chen, L.; Marx, D.; Ye, D.; and Zhang, G. 2017. Parameterized and Approximation Results for Scheduling with a Low Rank Processing Time Matrix. In Vollmer, H.; and Vallée, B., eds., *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, 22:1–22:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer.

Dvorák, P.; Eiben, E.; Ganian, R.; Knop, D.; and Ordyniak, S. 2017. Solving Integer Linear Programs with a Small Number of Global Variables and Constraints. In Sierra, C., ed., *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 607–613. ijcai.org.

Eiben, E.; Ganian, R.; Knop, D.; and Ordyniak, S. 2018. Unary Integer Linear Programming with Structural Restrictions. In Lang, J., ed., *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 1284–1290. ijcai.org.

Eisenbrand, F.; Hunkenschröder, C.; Klein, K.; Koutecký, M.; Levin, A.; and Onn, S. 2019. An Algorithmic Theory of Integer Programming. *CoRR* abs/1904.01361.

Eisenbrand, F.; Hunkenschröder, C.; and Klein, K.-M. 2018. Faster Algorithms for Integer Programs with Block Structure. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, 49:1–49:13.

Floudas, C.; and Lin, X. 2005. Mixed Integer Linear Programming in Process Scheduling: Modeling, Algorithms, and Applications. *Annals of Operations Research* 139(1): 131–162. ISSN 0254-5330.

Ganian, R.; and Ordyniak, S. 2018. The complexity landscape of decompositional parameters for ILP. *Artif. Intell.* 257: 61–71.

Ganian, R.; Ordyniak, S.; and Ramanujan, M. S. 2017. Going Beyond Primal Treewidth for (M)ILP. In Singh, S. P.; and Markovitch, S., eds., *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, 815–821. AAAI Press.

Guillemot, S. 2011. FPT algorithms for path-transversal and cycle-transversal problems. *Discrete Optimization* 8(1): 61–71.

Hemmecke, R. 2001. *On the decomposition of test sets*. Ph.D. thesis, Universität Duisburg.

Hemmecke, R. 2003. On the positive sum property and the computation of Graver test sets. *Math. Program.* 96(2): 247–269.

Hemmecke, R.; Köppe, M.; and Weismantel, R. 2014. Graver basis and proximity techniques for block-structured separable convex integer minimization problems. *Mathematical Programming* 145(1-2, Ser. A): 1–18. ISSN 0025-5610.

Hemmecke, R.; Onn, S.; and Romanchuk, L. 2013. N-fold integer programming in cubic time. *Mathematical Programming* 1–17.

Hochbaum, D. S.; and Shantikumar, J. G. 1990. Convex Separable Optimization Is Not Much Harder than Linear Optimization. *J. ACM* 37(4): 843–862.

Hu, T. C. 1963. Multi-Commodity Network Flows. *Operations Research* 11(3): 344–360.

Iwata, Y.; Wahlstrom, M.; and Yoshida, Y. 2016. Half-integrality, LP-branching, and FPT algorithms. *SIAM Journal on Computing* 45(4): 1377–1411.

Iwata, Y.; Yamaguchi, Y.; and Yoshida, Y. 2018. 0/1/All CSPs, Half-Integral A-Path Packing, and Linear-Time FPT Algorithms. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, 462–473.

Jansen, B. M. P.; and Kratsch, S. 2015. A Structural Approach to Kernels for ILPs: Treewidth and Total Unimodularity. In Bansal, N.; and Finocchi, I., eds., *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, 779–791. Springer.

Jansen, K.; Klein, K.; Maack, M.; and Rau, M. 2018. Empowering the Configuration-IP - New PTAS Results for Scheduling with Setups Times. In Blum, A., ed., *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, 44:1–44:19. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Kannan, R. 1987. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research* 12(3): 415–440. ISSN 0364-765X.

Karzanov, A. V. 1998. Minimum 0-extensions of graph metrics. *European Journal of Combinatorics* 19(1): 71–101.

Knop, D.; and Koutecký, M. 2018. Scheduling meets n-fold integer programming. *J. Scheduling* 21(5): 493–503.

Knop, D.; Koutecký, M.; and Mnich, M. 2017a. Combinatorial n-fold Integer Programming and Applications. In Pruhs, K.; and Sohler, C., eds., *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, 54:1–54:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Knop, D.; Koutecký, M.; and Mnich, M. 2017b. Voting and Bribing in Single-Exponential Time. In Vollmer, H.; and Vallée, B., eds., *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, 46:1–46:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Kotnyek, B. 2002. *A generalization of totally unimodular and network matrices.* Ph.D. thesis, London School of Economics and Political Science (United Kingdom).

Koutecký, M.; Levin, A.; and Onn, S. 2018. A Parameterized Strongly Polynomial Algorithm for Block Structured Integer Programs. In Chatzigiannakis, I.; Kaklamanis, C.; Marx, D.; and Sannella, D., eds., *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, 85:1–85:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

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

Lin, M.-H.; Carlsson, J. G.; Ge, D.; Shi, J.; and Tsai, J.-F. 2013. A review of piecewise linearization methods. *Mathematical problems in Engineering* 2013.

Lodi, A.; Martello, S.; and Monaci, M. 2002. Two-dimensional packing problems: A survey. *European Journal of Operational Research* 141(2): 241–252.

Nemhauser, G. L.; and Trotter, L. E. 1974. Properties of vertex packing and independence system polyhedra. *Mathematical Programming* 6(1): 48–61.

Papadimitriou, C. H. 1981. On the complexity of integer programming. *J. ACM* 28(4): 765–768.

Pilipczuk, M.; and Ziobro, M. 2018. Experimental Evaluation of Parameterized Algorithms for Graph Separation Problems: Half-Integral Relaxations and Matroid-based Kernelization. *arXiv preprint arXiv:1811.07779* .

Toth, P.; and Vigo, D., eds. 2001. *The Vehicle Routing Problem*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics. ISBN 0-89871-498-2.

van den Briel, M.; Vossen, T.; and Kambhampati, S. 2005. Reviving Integer Programming Approaches for AI Planning: A Branch-and-Cut Framework. In *Automated Planning and Scheduling, ICAPS 2005. Proceedings*, 310–319. AAAI.

Vossen, T.; Ball, M. O.; Lotem, A.; and Nau, D. S. 1999. On the Use of Integer Programming Models in AI Planning. In *International Joint Conference on Artificial Intelligence, IJCAI 99. Proceedings*, 304–309. Morgan Kaufmann.

Separable Convex Mixed-Integer Optimization: Improved Algorithms and Lower Bounds

Cornelius Brand
Chair of Algorithms and Complexity Theory, Faculty for Informatics and Computer Science, University of Regensburg, Germany

Martin Koutecký
Computer Science Institute, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Alexandra Lassota
Eindhoven University of Technology, The Netherlands

Sebastian Ordyniak
University of Leeds, UK

Abstract

We provide several novel algorithms and lower bounds in central settings of mixed-integer (non-)linear optimization, shedding new light on classic results in the field. This includes an improvement on record running time bounds obtained from a slight extension of Lenstra's 1983 algorithm [Math. Oper. Res. '83] to optimizing under few constraints with small coefficients. This is important for ubiquitous tasks like knapsack–, subset sum– or scheduling problems [Eisenbrand and Weismantel, SODA'18, Jansen and Rohwedder, ITCS'19].

Further, we extend our algorithm to an intermediate linear optimization problem when the matrix has many rows that exhibit 2-stage stochastic structure, which adds to a prominent line of recent results on this and similarly restricted cases [Jansen et al. ICALP'19, Cslovjecsek et al. SODA'21, Brand et al. AAAI'21, Klein, Reuter SODA'22, Cslovjecsek et al. SODA'24]. We also show that the generalization of two fundamental classes of structured constraints from these works (n-fold and 2-stage stochastic programs) to separable-convex mixed-integer optimization are harder than their mixed-integer, linear counterparts. This counters a widespread belief popularized initially by an influential paper of Hochbaum and Shanthikumar, namely that "convex separable optimization is not much harder than linear optimization" [J. ACM '90].

To obtain our algorithms, we employ the mixed Graver basis introduced by Hemmecke [Math. Prog. '03], and our work is the first to give bounds on the norm of its elements. Importantly, we use these bounds differently from how purely-integer Graver bounds are exploited in related approaches, and prove that, surprisingly, this cannot be avoided.

2012 ACM Subject Classification Theory of computation → Convex optimization

Keywords and phrases Mixed-Integer Programming, Separable Convex Optimization, Parameterized Algorithms, Parameterized Complexity

Digital Object Identifier 10.4230/LIPIcs.ESA.2024.32

Related Version *Full Version*: https://arxiv.org/abs/2111.08048v1

Funding *Martin Koutecký*: Partially supported by Charles Univ. project UNCE 24/SCI/008 and by the project 22-22997S of GA ČR.
Sebastian Ordyniak: Supported by the Engineering and Physical Sciences Research Council (EPSRC, project EP/V00252X/1).

1 Introduction

We study the Mixed Integer Programming problem, which asks to minimize a (linear or non-linear) objective function over a linearly constrained, *mixed-integer* set of numerical variables, that is, some of them are required to be integral, while others may be fractional. The case where all variables are required to remain integral is the *purely-integer* case. This problem is of enormous importance for applications, as Bixby [3] says in his famous analysis of speed-ups for *linear* programming solvers: *"[I]nteger programming, and most particularly the mixed-integer variant, is the dominant application of linear programming in practice[,]"*. Moreover, concerning non-linear optimization, Bertsimas et al. note in their spectacular work [2] on the notorious subset selection problem in statistical learning that, over the past decades, algorithmic and hardware advances have elevated *convex* (and therefore, in particular, non-linear) mixed-integer optimization to a comparable level of relevance in applications.

Despite its practical ubiquity, the algorithmic theory of both linear and non-linear mixed integer optimization is much less developed compared to purely-integer problems. Indeed, the little insight we do have into the algorithmics of the mixed-integer case so far derives mainly from the purely-integer case (see the discussion of related work below): The best algorithms relevant to the setting of our article follow as a corollary of a 40 year-old result, namely, Lenstra's famous algorithm [23] (which can be extended, e.g. using [14, Theorem 6.7.9], to the setting of arbitrary convex target functions).

In this article, we focus on central special cases of the problem. We embark from the important case when there are only few constraints with small entries, which plays a key role in ubiquitous algorithmic applications such as scheduling problems, subset sum problems and knapsack-type problems, and has increasingly come into focus in recent years [13, 19]. The gained insight in this domain is then supercharged to attack a linear, mixed-integer optimization problem that arises in analogy to a long and prominent line of recent works on purely-integer problems with structured sets of constraints [10, 5, 9, 8, 12, 20].

Our Contributions

We design faster algorithms for non-linear mixed-integer optimization in the case where there are only few constraints, and the coefficients appearing in them are of small absolute value (Theorem 1). Building on this, we give parameterized algorithms for a generalized, hard *linear* mixed-integer optimization problem (Theorem 2). In addition, we prove new lower bounds for separable convex optimization problems, showing that the result on polynomial-time solvability of mixed-integer linear programming [4] subject to constraints of bounded treedepth does *not* translate to the separable convex case (Theorems 3 and 4). This may come as a surprise, considering the generally observed, tight connection between tractability for separable convex and linear target functions in the case of both fully continuous and purely-integer optimization (see [16, 6]), which might make it tempting to conjecture a similarly close relationship also in the mixed-integer case.

In terms of technical innovation, our algorithmic results are based on novel insights into as-of-yet poorly understood mixed Graver bases. We prove that, somewhat unexpectedly, the usual manner in which these bases are employed in the literature to aid the design of algorithms for integer programming problems can *not* yield much in the mixed-integer setting. One key contribution of our algorithmic results is that they demonstrate how to circumvent this limitation, and how to exploit the mixed Graver bases algorithmically nonetheless.

Importantly, and in contrast to most results in the mixed-integer realm to date, our results do *not* rely on expanding the results for purely-integral cases, which we prove to have only limited power, but instead, use new insights about the structure of the mixed-integer problem itself.

Results on Algorithms and Complexity of MIPs. To set the stage somewhat more formally, we consider with the following problem:

$$\min f(\mathbf{x}) : E\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{Z}^{n_\mathbb{Z}} \times \mathbb{R}^{n_\mathbb{R}}. \tag{MIP}$$

Here the number of columns is $n = n_\mathbb{Z} + n_\mathbb{R}$, the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is *separable convex*, that is, $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i)$ for some sequence of *univariate* convex functions $f_1, \ldots, f_n : \mathbb{R} \to \mathbb{R}$, $E \in \mathbb{Z}^{m \times n}$, E denotes the constraint matrix, $\mathbf{b} \in \mathbb{R}^m$ is the right-hand side, and the lower and upper bounds are $\mathbf{l}, \mathbf{u} \in (\mathbb{R} \cup \{\pm\infty\})^n$.

We set $\mathbb{X} = \mathbb{Z}^{n_\mathbb{Z}} \times \mathbb{R}^{n_\mathbb{R}}$, where $n_\mathbb{Z}$ and $n_\mathbb{R}$ should be clear from the context. An important special case is that of an integral right-hand side $\mathbf{b} \in \mathbb{Z}^m$ and bounds $\mathbf{l}, \mathbf{u} \in \mathbb{Z}^n$, and a linear target function $f(\mathbf{x}) = \mathbf{w}\mathbf{x} = \sum_i w_i x_i$. In this setting, (MIP) specializes to

$$\min \mathbf{w}\mathbf{x} : E\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \quad \mathbf{x} \in \mathbb{X}. \tag{MILP}$$

On the front of algorithms for this problem, we improve the current, double-exponential record bound for mixed-integer programs with few rows and small coefficients to single-exponential, even when the target function is non-linear:[1]

▶ **Theorem 1** (Algorithm for MIPs with few rows). *The problem* (MIP) *can be solved in single-exponential time* $(m\|E\|_\infty)^{\mathcal{O}(m^2)} \cdot \mathcal{R}$*, where* $\mathcal{R}$ *is the time needed to solve the continuous relaxation of any* (MIP) *with the constraint matrix* E*.*

Until now, the best way to solve a (MIP) with few rows and small coefficients would be to remove duplicate columns from E in a preprocessing step, and then use Lenstra's 1983 algorithm for mixed integer programming [23]. Since there are $2\|E\|_\infty + 1$ numbers of absolute value at most $\|E\|_\infty$, the preprocessing ensures that there are at most $(2\|E\|_\infty + 1)^m$ columns in E. This, however, leads to a *double-exponential* running time in terms of m.

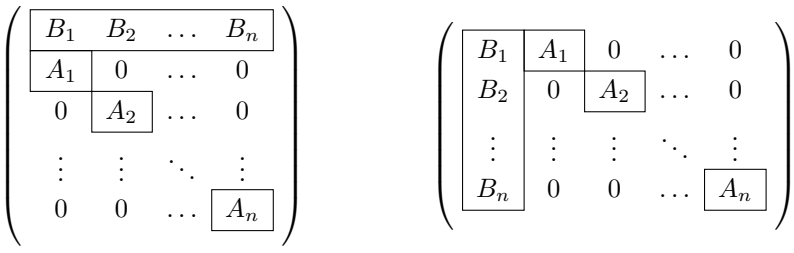
Moreover, we use the above algorithm as a starting point for developing a novel algorithm for an intermediate problem. Namely, we now allow the bounds $\mathbf{l}, \mathbf{u} \in \mathbb{X}$ and right-hand side $\mathbf{b} \in \mathbb{R}^m$ to be fractional, that is, we consider the problem

$$\min \mathbf{w}\mathbf{x} : E\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \quad \mathbf{x} \in \mathbb{X}. \tag{MILP$_{\text{frac}}$}$$

Already deciding feasibility of this variant has been shown to be NP-hard for totally unimodular matrices [7]. We are interested in algorithms that deal with constraint structures that were extensively treated in recent works in the purely integer setting [10, 5, 9, 8, 12, 20]. Namely, *n-fold* and *2-stage stochastic* matrices with bounded block-size, as depicted in Figure 1.[2] The matrices A_i and B_i in Figure 1 are called the *blocks* of the constraint matrices; furthermore, n denotes the number of blocks A_i and B_i. For the case of 2-stage stochastic constraints, we prove:

[1] As is common in the literature, we use the term *single-exponential* in x for functions of the form $2^{\text{poly}(x)}$, as opposed to e.g. $2^{O(x)}$. Similarly, we call exponential towers of height two, that is, $2^{2^{\text{poly}(x)}}$ *double-exponential* in x.

[2] Formal definitions of all terms used in the introduction will be given in the preliminaries.



(a) An n-fold matrix

(b) A 2-stage stochastic matrix

Figure 1 The A_i and B_i are matrices of dimension bounded by a parameter. Note that n-folds and 2-stage stochastic matrices are transpositions of each other.

▶ **Theorem 2** (Algorithm for 2-stage stochastic ($\text{MILP}_{\text{frac}}$))**.** *The problem* ($\text{MILP}_{\text{frac}}$) *where E is a 2-stage stochastic matrix with block-dimensions $B_i \in \mathbb{Z}^{t\times r}$ and $A_i \in \mathbb{Z}^{t\times s}$ can be solved in time $g(r, s, \|E\|_\infty) \cdot n^r$, for some computable function g.*

Turning to lower bounds, we show that this result is likely optimal:

▶ **Theorem 3** (Hardness for 2-stage stochastic MIP)**.** *The problem* (MIP) *with integral data is* W[1]*-hard when E is a 2-stage stochastic matrix with blocks of size bounded by a parameter and $\|E\|_\infty = 1$ already for linear objective functions.*

In particular, under the common parameterized complexity assumption that FPT $\neq$ W[1] holds, this rules out algorithms for (MIP) with running times of the form $g(k) \cdot \text{poly}(n)$, where k is the maximal block-dimension of each B_i, A_i in the 2-stage stochastic constraint matrices. Such a (double exponential) algorithm does exist for the pure integer case [12].

Moreover, we prove that the algorithm from Theorem 2 cannot be extended to the related case of n-fold constraint structure:

▶ **Theorem 4** (NP-hardness for n-fold MIP)**.** *The problem* (MIP) *with integral data is* NP*-hard when E is an n-fold matrix with blocks of constant dimensions and $\|E\|_\infty = 1$ already for linear objective functions.*

Interestingly, the above hardness results demonstrate that the relationship between n-folds and 2-stage stochastic programs in the mixed case is different from purely-integer case: In the purely integer case, n-folds are solvable faster (in time FPT and single-exponentially [8]) than 2-stage stochastic programs [18], while in the mixed-integer case, the situation seems to be reversed.

Results on Mixed Graver Bases. Our algorithmic approach uses the mixed Graver basis of the constraint matrix. This is a mixed analogue of the usual integral Graver basis, which is a central object in all the recent developments around block-structured integer programs. Deeper insights into the Graver basis have led to new dynamic data structures [12], proximity theorems [8, 9, 12, 20, 21] and better convergence rate analyses [12]. Intuitively speaking, the elements of the Graver basis comprise all possible improving directions that have to be considered by an algorithm that seeks to iteratively augment suboptimal solutions.

The mixed Graver basis was introduced by Hemmecke [15] already in 2003, but not understood well enough to be used. On our way to showing Theorem 2, we prove several results about the mixed Graver basis which are of independent interest, and disprove the

typical intuitions gained by studying the ordinary integral Graver basis. First, all elements of the *integral* Graver basis of an n-fold matrix with bounded block-dimension also have entries of bounded absolute value, whence they derive their algorithmic usefulness. We show that this is not true for the mixed Graver basis:

▶ **Theorem 5** (n-fold mixed Graver lower bound). *There is an n-fold matrix E with constant-sized blocks and $\|E\|_\infty = 1$ such that the mixed Graver basis of E contains an element with 1-norm of size $\Omega(n)$.*

On the other hand, for 2-stage stochastic matrices, the ∞-norm of its elements can be bounded by a function of the block-dimensions and $\|E\|_\infty$:

▶ **Theorem 6** (2-stage stochastic mixed Graver upper bound). *For any 2-stage stochastic matrix E, the maximum ∞-norm of an element of its mixed Graver basis is bounded by $h(r, s, \|E\|_\infty)$ for some computable function h.*

This bound also implies a proximity result: for any integer optimum $\mathbf{z}^*$, there is a nearby mixed optimum $\mathbf{x}^*$. Thus, we can first find $\mathbf{z}^*$ (which can be done efficiently), and then only search in a small neighborhood around $\mathbf{z}^*$.

Until now, a bound such as $h(r, s, \|E\|_\infty)$ on the Graver elements has always led to an algorithm with a corresponding running time $h(r, s, \|E\|_\infty)\,\mathrm{poly}(n)$. However, in the mixed case, such an algorithm is ruled out by Theorem 3. This shows that, in the mixed case, the common intuition of good bounds on the Graver norm directly leading to fast algorithms fails.

Related Work

We have already pointed to the most directly related recent works on block-structured (integer) linear programming. For an overview on the vast literature concerning practical attempts to deal with mixed-integer programming, the excellent article of Bertsimas et al. [2] provides pointers to relevant literature on this fascinating matter. We now sketch the theoretical literature in the field to contextualize the results obtained in the present paper, and in particular, how they contrast the typical, expected relationship between linear and convex optimization results observed heuristically in other situations. Since the limited insight we do have into the mixed-integer case so far derives mainly from the purely-integer case, we emphasize the comparison to the literature treating the latter setting to highlight patterns of lifting algorithmic results along two axes: From *purely-integer* to *mixed* domains, and from *linear* to *convex* objectives.

Towards the first axis, one first has to mention Lenstra's [23] algorithm for purely-integer *linear* optimization, which was seminal for the entire area. Notably, it extends to mixed-integer domains with a fixed number of integer coordinates. By the same token, there are other cases besides fixed integer dimension where tractability lifts from the purely-integer to the mixed case. For one, this includes the by-now classic polynomial-time solvability of linear integer optimization with totally unimodular constraint matrices [17]. In addition, it was shown more recently [4] how to obtain efficient algorithms for mixed-integer linear optimization subject to constraints of a particular structure (namely bounded treedepth, which includes n-fold and 2-stage stochastic programs), by leveraging the flurry of tractability results on purely-integer optimization in structurally restricted settings that we already pointed out above.

We now turn to the second axis, linear versus convex optimization. Indeed, as mentioned already above, the algorithm of Lenstra [23] for mixed- and purely-integer optimization with a fixed number of purely-integer coordinates can also be extended (e.g. using [14,

Theorem 6.7.9]) to the setting of arbitrary convex target functions. In a similar vein, a general result of Hochbaum and Shanthikumar [16] establishes the following: Whenever the linear integer optimization problem with constraint matrices of bounded subdeterminants is polynomial-time solvable, then so is also the non-linear integer optimization problem for target functions that are *separable convex* (that is, a sum of univariate convex functions on the coordinates). Their paper's eponymous heuristic observation that "convex separable optimization is not much harder than linear optimization" has since become common wisdom, further consolidated e.g. by Chubanov's result on reducing linear to separable convex optimization over the fully (non-mixed) continuous domain. What is more, also the series of results on integer optimization subject to constraints of bounded treedepth mentioned above apply, especially, to separable convex target functions. It is worth noting that, since it is already NP-hard to optimize a *quadratic* (non-separable) convex function over the boolean hypercube $\{0,1\}^n$ [12, Proposition 101], and the constraint matrix describing (the facets of) the hypercube is both totally unimodular and of bounded treedepth, the algorithms for separable convex optimization are most likely not extensible to general convex objectives.

We emphasize that our results shed light on the algorithmic properties of mixed-integer optimization that defies the intuition that the body of work outlined above may suggest. Indeed, our results can be interpreted to mean that mixed-integer separable-convex optimization behaves rather unexpectedly from this point of view.

Organization

We give all necessary preliminaries in Sect. 2. Then, we give new results on mixed Graver bases and algorithmic consequences for mixed-integer linear programs with few rows in Sect. 3. In Sect. 4, we then extend this to an algorithm for the 2-stage stochastic case, and Sect. 5 contains a matching lower bound. In Sects. 6 and 7, we prove both complexity and Graver norm lower bounds for the n-fold case.

Due to the page limit, we postpone the proofs of some statements to the full version.

2 Preliminaries

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). Any (MIP) instance with infinite bounds $\mathbf{l}, \mathbf{u}$ can be reduced to an instance with finite bounds using standard techniques in polynomial time (solving the continuous relaxation and using proximity bounds to restrict the relevant region). So from now on we assume finite bounds $\mathbf{l}, \mathbf{u} \in \mathbb{X}$ with $\mathbb{X} = \mathbb{Z}^{n_\mathbb{Z}} \times \mathbb{R}^{n_\mathbb{R}}$

The set of indices at which $\mathbf{x}$ is non-zero is the *support* of $\mathbf{x}$, denoted $\operatorname{supp}(\mathbf{x})$. For positive integers $m \leq n$ we set $[m, n] := \{m, \dots, n\}$ and $[n] := [1, n]$, and we extend this notation to vectors: for $\mathbf{l}, \mathbf{u} \in \mathbb{Z}^n$ with $\mathbf{l} \leq \mathbf{u}$, $[\mathbf{l}, \mathbf{u}] := \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$. 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 $\log := \log_2$. We define $\lfloor x \rceil$ to be $\lfloor x \rfloor$ if $x \geq 0$ and $\lceil x \rceil$ otherwise, and we define the *fractional part of* x to be $\{x\} := x - \lfloor x \rceil$. The division of variables into integer and continuous ones induces a division of the constraint matrix $E = (E_\mathbb{Z}\ E_\mathbb{R})$ where $E_\mathbb{Z} \in \mathbb{Z}^{m \times n_\mathbb{Z}}$ and $E_\mathbb{R} \in \mathbb{R}^{m \times n_\mathbb{R}}$, and analogously $\mathbf{x} = (\mathbf{x}_\mathbb{Z}, \mathbf{x}_\mathbb{R})$ and $f(\mathbf{x}) = f_\mathbb{Z}(\mathbf{x}_\mathbb{Z}) + f_\mathbb{R}(\mathbf{x}_\mathbb{R})$. More generally, whenever we make reference to any subset E' of columns or even submatrix of E, we will freely denote with $E'_\mathbb{Z}$ and $E'_\mathbb{R}$ the analogous division of E' into its integral and fractional part, respectively. Throughout, we assume that the rows of E are linearly independent.

We consider n-fold and 2-stage stochastic matrices. A matrix is of 2-stage stochastic structure if non-zero entries appear only in the first r columns and in n blocks of size $t \times s$ along the diagonal beside. The overall size is $nt \times (r + sn)$. An n-fold matrix is the transpose of a 2-stage stochastic matrix. It has thus $(r + sn)$ rows and nt columns. For an illustration, see Figure 1.

A vector $\mathbf{g} \in \ker(E) \setminus \{\mathbf{0}\}$ is a *circuit of* E if it is integral, its entries are co-prime, and it is support-minimal, that is, there is no vector $\mathbf{g}' \in \ker(E) \setminus \{\mathbf{0}\}$ with $\mathrm{supp}(\mathbf{g}') \subset \mathrm{supp}(\mathbf{g})$; let $\mathcal{C}(E)$ denote the set of circuits of E. For two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we say that $\mathbf{x}$ *is conformal to* $\mathbf{y}$ and write $\mathbf{x} \sqsubseteq \mathbf{y}$ if, for each $i \in [n]$, $|x_i| \leq |y_i|$ and $x_i \cdot y_i \geq 0$. Intuitively, $\mathbf{x}$ and $\mathbf{y}$ are in the same orthant, and $\mathbf{y}$ is at least as far from $\mathbf{0}$ as $\mathbf{x}$ in each coordinate. We say that $\mathbf{x} = \sum_i \mathbf{g}_i$ is a *conformal sum* or a *conformal decomposition of* $\mathbf{x}$ if, for all i, $\mathbf{g}_i \sqsubseteq \mathbf{x}$. For an arbitrary set S, we write $\ker_S(E)$ as a shorthand for $\ker(E) \cap S$. In particular, the *mixed kernel of* E is defined as $\ker_{\mathbb{X}}(E)$. The *Graver basis of* E, denoted $\mathcal{G}(E)$, is the set $\mathcal{G}(E) = \{\mathbf{g} \in \ker_{\mathbb{Z}^n}(E) \setminus \{\mathbf{0}\} \mid \mathbf{g} \text{ is } \sqsubseteq\text{-minimal}\}$.

▶ **Definition 7** (Mixed Graver basis [15]). *Let* $E = (E_{\mathbb{Z}}\ E_{\mathbb{R}}) \in \mathbb{Z}^{m \times n}$. *The* mixed Graver basis $\mathcal{G}_{\mathbb{X}}(E)$ of E with respect to $\mathbb{X}$ *consists of all vectors* $(\mathbf{0}, \mathbf{g}_{\mathbb{R}})$, *where* $\mathbf{g}_{\mathbb{R}} \in \mathcal{C}(E_{\mathbb{R}})$, *together with all vectors* $(\mathbf{g}_{\mathbb{Z}}, \mathbf{g}_{\mathbb{R}}) \in \ker_{\mathbb{X}}(E)$ *such that* $\mathbf{g}_{\mathbb{Z}} \neq \mathbf{0}$ *and there is no* $(\mathbf{g}'_{\mathbb{Z}}, \mathbf{g}'_{\mathbb{R}}) \in (\ker_{\mathbb{X}}(E) \setminus \{\mathbf{0}\})$ *(unequal to* $(\mathbf{g}_{\mathbb{Z}}, \mathbf{g}_{\mathbb{R}})$*) such that* $(\mathbf{g}'_{\mathbb{Z}}, \mathbf{g}'_{\mathbb{R}}) \sqsubseteq (\mathbf{g}_{\mathbb{Z}}, \mathbf{g}_{\mathbb{R}})$.

For any p, $1 \leq p \leq \infty$, define $g_p^{\mathbb{X}}(E) := \max_{\mathbf{g} \in \mathcal{G}_{\mathbb{X}}(E)} \|\mathbf{g}\|_p$.

The following is a helpful trick to reduce a ($\mathrm{MILP}_{\mathrm{frac}}$) to a (MIP) with integer input data and a constraint matrix $(E\ I)$.

▶ **Lemma 8.** *Let an* ($\mathrm{MILP}_{\mathrm{frac}}$) *instance be given. It is possible to construct an equivalent* (MIP) *instance in linear time with a constraint matrix* $E' = (E\ I)$, *bounds* $\mathbf{l}', \mathbf{u}' \in \mathbb{Z}^{n+m}$, *and a right-hand side* $\mathbf{b}' \in \mathbb{Z}^m$.

Proof sketch. The proof works by first moving fractional right-hand sides into the lower and upper bounds. Then, we relax any fractional lower and upper bounds to their closest integers, and penalize violations of the original bounds in the new objective function, which is what introduces non-linearity (the resulting function is piece-wise linear convex with 3 pieces in each coordinate). ◀

3 The Basic Case: Matrices with Few Rows and Small Coefficients

This section develops the basic version of our algorithmic result. We begin by giving upper bounds for a certain notion of decompositions of elements in the mixed Graver basis, and then employ these bounds to our algorithmic ends.

3.1 Mixed-Graver Bound

We begin with an upper bound on the 1-norm for matrices with few rows and small coefficients. For this, we will need the Steinitz lemma:

▶ **Proposition 9** (Steinitz [26], Sevastjanov, Banaszczyk [25]). *Let* $\|\cdot\|$ *be any norm, and let* $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ *be such that* $\|\mathbf{x}_i\| \leq 1$ *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* $k \in [n]$, $\|\sum_{i=1}^k \mathbf{x}_{\pi(i)}\| \leq d$.

▶ **Lemma 10.** *Let* $E \in \mathbb{Z}^{m \times (n_{\mathbb{Z}} + n_{\mathbb{R}})}$. *Then every* $\mathbf{g} \in \mathcal{G}_{\mathbb{X}}(E)$ *satisfies*

$$\|\mathbf{g}\|_1 \leq (2m\|E\|_\infty(2\|E\|_\infty + 1)^m + 1)^m + (2\|E\|_\infty + 1)^m$$

Proof. Let $\mathbf{g} \in \mathcal{G}_{\mathbb{X}}(E)$ and assume that all columns of E are distinct; we will show how to deal with doubled columns later. We define a sequence of vectors in the following manner: If $g_i \geq 0$, we add $\lfloor g_i \rfloor$ copies of the i-th column of E to the sequence, if $g_i < 0$ we add $|\lceil g_i \rceil|$ copies of the negation of column i to the sequence. Thus, for each $i \in [n]$, we obtained vectors $\mathbf{v}^i_1, \ldots, \mathbf{v}^i_{\lfloor g_i \rfloor}$. Finally, we add the vector $\mathbf{o} = \sum_{i=1}^n \{g_i\} E_{\bullet,i}$ to the sequence. Notice that this vector is integral. Let q be the number of vectors in this sequence.

Clearly, the sequence of vectors sums up to $\mathbf{0}$ as it exactly corresponds to $E\mathbf{g}$ and $\mathbf{g} \in \ker_{\mathbb{X}}(E)$. Moreover, their ℓ_∞-norm is bounded by $\|E\|_\infty(2\|E\|_\infty + 1)^m$ since there are at most $(2\|E\|_\infty + 1)^m$ distinct columns, $\|E\|_\infty$ is the largest number appearing in any of them, and this is an upper bound on any number appearing in $\mathbf{o} = \sum_{i=1}^n \{g_i\} E_{\bullet,i}$. The remaining vectors $\mathbf{v}^i_j$ are bounded by $\|E\|_\infty$ in ℓ_∞-norm.

Using the Steinitz Lemma, there is a reordering $\mathbf{u}^1, \ldots, \mathbf{u}^q$ (i. e., $\mathbf{v}^i_j = \mathbf{u}^{\pi(i,j)}$ for some permutation π) of this sequence such that each prefix sum $\mathbf{p}_k := \sum_{j=1}^k \mathbf{u}^j$ is bounded by $m\|E\|_\infty(2\|E\|_\infty + 1)^m$ in the l_∞-norm. Clearly,

$$\left|\{\mathbf{x} \in \mathbb{Z}^m \mid \|\mathbf{x}\|_\infty \leq m\|E\|_\infty(2\|E\|_\infty + 1)^m\}\right| = (2m\|E\|_\infty(2\|E\|_\infty + 1)^m + 1)^m =: P.$$

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

$$g'_i := \begin{cases} g'_i + 1 & \text{if } \pi^{-1}(\ell) = (i,j) \text{ and } g_i \geq 0 \\ g'_i - 1 & \text{if } \pi^{-1}(\ell) = (i,j) \text{ and } g_i < 0 \\ g'_i + \{g_i\} & \text{if } \mathbf{u}^\ell = \mathbf{o}, \text{ for each } i \in [n] \end{cases}.$$

Here, (i,j) indicates the j-th copy of the i-th vector. Similarly obtain $\mathbf{g}''$ from the sequence $\mathbf{u}^{\alpha+1} \ldots, \mathbf{u}^\beta$. We have $E\mathbf{g}'' = \mathbf{0}$, as $\mathbf{p}_\alpha - \mathbf{p}_\beta = \mathbf{0}$ and thus, $\mathbf{g}'' \in \ker_{\mathbb{X}}(E)$ and hence, $\mathbf{g}' \in \ker_{\mathbb{X}}(E)$. 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 $\mathbf{g}$ which is a condition needed for $\mathbf{g} \in \mathcal{G}_{\mathbb{X}}(E)$, hence $q \leq P$. Notice that only one of $\mathbf{g}'$ or $\mathbf{g}''$ may be fractional, as $\mathbf{o}$ will be in exactly one subsequence. For each of the at most $(2\|E\|_\infty + 1)^m$ columns, the respective fractional part in $\mathbf{g}$ contributes less than 1, so it follows that $\|\mathbf{g}\|_1 < P + (2\|E\|_\infty + 1)^m$ holds.

We are left to deal with the situation that E contains doubled columns. The solution is to adjust the construction of the sequence accordingly. Fix a column $E_{\bullet,i}$ and let S be the set of all indices j such that $E_{\bullet,i} = E_{\bullet,j}$. Let $u = \sum_{j \in S} g_j$. If $u > 0$, add $\lfloor u \rfloor$ copies of $E_{\bullet,i}$ into the sequence, else add $|\lceil u \rceil|$ copies of $-E_{\bullet,i}$ into the sequence. The contribution of this column type to $\mathbf{o}$ will be $\{u\}E_{\bullet,i}$. Since $-1 < \{u\} < 1$ for each column type, and the number of column types is bounded by $(2\|E\|_\infty + 1)^m$, our previous arguments hold. ◀

The proof of the above Lemma actually shows that there exists a particular decomposition of every element of $\ker_{\mathbb{X}}(E)$ into an element of $\ker_{\mathbb{Z}^n}(E)$ (which can be further decomposed into elements of $\mathcal{G}(E)$) and one element of $\ker_{\mathbb{X}}(E)$, which we can bound. This mixed element might not be an element of $\mathcal{G}_{\mathbb{X}}(E)$, and a bound on the elements of $\mathcal{G}_{\mathbb{X}}(E)$ does not imply a bound on this element. We crucially need this property in our proximity bound and the bounds on $\mathcal{G}_{\mathbb{X}}(E)$ for 2-stage matrices, as well as the prospect of extending these to multi-stage matrices. Thus, this emerges as an important feature:

▶ **Definition 11** (One-fat decomposition bound). *Let* $\mathbf{x} \in \ker_{\mathbb{X}}(E)$. *We say that* $\mathbf{x} = \mathbf{h} + \mathbf{g}$ *is a* one-fat decomposition *if it is a conformal decomposition,* $\mathbf{h} \in \ker_{\mathbb{X}}(E)$ *and* $\mathbf{g} \in \ker_{\mathbb{Z}^n}(E)$, *and we call* $\mathbf{h}$ *the* fat element *of the decomposition. For every* p, $1 \leq p \leq \infty$, *define* $\mathbf{wt}^{\mathbb{X}}_p(\mathbf{x}) = \min \|\mathbf{h}\|_p$, *where the minimum goes over all one-fat decompositions of* $\mathbf{x}$. *Define the* ℓ_p-weight of E with respect to $\mathbb{X}$ *as* $\mathbf{wt}^{\mathbb{X}}_p(E) = \max_{\mathbf{x} \in \ker_{\mathbb{X}}(E)} \mathbf{wt}^{\mathbb{X}}_p(\mathbf{x})$.

▶ **Corollary 12.** *For any matrix E, $\mathbf{wt}_1^{\mathbb{X}}(E) \leq (2m\|E\|_\infty(2\|E\|_\infty+1)^m+1)^m$.*

Proof. Note that if $\mathbf{x} \in \ker_{\mathbb{X}}(E)$ is decomposable, then it has a decomposition into conformal $\mathbf{g}', \mathbf{g}''$, only one of which is fractional. Iterating this, we obtain the decomposition of $\mathbf{x}$ into several elements of $\mathcal{G}_{\mathbb{Z}^n}(E)$, and one element of $\ker_{\mathbb{X}}(E)$ which is bounded as stated. ◀

We will obtain a better bound on both $g_1^{\mathbb{X}}(E)$ and the ℓ_1-weight of E, using a recent result:

▶ **Proposition 13** ([24, Lemma 1])**.** *Let $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{Z}^d$ and $\alpha_1, \ldots, \alpha_n \in \mathbb{R}_+$ such that $\sum_{i=1}^n \alpha_i \mathbf{x}^i \in \mathbb{Z}^d$. If $\sum_{i=1}^n \alpha_i > d$, then there exist numbers $\beta_1, \ldots, \beta_n \in \mathbb{R}_+$ such that, for all $i \in [n]$, $\beta_i \leq \alpha_i$ and $\sum_{i=1}^n \beta_i \leq d$, and $\sum_{i=1}^n \beta_i \mathbf{x}^i \in \mathbb{Z}^d$.*

An iterated use of this lemma gives rise to the following statement:

▶ **Lemma 14** (Packing Lemma)**.** *Let $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{Z}^d$ and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{R}_+^n$ such that $\sum_{i=1}^n \alpha_i \mathbf{x}^i \in \mathbb{Z}^d$. If $\sum_{i=1}^n \alpha_i > d$, there exist vectors $\boldsymbol{\beta}^1, \ldots, \boldsymbol{\beta}^m \in \mathbb{R}_+^n$ such that, for each $j \in [m]$, $\boldsymbol{\beta}^j \leq \boldsymbol{\alpha}$, $\sum_{i=1}^n \beta_i^j \mathbf{x}^i \in \mathbb{Z}^d$, $\|\boldsymbol{\beta}^j\|_1 \leq d$, and $\sum_{j=1}^m \boldsymbol{\beta}^j = \boldsymbol{\alpha}$. Moreover, for all but at most one $j \in [m]$, $\|\boldsymbol{\beta}^j\|_1 \geq d/2$.*

Proof. The only potentially non-obvious part is the last sentence of the statement. Notice that if there are $\boldsymbol{\beta}^j$ and $\boldsymbol{\beta}^{j'}$, $j \neq j'$, with $\|\boldsymbol{\beta}^j\|_1, \|\boldsymbol{\beta}^{j'}\|_1 \leq d/2$, then we can merge them. Formally, we set $\boldsymbol{\beta}^j := \boldsymbol{\beta}^j + \boldsymbol{\beta}^{j'}$, and delete $\boldsymbol{\beta}^{j'}$. ◀

Intuitively, the lemma allows us to take a non-negative linear combination of integer vectors whose result is an integer vector, and divide it into smaller such combinations while preserving the property that each smaller combination still results in an integer vector.

▶ **Lemma 15.** *Let $E \in \mathbb{Z}^{m \times (n_{\mathbb{Z}}+n_{\mathbb{R}})}$. Then $g_1^{\mathbb{X}}(E) \leq (2m^2\|E\|_\infty+1)^{m+1}$ and $\mathbf{wt}_1^{\mathbb{X}} \leq (2m^2\|E\|_\infty+1)^{2m+2}$.*

Proof sketch. As in Lemma 10, we will construct a sequence of vectors summing up to zero and then apply the Steinitz Lemma. However, this time we will use the Packing Lemma 14 to obtain a better bound on each element of the vector sequence and thus, a better bound on the elements of $\mathcal{G}_{\mathbb{X}}(E)$ overall. Unfortunately, this approach does not yield a one-fat decomposition, so we have to use the Steinitz Lemma in a more clever way to get a bound on $\mathbf{wt}_1^{\mathbb{X}}(E)$. ◀

The one-fat decomposition also allows us to prove a bound on the distance between an integer and mixed optimum, which we will use in both of our algorithmic results:

▶ **Lemma 16** (MIP Proximity)**.** *Let $\mathbf{z}^* \in \mathbb{Z}^n$ be an integer optimum of a* (MIP) *instance, and let $\mathbf{x}^*$ be a mixed optimum closest to $\mathbf{z}^*$ in ℓ_p-norm, $1 \leq p \leq \infty$. Then $\|\mathbf{z}^* - \mathbf{x}^*\|_p \leq \mathbf{wt}_p^{\mathbb{X}}(E)$.*

We will need a small technical proposition before we prove Lemma 16:

▶ **Proposition 17** ([12, Proposition 60])**.** *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 of Lemma 16. Assume for contradiction that $\|\mathbf{z}^* - \mathbf{x}^*\|_p > \mathbf{wt}_p^{\mathbb{X}}(E)$. Since $(\mathbf{z}^* - \mathbf{x}^*) \in \ker_{\mathbb{X}}(E)$, it has a one-fat decomposition $\mathbf{h}+\mathbf{g}$ where $\|\mathbf{h}\|_p \leq \mathbf{wt}_p^{\mathbb{X}}(E)$. As $\|\mathbf{z}^* - \mathbf{x}^*\|_p > \mathbf{wt}_p^{\mathbb{X}}(E)$, the integral part $\mathbf{g}$ is non-zero. Let $\hat{\mathbf{z}} := \mathbf{z}^* - \mathbf{g} = \mathbf{x}^* + \mathbf{h}$ and $\hat{\mathbf{x}} := \mathbf{x}^* + \mathbf{g} = \mathbf{z}^* - \mathbf{h}$. Thus, $\mathbf{z}^* - \mathbf{x}^* = \mathbf{h} + \mathbf{g} = (\mathbf{z}^* - \hat{\mathbf{x}}) + (\mathbf{z}^* - \hat{\mathbf{z}})$. Now Proposition 17 with $\mathbf{x} = \mathbf{x}^*$, $\mathbf{y}_1 = \mathbf{h}$, $\mathbf{y}_2 = \mathbf{g}$ shows

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

By the conformality of the decomposition, $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ are within the $\mathbf{l}, \mathbf{u}$ bounds. As $\mathbf{g} \in \ker_{\mathbb{Z}^n}(E)$, $\hat{\mathbf{z}}$ is an integer feasible solution, and because $\mathbf{h} \in \ker_{\mathbb{X}}(E)$, $\hat{\mathbf{x}}$ is a mixed feasible solution. Furthermore, because $\mathbf{z}^*$ was an integer optimum and $\hat{\mathbf{z}}$ is integer feasible, the left hand side is non-positive, and so is $f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)$, thus $\hat{\mathbf{x}}$ must be another mixed optimum and the right hand side must be zero, and so the left hand side, showing $\hat{\mathbf{z}}$ to be another integer optimum. However, $\hat{\mathbf{x}}$ is closer to $\mathbf{z}^*$, a contradiction. ◀

3.2 A Single-Exponential Algorithm

Armed with the bounds on the mixed Graver basis and our insights into one-fat decompositions, we are now ready to develop the single-exponential algorithm. Before we do so, however, a few general remarks are in order. These also apply to the two-stage stochastic algorithm for fixed block-dimensions later on.

▶ Remark 18. Both algorithmic results make use of the fact that if both the mixed and the integer version of the problem are feasible, then for every integral optimum, there is a mixed optimum nearby. It then suffices to first solve the (generally easier) integral version of the problem, and then solve an auxiliary mixed-integer program with the feasible region bounded by a small n-dimensional box around $\mathbf{x}$. Indeed, if $\mathbf{x}$ is an integral solution of $E\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$, then we will resort to solving the program $E(\mathbf{x}+\mathbf{y}) = \mathbf{b}$, $\|\mathbf{y}-\mathbf{x}\|_\infty \leq P$, $\mathbf{l} \leq \mathbf{x}+\mathbf{y} \leq \mathbf{u}$ for $\mathbf{y}$, which amounts to finding $\mathbf{y}$ with $E\mathbf{y} = \mathbf{0}$, $\mathbf{l}' \leq \mathbf{y} \leq \mathbf{u}'$ for some new bounds $\mathbf{l}', \mathbf{u}'$ such that $\|\mathbf{l}' - \mathbf{u}'\|_\infty$ is small. For general objectives, one optimizes the auxiliary objective $f'(\mathbf{y}) = f(\mathbf{x}+\mathbf{y})$, whereas for linear objectives no change is needed. Hence, all of the algorithmic heavy lifting will be done in order to solve problems of this form.

Of course, this strategy rests on the assumption that both the mixed and the integral variant of the problem are feasible. This assumption can in turn be removed by a standard two-phase approach, similar to what is customary e.g. for the Simplex algorithm, in order to find an initial feasible solution. In short, this is done by introducing slack variables that are penalized in the objective, but admit a trivial feasible solution. In the sequel, we will hence always assume feasibility.

We say that $\mathbf{x}_\epsilon$ is an *ϵ-accurate solution to* (MIP) if there exists an optimum $\mathbf{x}^*$ such that $\|\mathbf{x}^* - \mathbf{x}_\epsilon\|_\infty \leq \epsilon$. (For a discussion on the relationship of ϵ-accurate and ϵ-approximate optima and also the motivation to use the notion of ϵ-accuracy, see [16, Section 1.2].)

▶ **Theorem 1** (Algorithm for MIPs with few rows)**.** *The problem* (MIP) *can be solved in single-exponential time $(m\|E\|_\infty)^{\mathcal{O}(m^2)} \cdot \mathcal{R}$, where $\mathcal{R}$ is the time needed to solve the continuous relaxation of any* (MIP) *with the constraint matrix E.*

Proof of Theorem 1. The integer problem can be solved in time $(m\|E\|_\infty)^{\mathcal{O}(m^2)} + \mathcal{R}(\epsilon)$ by known techniques [12, 13] where $\mathcal{R}(\epsilon)$ is the ϵ-accurate solution to the continuous relaxation – essentially, first solve the continuous relaxation, then reduce $\mathbf{b}, \mathbf{l}, \mathbf{u}$ using proximity bounds, then solve a dynamic program. Now by Lemma 16, a mixed optimum $\mathbf{x}^*$ is at most $\mathbf{wt}_1^{\mathbb{X}}(E) \leq (2m^2\|E\|_\infty + 1)^{2m+2} =: P$ far in 1-norm. The proximity bound implies that all prefix sums of $\mathbf{x}^*_{\mathbb{Z}}$ with $E_{\mathbb{Z}}$ belong to the integer box $R := [-\|E\|_\infty \cdot P, \|E\|_\infty \cdot P]^m$, which has at most $(2\|E\|_\infty \cdot P + 1)^m = (m\|E\|_\infty)^{\mathcal{O}(m^2)}$ elements.

This allows us to construct a dynamic program with $n_{\mathbb{Z}} + 1$ stages. Our DP table D shall have an entry $D(i, \mathbf{r})$ for $i \in [n_{\mathbb{Z}}]$ and $\mathbf{r} \in R$ whose meaning is the minimum objective attainable if the prefix sum of $\mathbf{x}^*_{\mathbb{Z}}$ and $E_{\mathbb{Z}}$ restricted to the first i coordinates is $\mathbf{r}$. To that end, for all $\mathbf{r} \in R$, define $x_i^*(\mathbf{r})$ to be the choice of $x_i^* \in [-P, P]$ which minimizes f_i and such that $E_{\bullet,i}x_i^* = \mathbf{r}$; it is possible for the solution to be undefined if no number in $[-P, P]$

satisfies the conditions. Similarly, define $\mathbf{x}^*_{\mathbb{R}}(\mathbf{r})$ to be an ϵ-accurate minimizer of $f_{\mathbb{R}}$ satisfying $E_{\mathbb{R}}\mathbf{x}^*_{\mathbb{R}} = \mathbf{r}$. To compute D, set $D(0,\mathbf{r}) := 0$ for $\mathbf{r} = \mathbf{0}$ and $D(0,\mathbf{r}) := +\infty$ otherwise, and for $i \in [n_{\mathbb{Z}}]$, set

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

The last stage is defined as

$$D(n_{\mathbb{Z}}+1,\mathbf{0}) := \min_{\substack{\mathbf{r}',\mathbf{r}'' \in R: \\ \mathbf{r}'+\mathbf{r}''=\mathbf{0}}} D(n_{\mathbb{Z}},\mathbf{r}') + f_{\mathbb{R}}(\mathbf{x}^*_{\mathbb{R}}(\mathbf{r}'')) \ .$$

The value of the optimal solution is $D(n_{\mathbb{Z}}+1,\mathbf{0})$ and the solution $\mathbf{x}^*$ itself can be computed easily with a bit more bookkeeping in the table D.

As for complexity, the first $n_{\mathbb{Z}}$ stages of the DP can be computed in time at most $n_{\mathbb{Z}} \cdot |R|^2 = (m\|E\|_\infty)^{\mathcal{O}(m^2)} n_{\mathbb{Z}}$, and the last stage solves the continuous relaxation $|R|$ times, taking time $|R|\mathcal{R}(\epsilon)$. Altogether, the algorithm takes time at most $(m\|E\|_\infty)^{\mathcal{O}(m^2)}\mathcal{R}(\epsilon)$. Regarding correctness, note that any ϵ-accurate solution $\mathbf{x}^*$ is such that $\mathbf{x}^*_{\mathbb{R}}$ is an ϵ-accurate minimizer of $E_{\mathbb{R}}\mathbf{x}_{\mathbb{R}} = -E_{\mathbb{Z}}\mathbf{x}^*_{\mathbb{Z}}$, $\mathbf{l}_{\mathbb{R}} \le \mathbf{x}_{\mathbb{R}} \le \mathbf{u}_{\mathbb{R}}$, and $\mathbf{x}^*_{\mathbb{Z}}$ is an integer minimizer of $E_{\mathbb{Z}}\mathbf{x}_{\mathbb{Z}} = -E_{\mathbb{R}}\mathbf{x}^*_{\mathbb{R}}$, $\mathbf{l}_{\mathbb{Z}} \le \mathbf{x}_{\mathbb{Z}} \le \mathbf{u}_{\mathbb{Z}}$. Since the algorithm finds exactly such minimizers, its correctness follows. ◀

4 Algorithms for the 2-Stage Stochastic Case

After giving the basic version of our algorithm for the case of few rows, we now develop our algorithm for the case of fixed block-dimension. We first prove our bound of the mixed Graver basis:

▶ **Theorem 6** (2-stage stochastic mixed Graver upper bound). *For any 2-stage stochastic matrix E, the maximum ∞-norm of an element of its mixed Graver basis is bounded by $h(r,s,\|E\|_\infty)$ for some computable function h.*

Proof sketch. We first decompose the element of $\mathcal{G}_{\mathbb{X}}(E)$ blockwise according to Lemma 15. Then, we apply a recent result of [9] on the existence of submultisets with equal sums in certain multisets of vectors with similar (but, notably, not identical) sums. This is made possibly by the fact that we have not only a bound on $g^{\mathbb{X}}_\infty(E)$ but a bound on the weight of a one-fat decomposition. This gives us the desired one-fat decomposition for the 2-stage stochastic case. ◀

From Theorem 6 and Lemma 16, it follows that:

▶ **Corollary 19.** *Let $\mathbf{z}^* \in \mathbb{Z}^n$ be an integer optimum of a 2-stage stochastic* (MIP) *instance. Then there exists a mixed optimum $\mathbf{x}^* \in \mathbb{X}$ such that $\|\mathbf{z}^* - \mathbf{x}^*\|_\infty \le h(r,s,\|E\|_\infty)$ for a double exponential function h.*

4.1 A Polynomial Algorithm for Fixed Block-Dimension

Using the upper bounds for 2-stage stochastic MIPs on proximity and weight as combined in Corollary 19, we can now formulate an algorithm which solves the 2-stage stochastic MILP problem in polynomial time whenever the block-dimensions are fixed. We recall that h is the function from Theorem 6. In accordance with Remark 18, we note two things: Firstly, by following a standard two-phase approach, we may assume that the problem at hand is

integrally feasible. Then, secondly, the algorithm solves the integer program corresponding to the instance to optimality, which is fixed-parameter tractable [1, 22]. We thereby obtain an integer optimum $\mathbf{z}^*$, and we can now restrict ourselves to solving the following auxiliary MILP to optimality:

$$\min \mathbf{wx} : \ E\mathbf{x} = \mathbf{0},\ \hat{\mathbf{l}} \leq \mathbf{x} \leq \hat{\mathbf{u}},\ \mathbf{x} \in \mathbb{Z}^{n_{\mathbb{Z}}} \times \mathbb{R}^{n_{\mathbb{R}}}, \mathbf{l}, \mathbf{u} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{Z}^m. \tag{AuxMILP}$$

Here, $\hat{\ell}_i = \max\{\ell_i - z^*_i, -h(r,s,\|E\|_\infty)\}$ and $\hat{u}_i = \min\{u_i - z^*_i, h(r,s,\|E\|_\infty)\}$. Observe that

$$\|\hat{\mathbf{l}} - \hat{\mathbf{u}}\|_\infty \leq 2h(r,s,\|E\|_\infty) \tag{1}$$

holds. For an optimal solution $\mathbf{x}^*$ to (AuxMILP), the augmented solution $\mathbf{x}^* + \mathbf{z}^*$ is then an optimal solution to the original MILP, by Corollary 19.

What remains is to show how to solve (AuxMILP) in the claimed time bound. This is effected by proving the following Lemma:

▶ **Lemma 20.** *Let V be the set of vertices of all integer slices of the auxiliary mixed-integer program* (AuxMILP)*. There are at most $(8h(r,s,\|E\|_\infty))^{(r+1)(s+1)}n^r$ distinct global parts appearing in V, and they can be enumerated with polynomial delay.*

Proof sketch. The proof goes by analyzing the structure of invertible submatrices of two-stage stochastic matrices. Then, it becomes apparent that the global part of a basic solution is essentially determined by which subset of r blocks out of all n blocks influences the global part. The number of such choices is clearly bounded by n^r. The remainder of the bound stems from various guessing steps, including some of the values for the integer variables. Hence the appearance of h in the bound, making also the bounds from Theorem 6 crucially come into play. ◀

Lemma 20 now suggests an obvious strategy to solve the (AuxMILP) to optimality:

▶ **Proposition 21.** (AuxMILP) *can be solved in time $h(r,s,\|E\|_\infty)^{O(rs)} \cdot n^r$.*

Proof sketch. By Lemma 20, we may enumerate all possible global parts of vertices in the required time bound, guess the corresponding global integer values, and then solve the resulting block-diagonal mixed-integer system to optimality using the algorithm of Theorem 1 (notice that here we are in the special case of LP which can be solved exactly, i.e., with $\epsilon = 0$, and in strongly polynomial time since $\|E\|_\infty$ is small, so $\mathcal{R}(0) = \text{poly}(n)$). Among all choices of global parts, pick the one that yields the optimal value for the full program. ◀

We have now obtained:

▶ **Theorem 2** (Algorithm for 2-stage stochastic ($\text{MILP}_{\text{frac}}$))**.** *The problem* ($\text{MILP}_{\text{frac}}$) *where E is a 2-stage stochastic matrix with block-dimensions $B_i \in \mathbb{Z}^{t\times r}$ and $A_i \in \mathbb{Z}^{t\times s}$ can be solved in time $g(r,s,\|E\|_\infty) \cdot n^r$, for some computable function g.*

Proof. As mentioned before, it is enough to first solve the integer program corresponding to the MILP instances, and then solving the auxiliary problem using Proposition 21. ◀

▶ Remark 22. Let us note two things: Firstly, the exponent of n in our algorithm is only dependent on the number r of global variables. Hence, for values of s such that $h(r,s,\|E\|_\infty)^s \leq n^{f(r)}$ for some function f, our algorithm remains polynomial for fixed r.

Secondly, note that we may choose strongly polynomial (or rather, strongly fpt) subroutines to solve the arising integer and mixed-integer programs. In this case, also the algorithm we obtain is strongly polynomial for fixed block-dimensions.

5 W[1]-Hardness of 2-Stage Stochastic MILPs with Fractional Bounds

In the following we show that 2-stage stochastic ($\text{MILP}_{\text{frac}}$) and (MIP) with integral data is W[1]-hard parameterized by the block-dimension even if $\|E\|_\infty = 1$.

▶ **Theorem 3** (Hardness for 2-stage stochastic MIP). *The problem* (MIP) *with integral data is* W[1]*-hard when E is a 2-stage stochastic matrix with blocks of size bounded by a parameter and $\|E\|_\infty = 1$ already for linear objective functions.*

Proof Sketch of Theorem 3. We show the theorem using a parameterized reduction from the well-known Subset Sum problem, which is W[1]-hard when parameterized by the number of elements in a solution [11].

Subset Sum
Input: A set A of pairwise distinct natural numbers and two natural numbers k and t.
Goal: Decide whether there is a subset $S \subseteq A$ with $|S| = k$ and $\sum_{s \in S} s = t$?

Transformation: We give a formulation of Subset Sum as a 2-stage stochastic MILP. To do so, we first scale all input numbers $a_1, a_2, \ldots, a_n$ in A and t by $1/\max_i\{a_i\}$. Denote the new numbers as $a'_1, a'_2, \ldots, a'_n$ and t'. The scaling ensures that all considered sums are smaller or equal to 1, which comes in handy later on.

Let x^i_j be a binary variable that will indicate that a'_i is the jth number appearing in the sum for all $i \in [n]$ and $j \in [k]$. We collect those numbers not appearing in a solution in a binary slack variable x^i_{k+1} for each $i \in [n]$, yielding the constraints:

$$\sum_{j=1}^{k+1} x^i_j = 1 \qquad \forall i \in [n] \tag{2}$$

To express the condition on the sum of the solution being t', we introduce fractional variables y^i_j that take on the value a'_i if and only if $x^i_j = 1$ for $i \in [n]$ and $j \in [k]$. While this is trivially achieved by $y^i_j = a'_i x^i_j$, the crux is to model this without including a'_i as a coefficient, which would not be bounded by the parameter any more. This is accomplished by requiring the following:

$$y^i_j \leq x^i_j \qquad \forall i \in [n], \forall j \in [k] \tag{3}$$

$$\sum_{j=1}^{k+1} y^i_j = a'_i \qquad \forall i \in [n] \tag{4}$$

This has the intended effect since $a'_i \leq 1$ by construction. We will then store the solution indicated by the assignment to the x^i_j variables in yet another set of variables, denoted as z_j, where j ranges from 1 to k

$$\sum_{j=1}^{k} z_j = t' \tag{5}$$

While it is easy to project the y^i_j to z_j, the straightforward way to do so would blow up the block size to $\Omega(n)$. Indeed, to ensure that the z_j have the intended semantics, consider the following: The equality $z_j = y^i_j$ ought to be satisfied for exactly one choice of i, say when $i = i'$ (assuming distinct inputs); otherwise, $z_j = y^i_j + s^i_j$ holds for some non-zero compensation term s^i_j, whenever $i \neq i'$. Note that, while the s^i_j do satisfy a function similar

to slack variables, they may well need to be negative. In addition, we introduce binary variables r_j^i for all $i \in [n]$ and $j \in [k]$, indicating whether or not $s_j^i = 0$. The above semantics are captured in the following constraints:

$$z_j = y_j^i + s_j^i \qquad \forall i \in [n], \forall j \in [k] \tag{6}$$

$$z_j \geq \min_i a'_i \tag{7}$$

$$-r_j^i \leq s_j^i \leq r_j^i \qquad \forall i \in [n], \forall j \in [k] \tag{8}$$

Our aim is then to minimize the number of times any of the s_j^i are used, or conversely, to make $z_j = y_j^i$ for some i as often as possible, which is expressed in the choice of the objective function

$$\min \sum_{j=1}^{k} \sum_{i=1}^{n} r_j^i \tag{9}$$

As argued, note that in a solution of a yes instance, for a fixed j, $z_j = y_j^i \geq \min_i a'_i$ (equivalently, $r_j^i = 0$) holds for exactly one choice of i, making the optimum equal to $k(n-1)$.

The above constraints define a 2-stage stochastic MILP formulation with fractional variables z_j, y_j^i and s_j^i, and binary variables x_j^i and r_j^i. The global part is made up by the z_j, of which there are k. The remaining variables are distributed across n blocks of dimension $O(k)$ each, including the respective slack variables for the inequality constraints. The largest entry in the constraint matrix is $1 = O(k)$, and clearly, the transformation can be carried out in time polynomial in n and k. ◀

6 NP-hardness of n-Fold MIPs

The algorithmic upper bound for 2-stage stochastic programs stands in contrast to a much stronger bound for the n-fold case. Namely, we show NP-hardness of n-fold ($\text{MILP}_{\text{frac}}$) for constant parameter values. By Lemma 8, we immediately get that n-fold (MIP) is also NP-hard for constant parameter values.

▶ **Theorem 4** (NP-hardness for n-fold MIP)**.** *The problem* (MIP) *with integral data is* NP*-hard when E is an n-fold matrix with blocks of constant dimensions and $\|E\|_\infty = 1$ already for linear objective functions.*

Proof of Theorem 4. We reduce from the well-known Partition problem. That is, given integers $a_1, \ldots, a_n$ the Partition problems asks for the existence of a subset $I \subseteq [n]$ such that $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$.

Let an instance of Partition be given. Without loss of generality, assume that $a_{\max} := \max_i a_i \leq 1$; this can be achieved, e. g., by scaling every number of the original instance by $1/a_{\max}$. We will have n bricks, with brick $i \in [n]$ representing the choice whether $i \in I$ or $i \notin I$.

Specifically, for each $i \in [n]$, introduce integer variables $x_1^i, x_2^i \in \{0, 1\}$ and continuous variables y_1^i, y_2^i with bounds $0 \leq y_1^i, y_2^i \leq 1$. The local constraints (matrix A) are as follows. We enforce a disjunction on the x-variables by the constraint $x_1^i + x_2^i = 1$ for every i and we enforce that $y_1^i = a_i$ iff $x_1^i = 1$ and similarly $y_2^i = a_i$ iff $x_2^i = 1$ by the constraints $y_1^i + y_2^i = a_1$, $y_1^i \leq x_1^i$, and $y_2^i \leq x_2^i$ for every i.

It is now easy to see that the following global constraint encodes the requirement that $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$:

$$\sum_{i=1}^{n} y_1^i = \sum_{i=1}^{n} y_2^i. \tag{10}$$

Altogether, the instance has four variables per block, four local constraints and one global constraint, and is feasible if and only if the original PARTITION instance is. ◀

7 Lower Bound on the Graver Norm of n-fold MIPs

In this section, we show that the 1-norm of the mixed Graver norm can be unbounded even for n-fold matrices.

We start with the following auxiliary lemma, which is crucial for constructing an element of the mixed Graver basis with unbounded 1-norm.

▶ **Lemma 23.** *Let n be an integer. There are two sets S and T of natural numbers with $|S| = |T| = n$ such that:*
(1) *$\sum_{s \in S} s = \sum_{t \in T} t = 2^{n^2} - 1$ and*
(2) *for every two subsets $S' \subseteq S$ and $T' \subseteq T$, with $0 < |S' \cup T'| < 2n$, it holds that $\sum_{s \in S'} s \neq \sum_{t \in T'} t$.*

Proof. Let $X \subseteq \mathbb{N} \setminus \{0\}$. We denote by $N(X)$, the natural number whose binary representation has a 1 at the i-th bit (with 1 being the lowest-value bit) if and only if $i \in X$. Conversely, for a natural number x, let $B(x)$ be the set of all indices i such that the binary representation of x is 1 at the i-th bit. Note that $B(N(X)) = X$ for every $X \subseteq \mathbb{N} \setminus \{0\}$.

For every i and j with $1 \leq i, j \leq n$, let $p(i,j) = (i-1)n + j$. For every i with $1 \leq i \leq n$, we set:

- s_i is equal to $N(R_i)$, where $R_i = \{p(i,j) \mid 1 \leq j \leq n\}$,
- t_i is equal to $N(C_i)$, where $C_i = \{p(j,i) \mid 1 \leq j \leq n\}$.

We claim that setting $S = \{s_1, \ldots, s_n\}$ and $T = \{t_1, \ldots, t_n\}$ satisfies the statement of the lemma: As $\{B(s_1), \ldots, B(s_n)\}$ and $\{B(t_1), \ldots, B(t_n)\}$ form a partition of $[n^2]$, it holds that $\sum_{s \in S'} s = N(\bigcup_{s \in S'} B(s))$ and $\sum_{t \in T'} t = N(\bigcup_{t \in T'} B(t))$ for every subsets $S' \subseteq S$ and $T' \subseteq T$. Therefore, $\sum_{s \in S} s = \sum_{t \in T} t = N([n^2]) = 2^{n^2} - 1$, which shows (1).

Towards showing (2), let S' and T' be any two subsets with $S' \subseteq S$ and $T' \subseteq T$ such that $0 < |S' \cup T'| < 2n$. As $0 < |S' \cup T'| < 2n$, we obtain that either:

- there are i and j with $1 \leq i, j \leq n$ such that $s_i \in S \setminus S'$ and $t_j \in T'$ or
- there are i and j with $1 \leq i, j \leq n$ such that $t_i \in T \setminus T'$ and $s_j \in S'$.

Since the proofs for the two cases are analogous, we only give the proof for the former case. Let $O = B(s_i) \cap B(t_j)$ and note that $O = R_i \cap C_j = \{p(i,j)\} \neq \emptyset$. Since $t_i \in T'$, it holds that $O \in \bigcup_{t \in T'} B(t)$. However, due to $s_i \notin S'$, we have that $O \notin \bigcup_{s \in S'} B(s)$. Consequently, $\bigcup_{s \in S'} B(s) \neq \bigcup_{t \in T'} B(t)$ and therefore also $\sum_{s \in S'} s \neq \sum_{t \in T'} t$. ◀

▶ **Theorem 5** (n-fold mixed Graver lower bound). *There is an n-fold matrix E with constant-sized blocks and $\|E\|_\infty = 1$ such that the mixed Graver basis of E contains an element with 1-norm of size $\Omega(n)$.*

Proof sketch. Let n be an integer, $\mathbb{X}_n = (\mathbb{Z} \times \mathbb{R} \times \mathbb{R})^n$, and E_n be the matrix given by the n-fold of $\begin{pmatrix} 0 & I_3 \\ 0 & A \end{pmatrix}$, where I_3 is the identity matrix of dimension 3 and $A = (1,1,1)$. Note first that the structure of the matrix E together with the fact that the first coordinate in each

block is integer ensures that any vector of the form $(-1, s/V, 1-s/V)$ and $(1, -t/V, 1+t/V)$ for some s, t, V with $0 \leq s, t \leq V$ is contained in the mixed Graver basis $\mathcal{G}_{\mathbb{X}_1}(E)$ of E. This allows us to construct $\mathbf{g}$ by using $n/2$ blocks of the form $(-1, s/V, 1-s/V)$ and $n/2$ blocks of the form $(1, -t/V, 1+t/V)$, where $s \in S$ and $t \in T$ for some well-constructed sets S and T of integers. Moreover, because of the first three rows (given by n-repetitions of I_3) the sum of the i-th coordinate over all blocks has to be 0. Therefore, to force that all n blocks of $\mathbf{g}$ use non-zero kernel elements (of $\mathcal{G}_{\mathbb{X}_1}(E)$), it suffices to construct the sets S and T in such a way that $\sum_{s \in S'} s = \sum_{t \in T'} t$ for some subsets $S' \subseteq S$ and $T' \subseteq T$ if and only if $S' = S$ and $T' = T$. We show that this is possible in an auxiliary lemma. ◀

8 Open Questions

Our work points towards two main directions for further research. First, note that we formulate our algorithms in reference to 2-stage stochastic constraint matrices. As mentioned, these are generalized by matrices of bounded primal treedepth, so-called multi-stage stochastic matrices. They are structured in much the same way as in Fig. 1, but with diagonal blocks of recursive multi-stage stochastic form (and the depth of this recursion is bounded). Judging from previous results in the area, there is reason to believe that our algorithmic results generalize to multi-stage stochastic programs. However, some caution seems appropriate. After all, one key takeaway of both the lower bounds and the algorithms shown in this paper is that block-structured mixed-integer programs do not behave as predictably as one might hope.

A second natural, much more ambitious direction of investigation is to try to extend the present algorithmic results on linear optimization to arbitrary separable convex objective functions. Despite significant efforts, we were not able to push beyond the algorithms obtained here. As for some intuition on why extending Theorem 2 to the separable convex case in the style of Theorem 1 seems to fail: For the latter, the search space for optima is naturally restricted already by the fact that there are only few constraints. For the former, however, such a restriction is not possible based on rows alone; instead, we argue about vertices of the associated polytope, which are related to mixed-integer optimal solutions. In the separable convex case, this connection between optimal solutions and vertices disappears, and we are left with no handle on the size of the search space. We consider the problem of circumventing these roadblocks an intriguing and hard open question.

References

1 Matthias Aschenbrenner and Raymond Hemmecke. Finiteness theorems in stochastic integer programming. *Foundations of Computational Mathematics*, 7(2):183–227, 2007.

2 Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813–852, 2016. URL: `http://www.jstor.org/stable/43818629`.

3 Robert E Bixby. Solving real-world linear programs: A decade and more of progress. *Operations research*, 50(1):3–15, 2002.

4 Cornelius Brand, Martin Koutecký, and Sebastian Ordyniak. Parameterized algorithms for milps with small treedepth. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12249–12257. AAAI Press, 2021. URL: `https://ojs.aaai.org/index.php/AAAI/article/view/17454`, `doi:10.1609/AAAI.V35I14.17454`.

5 Timothy F. N. Chan, Jacob W. Cooper, Martin Koutecký, Daniel Král, and Kristýna Pekárková. Matrices of optimal tree-depth and a row-invariant parameterized algorithm for integer programming. *SIAM J. Comput.*, 51(3):664–700, 2022. doi:10.1137/20M1353502.

6 Sergei Chubanov. A polynomial-time descent method for separable convex optimization problems with linear constraints. *SIAM J. Optim.*, 26(1):856–889, 2016. doi:10.1137/14098524X.

7 Michele Conforti, Marco Di Summa, Friedrich Eisenbrand, and Laurence A. Wolsey. Network formulations of mixed-integer programs. *Math. Oper. Res.*, 34(1):194–209, 2009. doi:10.1287/moor.1080.0354.

8 Jana Cslovjecsek, Friedrich Eisenbrand, Christoph Hunkenschröder, Lars Rohwedder, and Robert Weismantel. Block-structured integer and linear programming in strongly polynomial and near linear time. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1666–1681. SIAM, 2021. doi:10.1137/1.9781611976465.101.

9 Jana Cslovjecsek, Friedrich Eisenbrand, Michal Pilipczuk, Moritz Venzin, and Robert Weismantel. Efficient sequential and parallel algorithms for multistage stochastic integer programming using proximity. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 33:1–33:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.33.

10 Jana Cslovjecsek, Martin Koutecky, Alexandra Lassota, Michal Pilipczuk, and Adam Polak. Parameterized algorithms for block-structured integer programs with large entries. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7 - 10, 2024*. SIAM, 2024.

11 Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: on completeness for W[1]. *Theor. Comput. Sci.*, 141(1&2):109–131, 1995.

12 Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. An algorithmic theory of integer programming. *CoRR*, abs/1904.01361, 2019.

13 Friedrich Eisenbrand and Robert Weismantel. Proximity results and faster algorithms for integer programming using the steinitz lemma. *ACM Trans. Algorithms*, 16(1):5:1–5:14, 2020. doi:10.1145/3340322.

14 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

15 Raymond Hemmecke. On the positive sum property and the computation of graver test sets. *Math. Program.*, 96(2):247–269, 2003. doi:10.1007/s10107-003-0385-7.

16 D. S. Hochbaum and J. G. Shantikumar. Convex separable optimization is not much harder than linear optimization. *J. ACM*, 37(4):843–862, 1990.

17 A Hoffman. Integral boundary points of convex polyhedra. *Linear Inequalities and Related Systems*, pages 223–246, 1956.

18 Klaus Jansen, Kim-Manuel Klein, and Alexandra Lassota. The double exponential runtime is tight for 2-stage stochastic ILPs. *Math. Program.*, 197(2):1145–1172, 2023. doi:10.1007/S10107-022-01837-0.

19 Klaus Jansen and Lars Rohwedder. On integer programming, discrepancy, and convolution. *Math. Oper. Res.*, 48(3):1481–1495, 2023. doi:10.1287/MOOR.2022.1308.

20 Kim-Manuel Klein and Janina Reuter. Collapsing the tower - on the complexity of multistage stochastic ips. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 348–358. SIAM, 2022. doi:10.1137/1.9781611977073.17.

21 Dusan Knop, Martin Koutecký, Asaf Levin, Matthias Mnich, and Shmuel Onn. High-multiplicity n-fold IP via configuration LP. *Math. Program.*, 200(1):199–227, 2023. doi:10.1007/S10107-022-01882-9.

22 Martin Koutecký, Asaf Levin, and Shmuel Onn. A parameterized strongly polynomial algorithm for block structured integer programs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 85:1–85:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

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

24 Joseph Paat, Robert Weismantel, and Stefan Weltge. Distances between optimal solutions of mixed-integer programs. *Math. Program.*, 179(1):455–468, 2020. doi:10.1007/s10107-018-1323-z.

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

26 E. Steinitz. Bedingt konvergente Reihen und konvexe Systeme. *J. Reine Angew. Math.*, 146:1–52, 1916.

SIAM J. COMPUT.
Vol. 51, No. 3, pp. 664–700

MATRICES OF OPTIMAL TREE-DEPTH AND A ROW-INVARIANT PARAMETERIZED ALGORITHM FOR INTEGER PROGRAMMING*

TIMOTHY F. CHAN†, JACOB W. COOPER‡, MARTIN KOUTECKÝ§, DANIEL KRÁL†, AND KRISTÝNA PEKÁRKOVÁ†

Abstract. A long line of research on fixed parameter tractability of integer programming culminated with showing that integer programs with n variables and a constraint matrix with dual tree-depth d and largest entry Δ are solvable in time $g(d,\Delta)\text{poly}(n)$ for some function g. However, the dual tree-depth of a constraint matrix is not preserved by row operations, i.e., a given integer program can be equivalent to another with a smaller dual tree-depth, and thus does not reflect its geometric structure. We prove that the minimum dual tree-depth of a row-equivalent matrix is equal to the *branch-depth* of the matroid defined by the columns of the matrix. We design a fixed parameter algorithm for computing branch-depth of matroids represented over a finite field and a fixed parameter algorithm for computing a row-equivalent matrix with minimum dual tree-depth. Finally, we use these results to obtain an algorithm for integer programming running in time $g(d^*,\Delta)\text{poly}(n)$ where d^* is the branch-depth of the constraint matrix; the branch-depth cannot be replaced by the more permissive notion of branch-width.

Key words. matroids, tree-depth, branch-depth, integer programming, fixed-parameter tractability

AMS subject classifications. 05B35, 90C10, 90C27

DOI. 10.1137/20M1353502

1. Introduction.

Integer programming is a fundamental problem of both theoretical and practical importance. It is well-known that integer programming in fixed dimension, i.e., with a bounded number of variables, is polynomially solvable since the work of Kannan [31] and Lenstra [36] from the 1980s. Much subsequent research has focused on studying extensions and speed-ups of the algorithm of Kannan and Lenstra. However, research on integer programs with many variables has been sparser. Until relatively recently, the most prominent tractable case has been that of totally unimodular constraint matrices, i.e., matrices with all subdeterminants equal to 0 or ± 1; in this case, all vertices of the feasible region are integral and algorithms for linear programming can be applied.

Besides total unimodularity, several recent results [24, 6, 14, 1, 23, 19, 18, 11] on algorithms for integer programming exploited various structural properties of the constraint matrix, yielding efficient algorithms for n-fold integer programs, tree-fold

*Received by the editors July 16, 2020; accepted for publication (in revised form) February 2, 2022; published electronically May 24, 2022. An extended abstract of this work has appeared in the Proceedings of ICALP'20.

https://doi.org/10.1137/20M1353502

Funding: The first, second, and fourth authors were supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement 648509). The second, fourth, and fifth authors were supported by the MUNI Award in Science and Humanities of the Grant Agency of Masaryk University. The third author was supported by Charles University project UNCE/SCI/004 and by 19-27871X of GA ČR.

†School of Mathematical Sciences, Monash University, Melbourne, Victoria, 3800, Australia, and Mathematics Institute, DIMAP, and Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK (timothy.chan@monash.edu).

‡Faculty of Informatics, Masaryk University, 625 00, Brno, Czech Republic (jcooper@mail.muni.cz, kral@fi.muni.cz, kristyna.pekarkova@mail.muni.cz).

§Computer Science Institute, Charles University, Prague, 116 36, Czech Republic (koutecky@iuuk.mff.cuni.cz).

integer programs, multistage stochastic integer programs, and integer programs with bounded fracture number and bounded tree-width. This research culminated in an algorithm by Koutecký, Levin, and Onn [35], who constructed a fixed parameter algorithm for integer programs with bounded (primal or dual) tree-depth and bounded coefficients.

These theoretical results well complement a long line of empirical research (see [5, 3, 33, 15, 2, 41, 40, 17, 39]), demonstrating that instances of integer programming can be solved efficiently when the constraint matrix is decomposable into blocks. So, the recent algorithm of Koutecký, Levin, and Onn [35] gives a theoretical explanation for this phenomenon. In particular, the Dantzig–Wolfe decomposition algorithm is known to work very well when the constraint matrix has a so-called bordered block-diagonal form with small blocks. It is usually necessary to describe this form explicitly but Bergner et al. [3] have shown that it can be constructed automatically (by permuting the rows of the constraint matrix), which has had significant performance benefits in important benchmark instances.

However, the bordered block-diagonal form of a matrix, and, more generally, the tree-depth of a constraint matrix, depends on the position of its nonzero entries. In particular, a matrix with large dual tree-depth, i.e., without any apparent bordered block-diagonal form, may be row-equivalent to another matrix with small dual tree-depth and thus amenable to efficient algorithms. We overcome this drawback with tools from matroid theory. To do so, we consider the *branch-depth* of the matroid defined by the columns of the constraint matrix and refer to this parameter as the branch-depth of the matrix. Since this matroid is invariant under row operations, the branch-depth of a matrix is row-invariant and better captures the true geometry of the instance, which can be obfuscated by the choice of basis. Our algorithm thus allows taking the "automated Dantzig–Wolfe" approach of Bergner et al. [3] one step further: it is possible to detect a block structure in a matrix even if it is obscured by row operations, not just by permuting the rows.

Our main results concerning integer programming can be summarized as follows (we state the results formally in the next subsection).

- The branch-depth of a matrix A is equal to the minimum dual tree-depth of a matrix row-equivalent to A (Theorem 1).
- There exists a fixed parameter algorithm for computing a matrix of minimum dual tree-depth that is row-equivalent to the input matrix and whose entry complexity stays bounded (Theorem 35).
- Integer programming is fixed parameter tractable when parameterized by the branch-depth and the entry complexity of the constraint matrix (Corollary 4).

Existing hardness results imply that the parameterization by both branch-depth and entry complexity in Corollary 4 is necessary unless $\mathsf{FPT} = \mathsf{W}[1]$, i.e., it is not sufficient to parameterize only by one of the two parameters. In particular, integer programming is $\mathsf{W}[1]$-hard when parameterized by tree-depth only [19, 34] and NP-hard for instances with bounded coefficients and dual tree-width (even dual path-width) bounded by two [13, Lemma 102] (also cf. [35, 19]). The latter also implies that integer programming is NP-hard when the branch-width and the entry complexity of input instances are bounded (also cf. [16]). On the positive side, Cunningham and Geelen [8] (also cf. [37] for detailed proofs and implementation) provided a slicewise pseudopolynomial algorithm for nonnegative matrices with bounded branch-width, i.e., the problem belongs to the complexity class XP for unary encoding of input. Finally, since the algorithm given in Corollary 4 is parameterized by the branch-depth of the vector matroid formed by the columns of the matrix A, it is natural to ask whether

the tractability also holds in the setting dual to this one, i.e., when the branch-depth of the vector matroid formed by the *rows* of A is bounded. This hope is dismissed in Proposition 16.

The algorithm from Theorem 35 is based on the following algorithmic result on matroid branch-depth, which we believe to be of independent interest.

- There exists a fixed parameter algorithm for computing an optimal branch-depth decomposition of a matroid represented over a finite field for the parameterization by the branch-depth and the order of the field (Theorem 31).

To apply this result, we show that every matroid represented by rational vectors that has bounded branch-depth is isomorphic to a matroid representable over a finite field (Lemma 34), where the order of the field depends on the branch-depth and the entry complexity of the rational vectors.

We would like to point out that the algorithm from Theorem 31 is fully combinatorial, similar to the recent algorithm for computing the branch-width of matroids represented over a finite field by Jeong, Kim, and Oum [30], which extends the classical algorithm for tree-width by Bodlaender and Kloks [4], and unlike the older algorithm by Hliněný and Oum [28, 29], which relies on an exponential upper bound on the size of excluded minors for branch-width by Geelen et al. [21] and needs to precompute the list of excluded minors. While it would likely be possible to follow a similar path in the setting of branch-depth, we chose the more challenging route of designing a fully combinatorial algorithm, i.e., one that is based on an explicit dynamic programming procedure. The benefit of a fully combinatorial approach is that the hidden constants are better, which is of importance to applications including those in model checking [20, 25, 26, 27]; in particular, Hliněný [25, 26, 27] (in the analogy of Courcelle's result [7] for graphs) proved that monadic second order model checking is fixed parameter tractable for matroids with bounded branch-width represented over finite fields.

1.1. Statement of integer programming results. To state our integer programming results precisely, we first need to fix some notation. Vectors throughout our exposition will be written in bold font. We consider the general integer programming problem in standard form:

$$\min\{f(\mathbf{x}) \mid A\mathbf{x} = \mathbf{b},\ \mathbf{l} \le \mathbf{x} \le \mathbf{u},\ \mathbf{x} \in \mathbb{Z}^n\}, \tag{1}$$

where $A \in \mathbb{Z}^{m\times n}$ is an integer $m \times n$ matrix, $\mathbf{b} \in \mathbb{Z}^m$, $\mathbf{l}, \mathbf{u} \in (\mathbb{Z} \cup \{\pm\infty\})^n$, and $f : \mathbb{Z}^n \to \mathbb{Z}$ is a separable convex function, i.e., $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i)$ where $f_i : \mathbb{Z} \to \mathbb{Z}$ are convex functions. In particular, each $f_i(x_i)$ can be a linear function of x_i. Integer programming is well-known to be NP-hard even when f is a constant function and either the entries of A are 0 and $+1$ (by a reduction from the vertex cover problem) or $m = 1$ (by a reduction from the subset sum problem).

We next demonstrate the previously mentioned drawback of parameterizing integer programs by tree-depth. The *tree-depth* of a graph G is the minimum depth of a rooted forest on the same vertex set such that the two end vertices of every edge of G are in ancestor-descendant relation. The *dual tree-depth* of a matrix A is the tree-depth of the graph with vertices corresponding to the rows of A and with two vertices being adjacent if the corresponding rows have a nonzero entry in the same column. We define the branch-depth, which requires definitions from matroid theory, and primal tree-depth of a matrix, in section 2. Consider the following matrices A and A':

$$A = \begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 2 & 1 & \cdots & 1 & 1 \\ 1 & 2 & \ddots & 1 & 1 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 1 & 1 & \ddots & 2 & 1 \\ 1 & 1 & \cdots & 1 & 2 \end{pmatrix}, \qquad A' = \begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

The dual tree-depth of the matrix A is equal to its number of rows while the dual tree-depth of A' is two; the graphs from the definition of the dual tree-depth are a complete graph and a star, respectively. We remark that the branch-depth of both matrices is two. Since the matrices A and A' are row-equivalent, the integer programs determined by them ought to be of the same computational difficulty. More precisely, consider the following matrix B:

$$B = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ -1 & 0 & 1 & \ddots & 0 & 0 \\ -1 & \vdots & \ddots & \ddots & \ddots & 0 \\ -1 & 0 & 0 & \ddots & 1 & 0 \\ -1 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Since $A' = BA$, it is possible to replace an integer program of the form (1) with an integer program with a constraint matrix $A' = BA$, right-hand-side $\mathbf{b}' = B\mathbf{b}$, and bounds $\mathbf{l}' = \mathbf{l}$ and $\mathbf{u}' = \mathbf{u}$, and solve this new instance of integer programming, which has dual tree-depth two and the same set of feasible solutions.

In section 4, we first observe that the branch-depth of a matrix A is at most its dual tree-depth, and we prove that the branch-depth of a matrix A is actually equal to the minimum dual tree-depth of a matrix A' that is row-equivalent to A.

THEOREM 1. *Let A be a matrix over a (finite or infinite) field $\mathbb{F}$. The branch-depth of A is equal to the minimum dual tree-depth of any matrix A' that is row-equivalent to A.*

We use the tools developed to prove Theorem 1 together with existing results on matroid branch-depth to obtain an algorithm that given a matrix A of small branch-depth outputs a matrix B that transforms A to a row-equivalent matrix with small dual tree-depth. The *entry complexity of a matrix A*, denoted by $\mathrm{ec}(A)$, is the maximum length of the binary encoding of an entry A_{ij} (the length of binary encoding a rational number $r = p/q$ with p and q being coprime is $\lceil \log_2(|p|+1) \rceil + \lceil (\log_2 |q| + 1) \rceil$). An algorithm is called *fixed parameter* if its running time for an instance of size n is bounded by $f(k)\mathrm{poly}(n)$ where $f : \mathbb{N} \to \mathbb{N}$ is a computable function and k is a parameter determined by the instance.

THEOREM 2. *There exists an algorithm with running time polynomial in $\mathrm{ec}(A)$, n and m that for an input $m \times n$ integer matrix A and an integer d either*

- *outputs that the branch-depth of A is larger than d, or*
- *outputs an invertible rational matrix $B \in \mathbb{Q}^{m \times m}$ such that the dual tree-depth of BA is at most 4^d and the entry complexity of BA is $O(d2^{2d}\,\mathrm{ec}(A))$.*

TABLE 1

Comparison of algorithms presented in Theorems 2 *and* 3. *The first line describes the dependence on the parameter* d, *the second and the third the tree-depth and the entry complexity of the constraint matrix output by the algorithm, and the last line time needed to solve the instance by the algorithm of Eisenbrand et al.* [13].

Algorithm from	Theorem 2	Theorem 3
Hidden constant	none	$2^{\mathrm{ec}(A)2^{2^{2d+2}+O(d)}}$
Tree-depth	at most 2^{2d}	d
Entry complexity	$O\left(d2^{2d}\,\mathrm{ec}(A)\right)$	$O\left(d^2 2^{2d}\,\mathrm{ec}(A)\right)$
IP algorithm constant	$2^{O\left(\mathrm{ec}(A)d2^{\mathrm{ec}(A)d2^{2d}+2^{2d}+4d}\right)}$	$2^{O\left(\mathrm{ec}(A)d^3 2^{\mathrm{ec}(A)d^2 2^{2d}+3d}\right)}$

However, we go further and design a fixed parameter algorithm for computing the branch-depth of a vector matroid (Theorem 35) and use this algorithm to prove the following strengthening of Theorem 2.

THEOREM 3. *There exists a fixed parameter algorithm parameterized by* d *and* e *with running time polynomial in* n *and* m *that for an input* $m \times n$ *integer matrix* A *with entry complexity at most* e *and an integer* d *either*

- *outputs that the branch-depth of* A *is larger than* d, *or*
- *outputs an invertible rational matrix* $B \in \mathbb{Q}^{m\times m}$ *such that the dual tree-depth of* BA *is equal to the branch-depth of* A *and the entry complexity of* BA *is* $O(d^2 2^{2d}\,\mathrm{ec}(A))$.

While the algorithm in Theorem 2 runs in polynomial time, the output matrix can have dual tree-depth (single) exponential in the minimum possible tree-depth; on the other hand, the running time of the algorithm from Theorem 3 is triple exponential in d but it always outputs a row-equivalent matrix with the minimum possible tree-depth. Also see Table 1 for a comparison of the algorithms from Theorems 2 and 3.

As explained above, Theorems 2 and 3 allow us to perform row operations to obtain an equivalent integer program with small dual tree-depth from an integer program with small branch-depth. The function g depends on which of the theorems is used to find the matrix B as displayed in Table 1. A proof of the corollary using Theorem 2 is presented in section 5; a proof using Theorem 3 is completely the same except that the obtained matrix A' has dual tree-depth at most d and its entry complexity is as given in Table 1.

COROLLARY 4. *Integer programming is fixed parameter tractable when parameterized by branch-depth and entry complexity, i.e., an integer program given as in* (1) *can be solved in time polynomial in* $g(\mathrm{bd}(A), \mathrm{ec}(A))$, n, $\mathrm{ec}(\mathbf{b})$, $\mathrm{ec}(\mathbf{l})$, *and* $\mathrm{ec}(\mathbf{u})$, *where* $\mathrm{bd}(A)$ *is the branch-depth of the matrix* A *and* $g : \mathbb{N}^2 \to \mathbb{N}$ *is a computable function.*

We remark that the results of [13, 35] give a strongly fixed parameter algorithm, i.e., an algorithm whose number of arithmetic operations does not depend on the size of the numbers involved, if the objective function f is a linear function, and so the algorithm from Corollary 4 is strongly polynomial in $g(\mathrm{bd}(A), \mathrm{ec}(A))$, and n when the objective function is linear.

We note that the dependence of g on $\mathrm{ec}(A)$ and $\mathrm{bd}(A)$ is double and triple exponential, respectively, regardless of whether we use Theorem 2 or Theorem 3 to prove Corollary 4. However, a rather pessimistic estimate on the entry complexity of the integer matrix A'' was used in the proof of Corollary 4 and it is likely that

the constraint matrix with a row-equivalent matrix with a significantly smaller dual tree-depth likely outweighs the increase of the entry complexity since the parameter dependence in the algorithm of [13], which is a refined version of the algorithm from [35], is $2^{(\mathrm{ec}(A)+\mathrm{td}_D(A))\,\mathrm{td}_D(A)2^{\mathrm{td}_D(A)}} = 2^{\mathrm{ec}(A)2^{O(\mathrm{td}_D(A))}}$.

1.2. Structure of the paper. We now briefly describe how the paper is organized. In section 2, we introduce notation used in this paper, in particular the notions of (dual) tree-depth and branch-depth of matrices, and provide the relevant background on matroids. We also give the definition of an extended depth-decomposition, which is a depth-decomposition of a matroid enhanced with information on the structure of subspaces represented by branches of the decomposition. Section 3 is devoted to proving a structural result on matroids, which establishes the existence of extended depth-decompositions with optimal depth. The results of section 3 are used to prove that the branch-depth of a matrix is equal to the minimum tree-depth of a row-equivalent matrix in section 4. We next apply these results in section 5 to prove Theorem 2 and Corollary 4, which implies that integer programming is fixed parameter tractable when parameterized by the branch-depth and the entry complexity of the constraint matrix.

The rest of the paper is devoted to computing optimal branch-depth of matroids and matrices. As a preparation for proofs of our main results, we develop additional tools to manipulate depth-decompositions of vector matroids in section 6. These tools are used in section 7 to construct a dynamic programming algorithm for computing optimal branch-depth decompositions of matroids represented over finite fields. Finally, in section 8, we use the algorithm from section 7 to design a fixed parameter algorithm for computing optimal branch-depth decompositions of matroids represented over rationals and for computing row-equivalent matrices with optimal branch-depth (Theorem 35).

2. Notation. In this section, we fix the notation used throughout the paper, present the notions of graph tree-depth and matroid branch-depth, and include relevant results concerning them that we will need later. To avoid our presentation becoming cumbersome through adding or subtracting one at various places, we define the *depth* of a rooted tree to be the maximum number of edges on a path from the root to a leaf, and define the *height* of a rooted tree to be the maximum number of vertices on a path from the root to a leaf, i.e., the height of a rooted tree is always equal to its depth plus one. The height of a rooted forest F is the maximum height of a rooted tree in F. The *depth* of a vertex in a rooted tree is the number of edges on the path from the root to that particular vertex; in particular, the depth of the root is zero. The *closure* $\mathrm{cl}(F)$ of a rooted forest F is the graph obtained by adding edges from each vertex to all its descendants. Finally, the *tree-depth* $\mathrm{td}(G)$ of a graph G is the minimum height of a rooted forest F such that the closure $\mathrm{cl}(F)$ of the rooted forest F contains G as a subgraph. It can be shown that the path-width of a graph G is at most its tree-depth $\mathrm{td}(G)$ minus one, and in particular, the tree-width of G is at most its tree-depth minus one (see, e.g., [9] for the definitions of path-width and tree-width). As [32] is one of our main references, we would like to highlight that the tree-depth as used in [32] is equal to the minimum depth of a rooted tree T such that $G \subseteq \mathrm{cl}(T)$; however, we here follow the definition of tree-depth that is standard.

The *primal graph* of an $m \times n$ matrix A is the graph $G_P(A)$ with vertices $\{1, \ldots, n\}$, i.e., its vertices one-to-one correspond to the columns of A, where vertices i and j are adjacent if A contains a row whose ith and jth entries are nonzero. The *primal*

tree-depth $\mathrm{td}_P(A)$ of a matrix A is the tree-depth of its primal graph. Analogously, the *dual graph* of A is the graph $G_D(A)$ with vertices $\{1,\dots,m\}$, i.e., its vertices one-to-one correspond to the rows of A, where vertices i and j are adjacent if A contains a column whose ith and jth entries are nonzero. Note that the dual graph $G_D(A)$ is isomorphic to the primal graph of the matrix transpose A^T. Finally, the *dual tree-depth* of A, which is denoted by $\mathrm{td}_D(A)$, is the tree-depth of the dual graph $G_D(A)$.

Before introducing the notion of the branch-depth of a matroid, we review basic definitions from matroid theory. A detailed introduction to matroid theory can be found in one of the standard textbooks on the topic, e.g., [38], but we include a brief overview of relevant concepts for completeness. A *matroid* M is a pair $(X,\mathcal{I})$, where $\mathcal{I}$ is a nonempty hereditary collection of subsets of X that satisfies the *augmentation axiom*. More specifically, the collection $\mathcal{I}$ is hereditary if $\mathcal{I}$ contains all subsets of X' for every $X' \in \mathcal{I}$, and the augmentation axiom asserts that for all $X' \in \mathcal{I}$ and $X'' \in \mathcal{I}$ with $|X'| < |X''|$, there exists an element $x \in X'' \setminus X'$ such that $X' \cup \{x\} \in \mathcal{I}$. The sets contained in $\mathcal{I}$ are referred to as *independent*. The *rank* $r(X')$ of a set $X' \subseteq X$ is the maximum size of an independent subset of X'; the rank $r(M)$ of a matroid $M = (X,\mathcal{I})$ is the rank of X and an independent set of size $r(M)$ is a *basis* of M. A *circuit* is a set $X' \subseteq X$ such that X' is not independent but every proper subset of X' is. Two elements x and x' of X are said to be *parallel* if $r(\{x\}) = r(\{x'\}) = r(\{x,x'\}) = 1$, and an element x is a *loop* if $r(\{x\}) = 0$.

Two particular examples of matroids are graphic matroids and vector matroids. If G is a graph, then the pair $(E(G),\mathcal{I})$ where $\mathcal{I}$ contains all acyclic subsets of edges of G is a matroid and is denoted by $M(G)$; matroids of this kind are called *graphic matroids*. If X is a set of vectors of a vector space and $\mathcal{I}$ contains all subsets of X that are linearly independent, then the pair $(X,\mathcal{I})$ is a matroid; matroids of this kind are *vector matroids*. In the setting of vector matroids, the rank of $X' \subseteq X$ is the dimension of the linear hull of X'. If $(X,\mathcal{I})$ is a vector matroid, we write $\mathcal{L}(X')$ for the linear hull of the vectors contained in $X' \subseteq X$ and abuse the notation by writing $\dim X'$ for $\dim \mathcal{L}(X')$.

In what follows, we will need a notion of a quotient of a vector space, which we now recall. If A is a vector space and K a subspace of A, the *quotient space* A/K is a vector space of dimension $\dim A - \dim K$ obtained from A by considering cosets of A given by K and inheriting addition and scalar multiplication from A; see, e.g., [22] for further details. One can show that for every subspace K of A, there exists a subspace B of A with dimension $\dim A - \dim K$ such that each coset contains a single vector from B, i.e., every vector w of A can be uniquely expressed as the sum of a vector w_B of B and a vector w_K of K. We call the vector w_B the *quotient* of w by K. Note that the quotient of a vector is not uniquely defined by K; however, it becomes uniquely defined when the subspace B, which intersects each coset at a single vector, is fixed.

If $M = (X,\mathcal{I})$ is a matroid and $X' \subseteq X$, then the *restriction* of M to X', which is denoted by $M[X']$, is the matroid $(X',\mathcal{I} \cap 2^{X'})$. The *contraction* of M by X', which is denoted by M/X', is the matroid with the elements $X \setminus X'$ such that a set $X'' \subseteq X \setminus X'$ is independent in M/X' if and only if $r(X'' \cup X') = |X''| + r(X')$. Analogously, if $M = (X,\mathcal{I})$ is a vector matroid and K a subspace of the linear hull of X, then the matroid M/K obtained by *contracting* along the subspace K is the matroid whose elements are the vectors of X, with a subset $X' \subseteq X$ being independent in M/K if X' is linearly independent in the quotient space $\mathcal{L}(X)/K$. Note that if $X' \subseteq X$, then M/X' is the matroid obtained by contracting along the linear hull of X' and then restricting to the subset $X \setminus X'$.

A matroid M is said to be *connected* if every two (distinct) nonloop elements of M are contained in a circuit. A set $X' \subseteq X$ of a matroid is a *component* of M if it is an inclusionwise maximal subset of X such that the matroid $M[X']$ is connected and X' is loop-free. Equivalently, X' is a component of M if it is an inclusionwise minimal nonempty subset of X such that $r(X') + r(X \setminus X') = r(M)$. If M is a vector matroid with rank at least one, then M is connected if and only if M has no loops and there do not exist two vector spaces A and B such that $A \cap B$ contains the zero vector only, both A and B contains a nonloop element of M, and every element of M is contained in A or B.

Since several algorithms presented in this paper work with matroids, we must fix the way that the time complexity of algorithms involving matroids is measured. All algorithms that we present work with vector matroids and we assume that matroids are given by their vector representation. It is then possible to use standard linear algebra algorithms to determine which subsets of the elements of such matroids are independent. Hence, we say that an algorithm is polynomial time if its running time is polynomial in the number of elements of the matroid and the complexity of its vector representation. We remark that some of the algorithms that we use, in particular the one in Theorem 8, are also polynomial in the more demanding setting when matroids are given by the independence oracle, which we do not consider here.

A *depth-decomposition* of a matroid $M = (X, \mathcal{I})$ is a pair (T, f), where T is a rooted tree and f is a mapping from X to the leaves of T such that the number of edges of T is the rank of M and the following holds for every subset $X' \subseteq X$: the rank of X' is at most the number of edges contained in paths from the root to the vertices $f(x)$, $x \in X'$. The *branch-depth* $\mathrm{bd}(M)$ of a matroid M is the smallest depth of a tree T that forms a depth-decomposition of M. For example, if $M = (X, \mathcal{I})$ is a matroid of rank r, T is a path with r edges rooted at one of its end vertices, and f is a mapping such that $f(x)$ is equal to the (nonroot) leaf of T for all $x \in X$, then the pair (T, f) is a depth-decomposition of M. In particular, the branch-depth of any matroid M is well-defined and is at most the rank of M. We remark that the notion of matroid branch-depth given here is the one defined in [32]; another matroid parameter, which is also called branch-depth but is different from the one that we use here, is defined in [10]. Finally, the *branch-depth* $\mathrm{bd}(A)$ of a matrix A is the branch-depth of the vector matroid formed by the columns of A. Since the vector matroid formed by the columns of a matrix A and the vector matroid formed by the columns of any matrix row-equivalent to A are the same, the branch-depth of A is invariant under row operations.

Similarly to the relation between tree-depth of graphs and long paths, branch-depth is related to the existence of long circuits in a matroid [32, Proposition 3.4].

PROPOSITION 5. *Let M be a matroid. If the branch-depth of M is d, then every circuit of M has at most 2^d elements.*

Kardoš et al. [32] also established the following relations between the tree-depth of a graph G and the branch-depth of the associated matroid $M(G)$. It is worth noting that Proposition 7 does not hold without the assumption on 2-connectivity of a graph G: the tree-depth of an n-vertex path is $\lfloor \log_2 n \rfloor$; however, its matroid is formed by $n - 1$ independent elements, i.e., its branch-depth is one.

PROPOSITION 6. *For any graph G, the branch-depth of the graphic matroid $M(G)$ is at most the tree-depth of the graph G decreased by one.*

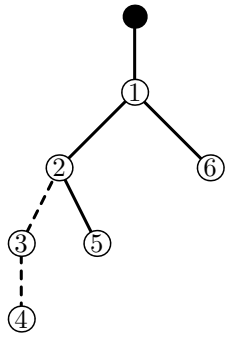


FIG. 1. *An illustration of the definition of solid branches and branches at capacity. The picture depicts the tree T of an extended depth-decomposition (T, f, g) such that the function g maps a nonroot vertex labeled with i to the ith unit vector. If the f-preimage of the leaf of the branch of T depicted by dashed edges is $\{(1,1,0,0,0,0),(0,1,1,0,0,0),(0,0,1,0,0,0),(0,0,0,1,0,0)\}$, then the branch is at capacity (regardless of the structure of the matroid and the choice of f); however, if the f-preimage is $\{(0,0,1,0,0,0),(0,0,0,1,0,0)\}$, then the branch is not at capacity. If the f-preimage is $\{(0,0,1,1,0,0),(0,1,1,0,0,0),(1,0,0,1,0,0)\}$, then the branch is solid, and if the f-preimage is $\{(0,0,1,0,0,0),(1,0,1,0,0,0),(1,0,0,1,0,0)\}$, then the branch is not solid.*

PROPOSITION 7. *For any 2-connected graph G, the branch-depth of the graphic matroid $M(G)$ is at least $\frac{1}{2}\log_2(\mathrm{td}(G)-1)$.*

Further properties of depth-decompositions and the branch-depth of matroids can be found in [32].

An *extended depth-decomposition* of a vector matroid $M = (X, \mathcal{I})$ is a triple (T, f, g) such that (T, f) is a depth-decomposition of M and g is a bijective mapping from the nonroot vertices of T to a basis of the linear hull of X that satisfies that every element $x \in X$ is contained in the linear hull of the g-image of the nonroot vertices on the path from $f(x)$ to the root of T. Note that g-images need not be elements of M in general; if all g-images are elements of the matroid M, we say that an extended depth-decomposition (T, f, g) is *principal*. Kardoš et al. [32, Corollary 3.17] designed an algorithm that outputs an approximation of an optimal depth-decomposition, which is a principal extended depth-decomposition; we state the result here for the case of vector matroids.

THEOREM 8. *There exists a polynomial-time algorithm that given a vector matroid M and an integer d either outputs that the branch-depth of M is larger than d or outputs a principal extended depth-decomposition of M of depth at most 4^d.*

If (T, f, g) is an extended depth-decomposition and u is a vertex of T, then K_u is the linear hull of the g-images of the vertices on the path from u to the root of T; in particular, if u is the root, then K_u contains the zero vector only. It will always be clear from the context for which extended depth-decomposition of M the spaces K_u are defined, in particular, the vertex u determines which rooted tree T is considered.

A *branch* of a rooted tree T is a subtree S rooted at a vertex u of T such that u has at least two children, and the subtree S contains exactly u, one child u' of u, and all descendants of u'. In particular, a rooted tree has a branch if and only if it has a vertex with at least two children. A branch S is *primary* if every ancestor of the root of S has exactly one child. Every rooted tree T that is not a rooted path has at least two primary branches and all primary branches are rooted at the same vertex. Let (T, f) be a depth-decomposition of a matroid M. We write $\widehat{S}$ for the set

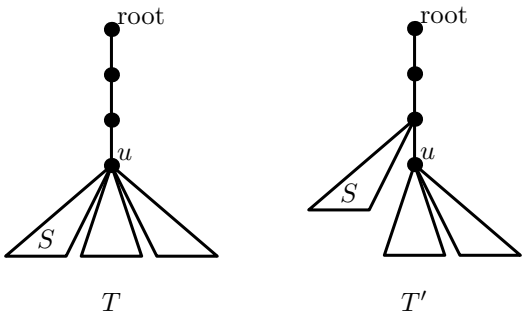


FIG. 2. *The trees T and T' from the statement of Lemma* 10.

of elements of the matroid M mapped by f to the leaves of S and $\|S\|$ for the number of edges of S. Let S be a branch of T and $S_1, \ldots, S_k$ be the other branches with the same root. The branch S is *at capacity* if

$$r\left(X \setminus \left(\widehat{S}_1 \cup \cdots \cup \widehat{S}_k\right)\right) = r(M) - \|S_1\| - \|S_2\| - \cdots - \|S_k\|,$$

where X is the set of all elements of the matroid M (see Figure 1 for an illustration). Note that if S is primary, then the left side of the equality is $r(\widehat{S})$ and the right side is $h + \|S\|$, where h is the depth of the root of S. In particular, a primary branch S is *at capacity* if and only if the rank of $\widehat{S}$ is equal to the sum of $\|S\|$ and h, i.e., if and only if the rank inequality from the definition of a depth-decomposition holds with equality for the set $\widehat{S}$. Finally, if (T, f, g) is an extended depth-decomposition of M, we say that a branch S rooted at a vertex u is *solid* if the matroid $(M/K_u)[\widehat{S}]$ after removal of its loops is connected, and that (T, f, g) is solid if all of its branches are; again, an illustration can be found in Figure 1.

3. Optimal extended depth-decompositions. The goal of this section is to show that every vector matroid has an extended depth-decomposition with depth equal to its branch-depth. To do so, we start with showing that branches rooted at the root of a decomposition tree are always at capacity.

LEMMA 9. *Let (T, f) be a depth-decomposition of a vector matroid M. If T has a branch S rooted at the root of T, then S is at capacity.*

Proof. Suppose that a branch S rooted at the root of T is not at capacity. This implies that $\dim \widehat{S} < \|S\|$. Let X' be the set of elements of M that are not contained in $\widehat{S}$. By the definition of a depth-decomposition, $\dim X'$ is at most $r(M) - \|S\|$. However, the submodularity of the dimension implies that $\dim \widehat{S} \cup X' < r(M)$, which is impossible. ∎

The following lemma is a core of our argument that every matroid has a depth-decomposition of optimal depth such that each primary branch is at capacity. An illustration of the operation described in the statement of the lemma is given in Figure 2.

LEMMA 10. *Let (T, f) be a depth-decomposition of a vector matroid M. Assume that T contains a primary branch S that is not at capacity. Let u be the root of S, and let T' be the rooted tree obtained from T by changing the root of S to be the parent of u. Then, (T', f) is a depth-decomposition of M.*

Proof. By Lemma 9, u has a parent and thus T' is well-defined. Let X be the set of elements of M and fix a subset X' of X. We need to show that $\dim X'$ is at most the number e_0 of edges on the paths in T' from the vertices in the f-image of X' to the root. If X' contains an element of $X \setminus \widehat{S}$, then the number of such edges is the same in the trees T and T' and the inequality follows from the fact that (T, f) is a depth-decomposition of M. Hence, we will assume that X' is a subset of $\widehat{S}$. Observe that collectively the primary branches of T different from S contain $r(M) - h - \|S\|$ edges, where h is the depth of u. We derive using the fact that (T, f) is a depth-decomposition the following:

$$\begin{aligned} e_0 + 1 + (r(M) - h - \|S\|) &\geq \dim X' \cup (X \setminus \widehat{S}) \\ &= \dim X' + \dim X \setminus \widehat{S} - \dim \mathcal{L}(X') \cap \mathcal{L}\left(X \setminus \widehat{S}\right) \\ &\geq \dim X' + \dim X \setminus \widehat{S} - \dim \mathcal{L}\left(\widehat{S}\right) \cap \mathcal{L}\left(X \setminus \widehat{S}\right) \\ &= \dim X' + \dim X \setminus \widehat{S} - (\dim \widehat{S} + \dim X \setminus \widehat{S} - \dim X) \\ &= \dim X' - \dim \widehat{S} + r(M). \end{aligned}$$

This implies that $\dim X'$ is at most

$$e_0 + \dim \widehat{S} + 1 - h - \|S\| \leq e_0,$$

where the inequality follows using that S is not at capacity, i.e., $\dim \widehat{S} < h + \|S\|$. Hence, (T', f) is a depth-decomposition of M. ∎

We can now show that every matroid has a depth-decomposition of optimal depth such that each primary branch is at capacity. We state the next two lemmas and the theorem that follows them for an arbitrary depth-decomposition (T, f), i.e., a depth-decomposition of not necessarily optimal depth, since we will need to apply the algorithmic arguments used in their proofs for arbitrary depth-decompositions later.

LEMMA 11. *Let (T, f) be a depth-decomposition of a vector matroid $M = (X, \mathcal{I})$ of depth d. There exists a depth-decomposition of M of depth at most d such that every primary branch is at capacity.*

Proof. If T is a rooted path, then the lemma holds vacuously. Suppose that T is not a rooted path. If all primary branches of (T, f) are at capacity, then we are done. If not, we consider a primary branch of T that is not at capacity and apply Lemma 10 to obtain a depth-decomposition (T', f). If all primary branches of (T', f) are at capacity, then we are done. If not, we consider a primary branch of T' that is not at capacity and iterate the process. Note that at each iteration, the sum of the lengths of the paths from the leaves to the root decreases, so the process eventually stops with a depth-decomposition such that all its primary branches are at capacity. ∎

Now we analyze depth-decompositions whose primary branches are at capacity.

LEMMA 12. *Let (T, f) be a depth-decomposition of a vector matroid $M = (X, \mathcal{I})$ such that T is not a rooted path and each primary branch of T is at capacity. Let $S_1, \ldots, S_k$ be the primary branches of T, and let $A_1, \ldots, A_k$ be the linear hulls of $\widehat{S_1}, \ldots, \widehat{S_k}$, respectively. Further, let h be the depth of the common root of $S_1, \ldots, S_k$ in T. There exists a subspace K of dimension h such that $A_i \cap A_j = K$ for all $1 \leq i < j \leq k$.*

Proof. Consider i and j such that $1 \leq i < j \leq k$. Since S_i is at capacity, we obtain that $\dim \widehat{S_i} = \dim A_i = h + \|S_i\|$. Analogously, it holds that $\dim \widehat{S_j} = \dim A_j = h + \|S_j\|$. Since (T, f) is a depth-decomposition, we deduce that

$$\begin{aligned} h + \|S_i\| + \|S_j\| &\geq \dim A_i \cup A_j \\ &= \dim A_i + \dim A_j - \dim A_i \cap A_j \\ &= (h + \|S_i\|) + (h + \|S_j\|) - \dim A_i \cap A_j, \end{aligned}$$

which implies that $\dim A_i \cap A_j \geq h$. On the other hand, it holds that

$$\begin{aligned} \dim A_i \cap A_j &\leq \dim A_i \cap \mathcal{L}\left(\bigcup_{j' \neq i} A_{j'}\right) \\ &= \dim A_i + \dim \bigcup_{j' \neq i} A_{j'} - \dim A_i \cup \bigcup_{j' \neq i} A_{j'} \\ &= h + \|S_i\| + \dim \bigcup_{j' \neq i} A_{j'} - h - \sum_{j'=1}^{k} \|S_{j'}\| \\ &\leq h + \|S_i\| + h + \sum_{j' \neq i} \|S_{j'}\| - h - \sum_{j'=1}^{k} \|S_{j'}\| = h. \end{aligned}$$

We conclude that $\dim A_i \cap A_j = h$. This implies that the first inequality in the expression above holds with equality, so

$$A_i \cap \mathcal{L}\left(\bigcup_{j' \neq i} A_{j'}\right) = A_i \cap A_j = A_j \cap A_i = A_j \cap \mathcal{L}\left(\bigcup_{i' \neq j} A_{i'}\right), \tag{2}$$

where the last step follows by the same argument above with indices i and j swapped. Since the left-hand side of (2) is independent of the choice of j and the right-hand side is independent of the choice of i, it must hold that $A_i \cap A_j$ is independent of the choice of both i and j. This intersection forms the required space K of dimension h.□

We are now ready to prove the main theorem of this section.

THEOREM 13. *Let (T, f) be a depth-decomposition of a vector matroid $M = (X, \mathcal{I})$ of depth d. There exists an extended depth-decomposition of M of depth at most d.*

Proof. The proof proceeds by induction on the rank of M. If T is a rooted path, we assign elements of a basis of $\mathcal{L}(X)$ to the nonroot vertices of T arbitrarily, i.e., we choose g to be any bijection to a basis of $\mathcal{L}(X)$, which yields an extended depth-decomposition (T, f, g) of M. Note that if the rank of M is one, then T is the one-edge rooted path, i.e., this case covers the base of the induction in particular.

We assume that T is not a rooted path for the rest of the proof. By Lemma 11, we can assume that all primary branches of T are at capacity. Let $S_1, \ldots, S_k$ be the primary branches of T, and let $h \geq 0$ be the depth of the common root of $S_1, \ldots, S_k$. By Lemma 12, there exists a subspace K of dimension h such that the intersection of linear hulls of $\widehat{S_i}$ and $\widehat{S_j}$ is K for all $1 \leq i < j \leq k$; let $b_1, \ldots, b_h$ be an arbitrary basis of K.

We define M_i, $i = 1, \ldots, k$, to be the matroid such that the elements of M_i are $\widehat{S_i}$ and $X' \subseteq \widehat{S_i}$ is independent if and only if the elements $X' \cup \{b_1, \ldots, b_h\}$ are linearly independent. In particular, the rank of $X' \subseteq \widehat{S_i}$ in M_i is equal to $\dim X' \cup K - h$. The matroid M_i can be viewed as obtained by taking the vector matroid with the elements $\widehat{S_i} \cup \{b_1, \ldots, b_h\}$ and contracting the elements $b_1, \ldots, b_h$. In particular, M_i

is a vector matroid, and the vector representation of M_i can be obtained from $\widehat{S}_i$ by taking quotients by K. Note that the rank of M_i is $\dim \widehat{S}_i \cup K - h$, i.e., its rank is smaller than the rank of M and we will be able to eventually apply induction to it.

Let f_i be the restriction of f to $\widehat{S}_i$. We claim that (S_i, f_i) is a depth-decomposition of M_i. Let X' be a subset of $\widehat{S}_i$, and let e_i be the number of edges contained in the union of paths from the elements $f_i(x)$, $x \in X'$, to the root of S_i. By the definition of M_i, the rank of X' in M_i is equal to $\dim X' \cup K - h$. Choose an arbitrary $j \neq i$, $1 \leq j \leq k$. Since the intersection of linear hulls of $\widehat{S}_i$ and $\widehat{S_j}$ is K, (T, f) is a depth-decomposition of M, and the branch S_j is at capacity, i.e., $\dim \widehat{S_j} = \|S_j\| + h$, we obtain that the rank of X' in M_i is equal to

$$\begin{aligned}\dim X' \cup K - h &= \dim X' \cup \widehat{S_j} - \dim \widehat{S_j} \\ &\leq e_i + \|S_j\| + h - \dim \widehat{S_j} = e_i.\end{aligned}$$

Hence, (S_i, f_i) is a depth-decomposition of M_i.

We apply induction to each matroid M_i and its depth-decomposition (S_i, f_i), $i = 1, \ldots, k$, to obtain extended depth-decompositions (S_i', f_i', g_i) of M_i such that the depth of S_i' is at most the depth of S_i. Let T' be a rooted tree obtained from a rooted path of length h by identifying its nonroot end with the roots of $S_1', \ldots, S_k'$. Observe that the depth of T' does not exceed the depth of T. Further, let f' be the unique function from X to the leaves of T such that the restriction of f' to the elements of M_i is f_i'. Finally, let g be any function from the nonroot vertices of T such that the h nonroot vertices of the path from the root are mapped by g to the vectors $b_1, \ldots, b_h$ by g and $g(v) = g_i(v)$ for every nonroot vertex v of S_i.

We claim that (T', f', g) is an extended depth-decomposition of M. We first verify that, for every $x \in X$, $f'(x)$ is contained in the linear hull of the g-image of the nonroot vertices on the path from $f'(x)$ to the root. Fix $x \in X$ and let i be such that $x \in \widehat{S}_i$. Since (S_i', f_i', g_i) is an extended depth-decomposition of M_i, x is contained in the linear hull of K and the g_i-images of the nonroot vertices on the path from $f'(x) = f_i(x)$ to the root of S_i'. Hence, x is contained in the linear hull of the g-image of the nonroot vertices on the path from $f'(x)$ to the root of T'.

Consider now an arbitrary subset $X' \subseteq X$. We have already established that all elements of X' are contained in the linear hull of the g-image of the nonroot vertices on the paths from $f'(x)$, $x \in X'$, to the root of T'. Since the dimension of this linear hull is at most the number of nonroot vertices on such paths, which is equal to the number of edges on the paths, it follows that (T', f') is a depth-decomposition of M. □

4. Optimal tree-depth of a matrix. In this section, we relate the optimal dual tree-depth of a matrix A to its branch-depth. We start with showing that the branch-depth of a matrix A is at most its dual tree-depth.

PROPOSITION 14. *If A is an $m \times n$ matrix, then* $\mathrm{bd}(A) \leq \mathrm{td}_D(A)$.

Proof. We assume without loss of generality that the rows of the matrix A are linearly independent. Indeed, deleting a row of A that can be expressed as a linear combination of other rows of A does not change the structure of the matroid formed by the columns of A. In particular, the branch-depth of A is preserved by deleting such a row, and the deletion cannot increase the tree-depth of the dual graph $G_D(A)$ (the dual graph of the new matrix is a subgraph of the original dual graph and the tree-depth is monotone under taking subgraphs).

Let X be the set of rows of the matrix A and Y be the set of its columns. Further, let T be a rooted forest of height $\mathrm{td}_D(A)$ with the vertex set X such that its closure

contains the dual graph $G_D(A)$ as a subgraph. Consider the rooted tree T' obtained from T by adding a new vertex w, making w adjacent to the roots of all trees in T and also making w to be the root of T'. Since the rows of A are linearly independent, the number of edges of T' is equal to the row rank of A, which is the same as its column rank. In particular, the number of edges of T' is the rank of the vector matroid formed by the columns of A.

We next define a function $f : Y \to V(T')$ such that the pair (T', f) is a depth-decomposition of the vector matroid formed by the columns of A. Let y be a column of A, and observe that all rows x such that the entry in the row x and the column y is nonzero form a complete subgraph of the dual graph $G_D(A)$. Hence they must lie on some path from a leaf to the root of T'; set $f(y)$ to be any such leaf.

Since the depth of T' is $\mathrm{td}_D(A)$ (the height of T' is $\mathrm{td}_D(A) + 1$), the proof will be completed by showing that (T', f) is a depth-decomposition of the vector matroid formed by the columns of A. Consider a subset $Y' \subseteq Y$ of columns of A and let X' be the set of rows (vertices of the dual graph) on the path from $f(y)$ for some $y \in Y'$ to the root of T'. Note that $|X'|$ is equal to the number of edges contained in such paths. The definition of f yields that every column $y \in Y'$ has nonzero entries only in the rows x such that $x \in X'$. Hence, the rank of Y' is at most $|X'|$. It follows that the pair (T', f) is a depth-decomposition of the vector matroid formed by the columns of A. ∎

We next prove the main theorem of this section.

THEOREM 15. *Let A be an $m \times n$ matrix of rank m, let M be the vector matroid formed by columns of A, and let (T, f, g) be an extended depth-decomposition of M. Further, let $\mathrm{Im}(g) = \{w_1, \ldots, w_m\}$. The dual tree-depth of the $m \times n$ matrix A' such that the jth column of A is equal to*

$$\sum_{i=1}^{m} A'_{ij} w_i$$

is at most the depth of the tree T.

Proof. Let F be the rooted forest obtained from T by removing the root and associate the ith row of A' with the vertex v of F such that $g(v) = w_i$. Note that the height of F is the depth of T. We will establish that the dual graph $G_D(A')$ is contained in the closure $\mathrm{cl}(F)$ of the forest F. Let i and i', $1 \le i < i' \le m$, be such that the vertices of F associated with the ith and i'th rows of A' are adjacent in $G_D(A')$. This means that there exists j, $1 \le j \le n$, such that $A'_{ij} \neq 0$ and $A'_{i'j} \neq 0$. Let v be the leaf of T that is the f-image of the jth column of A. The definition of an extended depth-decomposition yields that the jth column is a linear combination of the g-image of the nonroot vertices on the path from v to the root of T. In particular, the path contains the two vertices of T mapped by g to w_i and $w_{i'}$; these two vertices are associated with the ith and i'th rows of A'. Hence, the vertices associated with the ith and i'th rows are adjacent in $\mathrm{cl}(F)$. We conclude that $G_D(A')$ is a subgraph of $\mathrm{cl}(F)$. ∎

We are now ready to prove Theorem 1.

Proof of Theorem 1. Let $\mathrm{td}^*_D(A)$ be the smallest dual tree-depth of a matrix that is row-equivalent to A. By Proposition 14, it holds that $\mathrm{bd}(A) \le \mathrm{td}^*_D(A)$. We now prove the other inequality. We can assume without loss of generality that the rank of A is equal to the number of its rows; if this is not the case, we can apply the

following arguments to the matrix A restricted to a maximal linearly independent set of rows and then use row operations to make all entries of the remaining rows to be equal to zero. Since rows with all entries equal to zero correspond to isolated vertices in the dual graph, their presence does not affect the dual tree-depth of the matrix.

Let M be the vector matroid formed by the columns of A. By Theorem 13, the matroid M has an extended depth-decomposition (T, f, g) with depth $\mathrm{bd}(A) = \mathrm{bd}(M)$. Let A' be the matrix from the statement of Theorem 15. Note that $A' = B^{-1}A$ where B is the $m \times m$ matrix such that the columns of B are the vectors $w_1, \ldots, w_m$ from the statement of Theorem 15. In particular, A' is row-equivalent to A since $\{w_1, \ldots, w_m\}$ is a basis. As $\mathrm{td}_D(A')$ is at most the depth of T, which is $\mathrm{bd}(A)$, it follows that $\mathrm{td}^*_D(A) \leq \mathrm{bd}(A)$. ∎

5. Algorithms for integer programming. The main purpose of this section is to combine Theorem 1 with the existing approximation algorithm for branch-depth (Theorem 8) to obtain an approximation algorithm for computing a row-equivalent matrix with small dual tree-depth (if it exists).

Proof of Theorem 2. Let A be an $m \times n$ matrix. Without loss of generality, we can assume that the rows of A are linearly independent, i.e., the rank of A is m. This also implies that the rank of the column space of A is m, in particular, $n \geq m$. We apply the approximation algorithm described in Theorem 8 to the vector matroid M formed by the columns of the matrix A, and obtain an extended depth-decomposition (T, f, g) of M. If the depth of T is larger than 4^d, then the branch-depth of A is larger than d; we report this and stop. Let B_g be the matrix with the columns formed by the vectors in $\mathrm{Im}(g)$ and let $B = B_g^{-1}$. Note that the matrix A' from the statement of Theorem 15 is equal to BA. By Theorem 15, the dual tree-depth of A' is at most 4^d.

We will next show that the entry complexity of A' is at most $O(d \cdot 4^d \cdot \mathrm{ec}(A))$. Note that the classical implementation of the Gaussian elimination in strongly polynomial time by Edmonds [12] yields that the entry complexity of the matrix B is $O(m \log m \cdot \mathrm{ec}(A))$ and this estimate is not sufficient to bound the entry complexity of A' in the way that we need. Let x be a column of A, and let W be the set of indices i, $1 \leq i \leq m$, such that the ith column of B_g is $g(v)$ for some nonroot vertex v on the path from $f(x)$ to the root of T. Note that $|W| \leq 4^d$ since the depth T is at most 4^d. Since the column x is a linear combination of the g-images of nonroot vertices on the path from $f(x)$ to the root of T, the ith entry of the column of A' that corresponds to x is zero if $i \notin W$. The remaining $|W|$ entries of this column of A' form a solution of the following system of at most 4^d linear equations: the system is given by a matrix obtained from B_g by restricting B_g to the columns with indices in W and to $|W|$ rows such that the resulting matrix has rank $|W|$, and the right-hand side of the system is formed by the entries of the column x in A corresponding to these $|W|$ rows. It follows (using Cramer's rule for solving systems of linear equations involving determinants) that a solution of this system has entry complexity at most $O(\log(4^d)! \cdot \mathrm{ec}(A)) = O(d \cdot 4^d \cdot \mathrm{ec}(A))$. Hence, the entry complexity of the matrix $A' = BA$, after dividing the numerator and the denominator of each entry by their greatest common divisor, is $O(d \cdot 4^d \cdot \mathrm{ec}(A))$. ∎

As explained in section 1, Theorem 2 yields Corollary 4, which asserts that integer programming is fixed parameter tractable when parameterized by the branch-depth and the entry complexity of the constraint matrix.

Proof of Corollary 4 *using Theorem* 2. Consider an integer program as in (1) that has branch-depth at most d. We apply the algorithm from Theorem 2 to obtain

a rational matrix B such that the instance with $A' = BA$, $\mathbf{b}' = B\mathbf{b}$, $\mathbf{l}' = \mathbf{l}$, and $\mathbf{u}' = \mathbf{u}$ has dual tree-depth D at most 4^d in case of Theorem 2.

To apply the algorithm from [13], we need to transform the matrix A' into an integer matrix. We do so by multiplying each row by the least common multiple of the denominators of the fractions in this row. Since all denominators are integers between 1 and $2^{\mathrm{ec}(A')}$, the value of this least common multiple is at most $2^{\mathrm{ec}(A')2^{\mathrm{ec}(A')}}$. Hence, the entry complexity of the resulting integer matrix A'' is $O(\mathrm{ec}(A')2^{\mathrm{ec}(A')})$. Since the dependence of the algorithm from [13] on the entry complexity $\mathrm{ec}(A'')$ and the dual tree-depth D of A'' is $2^{\left(\mathrm{ec}(A'')+D\right)D2^D}$, we obtain that the running time of the resulting algorithm depends on $\mathrm{ec}(A)$ and d as given in Table 1. ∎

We complement Corollary 4 by showing that integer programming is not fixed parameter tractable when parameterized by the "primal" branch-depth.

PROPOSITION 16. *Integer programming is* NP*-hard for instances with constraint matrices A satisfying* $\mathrm{bd}(A^T) = 1$ *and* $\mathrm{ec}(A) = 1$*, i.e., for instances such that the vector matroid formed by rows of the constraint matrix has branch-depth one.*

Proof. An integer program as in (1) such that the rows of the matrix A are not linearly independent is equivalent to an integer program with a matrix A' obtained from A by a restriction to a maximal linearly independent set of rows unless the rank of the matrix A with the column $\mathbf{b}$ added is larger than the rank of A; in the latter case, the integer program is infeasible. Hence, it is possible in polynomial time either to determine that the input integer program is infeasible or to find an equivalent integer program such that the rows of the constraint matrix are linearly independent and the matrix is a submatrix of the original constraint matrix. However, the branch-depth of the matroid formed by rows of such a (nonzero) matrix is one. Since integer programming is already NP-hard for instances such that all the entries of the constraint matrix are 0 or ± 1 (cf. [13, Proposition 101, part 2]), the proposition follows. ∎

6. Structure of extended depth-decompositions. In this section, we present structural results on extended depth-decompositions that we need to design a fixed parameter algorithm to compute a depth-decomposition of a vector matroid with an optimal depth. We start with the following lemma, which can be viewed as a generalization of Lemma 12; indeed, if the set U in the statement contains only the common root of the primary branches, then the statement of the lemma is the same as that of Lemma 12.

LEMMA 17. *Let (T, f) be a depth-decomposition of a vector matroid M and let U be a set of vertices of T such that every vertex contained in U has at least two children and every ancestor u' of a vertex in U such that u' has at least two children is also contained in U. Assume that every branch of T rooted at a vertex from U is at capacity.*

Then, every vertex $u \in U$ can be associated with a subspace L_u of the linear hull of the elements of M such that the dimension of L_u is the depth of u and the following holds. Let $S_1, \ldots, S_k$ be all branches rooted at u. If each ancestor of u has a single child, let L_0 be the vector space containing the zero vector only; otherwise, let u' be the nearest ancestor of u with at least two children, and let L_0 be the space $L_{u'}$. It holds that

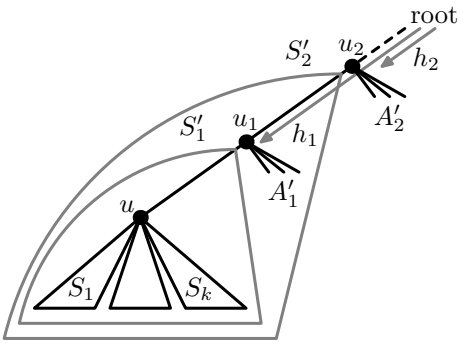


FIG. 3. *Notation used in the proof of Lemma* 17.

$$\dim \widehat{S_i} \cup L_0 = \|S_i\| + \dim L_u \qquad \textit{and} \qquad \mathcal{L}\left(\widehat{S_i} \cup L_0\right) \cap \mathcal{L}\left(\widehat{S_j} \cup L_0\right) = L_u$$

for all $1 \leq i < j \leq k$. *In particular,* $L_0 \subseteq L_u$.

Proof. We proceed by induction on the size of U. If U is empty, the lemma vacuously holds. Suppose that $|U| = 1$ and let u be the only vertex contained in U. By the assumption of the lemma, every branch rooted at u is primary. Hence, the statement of the lemma is implied by Lemma 12 and the fact that each branch rooted at u is at capacity.

Suppose that $|U| \geq 2$, and let u be any vertex of U with no descendant in U. Apply induction to the set $U \setminus \{u\}$ to get subspaces $L_{u'}$, $u' \in U \setminus \{u\}$, with properties given in the statement of the lemma. Let $S_1, \ldots, S_k$ be all branches rooted at u, and let $A_1, \ldots, A_k$ be the linear hulls of $\widehat{S_1}, \ldots, \widehat{S_k}$, respectively. Further, let $u_1, \ldots, u_\ell$ be all the vertices with at least two children on the path from the parent of u to the root (in this order), and let $h_1, \ldots, h_\ell$, be the depth of $u_1, \ldots, u_\ell$, respectively. See Figure 3 for an illustration of the notation. Note that $\{u_1, \ldots, u_\ell\} \subseteq U$; the choice of u implies that $\ell \geq 1$. Let S_i', $i = 1, \ldots, \ell$, be the branch rooted at u_i that contains u and A_i' be the linear hull of the elements assigned to leaves of the branches rooted at u_i different from S_i'. Note that $\widehat{S_1'} = \widehat{S_1} \cup \cdots \cup \widehat{S_k}$.

Set $L_i = L_{u_i}$ for $i = 1, \ldots, \ell$ and also set $L_{\ell+1}$ to be the vector space containing the zero vector only. Note that $L_{\ell+1} \subseteq L_\ell \subseteq \cdots \subseteq L_1$ and the dimension of L_i is h_i for $i = 1, \ldots, \ell$; the space L_1 is the vector space L_0 from the statement of the lemma with respect to u. The following holds by the induction assumption for every $i = 1, \ldots, \ell$: if R is a branch rooted at the vertex u_i, then

$$\dim \widehat{R} \cup L_{i+1} = \dim L_i + \|R\|, \tag{3}$$

and if R' is another branch rooted at the vertex u_i, then

$$\mathcal{L}\left(\widehat{R} \cup L_{i+1}\right) \cap \mathcal{L}\left(\widehat{R'} \cup L_{i+1}\right) = L_i. \tag{4}$$

The identity (3) for $i = 1$ and $R = S_1'$ yields that

$$\dim L_2 \cup \widehat{S_1'} = \dim L_1 + \|S_1'\|. \tag{3'}$$

Using (4) iteratively for $i = \ell, \ldots, 1$, we obtain the following; the iterative argument uses that $\widehat{S_i'} \cup A_i' \subseteq \mathcal{L}\left(\widehat{S_{i+1}'}\right)$,

$$L_i \subseteq \mathcal{L}(A'_i \cup L_{i+1}) \subseteq \cdots \subseteq \mathcal{L}(A'_i \cup \cdots \cup A'_\ell), \tag{5}$$

$$\mathcal{L}\left(\widehat{S'_i} \cup L_{i+1}\right) \cap \mathcal{L}(A'_i \cup L_{i+1}) = L_i,$$

$$\mathcal{L}\left(\widehat{S'_i} \cup L_{i+1}\right) \cap \mathcal{L}(A'_i \cup \cdots \cup A'_\ell) = L_i. \tag{6}$$

Let A' be the linear hull of $A'_i \cup \cdots \cup A'_\ell$. The relations (5) and (6) yield for $i = 1$ the following:

$$L_1 \subseteq \mathcal{L}(A'_1 \cup \cdots \cup A'_\ell) = A', \tag{5'}$$

$$\mathcal{L}\left(\widehat{S'_1}\right) \cap A' \subseteq L_1. \tag{6'}$$

Now we obtain using (6) that

$$\dim A'_i \cup \cdots \cup A'_\ell = r(M) - \|S'_i\|. \tag{7}$$

This implies for $i = 1$ that

$$\dim A' = r(M) - \|S'_1\|. \tag{7'}$$

The lemma requires establishing the existence of a vector space L_u such that

$$\dim A_i \cup L_1 = \|S_i\| + \dim L_1 + h \qquad \text{and} \qquad \mathcal{L}(A_i \cup L_1) \cap \mathcal{L}(A_j \cup L_1) = L_u$$

for all $1 \le i < j \le k$ and such that the dimension of L_u is $\dim L_1 + h$, where h is the distance between u and u_1. Since every branch rooted at u is at capacity, we obtain using (7') that

$$\dim A_i \cup A' = r(M) - (\|S'_1\| - h - \|S_i\|) = \dim A' + h + \|S_i\| \tag{8}$$

for every $i = 1, \ldots, k$. Since every A_i is a subspace of $\mathcal{L}\left(\widehat{S'_1}\right)$, we derive from (5'), (6'), and (8) that

$$\begin{aligned} \dim A_i \cup L_1 &= \dim(A_i \cup A') + \dim(\mathcal{L}(A_i \cup L_1) \cap A') - \dim(A') \\ &= \dim L_1 + h + \|S_i\|. \end{aligned} \tag{9}$$

Since (T, f) is a depth-decomposition, it holds that

$$\dim A' \cup \bigcup_{j \in J} A_j \le r(M) - \|S'_1\| + h + \sum_{j \in J} \|S_j\| = \dim A' + h + \sum_{j \in J} \|S_j\| \tag{10}$$

for all $J \subseteq \{1, \ldots, k\}$, and analogously to the proof of (9), we derive from (10) that

$$\dim L_1 \cup \bigcup_{j \in J} A_j \le \dim L_1 + h + \sum_{j \in J} \|S_j\|. \tag{11}$$

We now obtain using (9) and (11) for $J = \{i, j\}$ that

$$\begin{aligned} &\dim L_1 + h + \|S_i\| + \|S_j\| \\ &\ge \dim A_i \cup A_j \cup L_1 \\ &= \dim A_i \cup L_1 + \dim A_j \cup L_1 - \dim \mathcal{L}(A_i \cup L_1) \cap \mathcal{L}(A_j \cup L_1) \\ &= 2 \dim L_1 + 2h + \|S_i\| + \|S_j\| - \dim \mathcal{L}(A_i \cup L_1) \cap \mathcal{L}(A_j \cup L_1) \end{aligned}$$

for all $1 \leq i < j \leq k$. It follows that

$$\dim \mathcal{L}(A_i \cup L_1) \cap \mathcal{L}(A_j \cup L_1) \geq \dim L_1 + h \tag{12}$$

for all $1 \leq i < j \leq k$. On the other hand, it holds using (3'), (9), and (11) applied for $J = \{1, \ldots, k\} \setminus \{i\}$ that

$$\begin{aligned}
&\dim \mathcal{L}(A_i \cup L_1) \cap \mathcal{L}(A_j \cup L_1) \\
&\leq \dim \mathcal{L}(A_i \cup L_1) \cap \mathcal{L}\left(L_1 \cup \bigcup_{j' \neq i} A_{j'}\right) \\
&= \dim A_i \cup L_1 + \dim L_1 \cup \bigcup_{j' \neq i} A_{j'} - \dim L_1 \cup \widehat{S_1'} \\
&\leq \dim L_1 + h + \|S_i\| + \dim L_1 + h + \sum_{j' \neq i} \|S_{j'}\| - \dim L_1 - \|S_1'\| \\
&= \dim L_1 + h.
\end{aligned}$$

We conclude that equality always holds in (12). Hence, there exists a subspace L_u of dimension $\dim L_1 + h$ such that

$$\begin{aligned}
&\dim A_i \cup L_1 = \|S_i\| + \dim L_1 + h = \|S_i\| + \dim L_u \qquad \text{and} \\
&\mathcal{L}(A_i \cup L_1) \cap \mathcal{L}(A_j \cup L_1) = L_u
\end{aligned}$$

for all $1 \leq i < j \leq k$. ∎

Using Lemma 17, we prove the following.

LEMMA 18. *Let (T, f) be a depth-decomposition of a vector matroid M, u_1 a vertex of T with at least two children, and $u_2, \ldots, u_k$ all ancestors of u_1 with at least two children (listed in the increasing distance from u_1). Assume that every branch rooted at one the vertices $u_1, \ldots, u_k$ is at capacity, and let L_1 be the space L_{u_1} from the statement of Lemma 17 applied with $U = \{u_1, \ldots, u_k\}$. Further, let S_1 be any branch rooted at u_1 and f_1 the restriction of f to $\widehat{S_1}$.*

The pair (S_1, f_1) is a depth-decomposition of the vector matroid $(M/L_1)[\widehat{S_1}]$ and a branch of (S_1, f_1) is at capacity if and only if it is at capacity in (T, f). In addition, if (S_1', f_1') is another depth-decomposition of the matroid $(M/L_1)[\widehat{S_1}]$, then (T', f') is a depth-decomposition of the matroid M, where T' is obtained from T by replacing S_1 with S_1', and the function f' is defined as $f'(x) = f_1'(x)$ for $x \in \widehat{S_1}$, and $f'(x) = f(x)$ otherwise.

Proof. Let X be the set of elements of M, let A_1 be the linear hull of $\widehat{S_1}$, and let A_1' be the linear hull of $X \setminus \widehat{S_1}$. By Lemma 17, it holds that

$$A_1 \cap A_1' \subseteq L_1 \quad \text{and} \quad L_1 \subseteq A_1'. \tag{13}$$

It follows that the matroids $(M/L_1)[\widehat{S_1}]$ and $(M/A_1')[\widehat{S_1}]$ are the same. In addition, since all branches rooted at $u_1, \ldots, u_k$ are at capacity, it also holds that $\dim A_1' = \dim M - \|S_1\|$ by (7').

Instead of verifying that (S_1, f_1) is a depth-decomposition of $(M/L_1)[\widehat{S_1}]$, we verify that (S_1, f_1) is a depth-decomposition of $(M/A_1')[\widehat{S_1}]$, which is equivalent since the two matroids are the same. Let X' be any subset of $\widehat{S_1}$ and let e be the number of

edges on the paths from the f_1-image of X' to the root of S_1. Since (T, f) is a depth decomposition of the matroid M, we obtain that

$$\dim X' \cup A'_1 = \dim X' \cup \left(X \setminus \widehat{S_1}\right) \leq \dim M - \|S_1\| + e.$$

Since the rank of the set X' in $(M/A'_1)[\widehat{S_1}]$ is $\dim X' \cup A'_1 - \dim A'_1$ and $\dim A'_1 = \dim M - \|S_1\|$, we obtain that the rank of X' in $(M/A'_1)[\widehat{S_1}]$ is at most e as desired. Hence, (S_1, f_1) is a depth-decomposition of $(M/L_1)[\widehat{S_1}]$.

Let S be a branch of S_1 rooted at a vertex u, let X' be the elements assigned to the leaves of the other branches rooted at u, and let e' be the number of edges contained in the other branches rooted at u. The branch S is at capacity in the depth-decomposition (S_1, f_1) of the matroid $(M/L_1)[\widehat{S_1}]$ if and only if the rank of $\widehat{S_1} \setminus X'$ in the matroid $(M/L_1)[\widehat{S_1}]$ is $\|S_1\| - e'$. The rank of the set $\widehat{S_1} \setminus X'$ in the matroid $(M/L_1)[\widehat{S_1}]$ is equal to $\dim(\widehat{S_1} \setminus X') \cup L_1 - \dim L_1$, which is equal to $\dim(\widehat{S_1} \setminus X') \cup A'_1 - \dim A'_1$ by (13). Since $\dim A'_1 = \dim M - \|S_1\|$, we infer that the branch S is at capacity in the depth-decomposition (S_1, f_1) of the matroid $(M/L_1)[\widehat{S_1}]$ if and only if $\dim(\widehat{S_1} \setminus X') \cup A'_1$ is $\dim M - e'$. The latter holds if and only if S is at capacity in the depth-decomposition (T, f) of the matroid M.

It remains to prove the last part of the lemma. We proceed by induction of k. The base case is $k = 1$, i.e., the case when the branch S_1 is primary. Let (S'_1, f'_1) be another depth-decomposition of the matroid $(M/L_1)[\widehat{S_1}]$, and let (T', f') be obtained as described in the statement of the lemma. Denote by $S_2, \ldots, S_\ell$ the remaining branches rooted at u_1. Let X' be any subset of the elements of M, let e_i, $i = 1, \ldots, \ell$, be the number of edges on the paths from the vertices in the f'-image of $X' \cap \widehat{S_i}$ to u_1, and let f_i, $i \geq 2$ be the restriction of f to $\widehat{S_i}$. Since (S'_1, f'_1) is a depth-decomposition of the matroid $(M/L_1)[\widehat{S_1}]$, we obtain that

$$\dim \left(X' \cap \widehat{S_1}\right) \cup L_1 - \dim L_1 \leq e_1. \tag{14}$$

Similarly, as (S_i, f_i) is a depth-decomposition of the matroid $(M/L_1)[\widehat{S_i}]$ by the already proven part of this lemma, we obtain that

$$\dim \left(X' \cap \widehat{S_i}\right) \cup L_1 - \dim L_1 \leq e_i. \tag{15}$$

Using (14) and (15), we infer that

$$\dim X' \leq \dim X' \cup L_1 \ = \dim \bigcup_{i=1}^{\ell} \left[(X' \cap \widehat{S_i}) \cup L_1\right] \leq \dim L_1 + \sum_{i=1}^{\ell} e_i.$$

Since the choice of X' was arbitrary, (T', f') is a depth-decomposition of M.

The inductive step proceeds as follows. Let $k \geq 2$, let S_2 be the branch rooted at u_2 in T containing S_1, let S'_2 be the branch rooted at u_2 in T' containing S'_1, and let f_2 and f'_2 be the restrictions of f and f', respectively, to the elements of $\widehat{S_2} = \widehat{S'_2}$. Finally, let L_2 be the space L_{u_2} from the statement of Lemma 17 applied with $U = \{u_1, \ldots, u_k\}$. By the already proven part of the lemma, (S_2, f_2) is a depth-decomposition of the matroid $(M/L_2)[\widehat{S_2}]$. By the base of the induction, which we have already proven, (S'_2, f'_2) is also a depth-decomposition of the matroid $(M/L_2)[\widehat{S_2}]$. We now apply the induction to the vertex u_2 of T with replacing the branch S_2 with S'_2 in T and conclude that (T', f') is a depth-decomposition of the matroid M. ∎

We next extend Lemma 10 to all branches.

LEMMA 19. *Let (T, f) be a depth-decomposition of a vector matroid M and S_0 be a branch of T rooted at a vertex u_0 such that S_0 is not at capacity. Suppose that every branch rooted at an ancestor of u_0 is at capacity. Let T' be the rooted tree obtained from T by changing the root of S_0 to be the parent of u_0. Then, (T', f) is a depth-decomposition of M.*

Proof. If S_0 is primary, we apply Lemma 10. Hence, we can assume that S_0 is not primary. Let U be the set of ancestors of u_0 that have at least two children; note that U is nonempty since the branch S_0 is not primary. We next apply Lemma 17 to get subspaces L_u, $u \in U$, with properties described in the statement of the lemma. Let u_1 be the vertex of U nearest to u_0, L_1 the vector space L_{u_1}, S_1 the branch rooted at u_1 that contains u_0, and f_1 the function f restricted to the leaves of S_1. By Lemma 18, (S_1, f_1) is a depth-decomposition of the matroid $(M/L_1)[\widehat{S_1}]$ and the branch S_0 is not at capacity in $(M/L_1)[\widehat{S_1}]$.

Let S_1' be the rooted tree obtained from S_1 by changing the root of S_0 to be the parent of u_0. Since the branch S_0 is primary in the depth-decomposition (S_1, f_1), we obtain that (S_1', f_1) is a depth-decomposition of the matroid $(M/L_1)[\widehat{S_1}]$ by Lemma 10. The fact that (T', f) is a depth-decomposition of M now follows from Lemma 18. ∎

We are now ready to present one of the main results of this section.

THEOREM 20. *There exists a polynomial time algorithm that, given a vector matroid M and a depth-decomposition (T, f) of M, outputs an extended depth-decomposition (T', f', g) of M such that the depth of T' is at most the depth of T and every branch of T' is at capacity.*

Proof. The algorithm first modifies (T, f) to a depth-decomposition (T', f') such that every branch of T' is at capacity. This is done iteratively as follows. At each iteration, the algorithm searches in the increasing order given by the distance from the root for a vertex u with at least two children such that a branch rooted at u is not at capacity. The depth-decomposition is then modified by changing the root of the branch that is not at capacity to the parent of u (note that u cannot be the root of the whole tree by Lemma 9). Lemma 19 implies that the new tree is again a depth-decomposition of M. This finishes the iteration and the algorithm starts a new iteration (again searching in the order given by the distance from the root). Note that the number of leaves is preserved and at each iteration the sum of the distances of the leaves to the root of the tree decreases. Since T has $r(M)$ edges, it has at most $r(M)$ leaves and each leaf is at distance at most $r(M)$ from the root. It follows that the algorithm stops after at most $r(M)^2$ iterations producing a depth-decomposition (T', f') of M such that every branch of T' is at capacity. Since the depth of the tree is never increased by the algorithm, the depth of T' is at most the depth of T.

We next construct the function g. Let U be the set containing the root of T' and all vertices of (T', f') with at least two children, and let L_u be the vector spaces as described in Lemma 17 while setting L_u to be the space containing the zero vector only for the root u of T'. Observe that the spaces L_u, $u \in U$, can be algorithmically constructed. Indeed, if $u \in U$ and the space $L_{u'}$ has already been constructed for the nearest ancestor u' of u contained in U, then L_u is the intersection of the linear hulls of $\widehat{S_1} \cup L_{u'}$ and $\widehat{S_2} \cup L_{u'}$ where S_1 and S_2 are any two branches rooted at u.

The function g is defined for the vertices of T' in the order based on their distance from the root. Let v be a vertex of T' and assume that g has been defined for all

ancestors of v (except the root). We distinguish two cases. The first case is that v has at least two descendants that are leaves. If v has at least two children, then let u be the vertex v itself, and let u be the nearest descendant of v contained in U otherwise. We set $g(v)$ to be any vector of L_u that is linearly independent of the g-image of the vertices on the path from the parent of v to the root. Since the dimension of L_u is the depth of u, which is at least the depth of v, such a vector always exists. The other case is that v and all its descendants have at most one child. If v is a leaf, let v' be the vertex v itself. Otherwise, let v' be the only leaf descendant of v. We set $g(v)$ to be any vector in the f'-preimage of v' that is linearly independent of the g-image of the vertices on the path from the parent of v to the root. Since the branch S containing v that is rooted at the nearest ancestor u of v contained in U is a depth-decomposition of the matroid $(M/L_u)[\widehat{S}]$ (see Lemma 18), such a vector always exists. Note that the function g can be algorithmically constructed.

We now verify that (T', f', g) is an extended depth-decomposition of the matroid M. Observe that for every vertex $u \in U$, L_u contains all spaces $L_{u'}$ for ancestors u' of u contained in U. Hence, the g-image of the vertices on the path from u to the root form a basis of L_u for every $u \in U$. In particular, $K_u = L_u$ for every $u \in U$. Let v be a leaf of T' and u its nearest ancestor contained in U. By Lemma 18, the branch S rooted at u containing v together with the restriction of f' to $\widehat{S}$ is a depth-decomposition of the matroid $(M/L_u)[\widehat{S}]$. Note that the branch S is actually a path rooted at u. This implies that the g-image of the vertices on the path from v to the child of u contained in S form a basis of the linear hull of the f'-preimage of v quotiened by L_u. Since the g-image of the vertices on the path from u to the root form a basis of $K_u = L_u$, we conclude that every vector in the f'-preimage of v is contained in the linear hull of the g-image of the vertices on the path from v to the root of T'. ∎

We obtain the following two statements as corollaries of Theorem 20.

COROLLARY 21. *Every vector matroid M has a depth-decomposition (T, f) with depth* $\mathrm{bd}(M)$ *such that every branch of T is at capacity.*

COROLLARY 22. *If (T, f) is a depth-decomposition of a vector matroid M, then there exists g such that (T, f, g) is an extended depth-decomposition of M.*

Proof. Let (T', f', g) be the extended depth-decomposition of M constructed in Theorem 20. Since T' was obtained from T by rerooting some of the branches, the vertices of T and T' are in one-to-one correspondence. In particular, the roots of T and T' are the same vertex, the functions f and f' are identical, and g is a well-defined function from the nonroot vertices of T. Further, notice that any vertex on a given root-to-leaf path in T' is also on the path from the root to the corresponding leaf in T. Since (T', f', g) is an extended depth decomposition, any element x of M is contained in the linear hull of the g-image of the vertices on the path in T' from $f(x)$ to the root, and thus (T, f, g) is an extended depth decomposition as well. ∎

We conclude this section with a theorem that asserts that every vector matroid has a depth-decomposition of minimum depth such that every branch is both at capacity and solid. Before we can state and prove the theorem, we need three auxiliary lemmas.

LEMMA 23. *Let M be a vector matroid with no loops and $M_1, \ldots, M_k$ be its components. For each i, suppose (T_i, f_i, g_i) is an extended depth-decomposition of M_i. Let T be the rooted tree obtained from the trees $T_1, \ldots, T_k$ by identifying their roots, let f be the mapping from the elements of M to the leaves of T such that $f(x) = f_i(x)$*

if x belongs to M_i, and let g be the mapping such that $g(v) = g_i(v)$ if v is a nonroot vertex of T_i. The triple (T, f, g) is an extended depth-decomposition of M.

Proof. Let $X_1, \ldots, X_k$ be the elements of M contained in $M_1, \ldots, M_k$, respectively. Since $M_1, \ldots, M_k$ are components of M, the rank of M is the sum of the ranks of $M_1, \ldots, M_k$. In particular, the number of edges of T is the rank of M. Since every vertex $x \in X_i$ can be expressed as a linear combination of the g_i-image of the vertices on the path from $f_i(x)$ to the root of T_i, it is also a linear combination of the g-image of the vertices on the path from $f(x)$ to the root of T. In particular, the linear hull of the g-image of all nonroot vertices of T is equal to the linear hull of $X_1 \cup \cdots \cup X_k$. This implies that the vectors in the g-image of all nonroot vertices of T are linearly independent. Let X' be any set of vertices of M. Since any vector in X' can be expressed as a linear combination of the g-image of the nonroot vertices on the paths from $f(X')$ to the root, the dimension of the linear hull of X' is at most the number of such vertices, which is equal to the number of edges on the paths. It follows that (T, f, g) is an extended depth-decomposition of M. ∎

LEMMA 24. *Let (T, f, g) be an extended depth-decomposition of a vector matroid M, and let u be a vertex with at least two children. If S is a branch rooted at u, then $\widehat{S}$ is a union of some components and loops of M/K_u.*

Proof. Let X be all the vectors of M, A their linear hull, and A_S the linear hull of the g-images of the nonroot vertices of S. Since the vectors of $\mathrm{Im}(g)$ form a basis of the vector space A and every element x of the matroid M is a linear combination of the vectors in the g-image of the vertices on the path from $f(x)$ to the root, A_S is a subset of the linear hull of $\widehat{S}$, and the linear hull of $\widehat{S}$ is a subset of the linear hull of $A_S \cup K_u$. Since the dimension of A_S is $\|S\|$, we obtain that

$$\dim \widehat{S} \cup K_u - \dim K_u = \dim A_S \cup K_u - \dim K_u = \|S\|.$$

Along the same lines, we obtain that

$$\dim \left(X \setminus \widehat{S}\right) \cup K_u - \dim K_u = \dim X - \dim K_u - \|S\|.$$

Since the rank of M/K_u is $\dim X - \dim K_u$, the rank of $\widehat{S}$ in M/K_u is $\|S\|$, and the rank of $X \setminus \widehat{S}$ in M/K_i is $\dim X - \dim K_u - \|S\|$. Hence, the set $\widehat{S}$ is a union of components and loops of M/K_u. ∎

LEMMA 25. *Let (T, f, g) be an extended depth-decomposition of a vector matroid M, and let u be a vertex with at least two children. Further, let S be a branch rooted at u and (T', f', g') be an extended depth-decomposition of the matroid $(M/K_u)[\widehat{S}]$. Let T'' be the rooted tree obtained by removing from T the branch S and identifying the root of T' with u, setting $f''(x) = f'(x)$ for elements $x \in \widehat{S}$ and $f''(x) = f(x)$ for other elements x of M, and setting $g''(v) = g'(v)$ for nonroot vertices of T' and $g''(v) = g(v)$ for other nonroot vertices of T''. The triple (T'', f'', g'') is an extended depth-decomposition of M.*

Proof. Since the rank of the matroid $(M/K_u)[\widehat{S}]$ is equal to $\dim \widehat{S} \cup K_u - \dim K_u = \|S\|$, the trees T' and S have the same number of edges. This implies that the trees T and T'' also have the same number of edges. In order to establish that (T'', f'', g'') is an extended depth-decomposition of M, it is now enough to verify that every element x of the matroid M is a linear combination of the g''-image of the vertices on the path from $f''(x)$ to the root. If this is the case, then the g''-image of all nonroot vertices

of T'' form a basis of the vector space generated by the elements of M, and the rank of any subset X' of the elements of M is at most the number of nonroot vertices on the paths from the f''-image of X' to the root, which is equal to the number of edges on such paths.

Let x be any element of the matroid M. If $x \notin \widehat{S}$, then $f(x) = f''(x)$ and the g-image of the vertices on the path from $f(x) = f''(x)$ to the root is the same as the g''-image, in particular, x is a linear combination of the g''-image of the vertices on this path. Hence, we need to analyze the case when $x \in \widehat{S}$. In this case, the path from $f''(x)$ to the root contains all vertices on the path from u to the root of T'', which implies that the linear hull of the g''-image of the vertices on the path from $f''(x)$ to the root of T'' contains K_u. Since (T', f', g') is an extended depth-decomposition of the matroid $(M/K_u)[\widehat{S}]$, x is contained in the linear hull of the union of K_u and the g'-image of the vertices on the path from $f'(x)$ to the root of T'. Hence, x is contained in the linear hull of the g''-image of the vertices on the path from $f''(x)$ to the root of T''. We conclude that (T'', f'', g'') is an extended depth-decomposition of M. ∎

We are now ready to prove the final theorem of this section.

THEOREM 26. *Every vector matroid M has an extended depth-decomposition (T, f, g) of depth $\mathrm{bd}(M)$ such that every branch of T is both at capacity and solid.*

Proof. We start with a depth-decomposition (T, f, g) of M with depth $\mathrm{td}(M)$ and modify it iteratively as follows. At each iteration, we first apply Theorem 20 to obtain a depth-decomposition such that every branch is at capacity. If every branch is solid, we stop. If there is a branch S that is not solid, we proceed as follows. Since S is not solid, the matroid $(M/K_u)[\widehat{S}]$ is not connected, where u is the root of S. Let $M_1, \ldots, M_k$ be the components of the matroid $(M/K_u)[\widehat{S}]$ and let X_u be the set containing all loops of the matroid $(M/K_u)[\widehat{S}]$. Let (S_i, f_i, g_i) be an extended depth-decomposition of M_i, $i = 1, \ldots, k$, with depth $\mathrm{bd}(M_i)$. Since the branch-depth $\mathrm{bd}(M_i)$ of M_i is at most the branch-depth of $(M/K_u)[\widehat{S}]$ (as the branch-depth is a minor-monotone parameter), the depth of each of the trees $S_1, \ldots, S_k$ is at most the depth of S. By Lemmas 23 and 25, it is possible to replace the branch S with the branches $S_1, \ldots, S_k$ rooted at the root of S while assigning the elements of X_u to arbitrary leaves of the branches $S_1, \ldots, S_k$. Note that the depth of the new rooted tree does not exceed the depth of the original rooted tree. In this way, we obtain a new extended depth-decomposition of M, and we proceed to the next iteration.

We need to argue that the procedure described above eventually finishes. Let a_i be the sum of the degrees of the vertices at distance i from the root. During the iterations, the rooted tree T is modified by the algorithm presented in Theorem 20 and by the procedure described in the first paragraph. The algorithm presented in Theorem 20 selects a branch that is not at capacity and reroots it to the parent of its root. Hence, there always exists i_0 such that $a_0, \ldots, a_{i_0-1}$ are preserved and a_{i_0} has increased by one. Similarly, in the procedure described in the first paragraph, the degree of the vertex u has increased while the degrees of all vertices with distance to the root smaller than u have not changed, in particular, there exists i_0 such that $a_0, \ldots, a_{i_0-1}$ are preserved and a_{i_0} has increased. We conclude that the vector $(a_0, \ldots, a_{\mathrm{bd}(M)})$ lexicographically increases at each modification of the tree T. Since the sum of the degrees of the vertices at distance i from the root is bounded by the rank r of M, which is the total number of edges of T, there are at most $r^{\mathrm{bd}(M)+1}$ vectors that can

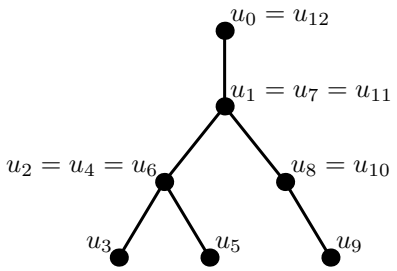


FIG. 4. *An example of a depth-first-search transversal of a rooted tree.*

represent a sequence of sums of degrees of vertices of T at distance $0, 1, \ldots, \mathrm{bd}(M)$ from the root. Hence, the procedure terminates after at most $r^{\mathrm{bd}(M)+1}$ iterations. ∎

7. Algorithm for finite fields. In this section, we design a fixed parameter algorithm for computing a depth-decomposition of a vector matroid over a fixed finite field. To do so, we need to introduce additional notation. Let (T, f, g) be an extended depth-decomposition of a vector matroid M, and let r be the rank of M. Let $u_0, \ldots, u_{2r}$ be a depth-first-search transversal of the tree T (see Figure 4 for an illustration). For $i \in \{0, \ldots, 2r\}$, we define A_i to be the linear hull of K_{u_i} and the f-preimage of the leaves among the vertices $u_0, \ldots, u_i$. Similarly, we define B_i to be the linear hull of K_{u_i} and the f-preimage of the leaves among the vertices $u_i, \ldots, u_{2r}$. The sequence $(u_i, A_i, B_i)_{i \in \{0,\ldots,2r\}}$ is called a *transversal sequence* for (T, f, g). Note that $A_i \cap B_i = K_{u_i}$ by the fact that $\mathrm{Im}(g)$ is a basis of the linear hull of elements of M. If (T, f, g) is principal and (T', f', g') is another extended depth-decomposition of M, we say that a branch S of T' is *i-crossed* if $\widehat{S}$ contains the g-image of a vertex on the path from u_i to the root of T.

LEMMA 27. *Let M be a vector matroid with rank r, (T, f, g) a principal extended depth-decomposition of M, and (T', f', g') a solid extended depth-decomposition of M. Further, let $(u_i, A_i, B_i)_{i \in \{0,\ldots,2r\}}$ be a transversal sequence for (T, f, g). If S is a branch of (T', f', g') that is not i-crossed, then $\widehat{S}$ is a subset of $A_i \cup K_v$ or $B_i \cup K_v$, where v is the root of S.*

Proof. Let v be the root of S, V the set of all vertices of T' that are not descendants of v in S, and C the linear hull of $g'(V)$. Since $\mathrm{Im}(g)$ is a base of the linear hull of elements of M, the matroids $(M/K_v)[\widehat{S}]$ and $(M/C)[\widehat{S}]$ are the same. Since the branch S is solid in (T', f', g'), it follows that $\widehat{S}$ is the union of a component of M/C and possibly some loops corresponding to vectors contained in $K_v \subseteq C$.

Let X be the set of elements of M. By the definition of an extended depth-decomposition, the sets $X \cap A_i$ and $X \cap B_i$ are unions of components and loops of the matroid M/K_{u_i}. Since S is not i-crossed, the leaf $f'(g(v'))$ for every vertex v' on the path from u_i to the root of T is contained in V. It follows that $g(v')$ is contained in C, and so K_{u_i} is a subspace of C. Hence, every component of the matroid M/K_{u_i} is a union of components and loops of the matroid M/C. In particular, each of the sets $X \cap A_i$ and $X \cap B_i$ is a union of components and loops of the matroid M/C. Since $\widehat{S}$ is the union of a component of M/C and possibly some vectors from K_v, it must hold that $\widehat{S}$ is a subset of the union of $X \cap A_i$ and K_v or the union of $X \cap B_i$ and K_v. The lemma follows. ∎

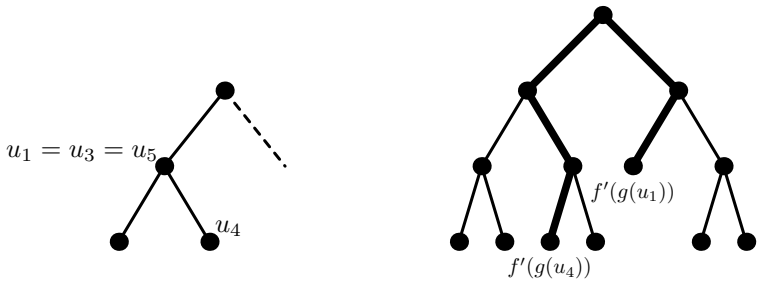


FIG. 5. *An example of the construction of an i-frontier. A part of the principal extended depth-decomposition (T, f, g) is depicted on the left and the extended depth-decomposition (T', f', g') is shown on the right; the subtree T_0 for the 4-frontier is depicted in bold.*

We will design a dynamic programming algorithm, which will construct an optimal depth-decomposition of a vector matroid M using the information on the structure of M captured by an extended depth-decomposition of M produced by an approximation algorithm given in Theorem 8. The depth-decomposition will be constructed iteratively for elements of M in the order that the leaves corresponding to them appear in the transversal sequence of the depth-decomposition produced by the approximation algorithm. Since it would not be feasible to store all possible "partial" depth-decompositions, we need a more succinct way of representing an already constructed part of a depth-decomposition, which we now formally introduce.

Let T_0 be a rooted tree; we say that a mapping h from the nonroot vertices to vectors is *k-matchable* for some $k \in \mathbb{N}$ if there exists a surjective mapping g from $\{1, \ldots, k\}$ to the leaves of T_0 such that for every $j = 1, \ldots, k$, the linear hull of the h-image of the vertices on the path from $g(j)$ to the root contains the jth unit vector. A *frontier* is a tuple (T_0, d, a, b, h) such that T_0 is a rooted tree, d, a, and b are nonnegative integers that sum to the number of edges of T_0 (in particular, T_0 has at most d leaves), and h is a mapping from the nonroot vertices of T_0 to $\mathbb{F}^{d+a+b}$ such that $\mathrm{Im}(h)$ is a basis of $\mathbb{F}^{d+a+b}$ and h is d-matchable. We will refer the middle a coordinates of h-images as *A-coordinates* and to the last b coordinates as *B-coordinates*.

Let (T, f, g) be a principal extended depth-decomposition of a vector matroid M with rank r over a field $\mathbb{F}$, $(u_i, A_i, B_i)_{i \in \{0,\ldots,2r\}}$ a transversal sequence for (T, f, g), and (T', f', g') another extended depth-decomposition of a matroid M. The *i-frontier* of (T', f', g') with respect to (T, f, g) and $(u_i, A_i, B_i)_{i \in \{0,\ldots,2r\}}$ is the frontier (T_0, d, a, b, h) obtained as described below; see Figure 5 for an illustration.

- The integer d is the depth of u_i in T.
- T_0 is the rooted subtree of T' formed by the paths from the root of T' to the f'-images of $v_1^u, \ldots, v_d^u$, where $v_1^u, \ldots, v_d^u$ are the g-images of the vertices on the path from the root of T to u_i (in this order).
- The integers a and b are the smallest integers for that there exists an a-dimensional subspace L_A of A_i and a b-dimensional subspace L_B of B_i such that the linear hull of the g'-images of the vertices of T_0 is a subspace of the linear hull of $v_1^u, \ldots, v_d^u$, L_A, and L_B.
- Finally, h is a mapping from the nonroot vertices of T_0 to $\mathbb{F}^{d+a+b}$ that satisfies the following: Let $v_1^A, \ldots, v_a^A$ be a basis of L_A, and let $v_1^B, \ldots, v_b^B$ be a basis of L_B. The value $h(v)$ for a nonroot vertex v of T_0 is equal

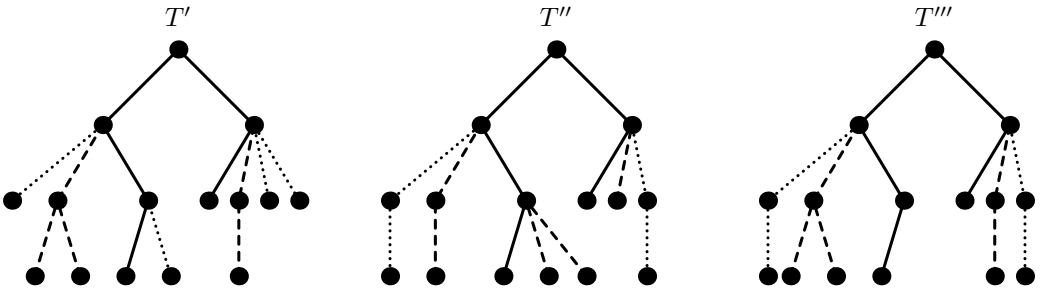


FIG. 6. *An illustration of the operation described in the statement of Lemma 28. The trees T', T'', and T''' are respectively depicted on the left, middle, and right. The edges of the i-frontier are drawn solid, the edges of branches S of T' such that $\widehat{S} \subseteq A_i \cup K_A$ and branches S of T'' such that $\widehat{S} \subseteq A_i \cup K_B$ are drawn dashed, and the edges of branches S of T' such that $\widehat{S} \subseteq B_i \cup K_A$ and branches S of T'' such that $\widehat{S} \subseteq B_i \cup K_B$ are drawn dotted; branches of T''' are drawn in the same way as the branches of T' and T'' corresponding to them.*

to the coordinates of $g'(v)$ with respect to the (linearly independent) vectors $v_1^u, \dots, v_d^u, v_1^A, \dots, v_a^A, v_1^B, \dots, v_b^B$.

Note that we use i both as an index for the transversal sequence and as an index of the frontier in order to emphasize the link between the two indices. To see that i-frontiers are indeed frontiers, observe first that K_{u_i} is the linear hull of $v_1^u, \dots, v_d^u$. Since $K_{u_i} = A_i \cap B_i$ and K_{u_i} is contained in the linear hull of the g'-images of the vertices of T_0, the subspaces $L_A \subseteq A_i$ and $L_B \subseteq B_i$ are uniquely determined and the dimension of the linear hull of the g'-images of the vertices of T_0 is equal to $d+a+b$. Since g' is a bijection from the nonroot vertices of T to a basis of the linear hull of elements of M, the number of edges of T_0 must be $d+a+b$. Finally, since T_0 contains the f'-images of $v_1^u, \dots, v_d^u$, the function h is d-matchable.

The following lemma justifies the definition of an *i-frontier*. Informally speaking, the lemma says that an i-frontier splits an extended depth-decomposition into a left and a right side, and that two depth-decompositions with the same i-frontier can be glued together on it, taking one side from each decomposition; also see Figure 6 for an illustration. In this way, the i-frontier contains all information that needs to be stored when iteratively constructing a depth-decomposition of M in a dynamic way for the elements of contained in $A_0, A_1, \dots, A_{2r}$.

LEMMA 28. *Let (T, f, g) be a principal extended depth-decomposition of a vector matroid M with rank r, let $(u_i, A_i, B_i)_{i\in\{0,\dots,2r\}}$ be a transversal sequence for (T, f, g), and let (T', f', g') and (T'', f'', g'') be two solid extended depth-decompositions of M. Suppose that $i \in \{0, \dots, 2r\}$ is such that the i-frontiers of (T', f', g') and (T'', f'', g'') with respect to (T, f, g) and $(u_i, A_i, B_i)_{i\in\{0,\dots,2r\}}$ are the same. Let T_0 be the rooted tree of the i-frontier, which we identify with the corresponding subtrees of T' and T''. Further, let K_A be the linear hull of the g'-image of the vertices of T_0, let K_B be the linear hull of the g''-image of the vertices of T_0, and set $C_A = K_A \cap A_i$ and $C_B = K_B \cap B_i$.*

Obtain T'_A from T' by removing all branches S with $\widehat{S} \subseteq B_i \cup K_A$ that are not i-crossed, T''_B from T'' by removing all branches S with $\widehat{S} \subseteq A_i \cup K_B$ that are not i-crossed, and T''' by gluing T'_A and T''_B together on the vertices of T_0 (note that both T'_A and T''_B contain T_0). Finally, let f''' be a function from the elements of M to the leaves of T''' defined as follows. If $x \in A_i \setminus C_A$, then $f'''(x) = f'(x)$. If $x \in B_i \setminus C_B$, then $f'''(x) = f''(x)$. If $x \in C_A$, then let $f'''(x)$ be any leaf u of T_0 such that x

is contained in the linear hull of the g'-image of the vertices on the path from u to the root. Finally, if $x \in C_B \setminus C_A$, then let $f'''(x)$ be any leaf u of T_0 such that x is contained in the linear hull of the g''-image of the vertices on the path from u to the root. Then, (T''', f''') is a depth-decomposition of M.

Proof. We first verify that f''' is well-defined. Consider an element x of M and suppose that $x \in A_i \setminus C_A$. Since the space C_A is the intersection of A_i and K_A, the element x does not belong to K_A and so $f'(x)$ is not contained in T_0, i.e., $f'(x)$ is contained in a branch of T' that is not i-crossed. Consider a maximal such branch S. Lemma 27 yields that $\widehat{S} \subseteq A_i \cup K_A$ or $\widehat{S} \subseteq B_i \cup K_A$, and since $x \in A_i \setminus K_A$ and $A_i \cap B_i = K_{u_i} \subseteq K_A$, it follows that $\widehat{S} \not\subseteq B_i \cup K_A$. Hence, the branch S is contained in T'_A and $f'''(x)$ is well-defined. The case that $x \in B_i \setminus C_B$ is symmetric.

The next case that we analyze is that $x \in C_A$. Since $C_A \subseteq K_A$ and the g'-image of T_0 is a basis of K_A, the element x can be uniquely expressed as a linear combination of the elements of the g'-image of T_0. Moreover, as (T', f', g') is an extended depth-decomposition of M, the g'-preimage of the basis elements with nonzero coefficients in this linear combination is contained in the path from $f'(x)$ to the root of T'. Hence, this preimage is contained in a path from the root of T_0 to one of its leaves, say, u. This implies that $x \in K_u$ in T', so $f'''(x)$ can be set to be u. The final case is that $x \in C_B \setminus C_A$. Since $x \in C_B$, the argument above yields that there exists a leaf u of T_0 such that $x \in K_u$ in T'', and so $f'''(x)$ can be set to be u.

For completeness, we check that the sets $A_i \setminus C_A$, $B_i \setminus C_B$, C_A, and $C_B \setminus C_A$ are pairwise disjoint so that $f'''(x)$ is never multiply defined. Observe that $A_i \setminus C_A = A_i \setminus K_A$ is disjoint from B_i as $A_i \cap B_i = K_{u_i} \subseteq K_A$. Hence $A_i \setminus C_A$ is disjoint from both $B_i \setminus C_B$ and $C_B \setminus C_A \subseteq B_i$. Similarly, $B_i \setminus C_B = B_i \setminus K_B$ is disjoint from both $A_i \setminus C_A$ and C_A. The remaining pairs are disjoint by definition, so indeed f''' is a well-defined function.

We next verify that the number of edges of T''' is the rank of M, which is

$$\dim A_i + \dim B_i - \dim A_i \cap B_i = \dim A_i + \dim B_i - \dim K_{u_i} = \dim A_i + \dim B_i - d.$$

Let (T_0, d, a, b, h) be the common i-frontier of (T', f', g') and (T'', f'', g''). Observe that $\dim A_i \cap K_A = d + a$ and $\dim B_i \cap K_A = d + b$. By Lemma 27, it holds that $\widehat{S} \subseteq A_i \cup K_A$ or $\widehat{S} \subseteq B_i \cup K_A$ for every branch S of T' that is not i-crossed. Hence, A_i is contained in the linear hull of the g'-image of T'_A and the dimension of this linear hull is $\dim A_i + b$. Since $\mathrm{Im}(g')$ is a basis, the number of edges of T'_A is $\dim A_i + b$. A symmetric argument yields that the number of edges of T''_B is $\dim B_i + a$. The tree T_0 has $d + a + b$ edges since (T_0, d, a, b, h) is a frontier, so T''' has

$$(\dim A_i + b) + (\dim B_i + a) - (d + a + b) = \dim A_i + \dim B_i - d$$

edges, as desired.

To finish the proof of the lemma, we need to show that (T''', f''') is a depth-decomposition of M. To do so, we define a function g''' such that (T''', f''', g''') is an extended depth-decomposition of M. Let u be a nonroot vertex of T'''. If u is a nonroot vertex of a branch S of T' with $\widehat{S} \subseteq A_i \cup K_A$ that is not i-crossed, we set $g'''(u)$ to be any nonzero vector of A_i such that $g'(u) - g'''(u) \in K_A$, i.e., the vectors $g'(u)$ and $g'''(u)$ are the same in the space quotiened by K_A. Similarly, if u is a nonroot vertex of a branch S of T'' with $\widehat{S} \subseteq B_i \cup K_B$ that is not i-crossed, we set $g'''(u)$ to be any nonzero vector of B_i such that $g''(u) - g'''(u) \in K_B$. It remains to define the mapping g''' for nonroot vertices of the tree T_0. Let $v_1^u, \ldots, v_d^u$ be the

vectors assigned to the vertices on the path from u_i to the root in T, let $v_1^A, \ldots, v_a^A$ be the basis of the space L_A as in the definition of the i-frontier of T', and let $v_1^B, \ldots, v_b^B$ be the basis of the space L_B as in the definition of the i-frontier of T''. If u is a nonroot vertex of T_0, we set $g'''(u)$ to be the linear combination of the vectors $v_1^u, \ldots, v_d^u$, $v_1^A, \ldots, v_a^A$, $v_1^B, \ldots, v_b^B$ with coefficients $h(u)$. Finally, let K'_u, K''_u, and K'''_u be the spaces K_u defined for trees T', T'', and T'''; we need to make this distinction here since the spaces K'_u, K''_u, and K'''_u may differ. Observe that for every nonroot vertex u of T_0, the intersection of K'_u and A_i is the same as the intersection of K'''_u and A_i. The choice of g''' for the vertices of T'_A not contained in T_0 implies that $K'_u \cap A_i \subseteq K'''_u \cap A_i$ for every vertex u of T'_A (indeed, even equality can be shown). Similarly, it holds that $K''_u \cap B_i \subseteq K'''_u \cap B_i$ for every vertex u of T''_B.

It remains to show that every element x of M is a linear combination of the g'''-image of the vertices on the path from $f'''(x)$ to the root in T'''. Fix an element x of M. If $x \in A_i \setminus C_A$, let u be the vertex $f'''(x) = f'(x)$ and observe that x belongs to T'_A (as we have already established that $f'''(x)$ is well-defined). Since $K'_u \cap A_i \subseteq K'''_u \cap A_i$, it follows that x is contained in the linear hull of the g'''-image of the vertices on the path from $f'''(x)$ to the root. If $x \in C_A$, the vertex $f'''(x)$ is a leaf of T_0 such that $x \in K'_u$ (as we have already established that $f'''(x)$ is well-defined). Since $C_A \subseteq A_i$, it follows that $x \in K'_u \cap A_i \subseteq K'''_u \cap A_i$ and x is contained in the linear hull of the g'''-image of the vertices on the path from $f'''(x)$ to the root in T'''. The cases $x \in B_i$ and $x \in C_B \setminus C_A$ follow the same line of reasoning. ∎

To prove the main result of this section, we will need the following auxiliary lemma.

LEMMA 29. *Let (T, f, g) be a principal extended depth-decomposition of a vector matroid M with rank r, $(u_i, A_i, B_i)_{i \in \{0,\ldots,2r\}}$ a transversal sequence for (T, f, g), and (T', f', g') a solid extended depth-decomposition of M. The following holds for every $i \in \{0, \ldots, 2r\}$. Let T_0 be the rooted tree of the i-frontier, K the linear hull of the g'-image of the vertices of T_0, T_A the rooted tree obtained from T' by removing all branches S with $\widehat{S} \subseteq B_i \cup K$ that are not i-crossed, g_A the restriction of g' to T_A, and M_A the vector matroid obtained from the restriction of M to the elements of A_i by adding the g'-image of the vertices of T_0 (note that parallel elements may be added to M_A by this operation). There exists a function f_A from the elements of M_A to the leaves of T_A such that the triple (T_A, f_A, g_A) is an extended depth-decomposition of M_A.*

Proof. We define f_A as follows. If x is contained in the restriction of M to the elements of A_i and $f'(x)$ is a leaf of T_A, we set $f_A(x)$ to $f'(x)$. If x is contained in the restriction of M to the elements of A_i but $f'(x)$ is not a leaf of T_A, we set $f_A(x)$ to any leaf u such that $x \in K_u$. Finally, if x is not contained in A_i, then $x = g'(v)$ for a vertex v of T_0 and we set $f_A(x)$ to any leaf descended from v.

We need to argue that $f_A(x)$ is well-defined for every element x of M_A. This is clear unless x is contained in the restriction of M to the elements of A_i and $f'(x)$ is not a leaf of T_A. In such a case, Lemma 27 implies that $f'(x)$ is a leaf of a branch S with $\widehat{S} \subseteq B_i \cup K$. It follows that the element x is contained in

$$A_i \cap (B_i \cup K) = (A_i \cap B_i) \cup (A_i \cap K) = K_{u_i} \cup (A_i \cap K) \subseteq K_{u_i} \cup K = K.$$

Since the g'-image of T_0 is a basis of K, the element x can be uniquely expressed as a linear combination of the elements of the g'-image of T_0. Moreover, as (T', f', g') is an extended depth-decomposition of M, the g'-preimage of the basis elements with

nonzero coefficients in this linear combination is contained in the path from $f'(x)$ to the root of T'. The subpath of this path containing the preimage is also contained in T_0, and so T_0 contains a path from one of its leaves, say, u, to its root such that the preimage is also contained in this path. It follows that $x \in K_u$ and $f_A(x)$ can be set to u.

To complete the proof, we need to show that (T_A, f_A, g_A) is an extended depth-decomposition of the matroid M_A. Observe that for every leaf u of T_A, the g-image and g_A-image of the vertices on the path from u to the root are the same. Hence, the space K_u is the same with respect to g and g_A and it follows that $x \in K_{f_A(x)}$ for every element x of M_A. Finally, since M_A contains all elements of A_i and a basis of K and the g_A-image of the vertices of T_A is a basis of the linear hull of $A_i \cup K$, the number of edges of T_A is the rank of M_A. It follows that (T_A, f_A, g_A) is an extended depth-decomposition of the matroid M_A. ∎

Before stating the main result of this section, we need to establish that the number of frontiers for any fixed d is bounded.

LEMMA 30. *For all integers d and D and any finite field $\mathbb{F}$, there exist at most $d^{2D+1}D|\mathbb{F}|^{(dD)^2}$ choices of a rooted tree T of depth at most d, integers a and b, and a mapping h from the nonroot vertices of T to $\mathbb{F}^{D+a+b}$ such that (T, D, a, b, h) is a frontier.*

Proof. We first show that there are at most d^{2k-1} rooted trees T with k leaves and depth at most d. Such a tree can be encoded as follows: enumerate the k leaves of T in the depth-first-search order. Let d_1 be the depth of the first leaf, and for $i = 2, \ldots, k$, let d_i be the depth of the ith leaf and m_i be the number of edges shared by the paths from the root to the $(i-1)$th and ith leaves. Note that the numbers $d_1, \ldots, d_k$ and $m_2, \ldots, m_k$ determine the tree T. Since each d_i is a positive integer that is at most d and each m_i is a nonnegative integer that is at most $d-1$, it follows that there are at most d^{2k-1} rooted trees T with k leaves and depth at most d. Summing over all the choices $k = 1, \ldots, D$, we obtain that there are at most d^{2D} rooted trees with at most D leaves and depth at most d.

Fix a tree T with at most D leaves and depth at most d, and let m be the number of edges of T. Note that $m \leq dD$, and $D + a + b = m$ if (T, D, a, b, h) is to be a frontier. Hence, the integers a and b can be chosen in at most $m \leq dD$ ways, and the mapping h can be chosen in at most $|\mathbb{F}|^{m^2} \leq |\mathbb{F}|^{(dD)^2}$ ways (although some of these choices would not yield a frontier). It follows that the number of choices of a, b, and h, when T is fixed (which determines d), is at most $dD|\mathbb{F}|^{(dD)^2}$. We conclude that for a fixed D, the number of frontiers (T, D, a, b, h) such that the depth T is at most d does not exceed $d^{2D+1}D|\mathbb{F}|^{(dD)^2}$. ∎

We are now ready to prove the main theorem of this section.

THEOREM 31. *For the parameterization by a positive integer d and a prime power q, there exists a fixed parameter algorithm that for a vector matroid M over the q-element field either outputs that* $\mathrm{bd}(M)$ *is larger than d or outputs an extended depth-decomposition of M with depth at most d.*

Proof. We first apply the algorithm from Theorem 8. The algorithm either outputs that the branch-depth of M is larger than d or outputs a principal extended depth-decomposition of M with depth at most 4^d. If the former applies, we stop and report that the branch-depth of M is larger than d. Otherwise, let (T, f, g) be the principal extended depth-decomposition returned by the algorithm.

Let r be the rank of the matroid M, $(u_i, A_i, B_i)_{i\in\{0,\dots,2r\}}$ a transversal sequence for (T, f, g), and d_i the depth of u_i in T. For $i = 0, \dots, 2r$, the algorithm iteratively computes the list of all frontiers (T_0, d_i, a, b, h) with depth at most d for which the following hold:

- there exists a vector matroid M' with rank $\dim A_i + b$ such that the linear hull of its elements is contained in $\mathcal{L}(A_i \cup B_i)$,
- M' contains the restriction of M to the elements of A_i,
- the matroid M' has an extended depth-decomposition (T_A, f_A, g_A) with depth at most d, and
- (T_0, d_i, a, b, h) is yielded by the procedure for creating i-frontiers performed on (T_A, f_A, g_A) with respect to (T, f, g) and (u_i, A_i, B'_i), where B'_i is the intersection of the linear hull of the elements of M' and B_i. (Note that we cannot formally take i-frontiers of (T_A, f_A, g_A) since (T, f, g) is not a depth-decomposition of M'.)

If the branch-depth of M is at most d, then M has a solid extended depth-decomposition with depth $\mathrm{bd}(M)$ by Theorem 26 and the list is nonempty for every $i = 0, \dots, 2r$ by Lemma 29. By Lemma 30, the size of the list computed in the ith iteration does not exceed $d^{2d_i+1} d_i |\mathbb{F}|^{(dd_i)^2}$ and is therefore bounded by a function of d and $|\mathbb{F}|$ only (since $d_i \le 4^d$ for every i). We emphasize that (T_0, d_i, a, b, h) is not required to be an i-frontier of M' with respect to a principal depth-decomposition of M'.

We now describe the iterations of the algorithm in detail. For $i = 0$, the list of frontiers contains a single element $(R, 0, 0, 0, h)$, where R is the rooted tree that contains the root only and h is the null function. Hence, assume that $i > 0$ and we have already computed the list for $i - 1$. The iteration of the algorithm differs according to whether u_i is the parent or a child of u_{i-1}.

We start with the case where u_i is the parent of u_{i-1}. Then the depth of u_{i-1} is $d_i + 1$, and the following is performed for every frontier $(T_0, d_i + 1, a, b, h)$ in the list from the previous iteration:

- If h is d_i-matchable, we add $(T_0, d_i, a+1, b, h')$ to the list for the ith iteration, where h' is a mapping from the nonroot vertices of T_0 obtained from h by changing the $(d_i + 1)$th coordinate into an A-coordinate and applying an invertible linear transformation to the $a + 1$ A-coordinates, i.e., we fix such a linear transformation L and set $h'(v) = L(h(v))$ for all vertices v of T_0.
- For every leaf v of T_0 such that the linear hull of the h-image of the vertices from v to the root contains the $(d_i + 1)$th unit vector, we proceed as follows. Let T'_0 be the tree obtained by removing the path from v to the first ancestor with at least two children, or to the root if there is no such ancestor. Let c be the number of edges on this path, and let h' be the restriction of h to the nonroot vertices of T'_0. If h' is d_i-matchable and the linear hull of $\mathrm{Im}(h')$ restricted to the last b coordinates has dimension b, we add $(T'_0, d_i, a + 1 - c, b, h'')$ to the list for the ith iteration, where h'' is a mapping from the nonroot vertices of T'_0 to $\mathbb{F}^{d_i+a+1-c+b}$ obtained from h' by changing the $(d_i + 1)$th coordinate into an A-coordinate and applying any full rank linear transformation $L : \mathbb{F}^{a+1} \to \mathbb{F}^{a+1-c}$ to its $a + 1$ A-coordinates.

We next describe the case that u_i is a child of u_{i-1}; note that $d_i = d_{i-1} + 1$ in this case. Let X_i be the set of d_i-dimensional vectors that formed by d_i coordinates of the elements of M contained in K_{u_i}, expressed with respect to the basis formed by the g-image of the vertices on the path from the root of T to u_i (in this order). The following is performed for every frontier $(T_0, d_i - 1, a, b, h)$ in the list from the previous iteration:

- For every invertible linear transformation L of the B-coordinates of h and every leaf v of T_0 such that the linear hull of the $L(h)$-image of the vertices on the path from v to the root contains the $(d_i - 1 + a + b)$th unit vector, we proceed as follows. Let h' be the mapping obtained from $L(h)$ by changing the last coordinate into a new d_ith coordinate. If for every $x \in X_i$ there exists a vertex v' of T_0 such that x is contained in the linear hull of the h'-image of the vertices on the path from v' to the root of T_0 restricted to the first d_i coordinates, we add $(T_0, d_i, a, b-1, h')$ to the list for the ith iteration.
- For every rooted tree T_0' obtained from T_0 by adding a new leaf v joined by a path with $c \geq 1$ edges to T_0 in a way that the depth of T_0' is at most d, we proceed as follows. Let h' be the mapping obtained from h by adding new c zero B-coordinates and extending it to each vertex on the newly added path by mapping the vertex to one of the unit vector for the new coordinates in a way that the dimension of of the h'-image is $d_i + a + b + c - 1$. We next consider all invertible linear transformations L of the $b + c$ B-coordinates such that the $L(h')$-image of the vertices on the path from v to the root contains the $(d_i + a + b + c - 1)$th unit vector, and, for each such L, we obtain the mapping h'' from $L(h')$ by turning the last coordinate into a new d_ith coordinate. If for every $x \in X_i$ there exists a vertex v' of T_0' such that x is contained in the linear hull of the h''-image of the vertices on the path from v' to the root of T_0' restricted to the first d_i coordinates, we add $(T_0', d_i, a, b + c - 1, h'')$ to the list for the ith iteration.

The inspection of the steps of the algorithm yields that if (T_0, d_i, a, b, h) is added to the list in the ith iteration, then h is d_i-matchable and the dimension of the h-image is $d_i + a + b$. Hence, the computed list of frontiers in the ith iteration contains exactly the frontiers described at the beginning of the proof.

As we have already argued, if the branch-depth of M is at most d, the list of frontiers is nonempty in all iterations. In particular, if the list becomes empty, we stop and report that the branch-depth of M exceeds d. Hence, we assume that the list is nonempty in all iterations. If the final list is nonempty, it contains a single element $(T_{2r}, d_{2r}, a_{2r}, b_{2r}, h_{2r}) = (R, 0, 0, 0, h)$ where R is the rooted tree that contains the root only and h is the null mapping. We now define $(T_i, d_i, a_i, b_i, h_i)$ for all $i = 0, \ldots, 2r$ by tracing back the lists of frontiers for $i = 2r, 2r-1, \ldots, 1$: if $(T_i, d_i, a_i, b_i, h_i)$ was added to the list in the iteration i, then let $(T_{i-1}, d_{i-1}, a_{i-1}, b_{i-1}, h_{i-1})$ be a frontier that triggered $(T_i, d_i, a_i, b_i, h_i)$ to be added to the list.

We next use this sequence of frontiers to construct an extended depth-decomposition of depth at most d. We first define inductively for $i = 0, \ldots, 2r$ linear maps h_i^M from $\mathbb{F}^{d_i + a_i}$ to the linear hull of the elements of M. The mapping h_0^M is the null mapping (note that $d_0 + a_0 = 0$). If u_i is a child of u_{i-1}, we obtain h_i^M from h_{i-1}^M by inserting the d_ith coordinate, mapping the d_ith unit vector to $g(u_i)$, and extending linearly to $\mathbb{F}^{d_i + a_i}$. If u_i is the parent of u_{i-1}, there exists a linear mapping L from $\mathbb{F}^{d_{i-1} + a_{i-1}}$ to $\mathbb{F}^{d_i + a_i}$ such that the restriction of h_i to the first $d_i + a_i$ coordinates is the L-transformation of the restriction of h_{i-1} to the first $d_{i-1} + a_{i-1}$ coordinates; we set h_i^M to be the L-transformation of h_{i-1}^M, i.e., $h_i^M = L(h_{i-1}^M)$.

As the next step, we define inductively for $i = 0, \ldots, 2r$ rooted trees T_i^M such that T_i^M contains T_i as a subtree, and mappings g_i^M from the vertices of T_i^M that are not contained in T_i to the linear hull of the elements of M. The tree T_0^M is the rooted tree that contains the root only and g_0^M is the null mapping. If $T_i = T_{i-1}$, we set $T_i^M = T_{i-1}^M$ and $g_i^M = g_{i-1}^M$. Otherwise, we proceed as follows. If u_i is a child of u_{i-1}, then T_i has been constructed from T_{i-1} by attaching a path to

one of its vertices, so we obtain T_i^M by attaching a path of the same length to the corresponding vertex of T_{i-1}^M and set $g_i^M = g_{i-1}^M$. If u_i is the parent of u_{i-1}, we set $T_i^M = T_{i-1}^M$ and define $g_i^M(v) = g_{i-1}^M(v)$ for vertices v of T_{i-1}^M not contained in T_{i-1} and $g_i^M(v) = h_{i-1}^M(h_{i-1}(v))$ for vertices v of T_{i-1} not contained in T_i. Observe that T_{i-1}^M is a subtree of T_i^M and g_{i-1}^M is a restriction of g_i^M for all $i = 1, \ldots, 2r$. Furthermore, the depth of all trees T_i^M, $i = 0, \ldots, 2r$, is at most d, and the number of edges of T_i^M is $\dim A_i + b_i$ for every $i = 0, \ldots, 2r$.

We now establish the existence of a mapping f^M such that $(T_{2r}^M, f^M, g_{2r}^M)$ is an extended depth-decomposition of M. Let x be an element of M and let i be the smallest index such that $x \in K_{u_i}$. It follows that u_{i-1} is the parent of u_i and $x \in K_{u_i} \setminus K_{u_{i-1}}$. Since $(T_i, d_i, a_i, b_i, h_i)$ was included in the list in the iteration i, there exists a vertex v of T_i such that the linear hull of the h_i-image of the vertices on the path from v to the root of T_i contains the vector defined by the d_i coordinates of x with respect to the basis formed by the g-image of the vertices on the path from the root to u_i in T (in this order). The definition of g_i^M and the construction of the list of frontiers imply that the the linear hull of the g_{2r}^M-image of the vertices of the path from v to the root of T_i contains x. Hence, $f^M(v)$ can be chosen to be any leaf descendant of v in T_{2r}^M.

We have shown that if the final list is nonempty, the matroid M has an extended depth-decomposition of depth at most d. Since the trees T_i^M and the mappings h_i^M and g_i^M can be constructed algorithmically, the extended depth-decomposition $(T_{2r}^M, f^M, g_{2r}^M)$ can also be constructed algorithmically. Finally, observe that the number of iterations in the algorithm is twice the rank of M, the size of the list computed in each iteration is bounded by a function of d and q only, and the number of steps needed for each element of these lists to be processed is bounded by a function of d and q times a polynomial in the size of M. Specifically, each list has at most $d^{2^{2d+1}+1} 4^d q^{(d4^d)^2} = q^{2^{O(d)}}$ elements and the number of steps needed to process each of their elements is bounded by a polynomial in the size of M times $q^{(d4^d)^2} = q^{2^{O(d)}}$. Hence the presented algorithm is a fixed parameter algorithm for parameterization by d and q. ∎

Theorems 20 and 31 yield the following corollary.

COROLLARY 32. *For the parameterization by a positive integer d and a prime power q, there exists a fixed parameter algorithm that for a vector matroid M over the q-element field either outputs that* $\mathrm{bd}(M)$ *is larger than d or computes* $\mathrm{bd}(M)$ *and outputs an extended depth-decomposition with this depth such that every branch is at capacity.*

8. Algorithm for rational matrices. In this section, we adopt the algorithm presented in section 7 to matroids over rationals. We start with an auxiliary lemma on linear combinations appearing in extended depth-decompositions. We remark that the bound of 2^{2d-1} in Lemma 33 can be replaced with $d \cdot 2^{d-1}$ using a slightly more careful analysis.

LEMMA 33. *Let M be a vector matroid and (T, f) a depth-decomposition of M with depth d such that every branch is at capacity. There exists a mapping g such that (T, f, g) is an extended depth-decomposition of M and every element of* $\mathrm{Im}(g)$ *is a linear combination of at most 2^{2d-1} elements of M.*

Proof. We show that it is possible to choose a mapping g in such a way that the g-image of a vertex at depth $i > 0$ is a linear combination of at most 2^{d+i-1} elements of M. Let $u_0, \ldots, u_k$ be the vertices on the path from the root of T to a leaf, in that order. Let $i_0 < \cdots < i_\ell$ be the sequence of indices such that $i_0 = 0$, $i_\ell = k$,

and $u_{i_1}, \ldots, u_{i_{\ell-1}}$ are exactly the vertices among $u_1, \ldots, u_{k-1}$ that have at least two children. By Lemma 17, each $K_{u_{i_j}}$, $j = 0, \ldots, \ell$, is the same for any function g such that (T, f, g) is an extended depth-decomposition of M. Let L_j be this space, which has dimension i_j and is determined by the choice of T and f only, and note that the elements $g(u_{i_{j-1}+1}), \ldots, g(u_{i_j})$ always form a basis of L_j/L_{j-1}. We will establish that it is possible to choose these elements in such a way that $g(u_i)$, $i = 1, \ldots, k$, is a linear combination of at most 2^{d+i-1} elements of M.

Suppose that we have already fixed $g(u_1), \ldots, g(u_{i-1})$, and let j be the smallest index such that $i \leq i_j$. Note that $\dim L_j > i - 1$. If $i_j = k$, then L_j/L_{j-1} is the linear hull of the elements in $f^{-1}(u_k)$ quotioned by L_{j-1} and so we choose $g(u_i)$ to be any element of $f^{-1}(u_k)$ that is linearly independent of $g(u_1), \ldots, g(u_{i-1})$. Hence, we assume that $i_j < k$ in the rest of the proof.

Let K be the linear hull of $g(u_1), \ldots, g(u_{i-1})$, and let C be the at most $(1 + \ldots + 2^{i-1})2^{d-1} = (2^i - 1)2^{d-1}$ elements of M appearing in the linear combinations used to express $g(u_1), \ldots, g(u_{i-1})$. Consider any two branches rooted at u_{i_j} and let A and B be the f-preimages of the leaves of the two branches. By Lemma 17, L_j is the intersection of the linear hulls of $A \cup L_{j-1}$ and $B \cup L_{j-1}$. Since the linear hulls of $A \cup L_{j-1}$ and $A \cup K$ are the same, and the linear hulls of $B \cup L_{j-1}$ and $B \cup K$ are the same, L_j is also the intersection of the linear hulls of $A \cup K$ and $B \cup K$. Since the dimension of L_j is larger than $\dim K = i - 1$, K is a proper subspace of L_j and so the matroid $(M/K)[A \cup B]$ contains a circuit X such that both $X \cap A$ and $X \cap B$ are nonempty. Hence, $L_j \setminus K$ contains a nonzero element w that is a linear combination of the elements of $(X \cap A) \cup K \subseteq (X \cap A) \cup \mathcal{L}(C)$ and also a linear combination of the elements of $(X \cap B) \cup K \subseteq (X \cap B) \cup \mathcal{L}(C)$. By Proposition 5, every circuit of M contains at most 2^d elements, and since every circuit of M/K is a subset of a circuit of M, we can by symmetry assume that $|X \cap A| \leq 2^{d-1}$. Hence, w is a linear combination of at most 2^{d+i-1} elements of M contained in $(X \cap A) \cup C$ and we can set $g(u_i)$ to be the vector w. ∎

The next lemma allows us to convert a representation of a matroid with small branch-depth over the rational numbers to a representation of an isomorphic matroid over a finite field.

LEMMA 34. *There exists an algorithm such that*

- *the input of the algorithm is an integer $d \geq 1$ and a matroid M represented over $\mathbb{Q}$ such that all entries of the vectors in the representation are integers between $-K$ and $+K$,*
- *the running time of the algorithm is polynomial in the number of elements of M and $\log K$, and*
- *the algorithm either outputs that the branch-depth of M is larger than d or computes a matroid M' represented over $\mathbb{F}_q$ for some $q \leq K^{2^{4^{d+1}}} 2^{2^{2^{d+2}}}$, along with an isomorphism between M and M'.*

Proof. We describe how the algorithm from the statement of the lemma proceeds. First, the algorithm from Theorem 8 is invoked for the input matroid M and an integer d. If the algorithm outputs that the branch-depth of M is larger than d, then we stop and report this. Otherwise, we obtain a principal depth-decomposition (T, f, g) of M with depth at most 4^d. For each element $x \in M$, we compute its representation x' with respect to the basis $\mathrm{Im}(g)$. Note that all entries of the vector x' are zero except for those that correspond to the g-image of the vertices on the path from $f(x)$ to the root. Hence, computing the entries of x' requires solving a system

of at most 4^d equations with integer coefficients between $-K$ and $+K$, which implies that the fractions appearing as the entries of x' have both numerator and denominator at most $(K4^d)^{4^d}$ by Cramer's rule. We define $\varphi(x)$ to be the integer vector obtained from x' by multiplying all its entries by the least common multiple of the denominators of the fractions that form the entries of x'. Since the vector x' has at most 4^d nonzero entries, all entries of $\varphi(x)$ are between $-K'$ and K' where $K' = (K4^d)^{4^{2d}}$.

Let q be any prime larger than $(2K'2^{4^d})^{2^{4^d}}$ and consider the vector matroid M' over $\mathbb{F}_q$ formed by the vectors $\mathrm{Im}(\varphi)$. Note that there exists such a prime q that is at most $2\cdot(2K'2^{4^d})^{2^{4^d}} \leq K^{2^{4^{d+1}}}2^{2^{2^{d+2}}}$. Observe that (T, f', g') is a depth-decomposition of M' where $f'(\varphi(x))$ is set to $f(x)$ and $g'(v)$ is the unit vector whose nonzero coordinate corresponds to $g(v)$. Indeed, the only nonzero coordinates of the vector $\varphi(x)$ are those corresponding to the g-images of the vertices on the path from $f(x)$ to the root. We conclude the branch-depth of M' is at most 4^d.

It is well-known that if a set of integer vectors is linearly dependent over $\mathbb{Q}$, then it is linearly dependent over any finite field; in particular, if X is a linearly dependent set of elements of M, then $\varphi(X)$ is linearly dependent over $\mathbb{F}_q$. On the other hand, the choice of q yields that if X is an independent set of at most 2^{4^d} elements of M, then $\varphi(X)$ is independent over $\mathbb{F}_q$. Since the branch-depths of both M and M' are at most 4^d, neither M nor M' has a circuit with more than 2^{4^d} elements by Proposition 5. Hence, the matroids M and M' have the same set of circuits. It follows that the matroids M and M' are isomorphic and φ is an isomorphism between them. ∎

Using Lemmas 33 and 34, we prove the main theorem of this section.

THEOREM 35. *For the parameterization by positive integers d and K, there exists a fixed parameter algorithm that, for a vector matroid M over $\mathbb{Q}$ such that the entries of all vectors in M are between $-K$ and $+K$, either outputs that* $\mathrm{bd}(M)$ *is larger than d, or computes* $\mathrm{bd}(M)$ *and outputs an extended depth-decomposition (T, f, g) of M with depth* $\mathrm{bd}(M)$*. Moreover, the entry complexity of the vectors in* $\mathrm{Im}(g)$ *is bounded by a function of d and K.*

Proof. Fix integers d and K. We first run the algorithm from Lemma 34 that either outputs that the branch-depth of M is larger than d or outputs a matroid M' with a representation over $\mathbb{F}_q$ for $q \leq K^{2^{4^{d+1}}}2^{2^{2^{d+2}}}$ that is isomorphic to M and an isomorphism φ between M and M'. Hence, we can use the algorithm from Corollary 32 to decide whether the branch-depth of M' is at most d and, if so, to construct a depth-decomposition (T, f') of depth $\mathrm{bd}(M')$ such that every branch of T is at capacity. If the branch-depth of M' exceeds d, then the branch-depth of M also exceeds d, so we stop and report this. Otherwise, (T, f) is a depth-decomposition of depth $\mathrm{bd}(M) = \mathrm{bd}(M')$ where f is obtained from f' using the isomorphism between M and M', i.e., $f(x) = f'(\varphi(x))$ for every element x of M. We now use the procedure described in the proof of Lemma 33 to compute a function g such that (T, f, g) is an extended depth-decomposition of M. Since computations given in the proof of Lemma 33 involve solving systems of equations with at most 2^{2d-1} variables, the entry complexity of the vectors in $\mathrm{Im}(g)$ is bounded by $O(d2^{2d}\log K)$. ∎

Theorem 35 implies Theorem 3 as follows.

Proof of Theorem 3. Let M be the matroid formed by columns of A. The algorithm from Theorem 35 either reports that $\mathrm{bd}(A) > d$ or outputs an extended depth-decomposition (T, f, g) of M with branch-depth $\mathrm{bd}(A)$ such that the entry complexity of $\mathrm{Im}(g)$ is bounded by a function of d and K. Let A' be the matrix from

Theorem 15. Since each element x of the matroid M is a linear combination of at most $\mathrm{bd}(A) \leq d$ elements from $\mathrm{Im}(g)$, which are those forming the g-image of the vertices on the path from $f(x)$ to the root of T, each entry of the matrix A' can be obtained by solving a system of at most d linear equations where the entry complexity of the coefficients and the right-hand side is bounded by $O(d2^{2d} \log K)$. Hence, the entry complexity of A' is bounded by $O(d^2 2^{2d} \log K)$. ∎

Acknowledgment. The authors would like to thank the anonymous reviewers for their comments, which significantly improved the presentation of the paper and its results.

REFERENCES

[1] M. Aschenbrenner and R. Hemmecke, *Finiteness theorems in stochastic integer programming*, Found. Comput. Math., 7 (2007), pp. 183–227.

[2] C. Aykanat, A. Pinar, and Ü. V. Çatalyürek, *Permuting sparse rectangular matrices into block-diagonal form*, SIAM J. Sci. Comput., 25 (2004), pp. 1860–1879, https://doi.org/10.1137/S1064827502401953.

[3] M. Bergner, A. Caprara, A. Ceselli, F. Furini, M. E. Lübbecke, E. Malaguti, and E. Traversi, *Automatic Dantzig–Wolfe reformulation of mixed integer programs*, Math. Program., 149 (2015), pp. 391–424.

[4] H. L. Bodlaender and T. Kloks, *Efficient and constructive algorithms for the pathwidth and treewidth of graphs*, J. Algorithms, 21 (1996), pp. 358–402.

[5] R. Borndörfer, C. E. Ferreira, and A. Martin, *Decomposing matrices into blocks*, SIAM J. Optim., 9 (1998), pp. 236–269.

[6] L. Chen and D. Marx, *Covering a tree with rooted subtrees–parameterized and approximation algorithms*, in Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 2018, pp. 2801–2820.

[7] B. Courcelle, *The monadic second-order logic of graphs.* I. *Recognizable sets of finite graphs*, Inform. Comput., 85 (1990), pp. 12–75.

[8] W. H. Cunningham and J. Geelen, *On integer programming and the branch-width of the constraint matrix*, in Proceedings of the 12th International IPCO Conference on Integer Programming and Combinatorial Optimization, 2007, pp. 158–166.

[9] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer, New York, 2015.

[10] M. DeVos, O. Kwon, and S. Oum, *Branch-Depth: Generalizing Tree-Depth of Graphs*, preprint, arXiv:1903.11988, 2019.

[11] P. Dvořák, E. Eiben, R. Ganian, D. Knop, and S. Ordyniak, *Solving integer linear programs with a small number of global variables and constraints*, in Proceedings of the 26th International Joint Conference on Artificial Intelligence, AAAI Press, 2017, pp. 607–613.

[12] J. Edmonds, *Systems of distinct representatives and linear algebra*, J. Res. National Bureau of Standards, 718 (1967), pp. 242–245.

[13] F. Eisenbrand, C. Hunkenschröder, K. Klein, M. Koutecký, A. Levin, and S. Onn, *An Algorithmic Theory of Integer Programming*, preprint, arXiv:1904.01361, 2019.

[14] F. Eisenbrand, C. Hunkenschröder, and K.-M. Klein, *Faster algorithms for integer programs with block structure*, in Proceedings of the 45th International Colloquium on Automata, Languages, and Programming, 2018, pp. 49:1–49:13.

[15] M. C. Ferris and J. D. Horn, *Partitioning mathematical programs for parallel solution*, Math. Program., 80 (1998), pp. 35–61, https://doi.org/10.1007/BF01582130.

[16] F. V. Fomin, F. Panolan, M. S. Ramanujan, and S. Saurabh, *On the optimality of pseudo-polynomial algorithms for integer programming*, in the Proceedings of the 26th Annual European Symposium on Algorithms, 2018, pp. 31:1–31:13.

[17] G. Gamrath and M. E. Lübbecke, *Experiments with a generic Dantzig–Wolfe decomposition for integer programs*, in Experimental Algorithms, P. Festa, ed., Springer, New York, 2010, pp. 239–252.

[18] R. Ganian and S. Ordyniak, *The complexity landscape of decompositional parameters for ILP*, in Proceedings of the 30th AAAI conference on Artificial Intelligence, 2016.

[19] R. Ganian, S. Ordyniak, and M. S. Ramanujan, *Going beyond primal treewidth for (M)ILP*, in Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017, pp. 815–821.

[20] T. Gavenčiak, D. Král', and S. Oum, *Deciding first order properties of matroids*, in Proceedings of the 39th International Colloquium Automata, Languages, and Programming, Lecture Notes in Comput. Sci. 7392, Springer, New York, 2012, pp. 239–250.

[21] J. Geelen, A. Gerards, N. Robertson, and G. Whittle, *On the excluded minors for the matroids of branch-width k*, J. Combin. Theory Ser. B, 88 (2003), pp. 261–265, https://doi.org/10.1016/S0095-8956(02)00046-1.

[22] P. Halmos, *Finite-Dimensional Vector Spaces*, Undergrad. Texts in Math., Springer, New York, 1993.

[23] R. Hemmecke, M. Köppe, and R. Weismantel, *Graver basis and proximity techniques for block-structured separable convex integer minimization problems*, Math. Program., 145 (2014), pp. 1–18.

[24] R. Hemmecke, S. Onn, and L. Romanchuk, *N-fold integer programming in cubic time*, Math. Program., 137 (2013), pp. 325–341.

[25] P. Hliněný, *Branch-width, parse trees, and monadic second-order logic for matroids*, in Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science, H. Alt and M. Habib, eds., Lecture Notes in Comput. Sci. 2607, Springer, New York, 2003, pp. 319–330.

[26] P. Hliněný, *On matroid properties definable in the MSO logic*, in Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science, B. Rovan and P. Vojtáš, eds., Lecture Notes in Comput. Sci. 2747, Springer, New York, 2003, pp. 470–479.

[27] P. Hliněný, *Branch-width, parse trees, and monadic second-order logic for matroids*, J. Combin. Theory Ser. B, 96 (2006), pp. 325–351.

[28] P. Hliněný and S. Oum, *Finding branch-decompositions and rank-decompositions*, in Proceedings of the 15th Annual European Symposium, L. Arge, M. Hoffmann, and E. Welzl, eds., Lecture Notes in Comput. Sci. 4698, Springer, New York, 2007, pp. 163–174.

[29] P. Hliněný and S. Oum, *Finding branch-decompositions and rank-decompositions*, SIAM J. Comput., 38 (2008), pp. 1012–1032.

[30] J. Jeong, E. J. Kim, and S. il Oum, *Finding branch-decompositions of matroids, hypergraphs, and more*, in Proceedings of the 45th International Colloquium on Automata, Languages, and Programming, I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella, eds., LIPIcs Leibniz Int. Proc. Inform. 107, Schloss Dagstuhl, 2018, pp. 80:1–80:14.

[31] R. Kannan, *Minkowski's convex body theorem and integer programming*, Math. Oper. Res., 12 (1987), p. 415–440.

[32] F. Kardoš, D. Král', A. Liebenau, and L. Mach, *First order convergence of matroids*, European J. Combin., 59 (2017), pp. 150–168.

[33] T. Khaniyev, S. Elhedhli, and F. S. Erenay, *Structure detection in mixed-integer programs*, INFORMS J. Comput., 30 (2018), pp. 570–587.

[34] D. Knop, M. Koutecký, and M. Mnich, *Voting and bribing in single-exponential time*, in Proceedings of the 34th Symposium on Theoretical Aspects of Computer Science, 2017, pp. 46:1–46:14.

[35] M. Koutecký, A. Levin, and S. Onn, *A parameterized strongly polynomial algorithm for block structured integer programs*, in Proceedings of the 45th International Colloquium on Automata, Languages, and Programming, 2018, pp. 85:1–85:14.

[36] H. W. Lenstra, Jr., *Integer programming with a fixed number of variables*, Math. Oper. Res., 8 (1983), pp. 538–548.

[37] S. Margulies, J. Ma, and I. V. Hicks, *The Cunningham-Geelen method in practice: Branch-decompositions and integer programming*, INFORMS J. Comput., 25 (2013), pp. 599–610.

[38] J. Oxley, *Matroid Theory*, Oxford Grad. Texts Math. 21, Oxford University Press, New York, 2011.

[39] F. Vanderbeck and L. A. Wolsey, *Reformulation and decomposition of integer programs*, in 50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art, Springer, New York, 2010, pp. 431–502, https://doi.org/10.1007/978-3-540-68279-0_13.

[40] J. Wang and T. Ralphs, *Computational experience with hypergraph-based methods for automatic decomposition in discrete optimization*, in Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Springer, New York, 2013, pp. 394–402.

[41] R. L. Weil and P. C. Kettler, *Rearranging matrices to block-angular form for decomposition (and other) algorithms*, Management Sci., 18 (1971), pp. 98–108, http://www.jstor.org/stable/2629299.

Mathematical Programming
https://doi.org/10.1007/s10107-023-02048-x

FULL LENGTH PAPER

Series A

Characterization of matrices with bounded Graver bases and depth parameters and applications to integer programming

Marcin Briański[1] · Martin Koutecký[2] · Daniel Král'[3] · Kristýna Pekárková[3] · Felix Schröder[4]

Received: 5 August 2022 / Accepted: 2 December 2023

Abstract

An intensive line of research on fixed parameter tractability of integer programming is focused on exploiting the relation between the sparsity of a constraint matrix A and the norm of the elements of its Graver basis. In particular, integer programming is fixed parameter tractable when parameterized by the primal tree-depth and the entry complexity of A, and when parameterized by the dual tree-depth and the entry complexity of A; both these parameterization imply that A is sparse, in particular, the number of its non-zero entries is linear in the number of columns or rows, respectively. We study preconditioners transforming a given matrix to a row-equivalent sparse matrix if it exists and provide structural results characterizing the existence of a sparse row-equivalent matrix in terms of the structural properties of the associated column matroid. In particular, our results imply that the ℓ_1-norm of the Graver basis is bounded by a function of the maximum ℓ_1-norm of a circuit of A. We use our results to design a parameterized algorithm that constructs a matrix row-equivalent to an input matrix A that has small primal/dual tree-depth and entry complexity if such a row-equivalent matrix exists. Our results yield parameterized algorithms for integer programming when parameterized by the ℓ_1-norm of the Graver basis of the constraint matrix, when parameterized by the ℓ_1-norm of the circuits of the constraint matrix, when

The first author was partially supported by the Polish National Science Center grant (BEETHOVEN; UMO-2018/31/G/ST1/03718). The second author was partially supported by Charles University project UNCE 24/SCI/008 and by the project 19-27871X of GA ČR. The third author was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 648509). This publication reflects only its authors' view; the ERC Executive Agency is not responsible for any use that may be made of the information it contains. The third and fourth authors were supported by the MUNI Award in Science and Humanities (MUNI/I/1677/2018) of the Grant Agency of Masaryk University.

✉ Kristýna Pekárková
kristyna.pekarkova@mail.muni.cz

Extended author information available on the last page of the article

Published online: 20 January 2024

parameterized by the smallest primal tree-depth and entry complexity of a matrix row-equivalent to the constraint matrix, and when parameterized by the smallest dual tree-depth and entry complexity of a matrix row-equivalent to the constraint matrix.

Keywords Integer programming · Width parameters · Matroids · Graver basis · Tree-depth · Fixed parameter tractability

Mathematics Subject Classification 90C10 · 05B35 · 90C27

1 Introduction

Integer programming is a problem of fundamental importance in combinatorial optimization with many theoretical and practical applications. From the computational complexity point of view, integer programming is very hard: it is one of the 21 problems shown to be NP-complete in the original paper on NP-completeness by Karp [37] and remains NP-complete even when the entries of the constraint matrix are zero and one only. On the positive side, Kannan and Lenstra [35, 45] showed that integer programming is polynomially solvable in fixed dimension, i.e., with a fixed number of variables. Another prominent tractable case is when the constraint matrix is totally unimodular, i.e., all determinants of its submatrices are equal to 0 or ± 1, in which case all vertices of the feasible region are integral and so linear programming algorithms can be applied.

Integer programming (IP) is known to be tractable for instances where the constraint matrix of an input instance enjoys a certain block structure. The two most important cases are the cases of 2-stage IPs due to Hemmecke and Schultz [27], further investigated in particular in [1, 13, 31, 39, 40, 44], and n-fold IPs introduced by De Loera et al. [15] and further investigated in particular in [11, 12, 19, 26, 34, 44]. IPs of this kind appear in various contexts, see e.g. [32, 41, 42, 48]. These (theoretical) tractability results complement well a vast number of empirical results demonstrating tractability of instances with a block structure, e.g. [2–4, 22, 23, 38, 49–51].

There tractability results on IPs with sparse constraint matrices can be unified and generalized using depth and width parameters of graphs derived from constraint matrices. Ganian and Ordyniak [24] initiated this line of study by showing that IPs with bounded primal tree-depth $\mathrm{td}_P(A)$ of a constraint matrix A and bounded coefficients of the constraint matrix A and the right hand side b can be solved efficiently. Levin, Onn and the second author [44] widely generalized this result by showing that IPs with bounded $\|A\|_\infty$ and bounded primal tree-depth $\mathrm{td}_P(A)$ or dual tree-depth $\mathrm{td}_D(A)$ of the constraint matrix A can be solved efficiently; such IPs include 2-stage IPs, n-fold IPs, and their generalizations.

Most of the existing algorithms for IPs assume that the input matrix is already given in its sparse form. This is a substantial drawback as existing algorithms cannot be applied to instances that are not sparse but can be transformed to an equivalent sparse instance. For example, the matrix in the left below, whose dual tree-depth is 5, can be transformed by elementary row operations to the matrix with dual tree-depth 2 given in the right; a formal definition of tree-depth is given in Sect. 2.1, however,

just the visual appearance of the two matrices indicates which is likely to be more amenable to algorithmic techniques.

$$\begin{pmatrix} 2&2&1&2&1&3&1\\ 2&1&1&1&2&1&1\\ 2&2&2&2&2&2&1\\ 2&1&1&2&2&1&1\\ 2&2&1&2&1&3&2 \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} 2&1&0&1&1&2&1\\ 0&1&1&0&0&1&0\\ 1&0&0&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&2&0&1 \end{pmatrix}$$

This transformation is an example of a preconditioner that transforms an instance of integer programming to an equivalent one that is more amenable to existing methods for solving integer programming and whose existence we investigate in this paper.

Preconditioning a problem to make it computationally simpler is a ubiquitous preprocessing step in mathematical programming solvers. An interesting link between matroid theory and preconditioners to sparsity of matrices was exhibited by Chan and Cooper together with the second, third and fourth authors [8, 9]. In particular, they proved the following structural characterization of matrices that are *row-equivalent*, i.e., can be transformed by elementary row operations, to a matrix with small dual tree-depth: a matrix is row-equivalent to one with small dual tree-depth if and only if the column matroid of the matrix has small contraction*-depth (see Theorem 1 below). In this paper, we further explore this uncharted territory by providing a structural characterization of matrices row-equivalent to matrices with small primal tree-depth, designing efficient algorithms for finding preconditioners with respect to both primal and dual tree-depth, and relating complexity of circuits and Graver basis of constraint matrices.

1.1 Our contribution

We now describe the results presented in this paper in detail. We opted not to interrupt the presentation of our results with various notions, some of which may be standard for some readers, and rather collect all definitions in a single section—Sect. 2. We remark that the primal tree-depth of a matrix A is a structural parameter that measures the complexity of interaction between the columns of A, and the dual tree-depth of a matrix A measures the complexity of interaction between the rows of A.

1.1.1 Characterization of depth parameters

Observe that the column matroid of the matrix is preserved by row operations, i.e., the column matroid of row-equivalent matrices is the same. The main structural result of [8, 9] is the following characterization of the existence of a row-equivalent matrix with small dual tree-depth in terms of the structural parameter of the column matroid [8, Theorem 1]. We remark that the term branch-depth was used in [8, 9] in line with the terminology from [36] but as there is a competing notion of branch-depth [16], we decided to use a different name for this depth parameter throughout the paper to avoid confusion.

Theorem 1 *For every non-zero matrix A, it holds that the smallest dual tree-depth of a matrix row-equivalent to A is equal to the contraction*-depth of $M(A)$, i.e.,* $\mathrm{td}^*_D(A) = \mathrm{c}^*\mathrm{d}(A)$.

We discover structural characterizations of the existence of a row-equivalent matrix with small primal tree-depth and the existence of a row-equivalent matrix with small incidence tree-depth.

Theorem 2 *For every matrix A, it holds that the smallest primal tree-depth of a matrix row-equivalent to A is equal to the deletion-depth of $M(A)$, i.e.,* $\mathrm{td}^*_P(A) = \mathrm{dd}(A)$.

Theorem 3 *For every matrix A, it holds that the smallest incidence tree-depth of a matrix row-equivalent to A is equal to contraction*-deletion-depth of $M(A)$ increased by one, i.e.,* $\mathrm{td}^*_I(A) = \mathrm{c}^*\mathrm{dd}(A) + 1$.

1.1.2 Interplay of circuit and graver basis complexity

Graver bases play an essential role in designing efficient algorithms for integer programming. We show that the maximum ℓ_1-norm of a circuit of a matrix A and the maximum ℓ_1-norm of an element of the Graver basis of A, which are denoted by $c_1(A)$ and $g_1(A)$, respectively, are functionally equivalent.

Theorem 4 *There exists a function $f_1 : \mathbb{N} \to \mathbb{N}$ such that the following holds for every rational matrix A with* $\dim\ker A > 0$*:*

$$c_1(A) \leq g_1(A) \leq f_1(c_1(A)).$$

The parameter $c_1(A)$ can be related to dual tree-depth and entry complexity as follows (we have opted throughout the paper to use entry complexity rather than $\|A\|_\infty$ as this permits to formulate our results for rational matrices rather than integral matrices, which is occasionally more convenient).

Theorem 5 *Every rational matrix A with* $\dim\ker A > 0$ *is row-equivalent to a rational matrix A' with* $\mathrm{td}_D(A') \leq c_1(A)^2$ *and* $\mathrm{ec}(A') \leq 2\lceil\log_2(c_1(A)+1)\rceil$.

Our results together with Theorem 9 imply that the following statements are equivalent for every rational matrix A:

- The ℓ_1-norm of every circuit of A, i.e., $c_1(A)$, is bounded.
- The ℓ_1-norm of every element of the Graver basis of A, i.e., $g_1(A)$, is bounded.
- The matrix A is row-equivalent to a matrix with bounded dual tree-depth and bounded entry complexity.
- The contraction*-depth of the matroid $M(A)$ is bounded, and the matrix A is row-equivalent to a matrix with bounded entry complexity (with any dual tree-depth).

1.1.3 Algorithms to compute matrices with small depth parameters

We also construct parameterized algorithms for transforming an input matrix to a row-equivalent matrix with small tree-depth and entry complexity if one exists. First, we design a parameterized algorithm for computing a row-equivalent matrix with small primal tree-depth and small entry complexity if one exists.

Theorem 6 *There exists a function $f : \mathbb{N}^2 \to \mathbb{N}$ and a fixed parameter algorithm for the parameterization by d and e that for a given rational matrix A:*

- *either outputs that A is not row-equivalent to a matrix with primal tree-depth at most d and entry complexity at most e, or*
- *outputs a matrix A' that is row-equivalent to A, its primal tree-depth is at most d and entry complexity is at most $f(d, e)$.*

The following algorithm for computing a row-equivalent matrix with small dual tree-depth was presented in [8, 9].

Theorem 7 *There exists a function $f : \mathbb{N}^2 \to \mathbb{N}$ and a fixed parameter algorithm for the parameterization by d and e that for a given rational matrix A with entry complexity at most e:*

- *either outputs that A is not row-equivalent to a matrix with dual tree-depth at most d, or*
- *outputs a matrix A' that is row-equivalent to A, its dual tree-depth is at most d and entry complexity is at most $f(d, e)$.*

We improve the algorithm by replacing the parameterization by the entry complexity of an input matrix with the parameterization by the entry complexity of the to be constructed matrix. Note that if a matrix A has entry complexity e and is row-equivalent to a matrix with dual tree-depth d, then Theorem 7 yields that A is row-equivalent to a matrix with dual tree-depth d and entry complexity bounded by a function of d and e. Hence, the algorithm given below applies to a wider set of input matrices than the algorithm from Theorem 7.

Theorem 8 *There exists a function $f : \mathbb{N}^2 \to \mathbb{N}$ and a fixed parameter algorithm for the parameterization by d and e that, for a given rational matrix A:*

- *either outputs that A is not row-equivalent to a matrix with dual tree-depth at most d and entry complexity at most e, or*
- *outputs a matrix A' that is row-equivalent to A, its dual tree-depth is at most d and entry complexity is at most $f(d, e)$.*

We point out the following difference between the cases of primal and dual tree-depth. As mentioned, if a matrix A has entry complexity e and is row-equivalent to a matrix with dual tree-depth d, then A is row-equivalent to a matrix with dual tree-depth d and entry complexity bounded by a function of d and e. However, the same is not true in the case of primal tree-depth. The entry complexity of every matrix with primal

tree-depth equal to one that is row-equivalent to the following matrix A is linear in the number of rows of A, quite in a contrast to the case of dual tree-depth.

$$\begin{pmatrix} 1 & 2 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & & \ddots & \ddots & & & \vdots & \vdots \\ \vdots & \vdots & & & \ddots & \ddots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 2 \end{pmatrix}$$

1.1.4 Fixed parameter algorithms for integer programming

One of the open problems in the area, e.g. discussed during the Dagstuhl workshop 19041 "New Horizons in Parameterized Complexity", has been whether integer programming is fixed parameter tractable when parameterized by $g_1(A)$, i.e., by the ℓ_1-norm of an element of the Graver basis of the constraint matrix A. Our results on the interplay of dual tree-depth, the circuit complexity and the Graver basis complexity of a matrix yield an affirmative answer. The existence of appropriate preconditioners that we establish in this paper implies that integer programming is fixed parameter tractable when parameterized by

- $g_1(A)$, i.e., the ℓ_1-norm of the Graver basis of the constraint matrix,
- $c_1(A)$, i.e., the ℓ_1-norm of the circuits of the constraint matrix,
- $\mathrm{td}^*_P(A)$ and $\mathrm{ec}(A)$, i.e., the smallest primal tree-depth and entry complexity of a matrix row-equivalent to the constraint matrix, and
- $\mathrm{td}^*_D(A)$ and $\mathrm{ec}(A)$, i.e., the smallest dual tree-depth and entry complexity of a matrix row-equivalent to the constraint matrix.

We believe that our new tractability results significantly enhance the toolbox of tractable IPs as the nature of our tractability conditions substantially differ from prevalent block-structured sparsity-based tractability conditions. The importance of availability of various forms of tractable IPs can be witnessed by n-fold IPs, which were shown fixed-parameter tractable in [26], and, about a decade later, their applications have become ubiquitous, see e.g. [6, 7, 10, 11, 28, 32, 33, 42, 43].

1.1.5 Hardness results

As our algorithmic results involve computing depth decompositions of matroids for various depth parameters in a parameterized way, we establish computational hardness of these parameters in Theorem 12, primarily for the sake of completeness of our exposition. In particular, computing the following matroid parameters is NP-complete:

- deletion-depth,
- contraction-depth,

- contraction-deletion-depth,
- contraction*-depth, and
- contraction*-deletion-depth.

2 Preliminaries

In this section, we fix the notation used throughout the paper. We start with general notation and we then fix the notation related to graphs, matrices and matroids.

The set of all positive integers is denoted by $\mathbb{N}$ and the set of the first k positive integers by $[k]$. If A is a linear space, we write $\dim A$ for its dimension. If K is a subspace of A, the *quotient space* A/K is the linear space of the dimension $\dim A - \dim K$ that consists of cosets of A given by K with the natural operations of addition and scalar multiplication; see e.g. [25] for further details. The quotient space A/K can be associated with a linear subspace of A of dimension $\dim A - \dim K$ formed by exactly a single vector from each coset of A given by K; we will often view the quotient space as such a subspace of A and write $w + K$ for the coset containing a vector w. For example, if A is $\mathbb{R}^3$ and K is the linear space generated by $(0, 0, 1)$, A/K can be associated with (or viewed as) the 2-dimensional space formed by vectors $(x, y, 0)$, $x, y \in \mathbb{R}$.

2.1 Graphs

All graphs considered in this paper are loopless simple graphs unless stated otherwise. If G is a graph, then we write $V(G)$ and $E(G)$ for the vertex set and the edge set of G, respectively. If W is a subset of vertices of a graph G, then $G \setminus W$ is the graph obtained by removing the vertices of W (and all edges incident with them), and $G[W]$ is the graph obtained by removing all vertices not contained in W (and all edges incident with them). If F is a subset of edges of a graph G, then $G \setminus F$ is the graph obtained by removing the edges contained in F and G/F is the graph obtained by contracting all edges contained in F and removing resulting loops and parallel edges (while keeping one edge from each group of parallel edges).

We next define the graph parameter *tree-depth*, which is the central graph parameter in this paper. The *height* of a rooted tree is the maximum number of vertices on a path from the root to a leaf, and the *height* of a rooted forest, i.e., a graph whose each component is a rooted tree, is the maximum height of its components. The *depth* of a rooted tree is the maximum number of edges on a path from the root to a leaf, and the *depth* of a rooted forest is the maximum depth of its components. Note that the height and the depth of a rooted tree always differ by one; we use both notions to avoid cumbersome way of expressing that would otherwise require adding or subtracting one. The *closure* $\mathrm{cl}(F)$ of a rooted forest F is the graph obtained by adding edges from each vertex to all its descendants. Finally, the *tree-depth* $\mathrm{td}(G)$ of a graph G is the minimum height of a rooted forest F such that the closure $\mathrm{cl}(F)$ of the rooted forest F contains G as a subgraph. See Fig. 1 for an example. It can be shown that the path-width, and so the tree-width, of any graph is at most its tree-depth decreased by one;

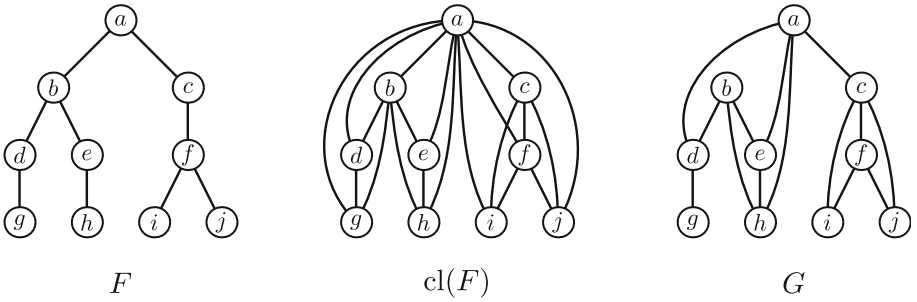


Fig. 1 A rooted forest F consisting of a single tree and its closure $\mathrm{cl}(F)$, which shows that tree-depth of the depicted graph G is (at most) four

see e.g. [14] for a more detailed discussion of the relation of tree-depth, path-width and tree-width, and their algorithmic applications.

2.2 Matroids

We next review basic definitions from matroid theory; we refer to the book of Oxley [46] for detailed exposition. A *hereditary* collection of subsets of a set is a collection closed under taking subsets; in particular, every non-empty hereditary collection of subsets contains the empty set. A *matroid* M is a pair $(X, \mathcal{I})$, where $\mathcal{I}$ is a non-empty hereditary collection of subsets of X that satisfies the *augmentation axiom*, i.e., if $X' \in \mathcal{I}$, $X'' \in \mathcal{I}$ and $|X'| < |X''|$, then there exists an element $x \in X'' \setminus X'$ such that $X' \cup \{x\} \in \mathcal{I}$. The set X is the *ground set* of M and the sets contained in $\mathcal{I}$ are referred to as *independent*. We often refer to elements of the ground set of M to as elements of the matroid M, and if e is an element of (the ground set of) M, we also write $e \in M$. Two important examples of matroids are vector matroids and graphic matroids. A *vector matroid* is a matroid whose ground set is formed by vectors and independent sets are precisely sets of linearly independent vectors (note that the augmentation axiom follows from the Steinitz exchange lemma). A *graphic matroid* is a matroid whose ground set is formed by edges of a graph and independent sets are precisely acyclic sets of edges, i.e., sets not containing a cycle.

The *rank* of a subset X' of the ground set X, which is denoted by $r_M(X')$ or simply by $r(X')$ if M is clear from the context, is the maximum size of an independent subset of X' (it can be shown that all maximal independent subsets of X' have the same cardinality); the *rank* of the matroid M, which is denoted by $r(M)$, is the rank of its ground set. Note that in the case of vector matroids, the rank of X' is exactly the dimension of linear space generated by X'. A *basis* of a matroid M is a maximal independent subset of the ground set of M and a *circuit* is a minimal subset of the ground set of M that is not independent. In particular, if X' is a circuit of M, then $r(X') = |X'| - 1$ and every proper subset of X' is independent. An element x of a matroid M is a *loop* if $r(\{x\}) = 0$, an element x is a *bridge* if it is contained in every basis of M, and two elements x and x' are *parallel* if $r(\{x\}) = r(\{x'\}) = r(\{x, x'\}) = 1$. Note that in the case of vector matroids, two non-loop elements are parallel if and only if they

are non-zero multiple of each other. If M is a matroid with ground set X, the *dual matroid*, which is denoted by M^*, is the matroid with the same ground set X such that $X' \subseteq X$ is independent in M^* if and only if $r_M(X \setminus X') = r(M)$; in particular, $r_{M^*}(X') = r_M(X \setminus X') + |X'| - r(M)$ for every $X' \subseteq X$.

For a field $\mathbb{F}$, we say that a matroid M is $\mathbb{F}$-*representable* if every element of M can be assigned a vector from $\mathbb{F}^{r(M)}$ in such a way that a subset of the ground set of M is independent if and only if the set of assigned vectors is linearly independent. In particular, an element of M is a loop if and only if it is assigned the zero vector and two non-loop elements of M are parallel if and only if they are non-zero multiples of each other. Such an assignment of vectors of $\mathbb{F}^{r(M)}$ to the elements of M is an $\mathbb{F}$-*representation* of M. Clearly, a matroid M is $\mathbb{F}$-representable if and only if it is isomorphic to the vector matroid given by its $\mathbb{F}$-representation. Matroids representable over the 2-element field are referred to as *binary* matroids. We say that a matroid M is $\mathbb{F}$-*represented* if the matroid M is given by its $\mathbb{F}$-representation. If a particular field $\mathbb{F}$ is not relevant in the context, we just say that a matroid M is *represented* to express that it is given by its representation.

Let M be a matroid with a ground set X. The matroid kM for $k \in \mathbb{N}$ is the matroid obtained from M by introducing $k-1$ parallel elements to each non-loop element and $k-1$ additional loops for each loop; informally speaking, every element of M is "cloned" to k copies. Note that a subset X' of the elements of kM is independent if and only if it does not contains two clones of the same element and the set of the elements of M corresponding to those contained in X' is independent. Observe that if M is a vector matroid, then kM is the vector matroid obtained by adding $k-1$ copies of each vector forming M. Similarly, if M is a graphic matroid associated with a graph G, then kM is the graphic matroid obtained from the graph G by duplicating each edge $k-1$ times.

If $X' \subseteq X$, then the *restriction* of M to X', which is denoted by $M\left[X'\right]$, is the matroid with the ground set X' such that a subset of X' is independent in $M\left[X'\right]$ if and only if it is independent in M. In particular, the rank of $M\left[X'\right]$ is $r_M(X')$. For example, if M is a graphic matroid associated with a graph G, then the restriction of M to X' is the graphic matroid associated with the spanning subgraph of G with edge set X'. The matroid obtained from M by *deleting* X' is the restriction of M to $X \setminus X'$ and is denoted by $M \setminus X'$.

The *contraction* of M by X', which is denoted by M/X', is the matroid with the ground set $X \setminus X'$ such that a subset X'' of $X \setminus X'$ is independent in M/X' if and only if $r_M(X'' \cup X') = |X''| + r_M(X')$. If X' is a single element set and e is its only element, we write $M \setminus e$ and M/e instead of $M \setminus \{e\}$ and $M/\{e\}$, respectively. If M is a graphic matroid associated with a graph G and e is an edge of G, then M/e is the graphic matroid associated with the graph obtained from G by contracting the edge e (while keeping all resulting loops and parallel edges). If an $\mathbb{F}$-representation of M is given and X' is a subset of the ground set of M, then an $\mathbb{F}$-representation of M/X' can be obtained from the $\mathbb{F}$-representation of M by considering the representation in the quotient space by the linear hull of the vectors representing the elements of X'. This leads us to the following definition: if M is an $\mathbb{F}$-represented matroid and A is a linear subspace of $\mathbb{F}^{r(M)}$, then the matroid M/A is the $\mathbb{F}$-represented matroid with the

representation of M in the quotient space by A. Note that the ground sets of M and M/A are the same, in particular, M and M/A have the same number of elements.

A matroid M is *connected* if every two distinct elements of M are contained in a common circuit. We remark that the property of being contained in a common circuit is transitive [46, Proposition 4.1.2], i.e., if the pair of elements e and e' is contained in a common circuit and the pair e' and e'' is also contained in a common circuit, then the pair e and e'' is also contained in a common circuit. If M is an $\mathbb{F}$-represented matroid with at least two elements, then M is connected if and only if M has no loops and there do not exist two non-trivial linear spaces A and B of $\mathbb{F}^{r(M)}$ such that $A \cap B$ contains the zero vector only and every element of M is contained in A or B (the linear space $\mathbb{F}^{r(M)}$ would be the direct sum of the linear spaces A and B). Also observe that if M is a graphic matroid associated with a graph G, then the matroid M is connected if and only if the graph G is 2-connected.

A *component* of a matroid M is an inclusion-wise maximal connected restriction of M; a component is *trivial* if it consists of a single loop, and it is *non-trivial* otherwise. If M is a vector matroid, then each non-trivial component of M can be associated with a linear space such that each element of M is contained in one of the linear spaces and the linear hull of all elements of M is the direct sum of the linear spaces. We often identify components of a matroid M with their element sets. Using this identification, it holds that a subset X' of a ground set of a matroid M is a component of M if and only if X' is a component of M^* (we use this equivalence to prove some of our hardness results in Sect. 6). We remark that $(M^*)^* = M$ for every matroid M, and if e is an element of a matroid M, then $(M/e)^* = M^* \setminus e$ and $(M\setminus e)^* = M^*/e$.

2.3 Matrices

In this section, we define notation related to matrices. If $\mathbb{F}$ is a field, we write $\mathbb{F}^{m\times n}$ for the set of matrices with m rows and n columns over the field $\mathbb{F}$. If A is a rational matrix, the entry complexity $\mathrm{ec}(A)$ is the maximum length of a binary encoding of its entries, i.e., the maximum of $\lceil \log_2(|p|+1)\rceil + \lceil(\log_2 |q|+1)\rceil$ taken over all entries p/q of A (where p and q are always assumed to be coprime). If A is an integral matrix, then $\mathrm{ec}(A) = \Theta(\log \|A\|_\infty)$. Throughout the paper, we use the entry complexity rather than the ℓ_∞-norm of matrices as this permits formulating our results for rational matrices rather than integral matrices only.

A rational matrix A is *z-integral* for $z \in \mathbb{Q}$ if every entry of A is an integral multiple of z. We say that two matrices A and A' are *row-equivalent* if one can be obtained from another by *elementary row operations*, i.e., by repeatedly adding a multiple of one row to another and multiplying a row by a non-zero element. Observe that if A and A' are row-equivalent matrices, then their kernels are the same. For a matrix A, we define $M(A)$ to be the represented matroid whose elements are the columns of A. Again, if matrices A and A' are row-equivalent, then the matroids $M(A)$ and $M(A')$ are the same.

If A is a matrix, the *primal graph* of A is the graph whose vertices are columns of A and two vertices are adjacent if there exists a row having non-zero elements in the two columns associated with the vertices; the *dual graph* of A is the graph

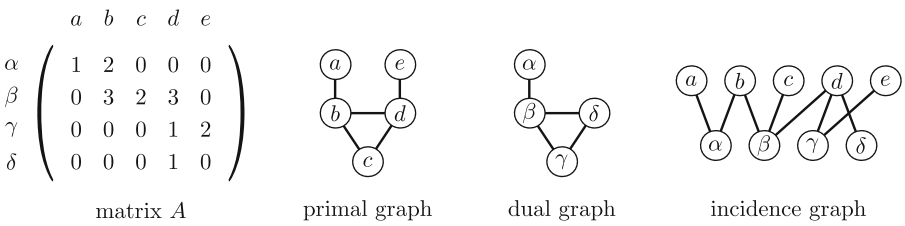


Fig. 2 The primal graph, the dual graph and the incidence graph of the depicted matrix A

whose vertices are rows of A and two vertices are adjacent if there exists a column having non-zero elements in the two associated rows; the *incidence graph* of A is the bipartite graph with one part formed by rows of A and the other part by columns of A and two vertices are adjacent if the entry in the associated row and in the associated column is non-zero. See Fig. 2 for an example. The *primal tree-depth* of A, denoted by $\mathrm{td}_P(A)$, is the tree-depth of the primal graph of A, the *dual tree-depth* of A, denoted by $\mathrm{td}_D(A)$, is the tree-depth of the dual graph of A, and the *incidence tree-depth* of A, denoted by $\mathrm{td}_I(A)$, is the tree-depth of the incidence graph of A. Finally, $\mathrm{td}^*_P(A)$ is the smallest primal tree-depth of a matrix row-equivalent to A, $\mathrm{td}^*_D(A)$ is the smallest dual tree-depth of a matrix row-equivalent to A, and $\mathrm{td}^*_I(A)$ is the smallest incidence tree-depth of a matrix row-equivalent to A.

A *circuit* of a rational matrix A is a support-wise minimal integral vector contained in the kernel of A such that all its entries are coprime; the set of circuits of A is denoted by $\mathcal{C}(A)$. Note that a set X of columns is a circuit in the matroid $M(A)$ if and only if $\mathcal{C}(A)$ contains a vector with the support exactly equal to X. We write $c_1(A)$ for the maximum ℓ_1-norm of a circuit of A and $c_\infty(A)$ for the maximum ℓ_∞-norm of a circuit of A. Note if A and A' are row-equivalent rational matrices, then $\mathcal{C}(A) = \mathcal{C}(A')$ and so the parameters $c_1(\cdot)$ and $c_\infty(\cdot)$ are invariant under elementary row operations. Following the notation from [21], we write $\dot{\kappa}_A$ for the least common multiple of the entries of the circuits of A. Observe that there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that $\dot{\kappa}_A \leq f(c_\infty(A))$ for every matrix A.

If $\mathbf{x}$ and $\mathbf{y}$ are two d-dimensional vectors, we write $\mathbf{x} \sqsubseteq \mathbf{y}$ if $|\mathbf{x}_i| \leq |\mathbf{y}_i|$ for all $i \in [d]$ and $\mathbf{x}$ and $\mathbf{y}$ are in the same orthant, i.e., $\mathbf{x}_i$ and $\mathbf{y}_i$ have the same sign (or one or both are zero) for all $i \in [d]$. The *Graver basis* of a matrix A, denoted by $\mathcal{G}(A)$, is the set of the $\sqsubseteq$-minimal non-zero elements of the integer kernel $\ker_{\mathbb{Z}}(A)$. We use $g_1(A)$ and $g_\infty(A)$ for the Graver basis of A analogously to the set of circuits, i.e., $g_1(A)$ is the maximum ℓ_1-norm of a vector in $\mathcal{G}(A)$ and $g_\infty(A)$ is the maximum ℓ_∞-norm of a vector in $\mathcal{G}(A)$. Again, the parameters $g_1(\cdot)$ and $g_\infty(\cdot)$ are invariant under elementary row operations as the Graver bases of row-equivalent matrices are the same. Note that every circuit of a matrix A belongs to the Graver basis of A, i.e., $\mathcal{C}(A) \subseteq \mathcal{G}(A)$, and so it holds that $c_1(A) \leq g_1(A)$ and $c_\infty(A) \leq g_\infty(A)$ for every matrix A.

The existence of efficient algorithms for integer programming with of constraint matrices A with bounded primal and dual tree-depth is closely linked to bounds on the norm of elements of the Graver basis of A. In particular, Koutecký, Levin and Onn [44] established the following.

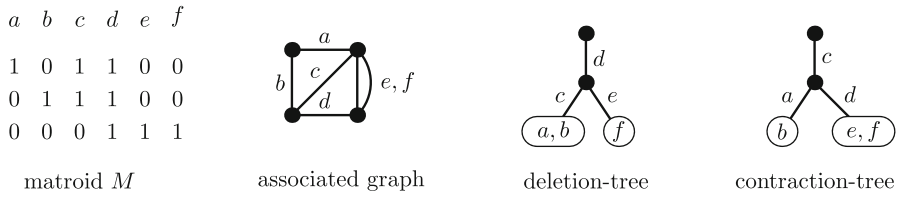


Fig. 3 A deletion-tree and a contraction-tree of the depicted binary matroid M, which is also the graphic matroid associated with the depicted graph

Theorem 9 *There exist functions* $f_P, f_D : \mathbb{N}^2 \to \mathbb{N}$ *such that the following holds for every rational matrix* A*:* $g_\infty(A) \leq f_P(\mathrm{td}_P(A), \mathrm{ec}(A))$ *and* $g_1(A) \leq f_D(\mathrm{td}_D(A), \mathrm{ec}(A))$.

2.4 Matroid depth parameters

We now define matroid depth parameters that will be of importance further. We start with the notion of deletion-depth and contraction-depth, which were introduced by DeVos, Kwon and Oum [16].

The *deletion-depth* of a matroid M, denoted by $\mathrm{dd}(M)$, is defined recursively as follows:

- If M has a single element, then $\mathrm{dd}(M) = 1$.
- If M is not connected, then $\mathrm{dd}(M)$ is the maximum deletion-depth of a component of M.
- Otherwise, $\mathrm{dd}(M) = 1 + \min_{e \in M} \mathrm{dd}(M \setminus e)$, i.e., $\mathrm{dd}(M)$ is 1 plus the minimum deletion-depth of $M \setminus e$ where the minimum is taken over all elements e of M.

A sequence of deletions of elements witnessing that the deletion-depth of a matroid M is $\mathrm{dd}(M)$ can be visualized by a rooted tree, which we call a *deletion-tree*, defined as follows. If M has a single element, then the deletion-tree of M consists of a single vertex labeled with the single element of M. If M is not connected, then the deletion-tree is obtained by identifying the roots of deletion-trees of the components of M. Otherwise, there exists an element e of the matroid M such that $\mathrm{dd}(M) = \mathrm{dd}(M \setminus e) + 1$ and the deletion-tree of M is obtained from the deletion-tree of $M \setminus e$ by adding a new vertex adjacent to the root of the deletion-tree of $M \setminus e$, changing the root of the tree to the newly added vertex and labeling the edge incident with it with the element e. See Fig. 3 for an example. Observe that the height of the deletion-tree is equal to the deletion-depth of M. In what follows, we consider deletion-trees that need not to be of optimal height, i.e., its edges can be labeled by a sequence of elements that decomposes a matroid M in a way described in the definition of the deletion-depth but its height is larger than $\mathrm{dd}(M)$. In this more general setting, the deletion-depth of a matroid M is the smallest height of a deletion-tree of M.

The *contraction-depth* of a matroid M, denoted by $\mathrm{cd}(M)$, is defined recursively as follows:

- If M has a single element, then $\mathrm{cd}(M) = 1$.

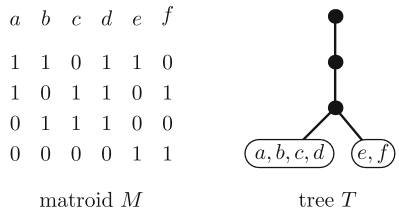


Fig. 4 A binary matroid M of rank four given by its representation and a tree T as in the definition of the contraction*-depth

- If M is not connected, then $\mathrm{cd}(M)$ is the maximum contraction-depth of a component of M.
- Otherwise, $\mathrm{cd}(M) = 1 + \min_{e \in M} \mathrm{cd}(M/e)$, i.e., $\mathrm{cd}(M)$ is 1 plus the minimum contraction-depth of M/e where the minimum is taken over all elements e of M.

It is not hard to show that $\mathrm{dd}(M) = \mathrm{cd}(M^*)$ and $\mathrm{cd}(M) = \mathrm{dd}(M^*)$ for every matroid M. We define a *contraction-tree* analogously to a deletion-tree; the contraction-depth of a matroid M is the smallest height of a contraction-tree of M (an example is given in Fig. 3).

We next introduce the contraction-deletion-depth; this parameter was studied under the name *type* in [17], however, we decided to adopt the name contraction-deletion-depth from [16], which we find to better fit the context considered here. The *contraction-deletion-depth* of a matroid M, denoted by $\mathrm{cdd}(M)$, is defined recursively as follows:

- If M has a single element, then $\mathrm{cdd}(M) = 1$.
- If M is not connected, then $\mathrm{cdd}(M)$ is the maximum contraction-deletion-depth of a component of M.
- Otherwise, $\mathrm{cdd}(M) = 1 + \min_{e \in M} \min\{\mathrm{cdd}(M \setminus e), \mathrm{cdd}(M/e)\}$, i.e., $\mathrm{cdd}(M)$ is 1 plus the smaller among the minimum contraction-deletion-depth of the matroid $M \setminus e$ and the minimum contraction-deletion-depth of the matroid M/e where both minima are taken over all elements e of M.

Observe that it holds that $\mathrm{cdd}(M) = \mathrm{cdd}(M^*)$, $\mathrm{cdd}(M) \leq \mathrm{dd}(M)$ and $\mathrm{cdd}(M) \leq \mathrm{cd}(M)$ for every matroid M.

One of the key parameters in our setting is that of contraction*-depth; this parameter was introduced under the name branch-depth in [36] and further studied in [8] but we decided to use a different name to avoid a possible confusion with the notion of branch-depth introduced in [16]. We first introduce the parameter for general matroids, and then present an equivalent definition for represented matroids, which is more convenient to work in our setting. The *contraction*-depth* of a matroid M, denoted by $\mathrm{c}^*\mathrm{d}(M)$, is the smallest depth of a rooted tree T with exactly $r(M)$ edges with the following property: there exists a function f from the ground set of M to the leaves of T such that for every subset X of the ground set of M the total number of edges contained in paths from the root to vertices of X is at least $r(X)$. An example is given in Fig. 4.

There is an alternative definition of the parameter for represented matroids, which also justifies the name that we use for the parameter. The *contraction***-depth* of a represented matroid M can be defined recursively as follows:

- If M has rank zero, then $\mathrm{c}^*\mathrm{d}(M) = 0$.
- If M is not connected, then $\mathrm{c}^*\mathrm{d}(M)$ is the maximum contraction*-depth of a component of M.
- Otherwise, $\mathrm{c}^*\mathrm{d}(M)$ is 1 plus the minimum contraction*-depth of a matroid obtained from the matroid M by factoring along an arbitrary one-dimensional subspace.

As the contraction in the definition is allowed to be by an arbitrary one-dimensional subspace, not only by a subspace generated by an element of M, it follows that $\mathrm{c}^*\mathrm{d}(M) \leq \mathrm{cd}(M)$.

The sequence of such contractions can be visualized by a *contraction***-tree* that is defined in the same way as a contraction-tree except that one-vertex trees are associated with matroids of rank zero (rather than matroids consisting of a single element) and the edges are labeled by one-dimensional subspaces. If each one-dimensional subspace that is the label of an edge of the tree is generated by an element of the matroid M, we say that the contraction*-tree is *principal* and we view the edges of the tree as labeled by the corresponding elements of M. Note that the minimum depth of a principal contraction*-tree of a matroid M is an upper bound on its contraction*-depth, however, in general, the contraction*-depth of a matroid M can be smaller than the minimum depth of a principal contraction*-tree of M. We point out that the notions of principal contraction*-trees and contraction-trees differ in a subtle but important way. For example, if M is a vector matroid of rank one containing a single element e, its only contraction-tree consists of a root labeled by e while its only contraction*-tree has a root and a leaf adjacent to it and the edge joining them is labeled with (the subspace generated by) e. However, if M is a vector matroid of rank one containing two parallel elements e and e', any contraction-tree and any contraction*-tree of M has depth one. Still, the minimum depth of a principal contraction*-tree of M is either $\mathrm{cd}(M) - 1$ or $\mathrm{cd}(M)$.

Kardoš et al. [36] established the connection between the contraction*-depth and the existence of a long circuit, which is described in Theorem 10 below; Theorem 10 implies that $\mathrm{cd}(M) \leq k^2 + 1$ where k is the size of the largest circuit of M.

Theorem 10 *Let M be a matroid and k the size of its largest circuit. It holds that $\log_2 k \leq \mathrm{c}^*\mathrm{d}(M) \leq k^2$. Moreover, there exists a polynomial-time algorithm that for an input oracle-given matroid M outputs a principal contraction*-tree of depth at most k^2.*

We next introduce the parameter of contraction*-deletion-depth, which we believe to have not been yet studied previously, but which is particularly relevant in our context. To avoid unnecessary technical issues, we introduce the parameter for represented matroids only. The *contraction***-deletion-depth* of a represented matroid M, denoted by $\mathrm{c}^*\mathrm{dd}(M)$, is defined recursively as follows:

- If M has rank zero, then $\mathrm{c}^*\mathrm{dd}(M) = 0$;
- if M has a single non-loop element, then $\mathrm{c}^*\mathrm{dd}(M) = 1$.

- If M is not connected, then $\mathrm{c}^*\mathrm{dd}(M)$ is the maximum contraction*-deletion-depth of a component of M.
- Otherwise, $\mathrm{c}^*\mathrm{dd}(M)$ is 1 plus the smaller among the minimum contraction*-deletion-depth of the matroid $M \setminus e$, where the minimum is taken over all elements of M, and the minimum contraction*-deletion-depth of a matroid obtained from M by factoring along an arbitrary one-dimensional subspace.

Observe that $\mathrm{c}^*\mathrm{dd}(M) \leq \mathrm{cdd}(M)$ and $\mathrm{c}^*\mathrm{dd}(M) \leq \mathrm{c}^*\mathrm{d}(M)$ for every matroid M.

Finally, if A is a matrix, the *deletion-depth*, *contraction-depth*, etc. of A is the corresponding parameter of the vector matroid $M(A)$ formed by the columns of A, and we write $\mathrm{dd}(A)$, $\mathrm{cd}(A)$, etc. for the deletion-depth, contraction-depth, etc. of the matrix A. Observe that the deletion-depth, contraction-depth etc. of a matrix A is invariant under elementary row operations as elementary row operations preserve the matroid $M(A)$.

3 Structural results

In this section, we prove our structural results concerning optimal primal tree-depth and optimal incidence tree-depth of a matrix. We start with presenting an algorithm, which uses a deletion-tree of the matroid associated with a given matrix to construct a row-equivalent matrix with small primal tree-depth.

Lemma 1 *There exists a polynomial-time algorithm that for an input matrix A and a deletion-tree of $M(A)$ with height d outputs a matrix A' row-equivalent to A such that $\mathrm{td}_P(A') \leq d$.*

Proof We establish the existence of the algorithm by proving that $\mathrm{td}_P(A') \leq d$ in a constructive (algorithmic) way. Fix a matrix A and a deletion-tree T of $M(A)$ with height d.

Let X be the set of non-zero columns that are labels of the vertices of T. We show that the columns contained in X form a basis of the column space of the matrix A. As the matroid obtained from $M(A)$ by deleting the labels of the edges of T has no component of size two or more, the columns contained in X are linearly independent. Suppose that there exists a non-zero column x that is not a linear combination of the columns contained in X, and choose among all such columns the label of an edge e as far from the root of T as possible. Since the element x does not form its own component in the matroid obtained from M by deleting the labels of all edges on the path from the root to e (excluding e), x is a linear combination of the labels of the vertices and edges of the subtree of T delimited by e. This implies that either x is a linear combination of the columns in X or there is a label of an edge of this subtree that is not a linear combination of the columns in X contrary to the choice of x. We conclude that X is a basis of the column space of A. In particular, unless A is the zero matrix, the set X is non-empty.

Let A' be the matrix obtained from A by elementary row operations such that the submatrix of A' induced by the columns of X is the unit matrix and with some additional zero rows; note that the set X is determined by the input deletion-tree and

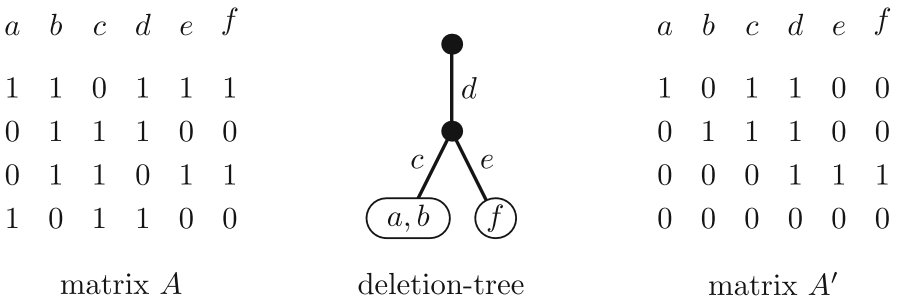


Fig. 5 A binary matrix A, a deletion-tree of the matroid $M(A)$ and the matrix A' as in the proof of Lemma 1. Note that $X = \{a, b, f\}$

so the matrix A' depends on the deletion-tree. See Fig. 5 for an example. This finishes the construction of A' and, in particular, the description of the algorithm to produce the output matrix A'. To complete the proof, it remains to show that the primal tree-depth of A' is at most d, i.e., the correctness of the algorithm. This will be proven by induction on the number of columns of an input matrix A.

The base of the induction is the case when A has a single column. In this case, the primal tree-depth of A' is one and the tree T is a single vertex labeled with the only column of A, and so its height is one. We next present the induction step. First observe that every label of a vertex of T is either in X or a loop in $M(A)$ (recall that only non-zero columns of A are included to X), and every label of an edge e is a linear combination of labels of the vertices in the subtree delimited by e.

Suppose that the root of T has a label and let x be one of its labels; note that x is either a loop or a bridge in the matroid $M(A)$. Let B be the matrix obtained from A by deleting the column x, and let T' be the deletion-tree of $M(B)$ obtained from T by removing the label x from the root. Let B' be the matrix produced by the algorithm described above for B and T'. If x is a loop, then the matrix A' is the matrix B' extended by a zero column with possibly permuted rows, and if x is a bridge (and so $x \in X$), then the matrix A' is, possibly after permuting rows, the matrix B' extended by a unit vector such that its non-zero entry is the only non-zero entry in its row. In either case, the vertex associated with the column x is isolated in the primal graph of A', and it follows that $\mathrm{td}_P(A') = \mathrm{td}_P(B') \leq d$ (the inequality holds by the induction hypothesis). Hence, we can assume that the root of T has no label.

We next analyze the case that the root of T has a single child and no label. Let x be the label of the single edge incident with the root of T. Let B be the matrix obtained from A by deleting the column x, and let T' be the deletion-tree of $M(B)$ obtained from T by deleting the edge incident with the root and rooting the tree at the remaining vertex of the deleted edge. Let B' be the matrix produced by the algorithm described above for B and T'; note that the primal tree-depth of B' is at most $d - 1$ by the induction hypothesis. Since B' is the submatrix of A' formed by the columns different from x (possibly after permuting rows), the primal tree-depth of A' is at most $\mathrm{td}_P(B') + 1 = d$.

The final case to analyze is the case when the root of T has $k \geq 2$ children (in addition to having no label). Let $T_1, \ldots, T_k$ be the k subtrees of T delimited by the k

edges incident with the root of T, let $Y_1, \ldots, Y_k$ be the labels of the vertices and edges of these subtrees, and let $B_1, \ldots, B_k$ be the submatrices of A formed by the columns contained in $Y_1, \ldots, Y_k$. Observe that T_i is a deletion-tree of the matroid $M(B_i)$ for $i = 1, \ldots, k$, and the matrix B_i' produced by the algorithm described above for B_i and T_i is the submatrix of A' formed by the columns contained in Y_i (possibly after permuting rows). Since the support of the columns contained in Y_i contains only the unit entries of the columns of A' contained in $X \cap Y_i$, the primal graph of A' contains no edge joining a column of Y_i and a column of Y_j for $i \neq j$. It follows that the primal tree-depth of A' is at most the maximum primal tree-depth of B_i, which is at most d by the induction hypothesis. It follows that $\mathrm{td}_P(A') \leq d$ as desired. The proof of the correctness of the algorithm is now completed and so is the proof of the lemma. □

We are now ready to establish the link between the optimal primal tree-depth and the deletion-depth of the matroid associated with the matrix.

Proof of Theorem 2 Fix a matrix A. By Lemma 1, it holds that $\mathrm{td}^*_P(A) \leq \mathrm{dd}(A)$ as there exists a deletion-tree of the matroid $M(A)$ with height $\mathrm{dd}(A)$. So, we focus on proving that $\mathrm{dd}(A) \leq \mathrm{td}^*_P(A)$. We will show that every matrix B satisfies that $\mathrm{dd}(B) \leq \mathrm{td}_P(B)$; this implies that $\mathrm{dd}(A) \leq \mathrm{td}_P(A')$ for every matrix A' row-equivalent to A as $\mathrm{dd}(A) = \mathrm{dd}(A')$ and so implies that $\mathrm{dd}(A) \leq \mathrm{td}^*_P(A)$.

The proof that $\mathrm{dd}(B) \leq \mathrm{td}_P(B)$ proceeds by induction on the number of columns. If B has a single column, then both $\mathrm{dd}(B)$ and $\mathrm{td}_P(B)$ are equal to one. We next present the induction step. We first consider the case when the matroid $M(B)$ is not connected. Let $B_1, \ldots, B_k$ be the submatrices of B formed by columns corresponding to the components of $M(B)$; note that some of the submatrices may consist of a single zero column (if $M(B)$ has a loop). The definition of the deletion-depth implies that $\mathrm{dd}(B)$ is the maximum among $\mathrm{dd}(B_1), \ldots, \mathrm{dd}(B_k)$. On the other hand, the primal tree-depth of each of the matrices B_i is at most the primal tree-depth of the matrix B as the primal graph of B_i is a subgraph of the primal graph of B. It follows that $\mathrm{dd}(B_i) \leq \mathrm{td}_P(B)$, which implies that $\mathrm{dd}(B) \leq \mathrm{td}_P(B)$.

We next assume that the matroid $M(B)$ is connected and claim that the primal graph of B must also be connected. Suppose that the primal graph of B is not connected, i.e., there exists a partition of rows of B into R_1 and R_2 and a partition of the columns into C_1 and C_2, such that for each $i = 1, 2$, the support of each column in C_i is contained in R_i. Therefore, for any dependent set of columns of B, either its subset formed by columns contained in C_1 is dependent, or its subset formed by by columns contained in C_2 is dependent, or both these subsets are dependent. It follows that the support of every circuit of $M(B)$ is fully contained in either C_1 or C_2; in particular, no there is no circuit of $M(B)$ containing a column from C_1 and a column from C_2. This implies that $M(B)$ is not connected. Hence, the primal graph of B must be connected.

Since the primal graph of B is connected, there exists a column such that the matrix B' obtained by deleting this column satisfies that $\mathrm{td}_P(B') = \mathrm{td}_P(B) - 1$. The induction assumption yields that $\mathrm{dd}(B') \leq \mathrm{td}_P(B) - 1$ and the definition of the deletion-depth yields that the deletion-depth of $M(B)$ is at most the deletion-depth of $M(B')$ increased by one. This implies that $\mathrm{dd}(B) = \mathrm{dd}(M(B)) \leq \mathrm{td}_P(B)$ as desired. □

Before proceeding with our structural result concerning incidence tree-depth, we use the structural results presented in Lemma 1 and Theorem 2 to get a parameterized algorithm for computing an optimal primal tree-depth of a matrix over a finite field.

Corollary 1 *There exists a fixed parameter algorithm for the parameterization by a finite field $\mathbb{F}$ and an integer d that for an input matrix A over the field $\mathbb{F}$,*

- *either outputs that $\mathrm{td}^*_P(A) > d$, or*
- *computes a matrix A' row-equivalent to A with $\mathrm{td}_P(A') \leq d$ and also outputs the associated deletion-tree of $M(A)$ with height $\mathrm{td}_P(A')$.*

Proof We first show that the property that a matroid M has deletion depth at most d can be expressed in monadic second order logic. Recall that monadic second order formulas for matroids may contain quantification over elements and subsets of elements of a matroid, and the predicate $\psi_I(\cdot)$ used to test whether a particular subset is independent in addition to logic connectives and the equality $=$, the set inclusion $\in$ and the subset inclusion $\subseteq$. In the formulas that we present, small letters are used to denote elements of a matroid and capital letters subsets of the elements. We next present a monadic second order formula $\psi_d(X)$ that describes whether the deletion-depth of the restriction of the matroid M to a subset X of the elements of M is at most d, which would imply the statement. The following formula $\psi_c(\cdot, \cdot)$ describes the existence of a circuit containing two distinct elements:

$$\psi_c(x, y) \equiv (x \neq y) \wedge (\exists X : x \in X \wedge y \in X \wedge \neg\psi_I(X) \wedge \forall z \in X : \psi_I(X \setminus z)) .$$

The next formula $\psi_C(\cdot)$ describes whether a subset X is a component of a matroid (recall that the binary relation of two matroid elements being contained in a common circuit is transitive):

$$\psi_C(X) \equiv (\forall x, y \in X : x \neq y \Rightarrow \psi_c(x, y)) \wedge (\forall x \in X, y \notin X : \neg\psi_c(x, y)) .$$

The sought formula $\psi_d(\cdot)$ is defined inductively as follows (we remark that $\psi_d(\emptyset)$ is true for all d):

$$\begin{aligned} &\psi_1(X) \equiv \forall x, y \in X : x \neq y \Rightarrow \neg\psi_c(x, y) && \text{and} \\ &\psi_d(X) \equiv \forall X' \subseteq X : \psi_C(X') \Rightarrow \exists x \in X' : \psi_{d-1}(X' \setminus \{x\}) && \text{for } d \geq 2. \end{aligned}$$

Hliněný [29, 30] proved that all monadic second order logic properties can be tested in a fixed parameter way for matroids represented over a finite field $\mathbb{F}$ with branch-width at most d when parameterized by the property, the field $\mathbb{F}$ and the branch-width d. Since the branch-width of a matroid M is at most its deletion-depth, it follows that testing whether the deletion-depth of an input matroid represented over a finite field $\mathbb{F}$ is at most d is fixed parameter tractable when parameterized by the field $\mathbb{F}$ and the integer d. This establishes the existence of a fixed parameter algorithm deciding whether $\mathrm{td}^*_P(A) = \mathrm{dd}(M(A)) \leq d$ (the equality follows from Theorem 2). To obtain the algorithm claimed to exist in the statement of the corollary, we need to extend the algorithm for testing whether the deletion-depth of an input matroid M represented

over $\mathbb{F}$ is at most d to an algorithm that also outputs a deletion-tree of M with height at most d; this would yield an algorithm for computing A' by Lemma 1.

We now describe the extension of the algorithm from testing to constructing a deletion-decomposition tree as a recursive algorithm. Let d be the computed deletion-depth of an input matroid M. The deletion-depth of an input matroid M is one if and only if every element of M is a loop or a bridge. Since the latter is easy to algorithmically test, if $d = 1$, then the deletion-tree of height one consists of a single vertex labeled with all elements of M. If $d \geq 2$ and the matroid M is not connected, we first identify its components, which can be done in polynomial time even in the oracle model, then proceed recursively with each component of M and eventually merge the roots of all deletion-trees obtained recursively.

Finally, we discuss the case when $d \geq 2$ and the matroid M is connected. We loop over all elements e of M and test using the monadic second order checking algorithm of Hliněný [29, 30] whether $\mathrm{dd}(M \setminus e) \leq d-1$, i.e., whether $\psi_{d-1}(M \setminus e)$ is true. Such an element e must exist (otherwise, the deletion-depth of M cannot be d) and when e is found, we recursively apply the algorithm to $M \setminus e$ to obtain a deletion-tree T of $M \setminus e$ with height $d-1$. Note that there is a single recursive call as we invoke recursion for a single element e of the matroid M. The tree T returned by the recursive call is extended to a deletion-tree of M by introducing a new vertex, joining it by an edge to the root of T, rerooting the tree to the new vertex, and labeling the new edge with the element e. This completes the description of the algorithm for constructing a deletion-tree of height at most d if it exists. Observe that the running time of the described procedure for constructing a deletion-tree is bounded by the product of the number of elements of M and the running time of the test whether an input matroid is connected and the test whether an input matroid satisfies $\psi_1(\cdot), \ldots, \psi_d(\cdot)$; in particular, it is fixed parameter when parameterized by a finite field $\mathbb{F}$ and an integer d. □

We conclude this section by establishing a link between the optimal incidence tree-depth and the contraction*-deletion-depth of the matroid associated with the matrix.

Proof of Theorem 3 We prove the equality as two inequalities starting with the inequality $\mathrm{c}^*\mathrm{dd}(A) \leq \mathrm{td}^*_I(A) - 1$. To prove this inequality, we show that $\mathrm{c}^*\mathrm{dd}(A) \leq \mathrm{td}_I(A) - 1$ holds for every matrix A with m rows and n columns by induction on $m + n$. The base of the induction is formed by the cases when all entries of A are zero, $n = 1$ or $m = 1$. If all entries of A are zero, then $\mathrm{c}^*\mathrm{dd}(A) = 0$ and $\mathrm{td}_I(A) = 1$. If $n = 1$ and A is non-zero, then $M(A)$ has a single non-loop element and so $\mathrm{c}^*\mathrm{dd}(A) = 1$ while the incidence graph of A is formed by a star and possibly some isolated vertices and so $\mathrm{td}_I(A) = 2$. Finally, if $m = 1$ and A is non-zero, then $M(A)$ has rank 1, so contracting any non-loop element of $M(A)$ yields a matroid of rank zero and so $\mathrm{c}^*\mathrm{dd}(A) = 1$. On the other hand, the incidence graph of A is formed by a star and possibly some isolated vertices and so $\mathrm{td}_I(A) = 2$.

We now establish the induction step, i.e., we assume that the matrix A is non-zero, $m \geq 2$ and $n \geq 2$. First suppose that the matroid $M(A)$ is not connected. Let $X_1, \ldots, X_k$ be the components of $M(A)$ and let $A_1, \ldots, A_k$ be the submatrices of A formed by the columns $X_1, \ldots, X_k$, respectively. The definition of the contraction*-deletion-depth implies that $\mathrm{c}^*\mathrm{dd}(A)$ is the maximum of $\mathrm{c}^*\mathrm{dd}(A_i)$. The induction hypothesis yields that $\mathrm{c}^*\mathrm{dd}(A_i) \leq \mathrm{td}_I(A_i) - 1$. Since the incidence graph of

A_i is a subgraph of the incidence graph of A, it follows that $\mathrm{td}_I(A_i) \leq \mathrm{td}_I(A)$ and so $\mathrm{c}^*\mathrm{dd}(A_i) \leq \mathrm{td}_I(A) - 1$. We conclude that $\mathrm{c}^*\mathrm{dd}(A) \leq \mathrm{td}_I(A) - 1$.

Next suppose that the matroid $M(A)$ is connected but the incidence graph of A is not connected. As the columns associated with vertices contained in different components of the incidence graph of A have disjoint supports, such columns cannot be contained in the same component of the matroid $M(A)$. Hence, if the incidence graph of A is not connected despite the matroid $M(A)$ being connected, then the incidence graph of A consists of a single non-trivial component and isolated vertices associated with zero rows of A. Let x be one such row and let A' be the matrix obtained from A by deleting the row x. Since the matroids $M(A)$ and $M(A')$ are the same, it holds that $\mathrm{c}^*\mathrm{dd}(A) = \mathrm{c}^*\mathrm{dd}(A')$, and since the incidence graph of A is the incidence graph of A' with an isolated vertex added, it holds $\mathrm{td}_I(A) = \mathrm{td}_I(A')$. Hence, the induction hypothesis yields that $\mathrm{c}^*\mathrm{dd}(A') \leq \mathrm{td}_I(A')-1$, which implies that $\mathrm{c}^*\mathrm{dd}(A) \leq \mathrm{td}_I(A)-1$.

Finally, suppose that the matroid $M(A)$ is connected and the incidence graph of A is also connected. The definition of the tree-depth implies that there exists a vertex w of the incidence graph whose deletion decreases the tree-depth of the incidence graph by one. Let A' be the matrix obtained from A by deleting the row or the column associated with the vertex w and note that $\mathrm{td}_I(A') = \mathrm{td}_I(A)-1$. If the vertex w is associated with a column x, the matroid $M(A')$ is the matroid obtained from $M(A)$ by deleting the element x. If the vertex w is associated with a row x, the matroid $M(A')$ is the matroid obtained from $M(A)$ by contracting by the subspace generated by the unit vector with the entry in the row x. In either case, the definition of the contraction*-deletion-depth implies that $\mathrm{c}^*\mathrm{dd}(A) \leq \mathrm{c}^*\mathrm{dd}(A') + 1$. The induction hypothesis applied to A' yields that $\mathrm{c}^*\mathrm{dd}(A') \leq \mathrm{td}_I(A') - 1$, which yields that $\mathrm{c}^*\mathrm{dd}(A) \leq \mathrm{td}_I(A') = \mathrm{td}_I(A) - 1$.

To complete the proof of the theorem, it remains to show that the inequality $\mathrm{td}^*_I(A) \leq \mathrm{c}^*\mathrm{dd}(A) + 1$ holds for every matrix A. The proof proceeds by induction on the number n of columns of A. If $n = 1$ and the only column of A is zero, then the incidence graph of A is formed by isolated vertices and so $\mathrm{td}_I(A) = 1$ while $\mathrm{c}^*\mathrm{dd}(A) = 0$ since the rank of $M(A)$ is zero. If $n = 1$ and the only column of A is not zero, then the incidence graph of A is formed by a star and possibly some isolated vertices and so $\mathrm{td}_I(A) = 2$ while $\mathrm{c}^*\mathrm{dd}(A) = 1$. In either case, it holds that $\mathrm{td}^*_I(A) \leq \mathrm{td}_I(A) = \mathrm{c}^*\mathrm{dd}(A) + 1$.

We now establish the induction step. First suppose that the matroid $M(A)$ is not connected. Let $X_1, \ldots, X_k$ be the sets of columns forming the components of $M(A)$ and let A' be the matrix row-equivalent to A such that there exist sets of rows $Y_1, \ldots, Y_k$ such that $|Y_i| = r_{M(A)}(X_i)$ for $i = 1, \ldots, k$ and the only columns with non-zero entries in the rows Y_i are those of X_i and all rows not contained in $Y_1 \cup \cdots \cup Y_k$ are zero (such a matrix A' exists since the matroid $M(A)$ is union of its restrictions to $X_1, \ldots, X_k$). Let A'_i be the submatrix of A' formed by the rows of Y_i and the columns of X_i. Observe that all entries of the matrix A not contained in one of the matrices $A'_1, \ldots, A'_k$ are zero. By the induction hypothesis, for every $i = 1, \ldots, k$, there exists a matrix A''_i row-equivalent to A'_i such that $\mathrm{td}_I(A''_i) \leq \mathrm{c}^*\mathrm{dd}(M(A)[X_i])+1$, in particular, $\mathrm{td}_I(A''_i) \leq \mathrm{c}^*\mathrm{dd}(A) + 1$. Let A'' be the matrix obtained from A' by replacing A'_i with A''_i for $i = 1, \ldots, k$. Observe that A'' is row-equivalent to A' and so to A. Since the incidence graph of A'' is the union of the incidence graphs of A''_i, $i = 1, \ldots, k$,

and possibly some isolated vertices (which correspond to zero rows), it follows that $\mathrm{td}_I(A'') \leq \mathrm{c}^*\mathrm{dd}(A) + 1$. Hence, it holds that $\mathrm{td}^*_I(A) \leq \mathrm{c}^*\mathrm{dd}(A) + 1$.

To complete the proof, we need to consider the case that the matroid $M(A)$ is connected. The definition of the contraction*-deletion-depth implies that there exists an element x of $M(A)$ such that $\mathrm{c}^*\mathrm{dd}(M(A)\backslash x) = \mathrm{c}^*\mathrm{dd}(M(A)) - 1 = \mathrm{c}^*\mathrm{dd}(A) - 1$ or there exists a one-dimensional subspace such that the contraction by this subspace yields a matroid M' such that $\mathrm{c}^*\mathrm{dd}(M') = \mathrm{c}^*\mathrm{dd}(M(A)) - 1 = \mathrm{c}^*\mathrm{dd}(A) - 1$. In the former case, let A' be the matrix obtained from A by deleting the column x. By the induction hypothesis, there exists a matrix A'' row-equivalent to A' such that $\mathrm{td}_I(A'') \leq \mathrm{c}^*\mathrm{dd}(A') + 1 = \mathrm{c}^*\mathrm{dd}(A)$, and let A''' be the matrix obtained from A by the same elementary row operations that A'' is obtained from A'. Observe that the incidence graph of A'' can be obtained from the incidence graph of A''' by deleting the vertex associated with the column x. Hence, $\mathrm{td}_I(A''') \leq \mathrm{td}_I(A'') + 1$. Since A''' is row-equivalent to A, it follows that

$$\mathrm{td}^*_I(A) \leq \mathrm{td}_I(A''') \leq \mathrm{td}_I(A'') + 1 \leq \mathrm{c}^*\mathrm{dd}(A) + 1.$$

We now analyze the latter case, i.e., the case that there exists a one-dimensional subspace such that the contraction by this subspace yields a matroid M' with $\mathrm{c}^*\mathrm{dd}(M') = \mathrm{c}^*\mathrm{dd}(A) - 1$. Let A' be the matrix obtained from A by elementary row operations such that the contracted subspace used to obtain M' is generated by the unit vector with the non-zero entry being its first entry, and let B be the matrix obtained from A' by deleting the first row. By the induction hypothesis, there exists a matrix B' row-equivalent to B such that $\mathrm{td}_I(B') \leq \mathrm{c}^*\mathrm{dd}(A') + 1 = \mathrm{c}^*\mathrm{dd}(A)$, and let A'' be the matrix consisting of the first row of A and the matrix B'. Observe that A'' is row-equivalent to A. Since the incidence graph of B' can be obtained from the incidence graph of A'' by deleting the vertex associated with the first row, it holds that $\mathrm{td}_I(A'') \leq \mathrm{td}_I(B') + 1$. Hence, it follows that

$$\mathrm{td}^*_I(A) \leq \mathrm{td}_I(A'') \leq \mathrm{td}_I(B') + 1 \leq \mathrm{c}^*\mathrm{dd}(A) + 1.$$

The proof of the theorem is now completed. □

4 Primal tree-depth

In this section, we present a parameterized algorithm for computing a row-equivalent matrix with small primal tree-depth and bounded entry complexity if such a matrix exists.

Proof of Theorem 6 We first find a bound on $\dot{\kappa}_B$ when B is an integer matrix with $\mathrm{td}_P(B) \leq d$ and $\mathrm{ec}(B) \leq e$ (recall that $\dot{\kappa}_B$ is the least common multiple of the entries of the circuits of B). Consider such a square invertible matrix C with $\mathrm{td}_P(C) \leq d$ and $\mathrm{ec}(C) \leq e$. By the result of Brand, Ordyniak and the second author [5], the maximum denominator appearing over all entries of C^{-1} can be bounded by a function of d and e; in particular, there exists $k_0 \leq (2^e)^{d!}(d!)^{d!/2}$ such that every entry of C^{-1} is

$1/k$-integral for some $k \leq k_0$. Set κ_0 to be the least common multiple of the integers $1, \ldots, k_0$. By [21, Theorem 3.8], $\dot{\kappa}_B$ is the smallest integer such that every denominator of the inverse of every invertible square submatrix of B divides $\dot{\kappa}_B$. Since the primal tree-depth of any square submatrix of B at most the primal tree-depth of B, $\dot{\kappa}_B$ divides κ_0 for every matrix B with $\mathrm{td}_P(B) \leq d$ and $\mathrm{ec}(B) \leq e$. In particular, $\dot{\kappa}_B \leq \kappa_0$ for every matrix B with $\mathrm{td}_P(B) \leq d$ and $\mathrm{ec}(B) \leq e$.

We next describe the algorithm from the statement of the theorem. Without loss of generality, we can assume that the rank of the input matrix A is equal to the number of its rows, in particular, A is non-zero. The algorithm starts with diagonalizing the square submatrix of the input matrix A formed by an arbitrary basis of the column space, i.e., performing elementary row operations so that the selected columns form the identity matrix. The resulting matrix is denoted by A_I. If the numerator or the denominator of any (non-zero) entry of A_I does not divide κ_0, the algorithm arrives at the first conclusion of the theorem as every (non-zero) entry of A_I is a fraction that can be obtained by dividing two entries of a circuit of A (indeed, consider the circuit of the matrix A with support formed by a column x and some of the columns of the chosen basis, and observe that each entry in the column x is equal to the x-entry of the circuit divided by one of its other entries). Hence, we assume that both numerator and denominator of each (non-zero) entry of A_I divides κ_0 in the rest. The algorithm multiplies A_I by κ_0, which yields an integral matrix A_0 with entries between $-\kappa_0^2$ and κ_0^2.

Let $M_{\mathbb{Q}}$ be the column matroid of A_0 when viewed as a matrix over rationals and let M_p be the column matroid of A_0 when viewed as a matrix over a p-element field $\mathbb{F}_p$ for a prime $p > \kappa_0^2$; note that such a prime p can be found algorithmically as the algorithm is parameterized by d and e and κ_0 depends on d and e only. Note that the elements of both matroids $M_{\mathbb{Q}}$ and M_p are the columns of the matrix A_0, i.e., we can assume that their ground sets are the same, and the matroid $M_{\mathbb{Q}}$ is the column matroid of A, which is a matrix over rationals.

We now establish the following claim: *if A is row-equivalent to a matrix with primal tree-depth at most d and entry complexity at most e, then the matroids $M_{\mathbb{Q}}$ and M_p are the same*. If a set X of columns forms a circuit in $M_{\mathbb{Q}}$, then there exists a linear combination of the columns of X that has all coefficients integral and coprime, i.e., not all are divisible by p, and that is equal to the zero vector (in fact, there exist such coefficients that they all divide κ_0 by the definition of κ_0); it follows that the set X is also dependent in M_p. If a set X of columns is independent in $M_{\mathbb{Q}}$, let B_I be an invertible square submatrix of A_I formed by the columns X and $|X|$ rows, and let B_0 be the square submatrix of A_0 formed by the same rows and columns. By [21, Lemma 3.3], the matrix B_I^{-1} is $1/\dot{\kappa}_A$-integral and the absolute value of both numerator and denominator of each entry of B_I^{-1} is at most $\dot{\kappa}_A$. Note that $\dot{\kappa}_A$ divides κ_0 (here, we use the definition of κ_0 and the assumption that A is row-equivalent to a matrix with primal tree-depth at most d and entry complexity at most e) and so the matrix B_I^{-1} is $1/\kappa_0$-integral and the absolute value of both numerator and denominator of each entry of B_I^{-1} is at most κ_0. Let B' be the matrix obtained from B_I^{-1} by multiplying each entry by κ_0; note that B' is an integer matrix with entries between $-\kappa_0^2$ and κ_0^2. The definitions of the matrices B_I, B_0 and B' yield that the product of the matrix B'

when viewed as over $\mathbb{F}_p$ and the matrix B_0 has numerically the same entries as the matrix $(\kappa_0 B_I^{-1})(\kappa_0 B_I)$, which is a diagonal matrix with all diagonal entries equal to κ_0^2. Hence, the matrix B_0 is full rank (over the field $\mathbb{F}_p$) and so the set X is independent in M_p.

We now continue the description of the algorithm that is asserted to exist in the statement of the theorem. As the next step, we apply the algorithm from Corollary 1 to the matrix A_0 viewed as over the field $\mathbb{F}_p$. If the algorithm determines that the deletion-depth of A_0 is larger than d, we arrive at the first conclusion of the theorem: either the matroids $M_{\mathbb{Q}}$ and M_p are different (and so A is not row-equivalent to a matrix with primal tree-depth at most d and entry complexity at most e), or the matroids $M_{\mathbb{Q}}$ and M_p are the same but $\mathrm{dd}(A) = \mathrm{td}^*_P(A) > d$. If the algorithm determines that the deletion-depth of A_0 is at most d, it also outputs a deletion-tree of M_p with height at most d. If the output deletion-tree is not valid for the matroid $M_{\mathbb{Q}}$, which can be verified in polynomial time, the matroids $M_{\mathbb{Q}}$ and M_p are different and we again arrive at the first conclusion of the theorem. If the output deletion-tree is also a deletion-tree of the matroid $M_{\mathbb{Q}}$, we use the algorithm from Lemma 1 to obtain a matrix A' row-equivalent to A such that the primal tree-depth of A' at most the height of the deletion-tree, i.e., $\mathrm{td}_P(A') \leq d$. As the matrix A' contains a unit submatrix formed by m rows and m columns, each (non-zero) entry of A' is a fraction that can be obtained by dividing two entries of a circuit of A (as argued earlier). If the absolute value of the numerator or the denominator of any of these fractions exceeds κ_0, then $c_\infty(A) > \kappa_0$ and we again arrive at the first conclusion of the theorem. Otherwise, we output the matrix A'. Note that the primal tree-depth of A' is at most d and its entry complexity is at most $2\lceil \log_2(\kappa_0 + 1)\rceil$. As κ_0 depends on d and e only, the matrix A' has the properties given in the second conclusion of the theorem. □

5 Dual tree-depth, circuit complexity and Graver basis

In this section, we link minimum dual tree-depth of a matrix to its circuit complexity, and also present related algorithmic results. We start with proving Theorem 5; in fact, we show that a matrix row-equivalent to an input matrix A such that both its dual tree-depth and entry complexity bounded by a function of $c_1(A)$ can be found efficiently. Note that the algorithm presented in the next theorem is not a fixed parameter algorithm, i.e., its running time is polynomial in the size of an input matrix.

Theorem 11 *There exists a polynomial-time algorithm that for a given rational matrix A with $\dim\ker A > 0$ outputs a row-equivalent matrix A' such that $\mathrm{td}_D(A') \leq c_1(A)^2$ and $\mathrm{ec}(A') \leq 2\lceil \log_2(c_1(A) + 1)\rceil$.*

Proof We start with the description of the algorithm from the statement of the theorem. Let A be the input matrix. We apply the algorithm from Theorem 10 to the matroid $M(A)$ given by the columns of the matrix A. Let T be the principal contraction*-tree output by the algorithm and let X be the set of columns of A that are the labels of the edges of T, i.e., the elements of $M(A)$ used in the contractions. Observe that the definition of the contraction*-depth and the principal contraction*-tree yields that the labels of the edges of T form a basis X of $M(A)$ and for every element z of $M(A)$

there is a leaf v of T such that z is contained in the linear hull of the labels of the edges on the path from v to the root of T. Next perform row-operations on the matrix A in a way that the submatrix formed by the columns of X is an identity matrix (with additional zero rows if the rank of A is smaller than the number of its rows); let A' be the resulting matrix; see Fig. 6 for an example. The algorithm outputs the matrix A'. Note that the running time of the algorithm is indeed polynomial in the size of the input matrix A.

We next analyze the matrix A' that is output by the algorithm. Since $\dim \ker A > 0$, the matrix A has at least one circuit. Recall that for every circuit C of the matroid $M(A)$, there exists a circuit of A whose support is exactly formed by the elements of C. This implies that every circuit of $M(A)$ contains at most $c_1(A)$ elements and so Theorem 10 implies that the depth of the principal contraction*-tree T is at most $c_1(A)^2$.

Let F be a rooted forest obtained from the tree T as follows. For each edge e, the vertex of e farther from the root is identified with the (unique) row of A' that is non-zero in the column that is the label of the edge e (recall that the columns of X form an identity matrix), and then remove the root of T; also add an isolated vertex for each zero row of A'. In this way, we obtain a rooted forest F with vertex set formed by the rows of A'. Note that the height of F is at most $c_1(A)^2$. We will show that the dual graph of A' is a subgraph of $\mathrm{cl}(F)$. As no column of X contributes any edges to the dual graph of A', it is enough to consider columns not contained in X. Let z be a column of A' that is not contained in X and let v be a leaf of T such that the column z of A, which is an element of $M(A)$, is contained in the linear hull of the labels of the edges on the path from v to the root of T. Hence, the column z of A' contains non-zero entries only in the rows with non-zero entries in the columns that are labels of the edges on the path from v to the root of T. Consequently, all edges contained in the dual graph of A' because of non-zero entries in the column z are between vertices on the path from the vertex v in F to the root of the corresponding tree of F. It follows that the dual graph of A' is a subgraph of $\mathrm{cl}(F)$ and so its tree-depth is at most the height of F, i.e., it is at most $c_1(A)^2$. We conclude that $\mathrm{td}_D(A') \leq c_1(A)^2$.

It remains to analyze the entry complexity of A'. The entries of A' in the columns of X are zero or one. Next consider a column z of A' that is not contained in X and consider a circuit c of A' (and so of A) whose support contains z and some elements of X (such a circuit exists as the columns of X form a basis of the column space of A'). Observe that the entries in the column z are equal to $-c_x/c_z$ (otherwise, c would not be a circuit of A'). We conclude that the entry complexity of A' is at most $2\lceil \log_2(c_1(A)+1)\rceil$. □

We are now ready to prove Theorem 4. Note that the condition $\dim \mathrm{Ker} A > 0$ in the statement of the theorem is necessary as otherwise A has no circuit and so $c_1(A)$ is not defined.

Proof of Theorem 4 Consider a rational matrix A with $\dim \mathrm{Ker} A > 0$. Note that $c_1(A) \leq g_1(A)$ as every circuit of A is also an element of the Graver basis of A. To prove the existence of the function f_1, let f_D be the function from Theorem 9 and note that Theorem 11 implies that $g_1(A) \leq f_D(c_1(A)^2, 2\lceil \log_2(c_1(A)+1)\rceil)$. □

We next combine the algorithms from Theorems 7 and 11.

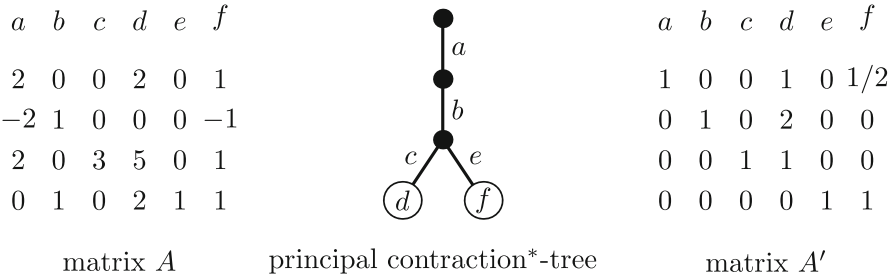


Fig. 6 A rational matrix A, a principal contraction*-tree T of the matroid $M(A)$ and the matrix A' as in the proof of Theorem 11

Corollary 2 *There exists a function $f : \mathbb{N} \to \mathbb{N}$ and a fixed parameter algorithm for the parameterization by k that for a given rational matrix A:*

- *either outputs that $c_1(A) > k$, or*
- *outputs a matrix A' that is row-equivalent to A, its dual tree-depth is $\mathrm{td}^*_D(A)$ and its entry complexity is at most $f(k)$.*

Proof If $\dim \mathrm{Ker} A = 0$ for an input matrix A, i.e., the rank of A is equal to the number of the columns, which can be easily verified in polynomial time, then A is row-equivalent to the unit matrix possibly with some zero rows added, i.e., to a matrix with dual tree-depth one and entry complexity one. If $\dim \mathrm{Ker} A > 0$, we apply the algorithm from Theorem 11 to get a matrix A' row-equivalent to A that has properties given in the statement of Theorem 11. If the dual tree-depth of A' is larger than k^2 or the entry complexity of A' is larger than $2\lceil \log_2(k+1) \rceil$, then $c_1(A) > k$ (by Theorem 11) and we arrive at the first conclusion. Otherwise, we apply the algorithm from Theorem 7 with parameters $d = k^2$ and $e = 2\lceil \log_2(k+1) \rceil$ to compute a matrix A'' row-equivalent to A' and so to A such that the dual tree-depth of A'' is $\mathrm{td}^*_D(A)$ and the entry complexity of A'' is bounded by a function of k only. □

Finally, the previous corollary together with Theorem 9 yields the parameterized algorithm for testing whether an input matrix is row-equivalent to a matrix with small dual tree-depth and small entry complexity as given in Theorem 8.

Proof of Theorem 8 Let f_D be the function from the statement of Theorem 9 and set $k = f_D(d, e)$; note $f_D(d, e) \leq 2^{2^{(d \log e)^{O(1)}}}$ by Eisenbrand et al. [20]. Apply the algorithm from Corollary 2 with the parameter k to an input matrix A. If the algorithm reports that $c_1(A) > k$, then A is not row-equivalent to a matrix with dual tree-depth at most d and entry complexity at most e. If the algorithm outputs a matrix A' and $\mathrm{td}_D(A') > d$, then $\mathrm{td}^*_D(A) > d$ and so the matrix A is not row-equivalent to a matrix with dual tree-depth at most d. Otherwise, the dual tree-depth of A' is at most d and its entry complexity is bounded by $f(k) = f(f_D(d, e))$ where $f(\cdot)$ is the function from Corollary 2, i.e., the entry complexity of A' is bounded by a function of d and e only as required. □

6 Computational hardness of depth parameters

In this section, we complement our algorithmic results by establishing computational hardness of matroid depth parameters that we have discussed in this paper. The hardness results apply even when the input matroid is given by its representation over a fixed (finite or infinite) field.

We start with defining a matroid $M_{\mathbb{F}}(G)$ derived from a graph G. Fix a field $\mathbb{F}$. For a graph G, we define an $\mathbb{F}$-represented matroid $M_{\mathbb{F}}(G)$ as follows. The matroid $M_{\mathbb{F}}(G)$ contains $|V(G)| + |E(G)|$ elements, each corresponding to a vertex or an edge of G. We associate each element of $M_{\mathbb{F}}(G)$ with a vector of $\mathbb{F}^{V(G)}$. An element of $M_{\mathbb{F}}(G)$ corresponding to a vertex w of G is represented by e_w and an element of $M_{\mathbb{F}}(G)$ corresponding to an edge ww' of G is represented by $e_w - e_{w'}$ or $e_{w'} - e_w$ (an arbitrary of the two vectors can be chosen as the choice does not affect the matroid).

We next define a graph G/A for a graph G and a linear subspace A of $\mathbb{F}^{V(G)}$. Let W be the subset of vertices of $V(G)$ such that $e_w \in A$ for $w \in W$, and let F be the set of edges ww' of G such that neither w nor w' is contained in W and A contains a vector $e_w + \alpha e_{w'}$ for a non-zero element $\alpha \in \mathbb{F}$. The graph G/A is obtained by deleting all vertices of W and then contracting a maximal acyclic subset of edges contained in F (the remaining edges of F become loops and so get removed); note that we use an acyclic subset of edges for contraction so that the resulting graph is well-defined.

The next lemma relates the number of components of the matroid $M_{\mathbb{F}}(G)/A$ and the number of components of the graph G/A for a graph G and a linear subspace A of $\mathbb{F}^{V(G)}$.

Lemma 2 *Let G be a graph, $\mathbb{F}$ a field and A a linear subspace of $\mathbb{F}^{V(G)}$. The number of components of $M_{\mathbb{F}}(G)/A$ is at most the number of components of the graph G/A.*

Proof Fix a graph G, a field $\mathbb{F}$ and a linear subspace of $\mathbb{F}^{V(G)}$. Let W and F be the subsets of vertices and edges of G as in the definition of G/A, respectively. Let A' be the linear subspace of $\mathbb{F}^{V(G)}$ generated by the vectors e_w, $w \in W$, and the vectors $e_w + \alpha e_{w'} \in A$ for $ww' \in F$; clearly, A' is a subspace of A. Note that the sets W and F defined with respect to A would be the same if defined with respect to A', and so the graphs G/A and G/A' are the same. We next describe an $\mathbb{F}$-representation of the matroid $M_{\mathbb{F}}(G)/A'$ using vectors of $\mathbb{F}^{V(G/A')}$. The matroid $M_{\mathbb{F}}(G)$ contains elements corresponding to vertices and to edges of G. Consider a vertex w of $V(G)$. If $w \in W$, then the element corresponding to w is a loop in $M_{\mathbb{F}}(G)/A'$ and so represented by the zero vector. If $w \notin W$, then the element corresponding to w is represented by the vector e_u where u is the vertex of G/A' that the vertex w was contracted to. Next consider an edge ww' of G.

- If both w and w' belong to W, then the element corresponding to ww' is a loop in $M_{\mathbb{F}}(G)/A'$ and so represented by the zero vector.
- If exactly one of w and w' belong to W, say $w \in W$ and $w' \notin W$, then the element corresponding to ww' is represented by the vector e_u where u is the vertex of G/A' that the vertex w' was contracted to.
- If neither w nor w' belongs to W and $e_w - e_{w'} \in A'$ (and so $ww' \in F$), then the element corresponding to ww' is a loop in $M_{\mathbb{F}}(G)/A'$ and so represented by the zero vector.

- If neither w nor w' belongs to W, $e_w - e_{w'} \notin A'$ but $ww' \in F$, then the element corresponding to ww' is represented by the vector e_u where u is the vertex of G/A' that the vertex w (and so the vertex w') was contracted to.
- Finally, if neither w nor w' belongs to W and $ww' \notin F$, the element corresponding to ww' is the vector $\alpha e_u + \alpha' e_{u'}$ where u is the vertex of G/A' that the vertex w was contracted to, u' is the vertex of G/A' that the vertex w' was contracted to, and the vector $\alpha e_u + \alpha' e_{u'}$ corresponds to the vector $e_w - e_{w'}$ in $\mathbb{F}^{V(G)}/A'$; note that uu' is an edge of G/A' and both coefficients α and α' are non-zero.

It is straightforward to verify that the just described representation is indeed a representation of the matroid $M_{\mathbb{F}}(G)/A'$; note that each non-loop element of $M_{\mathbb{F}}(G)/A'$ is associated to a vertex or an edge of G/A' and each vertex or an edge of G/A' with at least one (but possibly more) non-loop element of $M_{\mathbb{F}}(G)/A'$.

We now analyze the matroid $M_{\mathbb{F}}(G)/A$. The matroid $M_{\mathbb{F}}(G)/A$ can be viewed as the matroid $(M_{\mathbb{F}}(G)/A')/(A/A')$. Observe that the definition of A' implies that any non-loop element of $M_{\mathbb{F}}(G)/A'$ is non-loop in $M_{\mathbb{F}}(G)/A$. Also observe that the space A/A' viewed as a subspace of $\mathbb{F}^{V(G/A')}$ does not contain a vector e_w for $w \in V(G/A')$ or a vector $e_w + \alpha e_{w'}$ for a non-zero $\alpha \in \mathbb{F}$ such that ww' is an edge of $E(G/A')$. In particular, the support of every vector of A/A' is at least two and if the support has size two, then it does not correspond to an edge of $E(G/A')$. It follows that if ww' is an edge of $E(G/A')$, x is an element of $M_{\mathbb{F}}(G)/A'$ associated with the vertex w, x' is an element of $M_{\mathbb{F}}(G)/A'$ associated with the vertex w', and x'' is an element of $M_{\mathbb{F}}(G)/A'$ associated with the edge ww', then the elements x, x' and x'' form a circuit of $(M_{\mathbb{F}}(G)/A')/(A/A')$. Since the relation of being contained in a common circuit is transitive, it follows that all elements of $M_{\mathbb{F}}(G)/A'$ corresponding to the vertices and the edges of the same component of G/A' are contained in the same component of $(M_{\mathbb{F}}(G)/A')/(A/A')$. In particular, the number of components of the matroid $(M_{\mathbb{F}}(G)/A')/(A/A')$ is at most the number of components of the graph G/A'. Since the graphs G/A and G/A' are the same and the matroids $M_{\mathbb{F}}(G)/A$ and $(M_{\mathbb{F}}(G)/A')/(A/A')$ are also the same, the lemma follows. □

We next link the existence of a balanced independent set in a bipartite graph to the contraction*-depth of a suitably defined matroid. We remark that the idea of using a bipartite graph with cliques added between the vertices of its parts was used in [47] to establish that computing tree-depth of a graph is NP-complete.

Lemma 3 *Let G be a bipartite graph with parts X and Y, let $\mathbb{F}$ be a field, and let k be an integer. Let G' be the graph obtained from G by adding all edges between the vertices of X and between the vertices of Y. The following three statements are row-equivalent.*

- *The graph G has an independent set containing k elements of X and k elements of Y.*
- *The contraction*-depth of $M_{\mathbb{F}}(G')$ is at most $|X| + |Y| - k$.*
- *The contraction-depth of the matroid $2M_{\mathbb{F}}(G')$ is at most $|X| + |Y| - k + 1$.*

Proof Fix a bipartite graph G with parts X and Y, a field $\mathbb{F}$ and an integer k. We first show that if G has an independent set containing k elements of X and k elements of

Y, then the contraction*-depth of $M_{\mathbb{F}}(G')$ is at most $|X|+|Y|-k$ and the contraction-depth of $2M_{\mathbb{F}}(G')$ is at most $|X|+|Y|-k+1$. Let W be such an independent set and let W' be the set containing all elements e_w of $M_{\mathbb{F}}(G')$ such that $w \notin W$. Note that $|W'| = |X|+|Y|-2k$. The matroid $M_{\mathbb{F}}(G')/W'$ is the matroid obtained from $M_{\mathbb{F}}(G'[W])$ by adding

- a loop for every edge with both end vertices not contained in W, and
- an element represented by e_w for every edge joining a vertex $w \in W$ to a vertex not contained in W.

In particular, the matroid $M_{\mathbb{F}}(G')/W'$ has two non-trivial components, each of rank k, and so the contraction*-depth of $M_{\mathbb{F}}(G')$ is at most $|W'|+k = |X|+|Y|-k$. Similarly, the matroid $(2M_{\mathbb{F}}(G'))/W'$ is the matroid obtained from $2M_{\mathbb{F}}(G'[W])$ by adding

- a loop for every vertex not contained in W,
- two loops for every edge with both end vertices not contained in W, and
- two elements represented by e_w for every edge joining a vertex $w \in W$ to a vertex not contained in W.

Since the matroid $(2M_{\mathbb{F}}(G'))/W'$ has two non-trivial components, each of rank k, its contraction-depth is at most $|W'|+k+1 = |X|+|Y|-k+1$ (note that any rank r matroid has contraction-depth at most $r+1$).

We next argue that if the contraction*-depth of $M_{\mathbb{F}}(G')$ is at most $|X|+|Y|-k$ or the contraction-depth of $2M_{\mathbb{F}}(G')$ is at most $|X|+|Y|-k+1$, then there exists a subset W of $V(G) = V(G')$ that is independent in G, $|W \cap X| \geq k$ and $|W \cap Y| \geq k$. To do so, we first show that there is no linear subspace A such that $M_{\mathbb{F}}(G')/A$ would have more than two components. Consider a linear subspace A of $\mathbb{F}^{V(G')}$ such that the matroid $M_{\mathbb{F}}(G')/A$ is not connected. By Lemma 2, the graph G'/A is disconnected. Since the graph G'/A cannot have more than two components (one is formed by some of the vertices of X and another by some of the vertices of Y), it follows that the graph G'/A has exactly two components and so the matroid $M_{\mathbb{F}}(G')/A$ has exactly two components, too.

If the contraction*-depth of $M_{\mathbb{F}}(G')$ is at most $|X|+|Y|-k$, there exists a linear subspace A of $\mathbb{F}^{V(G')}$ such that the matroid $M_{\mathbb{F}}(G')/A$ is not connected and the rank of each of its two components is at most $|X|+|Y|-k-\dim A$. We will prove that the existence of such A is also implied by the assumption that the contraction-depth of $2M_{\mathbb{F}}(G')$ is at most $|X|+|Y|-k+1$. We next use that the matroid $(2M_{\mathbb{F}}(G'))/F$ has at most two non-trivial components for every subset F of the elements of $2M_{\mathbb{F}}(G')$. If the contraction-depth of $2M_{\mathbb{F}}(G')$ is at most $|X|+|Y|-k+1$, then there exists a subset F of the elements of $2M_{\mathbb{F}}(G')$ such that the matroid $(2M_{\mathbb{F}}(G'))/F$ is not connected and the rank of each of its two components is at most $|X|+|Y|-k-\operatorname{rank} F$ (as the contraction-depth of each of its two components is the rank of the component increased by one because each element is parallel to at least one other element). It follows that there exists a linear subspace A of $\mathbb{F}^{V(G')}$, which is the hull of the vectors representing the elements of the set F as above, such that the matroid $M_{\mathbb{F}}(G')/A$ is not connected and the rank of each of its two components is at most $|X|+|Y|-k-\dim A$. We conclude that if the contraction*-depth of $M_{\mathbb{F}}(G')$ is at most $|X|+|Y|-k$ or if the

contraction-depth of $2M_{\mathbb{F}}(G')$ is at most $|X| + |Y| - k + 1$, then there exists a linear subspace A of $\mathbb{F}^{V(G')}$ such that the matroid $M_{\mathbb{F}}(G')/A$ is not connected and the rank of each of its two components is at most $|X| + |Y| - k - \dim A$.

It remains to show that *the existence of a subspace A of $\mathbb{F}^{V(G')}$ such that the matroid $M_{\mathbb{F}}(G')/A$ is not connected and the rank of each of its two components is at most $|X| + |Y| - k - \dim A$ implies that there exists an independent set containing k elements of X and k elements of Y*. Fix such a subspace A. Let W be the set of vertices w such that e_w is contained in A and let A_W be the subspace of A generated by the vectors e_w, $w \in W$. Since G'/A is not connected, the graph $G' \setminus W$ is also not connected (recall that G'/A is obtained by removing the vertices of W and then contracting some edges). By Lemma 2 the matroid $M_{\mathbb{F}}(G')/A_W$ is also not connected. Since the space A_W is a subspace of A, the rank of each component of $M_{\mathbb{F}}(G')/A_W$ is larger by at most $\dim A - \dim A_W$ compared to the corresponding component of $M_{\mathbb{F}}(G')/A$. Hence, the rank of each of the two components of $M_{\mathbb{F}}(G')/A_W$ is at most $|X|+|Y|-k-\dim A_W = |X|+|Y|-k-|W|$. It follows that each component of the graph $G'/A_W = G' \setminus W$ contains at most $|X|+|Y|-k-|W|$ vertices. Since the sum of the sizes of the two components of $G' \setminus W$ is $|X| + |Y| - |W|$, each component of $G' \setminus W$ has at least k vertices. In addition, the vertex set of each component of $G' \setminus W$ is either a subset of X or a subset of Y, which implies that there is no edge joining a vertex of $X \setminus W$ and a vertex of $Y \setminus W$ and both sets $X \setminus W$ and $Y \setminus W$ have at least k vertices. Hence, the graph G has an independent set containing k elements of X and k elements of Y (such an independent set is a subset of $V(G) \setminus W$). □

We are now ready to state our hardness result.

Theorem 12 *For every field $\mathbb{F}$, each of the following five decision problems, whose input is an $\mathbb{F}$-represented matroid M and an integer d, is NP-complete:*

- *Is the contraction-depth of M at most d?*
- *Is the contraction*-depth of M at most d?*
- *Is the contraction-deletion-depth of M at most d?*
- *Is the contraction*-deletion-depth of M at most d?*
- *Is the deletion-depth of M at most d?*

Proof It is NP-complete to decide for a bipartite graph G with parts X and Y and an integer k whether there exist k-element subsets $X' \subseteq X$ and $Y' \subseteq Y$ such that $X' \cup Y'$ is independent [47]. For an input bipartite graph G, let G' be the graph obtained from G by adding all edges between the vertices of X and between the vertices of Y. We claim that the existence of such subsets X' and Y' is equivalent to each of the following four statements:

- The matroid $2M_{\mathbb{F}}(G')$ has contraction-depth at most $|X| + |Y| - k + 1$.
- The matroid $M_{\mathbb{F}}(G')$ has contraction*-depth at most $|X| + |Y| - k$.
- The matroid $(|V(G')| + 1)M_{\mathbb{F}}(G')$ has contraction-deletion-depth at most $|X| + |Y| - k + 1$.
- The matroid $(|V(G')| + 1)M_{\mathbb{F}}(G')$ has contraction*-deletion-depth at most $|X| + |Y| - k$.

The equivalences to the first and second statements follow directly from Lemma 3. Since the rank of the matroid $(|V(G')|+1)M_{\mathbb{F}}(G')$ is $|G'|$, its contraction-deletion-depth is at most $|G'|+1$ and its contraction*-deletion-depth is at most $|G'|$. As each element of the matroid $(|V(G')|+1)M_{\mathbb{F}}(G')$ is parallel to (at least) $|V(G')|$ elements of the matroid, it follows that the contraction-deletion-depth of $M_{\mathbb{F}}(G')$ is the same as its contraction-depth and its contraction*-deletion-depth is the same as its contraction*-depth. Lemma 3 now implies the equivalence of the third and fourth statements. As the matroids $2M_{\mathbb{F}}(G')$, $M_{\mathbb{F}}(G')$ and $(|V(G')|+1)M_{\mathbb{F}}(G')$ can be easily constructed from the input graph G in time polynomial in $|V(G)|$, the NP-completeness of the first four problems listed in the statement of the theorem follows.

For an $\mathbb{F}$-represented matroid M, it is easy to construct an $\mathbb{F}$-represented matroid M^* that is dual to M in time polynomial in the number of the elements of M [46, Chapter 2]. Since the contraction-depth of M is equal to the deletion-depth of M^*, it follows that the fifth problem listed in the statement of the theorem is also NP-complete. □

7 Concluding remarks

We would like to conclude with addressing three natural questions related to the work presented in this paper.

In Sect. 5, we have given a structural characterization of matrices A with $g_1(A)$ bounded by showing that $g_1(A)$ is bounded if and only if A is row-equivalent to a matrix with small dual tree-depth and small entry complexity. Unfortunately, a similar (if and only if) characterization of matrices A with $g_\infty(A)$ bounded does not seem to be in our reach.

Problem 1 Find a structural characterization of matrices A with $g_\infty(A)$ bounded.

In view of Theorem 9, it may be tempting to think that such a characterization can involve matrices with bounded incidence tree-depth as if a matrix A has bounded primal tree-depth or it has bounded dual tree-depth, then $g_\infty(A)$ is bounded. However, the following matrix A has incidence tree-depth equal to 4 and yet $g_\infty(A)$ grows with the number t of its columns; in particular, the vector $(t-1, 1, 1, \ldots, 1)$ is an element of its Graver basis as it can be readily verified. We remark that a similar matrix was used by Eiben et al. [18] in their NP-completeness argument.

$$\begin{pmatrix} 1 & -1 & -1 & -1 & \cdots & -1 & -1 \\ 0 & 1 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \ddots & & \vdots \\ 0 & 1 & 0 & 0 & \cdots & -1 & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 & -1 \end{pmatrix}$$

In Sect. 3, we have given structural characterizations of matrices that are row-equivalent to a matrix with small primal tree-depth or small incidence tree-depth,

which complements the characterization of matrices row-equivalent to a matrix with small dual tree-depth from [8, 9]. We have also presented fixed parameter algorithms (Theorems 6 and 8) for finding such a row-equivalent matrix with bounded entry complexity if one exists; both of these algorithms are based on fixed parameter algorithms for finding deletion-depth decompositions and contraction*-depth decompositions of matroids over finite fields, which are presented in Corollary 1 in the case of deletion-depth and in [8, 9] in the case of contraction*-depth. We believe that similar techniques would lead to a fixed parameter algorithm for contraction*-deletion-depth decompositions of matroids represented over a finite field (note that contraction*-deletion-depth does not have an obvious description in monadic second order logic and so the algorithmic results of Hliněný [29, 30] do not readily apply in this setting). However, it is unclear whether such an algorithm would yield a fixed parameter algorithm for rational matrices as we do not have structural results on the circuits of rational matrices with small incidence tree-depth, which would reduce the case of rational matrices to those over finite fields.

Another natural question is whether the upper bound on the depth of the principal contraction*-tree given in Theorem 10, which is quadratic in the length of the longest circuit of a represented matroid, can be improved. However, this turns out to be impossible as we now argue. Since the minimum depth of a principal contraction*-tree of a matroid M differs from $\mathrm{cd}(M)$, i.e. the minimum height of a contraction-tree of M, by at most one, it is enough to construct a sequence of matroids M_n such that

- the length of the longest circuit of M_n is is at most $\mathcal{O}(n)$, and
- the contraction-depth of M_n is at least $\Omega(n^2)$.

Hence, the quadratic dependence of the minimum depth in Theorem 10 is optimal up to a constant factor. Still, it can be the case that the bound on the contraction*-depth can be improved.

The matroids M_n are the graphic matroids of graphs G_n, which are constructed inductively. To facilitate the induction we will require slightly stronger properties. Each of the graphs G_n contains two distinguished vertices, denoted by r_n and b_n, and the following holds:

1. The length of any path in G_n between the vertices r_n and b_n is between n and $2n$.
2. The length of any circuit in M_n is at most $4n$.
3. The contraction-depth of M_n is at least $\binom{n}{2}$.

If $n = 1$, we set G_1 to be the two-vertex graph formed by two parallel edges, and r_1 and b_1 are chosen as the two vertices of G_1. Note that the graph G_1 and the matroid $M_1 = M(G_1)$ has the properties (7), (7), and (7). To obtain G_n, we start with a cycle of length $2n$ and choose any two vertices at distance n to be r_n and b_n. This cycle containing the vertices r_n and b_n will be referred to as the *root cycle*. We then add n copies of G_{n-1}, connect the vertex r_{n-1} in each copy to the vertex r_n, and connect the vertex b_{n-1} in each copy to b_n. The construction is illustrated in Fig. 7.

Assuming that the matroid M_{n-1} and the graph G_{n-1} have the properties (7), (7), and (7), and we will show that the matroid M_n and the graph G_n also have these properties.

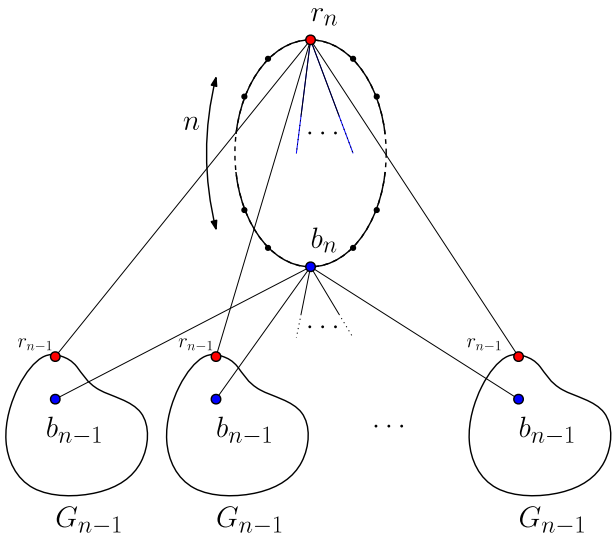


Fig. 7 The construction of the graph G_n. The vertices r_n and r_{n-1} are drawn red while b_n and b_{n-1} are drawn blue

We start with showing that G_n has the property (7). Indeed, a path from r_n to b_n is either contained in the root cycle or consists of a path between r_{n-1} and b_{n-1} in one of the copies of G_{n-1}, whose length is between $n-1$ and $2(n-1)$, together with two edges joining r_{n-1} to r_n and b_{n-1} to b_n. In either of the cases, the length of the path is between n and $2n$ as required.

Having established the property (7), we prove the property (7). Any circuit of the matroid M_n corresponds to a cycle in the graph G_n, thus we can simply investigate the lengths of cycles in G_n. First observe that a cycle of G_n contains either both vertices r_n and b_n or neither of them. Every cycle containing r_n and b_n consists of two paths between r_n and b_n and so its length is at most $4n$, and every cycle containing neither r_n nor b_n is contained entirely within a copy of G_{n-1} and so its length is at most $4(n-1) \leq 4n$.

Finally, we argue that contraction-depth of M_n is at least $\binom{n}{2}$. Recall that contracting an element of M_n corresponds to contracting the associated edge in the graph G_n, and components of a graphic matroid correspond to blocks, i.e., maximal 2-edge-connected components, of an associated graph. Since the length of any path between r_n to b_n is at least n, until at least n edge contractions are performed in graph G_n, the vertices r_n and b_n are distinct and are contained in the same block. Hence, after $n-1$ edge contractions followed by deleting all blocks not containing the vertices r_n and b_n (if such blocks appear), the graph still contains an intact copy G_{n-1}. It follows that $\mathrm{cd}(M_n) \geq (n-1) + \mathrm{cd}(M_{n-1})$, which implies that $\mathrm{cd}(M_n) \geq (n-1) + \binom{n-1}{2} = \binom{n}{2}$.

Acknowledgements All five authors would like to thank the Schloss Dagstuhl—Leibniz Center for Informatics for hospitality during the workshop "Sparsity in Algorithms, Combinatorics and Logic" in September 2021 where the work leading to the results contained in this paper was started. The authors are also indebted to the two anonymous reviewers for their detailed comments on the manuscript, which have helped to improve the presentation significantly and made the manuscript more accessible to a wider audience.

Funding Open access publishing supported by the National Technical Library in Prague.

References

1. Aschenbrenner, M., Hemmecke, R.: Finiteness theorems in stochastic integer programming. Found. Comput. Math. **7**(2), 183–227 (2007)
2. Aykanat, C., Pinar, A., Çatalyürek, Ü.V.: Permuting sparse rectangular matrices into block-diagonal form. SIAM J. Sci. Comput. **25**(6), 1860–1879 (2004)
3. Bergner, M., Caprara, A., Ceselli, A., Furini, F., Lübbecke, M.E., Malaguti, E., Traversi, E.: Automatic Dantzig–Wolfe reformulation of mixed integer programs. Math. Program. **149**(1–2), 391–424 (2015)
4. Borndörfer, R., Ferreira, C.E., Martin, A.: Decomposing matrices into blocks. SIAM J. Optim. **9**(1), 236–269 (1998)
5. Brand, C., Koutecký, M., Ordyniak, S.: Parameterized algorithms for MILPs with small treedepth. Proc. AAAI Conf. Artif. Intell. **35**(14), 12249–12257 (2021)
6. Bredereck, R., Figiel, A., Kaczmarczyk, A., Knop, D., Niedermeier, R.: High-multiplicity fair allocation made more practical. In: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS'21, pp. 260–268. International Foundation for Autonomous Agents and Multiagent Systems (2021)
7. Bredereck, R., Kaczmarczyk, A., Knop, D., Niedermeier, R.: High-multiplicity fair allocation: Lenstra empowered by n-fold integer programming. In: Proceedings of the 2019 ACM Conference on Economics and Computation, EC'19, pp. 505–523. Association for Computing Machinery (2019)
8. Chan, T.F.N., Cooper, J.W., Koutecký, M., Král', D., Pekárková, K.: Matrices of optimal tree-depth and row-invariant parameterized algorithm for integer programming. In: 47th International Colloquium on Automata, Languages, and Programming (ICALP 2020), Leibniz International Proceedings in Informatics (LIPIcs), vol. 168, pp. 26:1–26:19 (2020)
9. Chan, T.F.N., Cooper, J.W., Koutecký, M., Král', D., Pekárková, K.: Matrices of optimal tree-depth and a row-invariant parameterized algorithm for integer programming. SIAM J. Comput. **51**(3), 664–700 (2022)
10. Chen, H., Chen, L., Zhang, G.: FPT algorithms for a special block-structured integer program with applications in scheduling. preprint arXiv:2107.01373 (2021)
11. Chen, L., Marx, D.: Covering a tree with rooted subtrees–parameterized and approximation algorithms. In: 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018), pp. 2801–2820. SIAM (2018)
12. Cslovjecsek, J., Eisenbrand, F., Hunkenschröder, C., Rohwedder, L., Weismantel, R.: Block-structured integer and linear programming in strongly polynomial and near linear time. In: 32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2021), pp. 1666–1681. SIAM (2021)
13. Cslovjecsek, J., Eisenbrand, F., Pilipczuk, M., Venzin, M., Weismantel, R.: Efficient Sequential and Parallel Algorithms for Multistage Stochastic Integer Programming Using Proximity. In: 29th Annual European Symposium on Algorithms (ESA 2021), Leibniz International Proceedings in Informatics (LIPIcs), vol. 204, pp. 33:1–33:14 (2021)
14. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer, Berlin (2015)
15. De Loera, J.A., Hemmecke, R., Onn, S., Weismantel, R.: N-fold integer programming. Discret. Optim. **5**(2), 231–241 (2008)
16. DeVos, M., Kwon, O., Oum, S.: Branch-depth: Generalizing tree-depth of graphs. Eur. J. Comb. **90**, 103186 (2020)

17. Ding, G., Oporowski, B., Oxley, J.: On infinite antichains of matroids. J. Comb. Theory Ser. B **63**(1), 21–40 (1995)
18. Eiben, E., Ganian, R., Knop, D., Ordyniak, S., Pilipczuk, M., Wrochna, M.: Integer programming and incidence tree depth. In: Integer Programming and Combinatorial Optimization—20th International Conference (IPCO), LNCS vol. 11480, pp. 194–204. Springer International Publishing (2019)
19. Eisenbrand, F., Hunkenschröder, C., Klein, K.: Faster algorithms for integer programs with block structure. In: 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018), Leibniz International Proceedings in Informatics (LIPIcs), pp. 49:1–49:13 (2018)
20. Eisenbrand, F., Hunkenschröder, C., Klein, K., Koutecký, M., Levin, A., Onn, S.: An algorithmic theory of integer programming. preprint arXiv:1904.01361 (2019)
21. Ekbatani, F., Natura, B., Végh, L.A.: Circuit imbalance measures and linear programming. preprint arXiv:2108.03616 (2021)
22. Ferris, M.C., Horn, J.D.: Partitioning mathematical programs for parallel solution. Math. Program. **80**(1), 35–61 (1998)
23. Gamrath, G., Lübbecke, M.E.: Experiments with a generic Dantzig–Wolfe decomposition for integer programs. Exp. Algorithms **6049**, 239–252 (2010)
24. Ganian, R., Ordyniak, S.: The complexity landscape of decompositional parameters for ILP. Artif. Intell. **257**, 61–71 (2018)
25. Halmos, P.: Finite-Dimensional Vector Spaces. Undergraduate Texts in Mathematics, Springer, Berlin (1993)
26. Hemmecke, R., Onn, S., Romanchuk, L.: N-fold integer programming in cubic time. Math. Program. **137**, 325–341 (2013)
27. Hemmecke, R., Schultz, R.: Decomposition of test sets in stochastic integer programming. Math. Program. **94**, 323–341 (2003)
28. Hermelin, D., Molter, H., Niedermeier, R., Shabtay, D.: Equitable scheduling for the total completion time objective. preprint arXiv:2112.13824 (2021)
29. Hliněný, P.: Branch-width, parse trees, and monadic second-order logic for matroids. In: H. Alt, M. Habib (eds.) 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS), LNCS, vol. 2607, pp. 319–330 (2003)
30. Hliněný, P.: Branch-width, parse trees, and monadic second-order logic for matroids. J. Comb. Theory Ser. B **96**(3), 325–351 (2006)
31. Jansen, K., Klein, K., Lassota, A.: The double exponential runtime is tight for 2-stage stochastic ILPs. In: M. Singh, D.P. Williamson (eds.) Integer Programming and Combinatorial Optimization—22nd International Conference (IPCO), LNCS vol. 12707, Lecture Notes in Computer Science, vol. 12707, pp. 297–310. Springer (2021)
32. Jansen, K., Klein, K., Maack, M., Rau, M.: Empowering the configuration-IP: new PTAS results for scheduling with setup times. Math. Program. (2021)
33. Jansen, K., Lassota, A., Maack, M.: Approximation algorithms for scheduling with class constraints. In: Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures, pp. 349–357. Association for Computing Machinery (2020)
34. Jansen, K., Lassota, A., Rohwedder, L.: Near-linear time algorithm for n-fold ILPs via color coding. SIAM J. Discret. Math. **34**(4), 2282–2299 (2020)
35. Kannan, R.: Minkowski's convex body theorem and integer programming. Math. Oper. Res. **12**(3), 415–440 (1987)
36. Kardoš, F., Král', D., Liebenau, A., Mach, L.: First order convergence of matroids. Eur. J. Comb. **59**, 150–168 (2017)
37. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of Computer Computations. The IBM Research Symposia Series, pp. 85–103. Springer, Berlin (1972)
38. Khaniyev, T., Elhedhli, S., Erenay, F.S.: Structure detection in mixed-integer programs. INFORMS J. Comput. **30**(3), 570–587 (2018)
39. Klein, K.: About the complexity of two-stage stochastic IPs. Math. Program. **192**(1), 319–337 (2022)
40. Klein, K., Reuter, J.: Collapsing the tower—on the complexity of multistage stochastic IPs. In: 33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2022), pp. 348–358. SIAM (2022)
41. Knop, D., Koutecký, M.: Scheduling kernels via configuration LP. preprint arXiv:2003.02187 (2018)
42. Knop, D., Koutecký, M.: Scheduling meets n-fold integer programming. J. Sched. **21**(5), 493–503 (2018)

43. Knop, D., Koutecký, M., Mnich, M.: Combinatorial n-fold integer programming and applications. In: 25th Annual European Symposium on Algorithms (ESA), Leibniz International Proceedings in Informatics (LIPIcs), vol. 87, pp. 54:1–54:14 (2017)
44. Koutecký, M., Levin, A., Onn, S.: A parameterized strongly polynomial algorithm for block structured integer programs. In: 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018), Leibniz International Proceedings in Informatics (LIPIcs), vol. 107, pp. 85:1–85:14 (2018)
45. Lenstra, H.W., Jr.: Integer programming with a fixed number of variables. Math. Oper. Res. **8**(4), 538–548 (1983)
46. Oxley, J.: Matroid Theory. Oxford Graduate Texts in Mathematics, Oxford University Press, Oxford (2011)
47. Pothen, A.: The complexity of optimal elimination trees. Technical Report CS-88-13, Pennsylvania State University (1988)
48. Schultz, R., Stougie, L., van der Vlerk, M.H.: Solving stochastic programs with integer recourse by enumeration: a framework using gröbner basis reductions. Math. Program. **83**, 229–252 (1998)
49. Vanderbeck, F., Wolsey, L.A.: Reformulation and decomposition of integer programs. In: 50 Years of Integer Programming 1958–2008, pp. 431–502. Springer (2010)
50. Wang, J., Ralphs, T.: Computational experience with hypergraph-based methods for automatic decomposition in discrete optimization. In: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, pp. 394–402. Springer (2013)
51. Weil, R.L., Kettler, P.C.: Rearranging matrices to block-angular form for decomposition (and other) algorithms. Manag. Sci. **18**(1), 98–108 (1971)

Authors and Affiliations

Marcin Briański[1] · Martin Koutecký[2] · Daniel Král'[3] · Kristýna Pekárková[3] · Felix Schröder[4]

Marcin Briański
marcin.brianski@doctoral.uj.edu.pl

Martin Koutecký
koutecky@iuuk.mff.cuni.cz

Daniel Král'
dkral@fi.muni.cz

Felix Schröder
schroder@kam.mff.cuni.cz

1 Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

2 Computer Science Institute, Charles University, Prague, Czech Republic

3 Faculty of Informatics, Masaryk University, Brno, Czech Republic

4 Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Parameterized algorithms for block-structured integer programs with large entries*

Jana Cslovjecsek† Martin Koutecký‡ Alexandra Lassota§ Michał Pilipczuk¶ Adam Polak‖

Abstract

We study two classic variants of block-structured integer programming. *Two-stage stochastic programs* are integer programs of the form $\{A_i\mathbf{x} + D_i\mathbf{y}_i = \mathbf{b}_i \text{ for all } i = 1, \ldots, n\}$, where A_i and D_i are bounded-size matrices. Intuitively, this form corresponds to the setting when after setting a small set of global variables $\mathbf{x}$, the program can be decomposed into a possibly large number of bounded-size subprograms. On the other hand, *n-fold programs* are integer programs of the form $\{\sum_{i=1}^{n} C_i\mathbf{y}_i = \mathbf{a} \text{ and } D_i\mathbf{y}_i = \mathbf{b}_i \text{ for all } i = 1, \ldots, n\}$, where again C_i and D_i are bounded-size matrices. This form is natural for knapsack-like problems, where we have a large number of variables partitioned into small-size groups, each group needs to obey some set of local constraints, and there are only a few global constraints that link together all the variables.

A line of recent work established that the optimization problem for both two-stage stochastic programs and n-fold programs is fixed-parameter tractable when parameterized by the dimensions of relevant matrices A_i, C_i, D_i and by the maximum absolute value of any entry appearing in the constraint matrix. A fundamental tool used in these advances is the notion of the *Graver basis* of a matrix, and this tool heavily relies on the assumption that all the entries of the constraint matrix are bounded.

In this work, we prove that the parameterized tractability results for two-stage stochastic and n-fold programs persist even when one allows large entries in the global part of the program. More precisely, we prove the following: In this work, we prove that the parameterized tractability results for two-stage stochastic and n-fold programs persist even when one allows large entries in the global part of the program. More precisely, we prove the following:

- The feasibility problem for two-stage stochastic programs is fixed-parameter tractable when parameterized by the dimensions of matrices A_i, D_i and by the maximum absolute value of the entries of matrices D_i. That is, we allow matrices A_i to have arbitrarily large entries.
- The linear optimization problem for n-fold integer programs that are uniform – all matrices C_i are equal – is fixed-parameter tractable when parameterized by the dimensions of matrices C_i and D_i and by the maximum absolute value of the entries of matrices D_i. That is, we require that $C_i = C$ for all $i = 1, \ldots, n$, but we allow C to have arbitrarily large entries.

In the second result, the uniformity assumption is necessary; otherwise the problem is NP-hard already when the parameters take constant values. Both our algorithms are weakly polynomial: the running time is measured in the total bitsize of the input.

In both results, we depart from the approach that relies purely on Graver bases. Instead, for two-stage stochastic programs, we devise a reduction to integer programming with a bounded number of variables using new insights about the combinatorics of integer cones. For n-fold programs, we reduce a given n-fold program to an exponential-size program with bounded right-hand sides, which we consequently solve using a reduction to mixed integer programming with a bounded number of integral variables.

*The full version of the paper can be accessed at https://arxiv.org/abs/2311.01890. Martin Koutecký was partially supported by Charles University project UNCE/SCI/004 and by the project 22-22997S of GA ČR. The work of Michał Pilipczuk on this manuscript is a part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, Grant Agreement No 948057 — BOBR. Part of this work was done when Adam Polak and Alexandra Lassota were at École Polytechnique Fédérale de Lausanne, supported by the Swiss National Science Foundation projects *Lattice Algorithms and Integer Programming* (185030) and *Complexity of Integer Programming* (CRFS-2_207365). Part of this work was carried out while Alexandra Lassota was affiliated with Max Planck Institute for Informatics, SIC, Saarbrücken, Germany.

†École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
‡Computer Science Institute, Charles University, Prague, Czech Republic
§Eindhoven University of Technology, Eindhoven, The Netherlands
¶Institute of Informatics, University of Warsaw, Poland
‖Department of Computing Sciences, Bocconi University, Milan, Italy

1 Introduction

We study two variants of integer programming problems, where the specific structure of the constraint matrix can be exploited for the design of efficient parameterized algorithms. *Two-stage stochastic programs* are integer programs of the following form:

$$\begin{aligned} \mathbf{x} \in \mathbb{Z}^k_{\geqslant 0},\ \mathbf{y}_i \in \mathbb{Z}^k_{\geqslant 0}, \qquad & \text{and} \\ A_i\mathbf{x} + D_i\mathbf{y}_i = \mathbf{b}_i \qquad & \text{for all } i = 1, 2, \ldots, n. \end{aligned} \tag{♠}$$

Here, A_i, D_i are integer $k \times k$ matrices[1] and each $\mathbf{b}_i$ is an integer vector of length k. This form arises naturally when the given integer program can be decomposed into multiple independent subprograms on disjoint variable sets $\mathbf{y}_i$, except there are several global variables $\mathbf{x}$ that are involved in all the subprograms and thus link them. See the survey of Shultz et al. [41] as well as an exposition article by Gavenčiak et al. [19] for examples of applications.

We moreover study *n-fold programs* which are integer programs of the form

$$\begin{aligned} \mathbf{y}_i \in \mathbb{Z}^k_{\geqslant 0}, & \\ \sum_{i=1}^{n} C_i\mathbf{y}_i = \mathbf{a}, \qquad & \text{and} \\ D_i\mathbf{y}_i = \mathbf{b}_i \qquad & \text{for all } i = 1, 2, \ldots, n, \end{aligned} \tag{♣}$$

where again C_i, D_i are integer $k \times k$ matrices and $\mathbf{a}, \mathbf{b}_i$ are integer vectors of length k. These kind of programs appear for knapsack-like and scheduling problems, where blocks of variables $\mathbf{y}_i$ correspond to some independent local decisions (for instance, whether to pack an item into the knapsack or not), and there are only a few linear constraints that involve all variables (for instance, that the capacity of the knapsack is not exceeded). See [9, 14, 19, 23, 28, 30, 32, 33] for examples of n-fold programs appearing naturally "in the wild". Figure 1 depicts constraint matrices of two-stage stochastic and n-fold programs.

$$\begin{bmatrix} A_1 & D_1 & & & \\ A_2 & & D_2 & & \\ \vdots & & & \ddots & \\ A_n & & & & D_n \end{bmatrix} \qquad \begin{bmatrix} C_1 & C_2 & \ldots & C_n \\ D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_n \end{bmatrix} \qquad \begin{bmatrix} B & C_1 & C_2 & \ldots & C_n \\ A_1 & D_1 & & & \\ A_2 & & D_2 & & \\ \vdots & & & \ddots & \\ A_n & & & & D_n \end{bmatrix}$$

Figure 1: Constraint matrices in two-stage stochastic, n-fold, and 4-block integer programs, respectively. (The last kind will be discussed later.) Every block is a $k \times k$ matrix, where k is the parameter. Empty spaces denote blocks of zeroes.

Both for two-stage stochastic programs and for n-fold programs, we can consider two computational problems. The simpler *feasibility problem* just asks whether the given program has a *solution*: an evaluation of variables in nonnegative integers that satisfies all the constraints. In the harder *(linear) optimization problem*, we are additionally given an integer weight w_x for every variable x appearing in the program, and the task is to minimize $\sum_{x\colon \text{variable}} w_x \cdot x$ over all solutions.

Two-stage stochastic and n-fold programs have recently gathered significant interest in the theoretical community for two reasons. On one hand, it turns out that for multiple combinatorial problems, their natural integer programming formulations take either of the two forms. On the other hand, one can actually design highly non-trivial fixed-parameter algorithms for the optimization problem for both two-stage stochastic and n-fold programs; we will review them in a minute. Combining this two points yields a powerful algorithmic technique that allowed multiple new tractability results and running times improvements for various problems of combinatorial optimization; see [9, 19, 23, 28, 29, 30, 32, 33, 37] for examples.

Delving more into technical details, if by Δ we denote the maximum absolute value of any entry in the constraint matrix, then the optimization problem for

[1] Reliance on square matrices is just for convenience of presentation. The setting where blocks are rectangular matrices with dimensions bounded by k can be reduced to the setting of $k \times k$ square matrices by just padding with 0s.

- two-stage stochastic programs (♠) can be solved in time $2^{\Delta^{\mathcal{O}(k^2)}} \cdot n \log^{\mathcal{O}(k^2)} n$ [11]; and
- n-fold programs (♣) can be solved in time $(k\Delta)^{\mathcal{O}(k^3)} \cdot n \log^{\mathcal{O}(k^2)} n$ [10].

The results above are in fact pinnacles of an over-a-decade-long sequence of developments, which gradually improved both the generality of the results and the running times [2, 10, 11, 14, 15, 16, 21, 24, 26, 35], as well as provided complexity lower bounds [22, 34]. We refer the interested reader to the monumental manuscript of Eisenbrand et al. [16] which provides a comprehensive perspective on this research area.

We remark that the tractability results presented above can be also extended to the setting where the global-local block structure present in two-stage stochastic and n-fold programs can be iterated further, roughly speaking to trees of bounded depth. This leads to the study of integer programs with bounded *primal* or *dual treedepth*, for which analogous tractability results have been established. Since these notions will not be of interest in this work, we refrain from providing further details and refer the interested reader to the works relevant for this direction [3, 4, 8, 10, 11, 15, 16, 26, 27, 34, 35].

All the abovementioned works, be it on two-stage stochastic or n-fold programs, or on programs of bounded primal or dual treedepth, crucially rely on one tool: the notion of the *Graver basis* of a matrix. Intuitively speaking, the Graver basis of a matrix A consists of "minimal steps" within the lattice of integer points belonging to the kernel of A. Therefore, the maximum norm of an element of the Graver basis reflects the "granularity" of this lattice. And so, the two fundamental observations underlying all the discussed developments are the following:

- in two-stage stochastic matrices (or more generally, matrices of bounded primal treedepth), the ℓ_∞ norm of the elements of the Graver basis is bounded in terms of k (the dimension of every block) and the maximum absolute value of any entry appearing in the matrix (see [16, Lemma 28]); and
- an analogous result holds for n-fold matrices (or more generally, matrices of bounded dual treedepth) and the ℓ_1 norm (see [16, Lemma 30]).

Based on these observations, various algorithmic strategies, including augmentation frameworks [24, 35] and proximity arguments [10, 11, 15], can be employed to solve respective integer programs.

The drawback of the Graver-based approach is that it heavily relies on the assumption that all the entries of the input matrices are (parametrically) bounded. Indeed, the norms of the elements of the Graver basis are typically at least as large as the entries, so lacking any upper bound on the latter renders the methodology inapplicable. This is in stark contrast with the results on fixed-parameter tractability of integer programming parameterized by the number of variables [12, 13, 18, 25, 38, 40], which rely on different tools and for which no bound on the absolute values of the entries is required. In fact, two-stage stochastic programs generalize programs with a bounded number of variables (just do not use variables $\mathbf{y}_i$), yet the current results for two-stage stochastic programs do *not* generalize those for integer programs with few variables, because they assume a bound on the absolute values of the entries.

The goal of this paper is to understand to what extent one can efficiently solve two-stage stochastic and n-fold programs while allowing large entries on input.

Our contribution. We prove that both the feasibility problem for two-stage stochastic programs and the optimization problem for uniform n-fold programs (that is, where $C_1 = C_2 = \ldots = C_n$) can be solved in fixed-parameter time when parameterized by the dimensions of the blocks and the maximum absolute value of any entry appearing in the diagonal blocks D_i. That is, we allow the entries of the global blocks (A_i and C_i, respectively) to be arbitrarily large, and in the case of n-fold programs, we require that all blocks C_i are equal. The statements below summarize our results. ($\|P\|$ denotes the total bitsize of a program P.)

THEOREM 1.1. *The feasibility problem for two-stage stochastic programs P of the form (♠) can be solved in time $f(k, \max_i \|D_i\|_\infty) \cdot \|P\|$ for a computable function f, where k is the dimension of all the blocks A_i, D_i.*

THEOREM 1.2. *The optimization problem for n-fold programs P of the form (♣) that are uniform (that is, satisfy $C_1 = \ldots = C_n$) can be solved in time $f(k, \max_i \|D_i\|_\infty) \cdot \|P\|^{\mathcal{O}(1)}$ for a computable function f, where k is the dimension of all the blocks C_i, D_i.*

The uniformity condition in Theorem 1.2 is necessary (unless $\mathsf{P} = \mathsf{NP}$), as one can very easily reduce SUBSET SUM to the feasibility problem for n-fold programs with $k = 2$ and D_i being $\{0, 1\}$-matrices. Indeed, given an instance of SUBSET SUM consisting of positive integers $a_1, \ldots, a_n$ and a target integer t, we can write the following n-fold program on variables $y_1, \ldots, y_n, y'_1, \ldots, y'_n \in \mathbb{Z}_{\geqslant 0}$: $y_i + y'_i = 1$ for all $i = 1, \ldots, n$ and $\sum_{i=1}^n a_i y_i = t$. We remark that uniform n-fold programs are actually of the highest importance, as this form typically arises in

applications. In fact, many of the previous works named such problems "n-fold", while our formulation (♣) would be called "generalized n-fold".

We also remark that the algorithm of Theorem 1.2 does not use the assumption that the number of *rows* of matrix C is bounded by k (formally, in the formal statement proved in the full version of this work, we do not consider this number among parameters). However, we stress that the bound on the number of *columns* of C is heavily exploited, which sets our approach apart from many of the previous works [10, 15, 35].

Further, observe that Theorem 1.1 applies only to the feasibility problem for two-stage stochastic programs. We actually do not know whether Theorem 1.1 can be extended to the optimization problem as well, and we consider determining this an outstanding open problem. We will discuss its motivation in more details later on. Also, we remark that Theorem 1.1 seems to be the first algorithm for feasibility of two-stage stochastic programs that achieves truly linear dependence of the running time on the total input size; the earlier algorithms of [11, 35] had at least some additional polylogarithmic factors.

Finally, note that the algorithms of Theorems 1.1 and 1.2 are not strongly polynomial (i.e., the running time depends on the total bitsize of the input, rather than is counted in the number of arithmetic operations), while the previous algorithms of [10, 11, 15, 35] for the stronger parameterization are. At least in the case of Theorem 1.1, this is justified, as the problem considered there generalizes integer programming parameterized by the number of variables, for which strongly polynomial FPT algorithms are not known.

Not surprisingly, the proofs of Theorems 1.1 and 1.2 depart from the by now standard approach through Graver bases; they are based on entirely new techniques, with some key Graver-based insight needed in the case of Theorem 1.2. In both cases, the problem is ultimately reduced to (mixed) integer programming with a bounded number of (integral) variables, and this allows us to cope with large entries on input. We expand the discussion of our techniques in Section 2, which contains a technical overview of the proofs.

4-block programs. Finally, we would like to discuss another motivation for investigating two-stage stochastic and n-fold programs with large entries, namely the open question about the parameterized complexity of 4-*block integer programming*. 4-block programs are programs in which the constraint matrix has the block-structured form depicted in the right panel of Figure 1; note that this form naturally generalizes both two-stage stochastic and n-fold programs. It is an important problem in the area to determine whether the feasibility problem for 4-block programs can be solved in fixed-parameter time when parameterized by the dimension of blocks k and the maximum absolute value of any entry in the input matrix. The question was asked by Hemmecke et al. [20], who proposed an XP algorithm for the problem. Improvements on the XP running time were reported by Chen et al. [7], and FPT algorithms for special cases were proposed by Chen et al. [5]; yet no FPT algorithm for the problem in full generality is known so far. We remark that recently, Chen et al. [6] studied the complexity of 4-block programming while allowing large entries in all the four blocks of the matrix. They showed that then the problem becomes NP-hard already for blocks of constant dimension, and they discussed a few special cases that lead to tractability.

We observe that in the context of the feasibility problem for uniform 4-block programs (i.e., with $A_i = A$ and $C_i = C$ for all $i = 1, \ldots, n$), it is possible to emulate large entries within the global blocks A, B, C using only small entries at the cost of adding a bounded number of auxiliary variables. This yields the following reduction, whose proof can be found in the full version of this work.

Observation 1. *Suppose the feasibility problem for uniform 4-block programs can be solved in time $f(k, \Delta) \cdot \|P\|^{\mathcal{O}(1)}$ for some computable function f, where k is the dimension of every block and Δ is the maximum absolute value of any entry in the constraint matrix. Then the feasibility problem for uniform 4-block programs can be also solved in time $g(k, \max_i \|D_i\|_\infty) \cdot \|P\|^{\mathcal{O}(1)}$ for some computable function g under the assumption that all the absolute values of the entries in matrices A, B, C are bounded by n.*

Consequently, to approach the problem of fixed-parameter tractability of 4-block integer programming, it is imperative to understand first the complexity of two-stage stochastic and n-fold programming with large entries allowed in the global blocks. And this is precisely what we do in this work.

We believe that the next natural step towards understanding the complexity of 4-block integer programming would be to extend Theorem 1.1 to the optimization problem; that is, to determine whether optimization of two-stage stochastic programs can be solved in fixed-parameter time when parameterized by k and $\max_i \|D_i\|_\infty$. Indeed, lifting the result from feasibility to the optimization problem roughly corresponds to adding a single constraint that links all the variables, and 4-block programs differ from two-stage stochastic programs precisely

in that there may be up to k such additional linking constraints. Thus, we hope that the new approach to block-structured integer programming presented in this work may pave the way towards understanding the complexity of solving 4-block integer programs.

Acknowledgements. This research has been initiated during the trimester on Discrete Optimization at the Hausdorff Research Institute for Mathematics (HIM) in Bonn, Germany. We thank the organizers of the trimester for creating a friendly and motivating research environment. We also thank Eleonore Bach, Fritz Eisenbrand, and Robert Weismantel, for pointing us to the work of Aliev and Henk [1].

2 Overview

In this section we provide a technical overview of our results aimed at presenting the main ideas and new conceptual contributions. Complete and formal proofs of all our results can be found in the full version of this paper.

2.1 Two-stage stochastic programming.

We start with an overview on the proof of Theorem 1.1. We will heavily rely on the combinatorics of integer and polyhedral cones, so let us recall basic definitions and properties.

Cones. Consider an integer matrix D with t columns and k rows. The *polyhedral cone* spanned by D is the set $\mathsf{cone}(D) \coloneqq \{D\mathbf{y} \colon \mathbf{y} \in \mathbb{R}^t_{\geqslant 0}\} \subseteq \mathbb{R}^k_{\geqslant 0}$, or equivalently, the set of all vectors in $\mathbb{R}^k_{\geqslant 0}$ expressible as nonnegative combinations of the columns of D. Within the polyhedral cone, we have the *integer cone* where we restrict attention to nonnegative integer combinations: $\mathsf{intCone}(D) \coloneqq \{D\mathbf{y} \colon \mathbf{y} \in \mathbb{Z}^t_{\geqslant 0}\} \subseteq \mathbb{Z}^k$. Finally, the *integer lattice* is the set $\mathsf{lattice}(D) \coloneqq \{D\mathbf{y} \colon \mathbf{y} \in \mathbb{Z}^t\} \subseteq \mathbb{Z}^k$ which comprises all integer combinations of columns of D with possibly negative coefficients.

Clearly, not every integer vector in $\mathsf{cone}(D)$ has to belong to $\mathsf{intCone}(D)$. It is not even necessarily the case that $\mathsf{intCone}(D) = \mathsf{cone}(D) \cap \mathsf{lattice}(D)$, as there might be vectors that can be obtained both as a nonnegative combination and as an integer combination of columns of D, but every such integer combination must necessarily contain negative coefficients. To see an example, note that in dimension $k = 1$, this is the Frobenius problem: supposing all entries of D are positive integers, the elements of $\mathsf{intCone}(D)$ are essentially all nonnegative numbers divisible by the gcd (greatest common divisor) of the entries of D, except that for small numbers there might be some aberrations: a positive integer of order $\mathcal{O}(\|D\|^2_\infty)$ may not be presentable as a nonnegative combination of the entries of D, even assuming it is divisible by the gcd of the entries of D.

However, the Frobenius example suggests that the equality $\mathsf{intCone}(D) = \mathsf{cone}(D) \cap \mathsf{lattice}(D)$ is almost true, except for aberrations near the boundary of $\mathsf{cone}(D)$. We forge this intuition into a formal statement presented below that says roughly the following: if one takes a look at $\mathsf{intCone}(D)$ at a large scale, by restricting attention to integer vectors $\mathbf{v} \in \mathbb{Z}^k$ with fixed remainders of entries modulo some large integer B, then $\mathsf{intCone}(D)$ behaves like a polyhedron. In the following, for a positive integer B and a vector $\mathbf{r} \in \{0, 1, \ldots, B-1\}^k$, we let $\Lambda^B_{\mathbf{r}}$ be the set of all vectors $\mathbf{v} \in \mathbb{Z}^k$ such that $\mathbf{v} \equiv \mathbf{r} \bmod B$, which means $v_i \equiv r_i \bmod B$ for all $i \in \{1, \ldots, k\}$.

Theorem 2.1. (Reduction to Polyhedral Constraints) *Let D be an integer matrix with t columns and k rows. Then there exists a positive integer B, computable from D, such that for every $\mathbf{r} \in \{0, 1, \ldots, B-1\}^k$, there exists a polyhedron $\mathcal{Q}_{\mathbf{r}}$ such that*

$$\Lambda^B_{\mathbf{r}} \cap \mathsf{intCone}(D) = \Lambda^B_{\mathbf{r}} \cap \mathcal{Q}_{\mathbf{r}}.$$

Moreover, a representation of such a polyhedron $\mathcal{Q}_{\mathbf{r}}$ can be computed given D and $\mathbf{r}$.

In other words, Theorem 2.1 states that if one fixes the remainders of entries modulo B, then membership in the integer cone can be equivalently expressed through a finite system of linear inequalities. Before we sketch the proof of Theorem 2.1, let us discuss how to use this to solve two-stage stochastic programs.

The algorithm. Consider a two-stage stochastic program $P = (A_i, D_i, \mathbf{b}_i \colon i \in \{1, \ldots, n\})$ such that blocks A_i, D_i are integer $k \times k$ matrices and all entries of blocks D_i are bounded in absolute value by Δ. The feasibility problem for P can be understood as the question about satisfaction of the following sentence, where all quantifications range over $\mathbb{Z}^k_{\geqslant 0}$:

$$\exists_{\mathbf{x}} \left(\bigwedge_{i=1}^{n} \exists_{\mathbf{y}_i} \; A_i\mathbf{x} + D_i\mathbf{y}_i = \mathbf{b}_i \right), \quad \text{or equivalently,} \quad \exists_{\mathbf{x}} \left(\bigwedge_{i=1}^{n} \; \mathbf{b}_i - A_i\mathbf{x} \in \mathsf{intCone}(D_i) \right). \tag{2.1}$$

Applying Theorem 2.1 to each matrix D_i yields a positive integer B_i. Note that there are only at most $(2\Delta+1)^{k^2}$ different matrices D_i appearing in P, which also bounds the number of different integers B_i. By replacing all B_is

with their least common multiple, we may assume that $B_1 = B_2 = \ldots = B_n = B$. Note that B is bounded by a computable function of Δ and k.

Consider a hypothetical solution $\mathbf{x}, (\mathbf{y}_i \colon i \in \{1, \ldots, n\})$ to P. We guess, by branching into B^k possibilities, a vector $\mathbf{r} \in \{0, 1, \ldots, B-1\}^k$ such that $\mathbf{x} \equiv \mathbf{r} \bmod B$. Having fixed $\mathbf{r}$, we know how the vectors $\mathbf{b}_i - A_i\mathbf{x}$ look like modulo B, hence by Theorem 2.1, we may replace the assertion $\mathbf{b}_i - A_i\mathbf{x} \in \mathsf{intCone}(D_i)$ with the assertion $\mathbf{b}_i - A_i\mathbf{x} \in \mathcal{Q}_{\mathbf{r}_i}$, where $\mathbf{r}_i \in \{0, 1, \ldots, B-1\}^k$ is the unique vector such that $\mathbf{b}_i - A_i\mathbf{r} \equiv \mathbf{r}_i \bmod B$. Thus, (2.1) can be rewritten to the sentence

$$\bigvee_{\mathbf{r} \in \{0,1,\ldots,B-1\}^k} \exists_{\mathbf{x}} \ (\mathbf{x} \equiv \mathbf{r} \bmod B) \wedge \left(\bigwedge_{i=1}^{n} \mathbf{b}_i - A_i\mathbf{x} \in \mathcal{Q}_{\mathbf{r}_i} \right),$$

which is equivalent to

$$\bigvee_{\mathbf{r} \in \{0,1,\ldots,B-1\}^k} \exists_{\mathbf{x}} \exists_{\mathbf{z}} \ (\mathbf{x} = B \cdot \mathbf{z} + \mathbf{r}) \wedge \left(\bigwedge_{i=1}^{n} \mathbf{b}_i - A_i\mathbf{x} \in \mathcal{Q}_{\mathbf{r}_i} \right). \tag{2.2}$$

Verifying satisfiability of (2.2) boils down to solving B^k integer programs on $2k$ variables $\mathbf{x}$ and $\mathbf{z}$ and linearly FPT many constraints, which can be done in linear fixed-parameter time using standard algorithms, for instance that of Kannan [25].

We remark that the explanation presented above highlights that Theorem 2.1 can be understood as a quantifier elimination result in the arithmetic theory of integers. This may be of independent interest, but we do not pursue this direction in this work.

Reduction to polyhedral constraints. We are left with sketching the proof of Theorem 2.1. Let $\mathcal{Z} \coloneqq \Lambda^B_{\mathbf{r}} \cap \mathsf{intCone}(D)$. Our goal is to understand that $\mathcal{Z}$ can be expressed as the points of $\Lambda^B_{\mathbf{r}}$ that are contained in some polyhedron $\mathcal{Q} = \mathcal{Q}_{\mathbf{r}}$.

The first step is to understand $\mathsf{cone}(D)$ itself as a polyhedron. This understanding is provided by a classic theorem of Weyl [42]: given D, one can compute a set of integer vectors $\mathcal{F} \subseteq \mathbb{Z}^k$ such that

$$\mathsf{cone}(D) = \{\mathbf{v} \in \mathbb{R}^k \mid \langle \mathbf{f}, \mathbf{v} \rangle \geqslant 0 \text{ for all } \mathbf{f} \in \mathcal{F}\}.$$

Here, $\langle \cdot, \cdot \rangle$ denotes the scalar product in $\mathbb{R}^k$. We will identify vectors $\mathbf{f} \in \mathcal{F}$ with their associated linear functionals $\mathbf{v} \mapsto \langle \mathbf{f}, \mathbf{v} \rangle$. Thus, $\mathsf{cone}(D)$ comprises all vectors $\mathbf{v}$ that have nonnegative evaluations on all functionals in $\mathcal{F}$. It is instructive to also think of the elements of $\mathcal{F}$ as of the facets of $\mathsf{cone}(D)$ understood as a polyhedron, where the functional associated with $\mathbf{f} \in \mathcal{F}$ measures the distance from the corresponding facet.

Recall that in the context of Theorem 2.1, we consider vectors of $\Lambda^B_{\mathbf{r}}$, that is, vectors $\mathbf{v} \in \mathbb{Z}^k$ such that $\mathbf{v} \equiv \mathbf{r} \bmod B$. Then $\langle \mathbf{f}, \mathbf{v} \rangle \equiv \langle \mathbf{f}, \mathbf{r} \rangle \bmod B$ for every $\mathbf{f} \in \mathcal{F}$, hence we can find a unique integer $p_{\mathbf{f}} \in \{0, 1, \ldots, B-1\}$, $p_{\mathbf{f}} \equiv \langle \mathbf{f}, \mathbf{r} \rangle \bmod B$, such that $\langle \mathbf{f}, \mathbf{v} \rangle \equiv p_{\mathbf{f}} \bmod B$ for all $\mathbf{v} \in \Lambda^B_{\mathbf{r}}$. Now $\langle \mathbf{f}, \mathbf{v} \rangle$ is also nonnegative provided $\mathbf{v} \in \mathsf{cone}(D)$, hence

$$\langle \mathbf{f}, \mathbf{v} \rangle \in \{p_{\mathbf{f}}, p_{\mathbf{f}} + B, p_{\mathbf{f}} + 2B, \ldots\} \qquad \text{for all } \mathbf{f} \in \mathcal{F} \text{ and } \mathbf{v} \in \Lambda^B_{\mathbf{r}} \cap \mathsf{cone}(D).$$

Now comes the key distinction about the behavior of $\mathbf{v} \in \Lambda^B_{\mathbf{r}} \cap \mathsf{cone}(D)$ with respect to $\mathbf{f} \in \mathcal{F}$: we say that $\mathbf{f}$ is *tight* with respect to $\mathbf{v}$ if $\langle \mathbf{f}, \mathbf{v} \rangle = p_{\mathbf{f}}$, and is *not tight* otherwise, that is, if $\langle \mathbf{f}, \mathbf{v} \rangle \geqslant p_{\mathbf{f}} + B$. Recall that in the context of Theorem 2.1, we are eventually free to choose B to be large enough. Intuitively, this means that if $\mathbf{f}$ is not tight for $\mathbf{v}$, then $\mathbf{v}$ lies far from the facet corresponding to $\mathbf{f}$ and there is a very large slack in the constraint posed by $\mathbf{f}$ understood as a functional. On the other hand, if $\mathbf{f}$ is tight with respect to $\mathbf{v}$, then $\mathbf{v}$ is close to the boundary of $\mathsf{cone}(D)$ at the facet corresponding to $\mathbf{f}$, and there is a potential danger of observing Frobenius-like aberrations at $\mathbf{v}$.

Thus, the set $\mathcal{R} \coloneqq \Lambda^B_{\mathbf{r}} \cap \mathsf{cone}(D)$ can be partitioned into subsets $\{\mathcal{R}_{\mathcal{G}} \colon \mathcal{G} \subseteq \mathcal{F}\}$ defined as follows: $\mathcal{R}_{\mathcal{G}}$ comprises all vectors $\mathbf{v} \in \mathcal{R}$ such that $\mathcal{G}$ is exactly the set of functionals $\mathbf{f} \in \mathcal{F}$ that are tight with respect to $\mathbf{v}$. Our goal is to prove that each set $\mathcal{R}_{\mathcal{G}}$ behaves uniformly with respect to $\mathcal{Z}$: it is either completely disjoint or completely contained in $\mathcal{Z}$. To start the discussion, let us look at the particular case of $\mathcal{R}_{\mathcal{G}}$ for $\mathcal{G} = \emptyset$. These are vectors that are deep inside $\mathsf{cone}(D)$, for which no functional in $\mathcal{F}$ is tight. For these vectors, we use the following lemma, which is the cornerstone of our proof.

LEMMA 2.1. (DEEP-IN-THE-CONE LEMMA, SIMPLIFIED VERSION) *There exists a constant M, depending only on D, such that the following holds. Suppose $\mathbf{v} \in \mathsf{cone}(D) \cap \mathbb{Z}^k$ is such that $\langle \mathbf{f}, \mathbf{v} \rangle > M$ for all $\mathbf{f} \in \mathcal{F}$. Then $\mathbf{v} \in \mathsf{intCone}(D)$ if and only if $\mathbf{v} \in \mathsf{lattice}(D)$.*

Proof. The left-to-right implication is obvious, hence let us focus on the right-to-left implication. Suppose then that $\mathbf{v} \in \mathsf{lattice}(D)$.

Let $\mathbf{w} = \sum_{\mathbf{d} \in D} L \cdot \mathbf{d}$, where the summation is over the columns of D and L is a positive integer to be fixed later. Observe that for every $\mathbf{f} \in \mathcal{F}$, we have $\langle \mathbf{f}, \mathbf{v} - \mathbf{w} \rangle > M - L \cdot \sum_{\mathbf{d} \in D} \langle \mathbf{f}, \mathbf{d} \rangle$. Therefore, if we choose M to be not smaller than $L \cdot \max_{\mathbf{f} \in \mathcal{F}} \|\mathbf{f}\|_1 \cdot \|D\|_\infty$, then we are certain that $\langle \mathbf{f}, \mathbf{v} - \mathbf{w} \rangle \geqslant 0$ for all $\mathbf{f} \in \mathcal{F}$, and hence $\mathbf{v} - \mathbf{w} \in \mathsf{cone}(D)$. Consequently, we can write $\mathbf{v} - \mathbf{w} = D\mathbf{y}$ for some $\mathbf{y} \in \mathbb{R}^t_{\geqslant 0}$. Let $\mathbf{y}' \in \mathbb{Z}^t_{\geqslant 0}$ be such that $y'_i = \lfloor y_i \rfloor$ for all $i \in \{1, \ldots, t\}$, and let $\mathbf{v}' = \mathbf{w} + D\mathbf{y}'$. Then

$$\|\mathbf{v} - \mathbf{v}'\|_\infty = \|D(\mathbf{y} - \mathbf{y}')\|_\infty \leqslant t \cdot \|D\|_\infty.$$

On the other hand, we clearly have $\mathbf{v}' \in \mathsf{intCone}(D)$ and by assumption, $\mathbf{v} \in \mathsf{lattice}(D)$. It follows that $\mathbf{v} - \mathbf{v}' \in \mathsf{lattice}(D)$. From standard bounds, see e.g. [39], it follows that there exists $\mathbf{z} \in \mathbb{Z}^t$ with $\mathbf{v} - \mathbf{v}' = D\mathbf{z}$ such that $\|\mathbf{z}\|_1$ is bounded by a function of D and $\|\mathbf{v} - \mathbf{v}'\|_\infty$, which in turn is again bounded by a function of D as explained above. (Note here that t is the number of columns of D, hence it also depends only on D.) This means that if we choose L large enough depending on D, we are certain that $\|\mathbf{z}\|_1 \leqslant L$. Now, it remains to observe that

$$\mathbf{v} = \mathbf{w} + D\mathbf{y}' + (\mathbf{v} - \mathbf{v}') = D(L \cdot \mathbf{1} + \mathbf{y}' + \mathbf{z}),$$

where $\mathbf{1}$ denotes the vector of t ones, and that all the entries of $L \cdot \mathbf{1} + \mathbf{y}' + \mathbf{z}$ are nonnegative integers. This proves that $\mathbf{v} \in \mathsf{intCone}(D)$. ∎

We remark that the statement of Lemma 2.1 actually follows from results present in the literature, concerning the notion of *diagonal Frobenius numbers*. See the work of Aliev and Henk [1] for a broader discussion and pointers to earlier works. As we will discuss in a moment, in this work we actually use a generalization of Lemma 2.1.

Consider any $\mathbf{u}, \mathbf{v} \in \mathcal{R}$. Since all the entries of $\mathbf{u} - \mathbf{v}$ are divisible by B, it is not hard to prove the following: if we choose B to be a large enough factorial, then $\mathbf{u} \in \mathsf{lattice}(D)$ if and only if $\mathbf{v} \in \mathsf{lattice}(D)$. Hence, from Lemma 2.1 it follows that $\mathcal{R}_\emptyset$ is either entirely disjoint or entirely contained in $\mathcal{Z}$.

A more involved reasoning based on the same fundamental ideas, but using a generalization of Lemma 2.1, yields the following lemma, which tackles also the case when some functionals of $\mathcal{F}$ are tight with respect to the considered vectors.

LEMMA 2.2. *Suppose $\mathbf{u}, \mathbf{v} \in \mathcal{R}$ are such that for every $\mathbf{f} \in \mathcal{F}$, if $\mathbf{f}$ is tight with respect to $\mathbf{u}$, then $\mathbf{f}$ is also tight with respect to $\mathbf{v}$. Then $\mathbf{u} \in \mathcal{Z}$ implies $\mathbf{v} \in \mathcal{Z}$.*

We remark that the proof of Lemma 2.2 actually requires more work and more ideas than those presented in the proof of Lemma 2.1. In essence, one needs to partition functionals that are tight with respect to $\mathbf{u}$ into those that are very tight (have very small $p_{\mathbf{f}}$) and those that are only slightly tight (have relatively large $p_{\mathbf{f}}$) in order to create a sufficient gap between very tight and slightly tight functionals. Having achieved this, a delicate variant of the reasoning from the proof of Lemma 2.1 can be applied. It is important that whenever a functional $\mathbf{f} \in \mathcal{F}$ is tight with respect to both $\mathbf{u}$ and $\mathbf{v}$, we actually know that $\langle \mathbf{f}, \mathbf{u} \rangle = \langle \mathbf{f}, \mathbf{v} \rangle = p_{\mathbf{f}}$. Note that this is exactly the benefit achieved by restricting attention to the vectors of $\Lambda^B_{\mathbf{r}}$.

Using Lemma 2.2, we can immediately describe how the structure of $\mathcal{Z}$ relates to that of $\mathcal{R}$.

COROLLARY 2.1. *For every $\mathcal{G} \subseteq \mathcal{F}$, either $\mathcal{R}_\mathcal{G} \cap \mathcal{Z} = \emptyset$ or $\mathcal{R}_\mathcal{G} \subseteq \mathcal{Z}$. Moreover, if $\mathcal{R}_\mathcal{G} \subseteq \mathcal{Z}$ and $\mathcal{R}_\mathcal{G}$ is non-empty, then $\mathcal{R}_{\mathcal{G}'} \subseteq \mathcal{Z}$ for all $\mathcal{G}' \subseteq \mathcal{G}$.*

Corollary 2.1 suggests now how to define the polyhedron $\mathcal{Q}$. Namely, $\mathcal{Q}$ is defined as the set of all $\mathbf{v} \in \mathbb{R}^k$ satisfying the following linear inequalities:

- inequalities $\langle \mathbf{f}, \mathbf{v} \rangle \geqslant 0$ for all $\mathbf{f} \in \mathcal{F}$ that define $\mathsf{cone}(D)$; and
- for every $\mathcal{G} \subseteq \mathcal{F}$ such that $\mathcal{R}_\mathcal{G}$ is nonempty and $\mathcal{R}_\mathcal{G} \cap \mathcal{Z} = \emptyset$, the inequality

$$\sum_{\mathbf{g} \in \mathcal{G}} \langle \mathbf{g}, \mathbf{v} \rangle \geqslant 1 + \sum_{\mathbf{g} \in \mathcal{G}} p_{\mathbf{g}}.$$

In essence, the inequalities from the second point "carve out" those parts $\mathcal{R}_\mathcal{G}$ that should not be included in $\mathcal{Z}$. We note that computing the inequalities defining $\mathcal{Q}$ requires solving several auxiliary integer programs to figure out for which $\mathcal{G} \subseteq \mathcal{F}$ the corresponding inequality should be included.

It is now straightforward to verify, using all the accumulated observations, that indeed $\mathcal{Z} = \mathcal{R} \cap \mathcal{Q}$ as required. This concludes a sketch of the proof of Theorem 2.1.

2.2 n-fold programming. We now give an overview of the proof of Theorem 1.2. For simplicity, we make the following assumptions.

- We focus on the feasibility problem instead of optimization. At the very end, we will remark on what additional ideas are needed to also tackle the optimization problem.
- We assume that all the diagonal blocks D_i are equal: $D_i = D$ for all $i \in \{1, \ldots, n\}$, where D is a $k \times k$ integer matrix with $\|D\|_\infty \leqslant \Delta$. This is only a minor simplification because there are only $(2\Delta+1)^{k^2}$ different matrices D_i with $\|D_i\|_\infty \leqslant \Delta$, and in the general case, we simply treat every such possible matrix "type" separately using the reasoning from the simplified case.

Breaking up bricks. Basic components of the given n-fold program $P = (C, D, \mathbf{a}, \mathbf{b}_i \colon i \in \{1, \ldots, n\})$ are *bricks*: programs $D\mathbf{y}_i = \mathbf{b}_i$ for $i \in \{1, \ldots, n\}$ that encode local constraints on the variables $\mathbf{y}_i$. While the entries of D are bounded in absolute values by the parameter Δ, we do not assume any bound on the entries of vectors $\mathbf{b}_i$. This poses an issue, as different bricks may possibly have very different behaviors.

The key idea in our approach is to simplify the program P by iteratively breaking up every brick $D\mathbf{y} = \mathbf{b}$ into two bricks $D\mathbf{y} = \mathbf{b}'$ and $D\mathbf{y} = \mathbf{b}''$ with strictly smaller right-hand sides $\mathbf{b}', \mathbf{b}''$, until eventually, we obtain an equivalent n-fold program P' in which all right-hand sides have ℓ_∞-norms bounded in terms of the parameters. The following lemma is the crucial new piece of technology used in our proof. (Here, we use the *conformal order* on $\mathbb{Z}^k$: we write $\mathbf{u} \sqsubseteq \mathbf{v}$ if $|\mathbf{u}[i]| \leqslant |\mathbf{v}[i]|$ and $\mathbf{u}[i] \cdot \mathbf{v}[i] \geqslant 0$ for all $i \in \{1, \ldots, k\}$.)

Lemma 2.3. (Brick Decomposition Lemma) *There exists a function $g(k, \Delta) \in 2^{(k\Delta)^{\mathcal{O}(k)}}$ such that the following holds. Let D be an integer matrix with t columns and k rows and all absolute values of its entries bounded by Δ. Further, let $\mathbf{b} \in \mathbb{Z}^k$ be an integer vector such that $\|\mathbf{b}\|_\infty > g(k, \Delta)$. Then there are non-zero vectors $\mathbf{b}', \mathbf{b}'' \in \mathbb{Z}^k$ such that:*

- *$\mathbf{b}', \mathbf{b}'' \sqsubseteq \mathbf{b}$ and $\mathbf{b} = \mathbf{b}' + \mathbf{b}''$; and*
- *for every $\mathbf{v} \in \mathbb{Z}^\mathbf{y}_{\geqslant 0}$ satisfying $D\mathbf{v} = \mathbf{b}$, there exist $\mathbf{v}', \mathbf{v}'' \in \mathbb{Z}^\mathbf{y}_{\geqslant 0}$ such that*

$$\mathbf{v} = \mathbf{v}' + \mathbf{v}'', \qquad D\mathbf{v}' = \mathbf{b}', \qquad \text{and} \qquad D\mathbf{v}'' = \mathbf{b}''.$$

In other words, Lemma 2.3 states that the brick $D\mathbf{y} = \mathbf{b}$ can be broken into two new bricks $D\mathbf{y}' = \mathbf{b}'$ and $D\mathbf{y}'' = \mathbf{b}''$ with conformally strictly smaller $\mathbf{b}', \mathbf{b}''$ so that *every* potential solution $\mathbf{v}$ to $D\mathbf{y} = \mathbf{b}$ can be decomposed into solutions $\mathbf{v}', \mathbf{v}''$ to the two new bricks. It is easy to see that this condition implies that in P, we may replace the brick $D\mathbf{y} = \mathbf{b}$ with $D\mathbf{y}' = \mathbf{b}'$ and $D\mathbf{y}'' = \mathbf{b}''$ without changing feasibility or, in the case of the optimization problem, the minimum value of the optimization goal. In the latter setting, both new bricks inherit the optimization vector $\mathbf{c}_i$ from the original brick.

Before we continue, let us comment on the proof of Lemma 2.3. We use two ingredients. The first one is the following fundamental result of Klein [26]. (Here, for a multiset of vectors A, by $\sum A$ we denote the sum of all the vectors in A.)

Lemma 2.4. (Klein Lemma, variant from [11]) *Let $T_1, \ldots, T_n$ be non-empty multisets of vectors in $\mathbb{Z}^k$ such that $\sum T_1 = \sum T_2 = \ldots = \sum T_n$ and all vectors contained in all multisets $T_1, \ldots, T_n$ have ℓ_∞-norm bounded by Δ. Then there are non-empty multisets $S_1 \subseteq T_1, \ldots, S_n \subseteq T_n$, each of size at most $2^{\mathcal{O}(k\Delta)^k}$, such that $\sum S_1 = \sum S_2 = \ldots = \sum S_n$.*

In the context of the proof of Lemma 2.3, we apply Lemma 2.4 to the family of all multisets T that consist of columns of D and satisfy $\sum T = \mathbf{b}$. By encoding multiplicities, such multisets correspond to vectors $\mathbf{v} \in \mathbb{Z}^k_{\geqslant 0}$ satisfying $D\mathbf{v} = \mathbf{b}$. (We hide here some technicalities regarding the fact that this family is infinite.) By Lemma 2.4, from each such multiset T, we can extract a submultiset S of bounded size such that all the submultisets S sum up to the same vector $\mathbf{b}'$. Denoting $\mathbf{b}'' = \mathbf{b} - \mathbf{b}'$, this means that every vector $\mathbf{v} \in \mathbb{Z}^k_{\geqslant 0}$ satisfying $D\mathbf{v} = \mathbf{b}$ can be decomposed as $\mathbf{v} = \mathbf{v}' + \mathbf{v}''$ with $\mathbf{v}', \mathbf{v}'' \in \mathbb{Z}^k_{\geqslant 0}$ so that $D\mathbf{v}' = \mathbf{b}'$ and $D\mathbf{v}'' = \mathbf{b}''$. Namely, $\mathbf{v}'$ corresponds

to the vectors contained in S and $\mathbf{v}''$ corresponds to the vectors contained in $T - S$, where T is the multiset corresponding to $\mathbf{v}$.

There is an issue in the above reasoning: we do not obtain the property $\mathbf{b}', \mathbf{b}'' \sqsubseteq \mathbf{b}$, which will be important in later applications of Lemma 2.3. To bridge this difficulty, we apply the argument above exhaustively to decompose $\mathbf{b}$ as $\mathbf{b}_1 + \ldots + \mathbf{b}_m$, for some integer m, so that every vector $\mathbf{b}_i$ has the ℓ_∞-norm bounded by $2^{\mathcal{O}(k\Delta)^k}$ and every vector $\mathbf{v} \in \mathbb{Z}^k_{\geqslant 0}$ satisfying $D\mathbf{v} = \mathbf{b}$ can be decomposed as $\mathbf{v} = \mathbf{v}_1 + \ldots + \mathbf{v}_m$ where $\mathbf{v}_i \in \mathbb{Z}^k_{\geqslant 0}$ satisfies $D\mathbf{v}_i = \mathbf{b}_i$. Then, we treat vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m$ with the following lemma.

LEMMA 2.5. *Let $\mathbf{u}_1, \ldots, \mathbf{u}_m$ be vectors in $\mathbb{Z}^k$ of ℓ_∞-norm bounded by Ξ, and let $\mathbf{b} = \sum_{i=1}^m \mathbf{u}_i$. Then the vectors $\mathbf{u}_1, \ldots, \mathbf{u}_m$ can be grouped into non-empty groups $U_1, \ldots, U_\ell$, each of size at most $\mathcal{O}(\Delta)^{2^{k-1}}$, so that $\sum U_i \sqsubseteq \mathbf{b}$ for all $i = 1, \ldots, \ell$.*

More precisely, Lemma 2.5 allows us to group vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m$ into groups of bounded size so that the sum within each group is sign-compatible with $\mathbf{b}$. Assuming $\|\mathbf{b}\|_\infty$ is large enough, there will be at least two groups. Then, any non-trivial partition of the groups translates into a suitable decomposition $\mathbf{b} = \mathbf{b}' + \mathbf{b}''$ with $\mathbf{b}', \mathbf{b}'' \sqsubseteq \mathbf{b}$.

The proof of Lemma 2.5 is by induction on k and uses arguments similar to standard proofs of Steinitz Lemma. This concludes a sketch of the proof of Lemma 2.3.

Once Lemma 2.3 is established, it is natural to use it iteratively: break $\mathbf{b}$ into $\mathbf{b}', \mathbf{b}''$, then break $\mathbf{b}'$ into two even smaller vectors, and so on. By applying the argument exhaustively, eventually we obtain a collection of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m \sqsubseteq \mathbf{b}$ such that $\mathbf{b} = \mathbf{b}_1 + \ldots + \mathbf{b}_m$, $\|\mathbf{b}_i\|_\infty \leqslant 2^{(k\Delta)^{\mathcal{O}(k)}}$ for all $i \in \{1, \ldots, m\}$, and every $\mathbf{v} \in \mathbb{Z}^k_{\geqslant 0}$ satisfying $D\mathbf{v} = \mathbf{b}$ can be decomposed as $\mathbf{v} = \mathbf{v}_1 + \ldots + \mathbf{v}_m$ with $\mathbf{v}_i \in \mathbb{Z}^k_{\geqslant 0}$ and $D\mathbf{v}_i = \mathbf{b}_i$ for all $i \in \{1, \ldots, m\}$. We call such a collection a *faithful decompostion* of $\mathbf{b}$ of *order* $2^{(k\Delta)^{\mathcal{O}(k)}}$.

There is an important technical caveat here. Observe that the size m of a faithful decomposition of a right-hand side $\mathbf{b}$ can be as large as $\Omega(\|\mathbf{b}\|_1)$, which is *exponential* in the bitsize of the program P. So we cannot hope to compute a faithful decomposition explicitly within the target time complexity. However, observe that all vectors $\mathbf{b}_i$ in a faithful decomposition $\mathcal{B}$ are bounded in ℓ_∞-norm by $\Xi := 2^{(k\Delta)^{\mathcal{O}(k)}}$, and there are only at most $(2\Xi + 1)^k$ different such vectors. Therefore, $\mathcal{B}$ can be encoded by storing, for each vector $\mathbf{b}'$ present in $\mathcal{B}$, the multiplicity of $\mathbf{b}'$ in $\mathcal{B}$. Thus, describing $\mathcal{B}$ takes $2^{(k\Delta)^{\mathcal{O}(k)}} \cdot \log \|\mathbf{b}\|_\infty$ bits.

With this encoding scheme in mind, we show that a faithful decomposition $\mathcal{B}$ of a given vector $\mathbf{b}$ of order at most Ξ can be computed in fixed-parameter time $f(\Delta, k) \cdot (\log \|\mathbf{b}\|_\infty)^{\mathcal{O}(1)}$, for a computable function f. For this, we show that one can extract parts of the decomposition in "larger chunks", at each step reducing the ℓ_1-norm of the decomposed vector by a constant fraction; this gives a total number of steps logarithmic in $\|\mathbf{b}\|_1$. In each step, to extract the next large chunk of the decomposition, we use the fixed-parameter algorithm for optimization problems definable in Presburger arithmetic, due to Koutecký and Talmon [36]. We remark that in our context, this tool could be also replaced by the fixed-parameter algorithm of Eisenbrand and Shmonin [17] for $\forall\exists$ integer programming.

Reduction to (mixed) integer programming with few variables. With faithful decompositions understood, we can compute, for every right-hand side $\mathbf{b}_i$ part of P, a faithful decomposition $\{\mathbf{b}_i^1, \ldots, \mathbf{b}_i^{m_i}\}$ of $\mathbf{b}_i$. This allows us to construct an equivalent (in terms of feasibility and optimization) n-fold program P' by replacing each brick $D\mathbf{y}_i = \mathbf{b}_i$ with bricks $D\mathbf{y}_i^j = \mathbf{b}_i^j$ for $j \in \{1, \ldots, m_i\}$. Thus, the program P' has an exponential number of bricks, but can be computed and described concisely: all right-hand sides are bounded in the ℓ_∞-norm by at most Ξ, so for every potential right-hand side $\mathbf{b}$, we just write the multiplicity in which $\mathbf{b}$ appears in P'. We remark that such *high-multiplicity encoding* of n-fold integer programs has already been studied by Knop et al. [31].

For convenience, let $\mathsf{RHS} := \{-\Xi, \ldots, \Xi\}^k$ be the set of all possible right-hand sides, and for $\mathbf{b} \in \mathsf{RHS}$, by $\mathsf{count}[\mathbf{b}]$ we denote the multiplicity of $\mathbf{b}$ in P'.

It is now important to better understand the set of solutions to a single brick $D\mathbf{y} = \mathbf{b}$ present in P'. Here comes a key insight stemming from the theory of Graver bases: as (essentially) proved by Pottier [39], every solution $\mathbf{w} \in \mathbb{Z}^k_{\geqslant 0}$ to $D\mathbf{w} = \mathbf{b}$ can be decomposed as $\mathbf{w} = \widehat{\mathbf{w}} + \mathbf{g}_1 + \ldots + \mathbf{g}_\ell$, where

- $\widehat{\mathbf{w}} \in \mathbb{Z}^k_{\geqslant 0}$ is a *base solution* that also satisfies $D\widehat{\mathbf{w}} = \mathbf{b}$, but $\|\widehat{\mathbf{w}}\|_\infty$ is bounded by a function of Δ and $\|\mathbf{b}\|_\infty$, and

- $\mathbf{g}_1, \ldots, \mathbf{g}_\ell \in \mathbb{Z}^k_{\geqslant 0}$ are elements of the Graver basis of D.

Here, the *Graver basis* of D consists of all conformally-minimal non-zero vectors $\mathbf{g}$ satisfying $D\mathbf{g} = \mathbf{0}$. In particular, it is known that the Graver basis is always finite and consists of vectors of ℓ_∞ norm bounded by $(2k\Delta + 1)^k$ [15]. The decomposition explained above will be called a *Graver decomposition* of $\mathbf{w}$.

For $\mathbf{b} \in \mathsf{RHS}$, let $\mathsf{Base}[\mathbf{b}]$ be the set of all possible base solutions $\widehat{\mathbf{w}}$ to $D\mathbf{y} = \mathbf{b}$. As $\|\mathbf{b}\|_\infty \leqslant \Xi$ and Ξ is bounded by a function of the parameters under consideration, it follows that $\mathsf{Base}[\mathbf{b}]$ consists only of vectors of bounded ℓ_∞-norms, and therefore it can be efficiently constructed.

Having this understanding, we can write an integer program M with few variables that is equivalent to P'. The variables are as follows:

- for every $\mathbf{b} \in \mathsf{RHS}$ and $\widehat{\mathbf{w}} \in \mathsf{Base}[\mathbf{b}]$, we introduce a variable $\zeta^{\mathbf{b}}_{\widehat{\mathbf{w}}} \in \mathbb{Z}_{\geqslant 0}$ that signifies how many times in total $\widehat{\mathbf{w}}$ is used in the Graver decompositions of solutions to individual bricks.
- for every nonnegative vector $\mathbf{g}$ in the Graver basis of D, we introduce a variable $\delta_{\mathbf{g}} \in \mathbb{Z}_{\geqslant 0}$ signifying how many times in total $\mathbf{g}$ appears in the Graver decompositions of solutions to individual bricks.

Note that since program P' is uniform, the guessed base solutions and elements of the Graver basis can be assigned to *any* brick with the same effect on the linking constraints of P'. Hence, it suffices to verify the cardinalities and the total effect on the linking constrains of P', yielding the following constraints of M:

- the translated linking constraints: $\sum_{\mathbf{b} \in \mathsf{RHS}} \sum_{\widehat{\mathbf{w}} \in \mathsf{Base}[\mathbf{b}]} \zeta^{\mathbf{b}}_{\widehat{\mathbf{w}}} \cdot C\widehat{\mathbf{w}} + \sum_{\mathbf{g} \in \mathsf{Graver}(D), \mathbf{g} \geqslant 0} \delta_{\mathbf{g}} \cdot C\mathbf{g} = \mathbf{a}$.
- for every $\mathbf{b} \in \mathsf{RHS}$, the cardinality constraint $\sum_{\widehat{\mathbf{w}} \in \mathsf{Base}[\mathbf{b}]} \zeta^{\mathbf{b}}_{\widehat{\mathbf{w}}} = \mathsf{count}[\mathbf{b}]$.

Noting that the number of variables of M is bounded in terms of the parameters, we may apply any fixed-parameter algorithm for integer programming parameterized by the number of variables, for instance that of Kannan [25], to solve M. This concludes the description of the algorithm for the feasibility problem.

In the case of the optimization problem, there is an issue that the optimization vectors $\mathbf{c}_i$ may differ between different bricks, and there may be as many as n different such vectors. While the Graver basis elements can be always greedily assigned to bricks in which their contribution to the optimization goal is the smallest, this is not so easy for the base solutions, as every brick may accommodate only one base solution. We may enrich M by suitable assignment variables $\omega^{\mathbf{b},i}_{\widehat{\mathbf{w}}}$ to express how many base solutions of each type are assigned to bricks with different optimization vectors; but this yields as many as $\Omega(n)$ additional variables. Fortunately, we observe that in the enriched program M, if one fixes any integral valuation of variables $\zeta^{\mathbf{b}}_{\widehat{\mathbf{w}}}$ and $\delta_{\mathbf{g}}$, the remaining problem on variables $\omega^{\mathbf{b},i}_{\widehat{\mathbf{w}}}$ corresponds to a flow problem, and hence its constraint matrix is totally unimodular. Thus, we may solve M as a mixed integer program where variables $\omega^{\mathbf{b},i}_{\widehat{\mathbf{w}}}$ are allowed to be fractional. The number of integral variables is bounded in terms of parameters, so we may apply the fixed-parameter algorithm for mixed integer programming of Lenstra [38].

References

[1] I. Aliev and M. Henk. Feasibility of integer knapsacks. *SIAM J. Optim.*, 20(6):2978–2993, 2010.

[2] M. Aschenbrenner and R. Hemmecke. Finiteness theorems in stochastic integer programming. *Found. Comput. Math.*, 7(2):183–227, 2007.

[3] M. Briański, M. Koutecký, D. Král', K. Pekárková, and F. Schröder. Characterization of matrices with bounded Graver bases and depth parameters and applications to integer programming. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022*, volume 229 of *LIPIcs*, pages 29:1–29:20. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2022.

[4] T. F. N. Chan, J. W. Cooper, M. Koutecký, D. Král, and K. Pekárková. Matrices of optimal tree-depth and a row-invariant parameterized algorithm for integer programming. *SIAM J. Comput.*, 51(3):664–700, 2022.

[5] H. Chen, L. Chen, and G. Zhang. FPT algorithms for a special block-structured integer program with applications in scheduling. *CoRR*, abs/2107.01373, 2021.

[6] H. Chen, L. Chen, and G. Zhang. Block-structured integer programming: Can we parameterize without the largest coefficient? *Discret. Optim.*, 46:100743, 2022.

[7] L. Chen, M. Koutecký, L. Xu, and W. Shi. New bounds on augmenting steps of block-structured integer programs. In *28th Annual European Symposium on Algorithms, ESA 2020*, volume 173 of *LIPIcs*, pages 33:1–33:19. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2020.

[8] L. Chen and D. Marx. Covering a tree with rooted subtrees — parameterized and approximation algorithms. In *29th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 2801–2820. SIAM, 2018.

[9] L. Chen, D. Marx, D. Ye, and G. Zhang. Parameterized and approximation results for scheduling with a low rank processing time matrix. In *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017*, volume 66 of *LIPIcs*, pages 22:1–22:14. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2017.

[10] J. Cslovjecsek, F. Eisenbrand, C. Hunkenschröder, L. Rohwedder, and R. Weismantel. Block-structured integer and linear programming in strongly polynomial and near linear time. In *32nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 1666–1681. SIAM, 2021.

[11] J. Cslovjecsek, F. Eisenbrand, M. Pilipczuk, M. Venzin, and R. Weismantel. Efficient sequential and parallel algorithms for multistage stochastic integer programming using proximity. In *29th Annual European Symposium on Algorithms, ESA 2021*, volume 204 of *LIPIcs*, pages 33:1–33:14. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2021.

[12] D. Dadush. *Integer Programming, Lattice Algorithms, and Deterministic Volume Estimation*. PhD thesis, Georgia Institute of Technology, USA, 2012.

[13] D. Dadush, F. Eisenbrand, and T. Rothvoss. From approximate to exact integer programming. In *24th International Conference on Integer Programming and Combinatorial Optimization, IPCO 2023*, volume 13904 of *Lecture Notes in Computer Science*, pages 100–114. Springer, 2023.

[14] J. A. De Loera, R. Hemmecke, S. Onn, and R. Weismantel. N-fold integer programming. *Discrete Optimization*, 5(2):231–241, 2008. In Memory of George B. Dantzig.

[15] F. Eisenbrand, C. Hunkenschröder, and K. Klein. Faster algorithms for integer programs with block structure. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, volume 107 of *LIPIcs*, pages 49:1–49:13. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2018.

[16] F. Eisenbrand, C. Hunkenschröder, K. Klein, M. Koutecký, A. Levin, and S. Onn. An algorithmic theory of integer programming. *CoRR*, abs/1904.01361, 2019.

[17] F. Eisenbrand and G. Shmonin. Parametric integer programming in fixed dimension. *Math. Oper. Res.*, 33(4):839–850, 2008.

[18] A. Frank and É. Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.

[19] T. Gavenčiak, M. Koutecký, and D. Knop. Integer programming in parameterized complexity: Five miniatures. *Discret. Optim.*, 44(Part):100596, 2022.

[20] R. Hemmecke, M. Köppe, and R. Weismantel. Graver basis and proximity techniques for block-structured separable convex integer minimization problems. *Math. Program.*, 145(1-2):1–18, 2014.

[21] R. Hemmecke, S. Onn, and L. Romanchuk. n-fold integer programming in cubic time. *Math. Program.*, 137(1-2):325–341, 2013.

[22] K. Jansen, K. Klein, and A. Lassota. The double exponential runtime is tight for 2-stage stochastic ILPs. *Math. Program.*, 197(2):1145–1172, 2023.

[23] K. Jansen, K. Klein, M. Maack, and M. Rau. Empowering the configuration-IP: new PTAS results for scheduling with setup times. *Math. Program.*, 195(1):367–401, 2022.

[24] K. Jansen, A. Lassota, and L. Rohwedder. Near-linear time algorithm for n-fold ILPs via color coding. *SIAM J. Discret. Math.*, 34(4):2282–2299, 2020.

[25] R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, 1987.

[26] K. Klein. About the complexity of two-stage stochastic IPs. *Math. Program.*, 192(1):319–337, 2022.

[27] K. Klein and J. Reuter. Collapsing the tower — On the complexity of multistage stochastic IPs. In *33rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 348–358. SIAM, 2022.

[28] D. Knop and M. Koutecký. Scheduling meets n-fold integer programming. *J. Sched.*, 21(5):493–503, 2018.

[29] D. Knop and M. Koutecký. Scheduling kernels via configuration LP. In *30th Annual European Symposium on Algorithms, ESA 2022*, volume 244 of *LIPIcs*, pages 73:1–73:15. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2022.

[30] D. Knop, M. Koutecký, A. Levin, M. Mnich, and S. Onn. Parameterized complexity of configuration integer programs. *Oper. Res. Lett.*, 49(6):908–913, 2021.

[31] D. Knop, M. Koutecký, A. Levin, M. Mnich, and S. Onn. High-multiplicity N-fold IP via configuration LP. *Math. Program.*, 200(1):199–227, 2023.

[32] D. Knop, M. Koutecký, and M. Mnich. Combinatorial n-fold integer programming and applications. *Math. Program.*, 184(1):1–34, 2020.

[33] D. Knop, M. Koutecký, and M. Mnich. Voting and bribing in single-exponential time. *ACM Trans. Economics and Comput.*, 8(3):12:1–12:28, 2020.

[34] D. Knop, M. Pilipczuk, and M. Wrochna. Tight complexity lower bounds for integer linear programming with few constraints. *ACM Trans. Comput. Theory*, 12(3):19:1–19:19, 2020.

[35] M. Koutecký, A. Levin, and S. Onn. A parameterized strongly polynomial algorithm for block structured integer

programs. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, volume 107 of *LIPIcs*, pages 85:1–85:14. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2018.

[36] M. Koutecký and N. Talmon. Multi-party campaigning. In *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 5506–5513. AAAI Press, 2021.

[37] M. Koutecký and J. Zink. Complexity of scheduling few types of jobs on related and unrelated machines. In *31st International Symposium on Algorithms and Computation, ISAAC 2020*, volume 181 of *LIPIcs*, pages 18:1–18:17. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2020.

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

[39] L. Pottier. Minimal solutions of linear diophantine systems: Bounds and algorithms. In *4th International Conference on Rewriting Techniques and Applications, RTA-91*, volume 488 of *Lecture Notes in Computer Science*, pages 162–173. Springer, 1991.

[40] V. Reis and T. Rothvoss. The subspace flatness conjecture and faster integer programming. *CoRR*, abs/2303.14605, 2023.

[41] R. Schultz, L. Stougie, and M. H. van der Vlerk. Two-stage stochastic integer programming: a survey. *Statistica Neerlandica*, 50(3):404–416, 1996.

[42] H. Weyl. Elementare Theorie der konvexen Polyeder. *Commentarii Mathematici Helveticii*, 7:290–306, 1935.