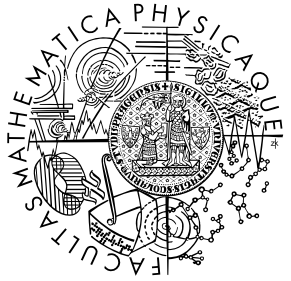

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

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

March 15, 2025

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, 19-27871X, 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, doi:10.1609/AAAI.V35I14.17454

[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, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms (ESA 2024)*, volume 308 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ESA.2024.32, doi:10.4230/LIPIcs.ESA.2024.32

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, Micha⟨*Pilipczuk*, *andAdamPolak*.*Parameterizedalgorithmsforblock*
structuredintegerprogramswithlargeentries.*InProceedingsofthe*2024*AnnualACM* − *SIAMSymposiumonDiscreteAlgorithm*
−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 inequalities involving the variables x, and each inequality is linear in the variables. 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. This function may be linear, but often, including in this thesis, more general functions are allowed. 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 time which is at most polynomial in the input length, and those which likely cannot. (The "likely" part of the previous sentence is the heart of the famous P $\neq$ 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, even for relaxed notions of "efficiency". 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 with both linear and more general objective functions.

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. In our work we have devised more effective

[1]We omit references in these first few introductory paragraphs to keep the 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.

algorithms, for more general classes of integer programs, and also a clearer understanding of the boundaries of efficiency, and the true underlying reasons for them.

1.1 Integer Programming

Formally, the **integer programming** problem is[2]

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

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, integer programming 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[3] page of the Gurobi solver.

Because of its importance, (IP) has been studied from many perspectives, including heuristics [47], or 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 with provable time complexity guarantees. 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* 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 -1, 0, or 1, then (ILP) coincides with its continuous relaxation[4]. 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 integers $-\Delta$ 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

[2]In contrast to the paragraphs above, we study the minimization version of the problem. This is without loss of generality if f is a linear function, as minimizing f is the same as maximizing $-f$, which is linear as well; however, for more general functions the difference between minimization and maximization may be substantial.

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

[4]The continuous relaxation of an (ILP) instance is the same instance, just without the $\mathbf{x} \in \mathbb{Z}^n$ constraint. This is then an instance of *linear programming*.

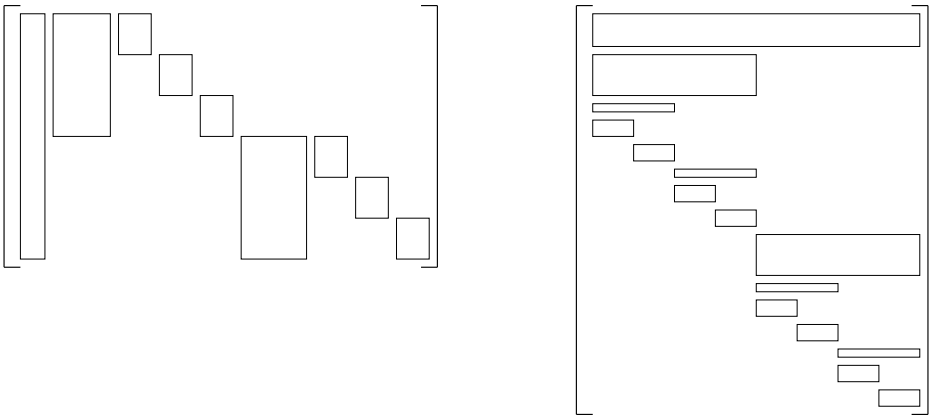

Fig. 1. On the left, a schematic multi-stage with three levels is presented. On the right, a schematic tree-fold with 4 layers is pictured. All entries within a rectangle can be non-zero, all entries outside of the rectangles must be zero.

set. The state-of-the-art is due to Reis and Rothvoss [98] who have shown that (ILP) can be solved in time $(\log n)^{O(n)}\text{poly}(L)$ (as before, L is the encoding length of the input instance); 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 $(m\|A\|_\infty)^{O(m^2)}\text{poly}(L)$; here, $\|A\|_\infty$ is the maximum absolute value of any entry of A. This is FPT parameterized by the combined parameter $\|A\|_\infty + m$, that is, when A has few rows and small coefficients in absolute value.

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 obtained time complexity improvements, generalizations 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), meaning that the key ideas are the same, 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. Consider the transportation problem, which asks for an optimal routing from several sources to several destinations. It has been defined by Hitchcock [58] in 1941 and independently studied by Kantorovich [69] in 1942, and Dantzig [21] studied it in 1951 in the context of the applicability of the simplex method. 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 the given 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 $k \in [l]$, objective function $f \equiv 0$, and the following constraints:

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

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

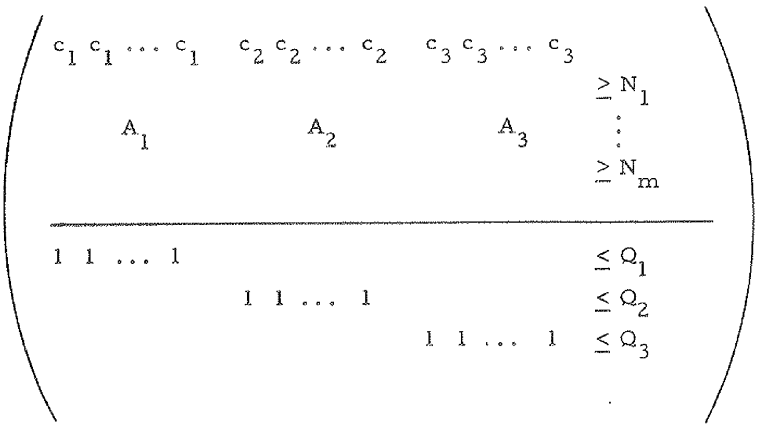

Fig. 2. A figure from "A Linear Programming Approach to the Cutting-Stock Problem—Part II" by Gilmore and Gomory [46] depicting the Configuration LP (not yet under this name) of the "Machine Balance Problem".

A different implicit appearance of the n-fold structure is in the famous[5] 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 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 and $\bigcup_i M_i$ contains all jobs, 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

[5]Over 2,800 and 1,700 citations, respectively, according to Google Scholar, as of September 2024.

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. While this approach is primarily studied in the context of continuous optimization, it can be fruitfully generalized to integer programming as well [122]. 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 [122] 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 sub-matrices, 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

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

Already deciding feasibility of this variant has been shown to be NP-hard for totally unimodular matrices [16]. In a way, (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* (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 ≠ 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: (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 (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 $\mathrm{c\mathring{d}d}(M(A))$? 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 $\mathrm{c\mathring{d}d}(M(A))$. 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, Micha⟨*Pilipczuk*, *MoritzVenzin*, *andRobertWeismantel*.*Efficientsequentialandparallelalgorithmsformultist*
1 – −33 : 14.*SchlossDagstuhl* − *Leibniz* − *ZentrumfrInformatik*, 2021..

Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Micha⟨*Pilipczuk*, *andSaketSaur*

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

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

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.

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

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

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.

Matt DeVos, O-joung Kwon, and Sang-il Oum. Branch-depth: Generalizing tree-depth of graphs. *European Journal of Combinatorics*, 90:103186, 2020.
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.
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.
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.
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.
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.
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.
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.
Michael C. Ferris and Jeffrey D. Horn. Partitioning mathematical programs for parallel solution. *Mathematical Programming*, 80(1):35–61, 1998.
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.
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.
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.
András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.
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.
Gerald Gamrath and Marco E. Lübbecke. Experiments with a generic Dantzig–Wolfe decomposition for integer programs. *Experimental Algorithms*, 6049:239 – 252, 2010.
Robert Ganian and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP. *Artificial Intelligence*, 2018.
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.
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.
Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961.
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.
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.
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.

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.

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

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.

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

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.

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.

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

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

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

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.

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.

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

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.

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

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.

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.

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.

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.

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

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

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

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

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.

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

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.

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

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.

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.

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.

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.

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.

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

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.

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.

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

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

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

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.

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

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

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.

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

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.

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

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.

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.

Marcin Pilipczuk, Micha⟨*Pilipczuk, and Sebastian Siebertz. Lecture notes from the course "Sparsity" given at the Faculty of Mathema*

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

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

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.

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

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.

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.

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

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.

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

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.

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.

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.

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.

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