A Brief History of Parameterized Algorithms for Block-Structured Integer Programs

Martin Koutecký

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

Abstract

Integer Programming (IP) is a fundamental but computationally hard problem. Still, certain efficiently solvable subclasses have been identified over time, most notably totally unimodular IPs in the 1950s, and fixed-dimension IPs in the 1980s. Starting around the year 2000, a stream of research has identified block-structured IPs as yet another tractable subclass. In this paper, we give a brief and incomplete review of this history, with a focus on several of the author's contributions.

2012 ACM Subject Classification Theory of computation → Convex optimization

Keywords and phrases Integer Programming, Parameterized Algorithm, Graver Basis, Treedepth, n-fold, tree-fold, 2-stage stochastic, multistage stochastic, Mixed-Integer Programming

Digital Object Identifier 10.4230/LIPIcs.IPEC.2025.1

Category Invited Talk

Related Version This paper is partially based on my habilitation. It contains some material which was omitted here due to space limitations, in particular an entire chapter on the papers [11, 13].
Habilitation: `https://research.koutecky.name/papers/habil.pdf`

Funding Partially supported by Charles Univ. project UNCE 24/SCI/008, by the ERC-CZ project LL2406 of the Ministry of Education of Czech Republic, and by the project 24-10306S of GA ČR.

1 Introduction

The **integer programming** problem is

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

with A an integer $m \times n$ matrix, $f : \mathbb{R}^n \to \mathbb{R}$ a separable convex function, $\mathbf{b} \in \mathbb{Z}^m$, and $\mathbf{l}, \mathbf{u} \in (\mathbb{Z} \cup \{\pm\infty\})^n$. IP is well-known to be strongly NP-hard (shown by Karp [64]) 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 [97], vehicle routing [95], process scheduling [39], packing [76], 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[1] 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 [78], branch-and-cut [99, 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*

[1] `https://www.gurobi.com/case_studies/`

complexity [22]: 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 $g(k)\mathrm{poly}(L)$ 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 [57] 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[2]. 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 [38,48,81], 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 [75]. His algorithm was subsequently improved by Kannan [62] and generalized by Grötschel, Lovász, and Schrijver [49] to optimize any convex function over the integers contained in any convex set. The state-of-the-art is due to Reis and Rothvoss [90] who have shown that ILP can be solved in time $(\log n)^{\mathcal{O}(n)}\mathrm{poly}(L)$ (as before, L is the encoding length of the input instance); the important question whether an algorithm with complexity $2^{\mathcal{O}(n)}\mathrm{poly}(L)$ exists remains open.

About the same time as Lenstra's result, Papadimitriou [86] has shown that the textbook dynamic programming algorithm for Knapsack can be generalized to ILP, achieving complexity $n^{\mathcal{O}(m)} \cdot (m\|A, \mathbf{b}\|_\infty)^{\mathcal{O}(m^2)}$, where $\|A, \mathbf{b}\|_\infty$ is the maximum absolute value of any entry appearing in A and $\mathbf{b}$. Looking back, it is not too difficult to tweak this algorithm to run in time $n \cdot (m\|A\|_\infty)^{\mathcal{O}(m^2)}\mathrm{poly}(L)$, which is FPT parameterized by the combined parameter $\|A\|_\infty + m$, that is, when A has few rows and small coefficients in absolute value. However, this complexity has only been stated explicitly recently [71].

1.1 Block-structured Integer Programs

Our focus is on how a stream of research spanning over two decade has blossomed into a unified theory including significant time complexity improvements, generalizations to wider classes, and complementary lower bound results. Besides the results discussed in this paper, we have also shown how this theory can be used to obtain efficient algorithms for problems in scheduling, stringology, graph theory, and computational social choice [36,44,67–72,74].

[2] 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*.

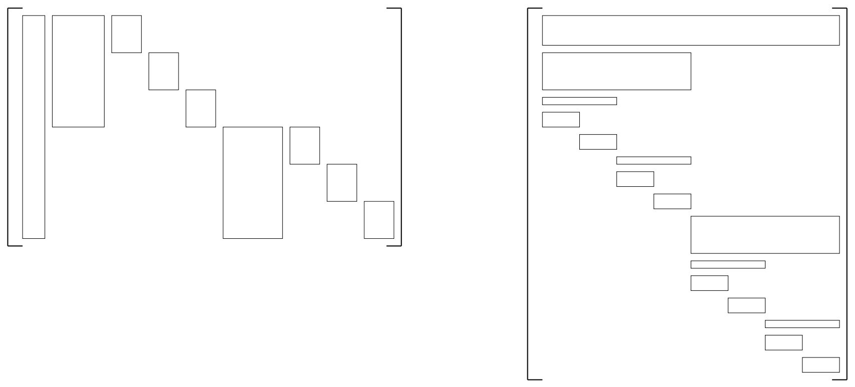

Figure 1 On the left, a schematic multi-stage with three levels is presented. On the right, a schematic tree-fold with 4 layers is pictured. All entries within a rectangle can be non-zero, all entries outside of the rectangles must be zero.

The purpose of this section is to introduce two important classes of block-structured IPs, and briefly explain the motivation and history of their study. 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) the key proof ideas usually extend to the generalized case,

(c) the uniform case is usually without loss of generality, because the generalized case can be embedded in it by suitable encoding tricks (though incurring some inefficiencies).

On the other hand, 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; see Figure 1.

1.2 N-fold Integer Programming

N-fold IP was introduced under this name by De Loera et al. [26] 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 [54] in 1941 and independently studied by Kantorovich [63] in 1942, and Dantzig [23] 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 [80]. It also has applications in privacy in databases and confidential data disclosure of statistical tables, see a survey by Fienberg and Rinaldo [37] 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 $l = 3$ [27]. 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 [28]. 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

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

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

A different implicit appearance of the n-fold structure is in the famous[3] 1961 and 1963 papers by Gilmore and Gomory [45,46] on solving the Cutting Stock problem. Their work has been essential in regards to fundamental notions like column generation, cutting planes, and the *Configuration LP* whose depiction from the 1963 paper [46] appears in Figure 2. It is clear that the presented matrix is n-fold, and we will return to it in Section 3. In fact, the significance of the n-fold structure in packing and scheduling problems seems to have been lost until a brief mention in the paper introducing n-fold IP [26], and has only been taken up in earnest [14,15,50,60,65,68,69] in the wake of the paper of Knop and Koutecký from 2018 [67]. Let us now briefly describe this connection.

[3] Over 3,000 and 1,800 citations, respectively, according to Google Scholar, as of October 2025.

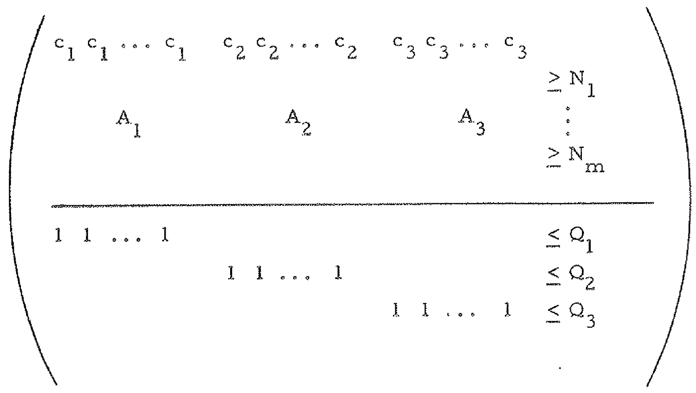

Figure 2 A figure from "A Linear Programming Approach to the Cutting-Stock Problem – Part II" by Gilmore and Gomory [46] depicting the Configuration LP (not yet under this name) of the "Machine Balance Problem".

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 $s_i > 0$, 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^i_j be a variable representing the number of jobs of length j assigned to machine i, Knop and Koutecký [67] give the following n-fold formulation:

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

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

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 [67] 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 is that they allow the handling of many "types" of machines (such as 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: a time complexity which is doubly-exponential in the parameter, and the fact that voters have to be "uniform" and cannot have distinct cost functions. Knop et al. [72] 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 [24] is an fundamental 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 [98].

1.3 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 $\mathbf{b}^1, \ldots, \mathbf{b}^n \in \mathbb{Z}^t$, their probabilities by $p_1, \ldots, p_n \in (0,1]$, 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 $\mathbf{x} := (\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^n)$ and $\mathbf{b} := (\mathbf{b}^1, \ldots, \mathbf{b}^n)$ 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 [7, 61, 88].

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 [5] in 1962. Where Dantzig-Wolfe introduces new columns in each iteration (leading to "column generation"), Benders introduces new rows (leading to "row generation"). A generalization to the (mixed) integer case has been also studied [94].

Integer programs with block structure have also been the subject of at least two habilitations, that of Martin [79] and Nowak [83], which contain many other relevant references.

2 (Strongly) Fixed-parameter Tractable Algorithms for ILP and IP

This section is based on the papers [73] and [31]. Note that even though [73] has been published in 2018 and [31] only in 2024, the results of both papers have been obtained in 2017-2018 and both logically as well as chronologically precede the papers reviewed in subsequent sections.

2.1 State of the Art in 2018

We will now review the developments regarding parameterized complexity of block-structured IPs until 2018. Whenever we talk about FPT algorithms here, 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.

We now discuss three areas where the main developments took place.

2-stage and multi-stage stochastic IP. Already in 2003, Hemmecke and Schultz have shown that 2-stage stochastic IP is FPT [53], 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 [77] 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. [26] 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 [91]) and 2007 (by Hoşten and Sullivant [58]). A particularly ingenious trick appears in the paper by Santos and Sturmfels [91], 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 [91], *"The phrase 'the Graver basis of the Graver basis' is not a typo but it is the punchline"*.

Still, the 2008 algorithm of [26] 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 [52] in 2013, where they combined the ideas from [26] with a Knapsack-like dynamic programming algorithm (in the spirit of Papadimitriou [86]) to obtain the first FPT algorithm for n-fold IP. This algorithm was highlighted in the textbook of Downey and Fellows [29], and it attracted much attention after we have shown its wide applicability in scheduling [67], computational social choice [72], and other areas [71]. Spurred by these results, Chen and Marx [14] 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 [82, 87], and it is natural to ask about its applicability to IP. We focus on two graphs 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 [41] and was reproven by Jansen and Kratsch [59] 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 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. [30] 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 matrix. Indeed, the algorithms presented by [30] 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 [73], we have fully settled this with the following result:

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

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

Note that for this 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 $\min\{\mathrm{td}_P(A_I), \mathrm{td}_D(A_I)\} \leq \min\{\mathrm{td}_P(A) + 1, \mathrm{td}_D(A)\}$.

Theorem 1 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 $a = 2$ [33]. Second, the parameterization cannot be relaxed by removing the parameter $\|A\|_\infty$: IP is para-NP-hard parameterized by $\mathrm{td}_P(A)$ [30, Thm 21] and strongly W[1]-hard parameterized by $\mathrm{td}_D(A)$ [72, 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 [33]. Let us now sketch our approach towards proving Theorem 1.

Treedepth is Equivalent to Block Structure. The first ingredient is showing that, assuming small $\|A\|_\infty$, matrices with small primal treedepth are essentially equivalent to multi-stage Wmatrices, 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 [52] 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.

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 [14] 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 Knapsack-style 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* [93]. 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 [56], bimodular ILP [3], so-called binet ILP [2], and n-fold ILP with constant block dimensions [25]. All remaining results, such as Lenstra's famous algorithm or the FPT algorithm for n-fold IP of Hemmecke et al. [52], are not strongly polynomial.

A second contribution of [73] 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 ILP can be solved with an algorithm whose number of arithmetic operations is bounded by*

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

The proof of Theorem 2 proceeds in four steps:

1. **LP relaxation:** the LP relaxation can be solved in time poly$(n \cdot \langle A \rangle)$ by Tardos' algorithm [93], obtaining a fractional optimum $\mathbf{x}^*$.
2. **Proximity:** the integer optimum $\mathbf{z}^*$ which we seek is at an ℓ_∞-distance at most $\mathcal{O}(g_\infty(A) \cdot n)$ 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 $\mathcal{O}(g_\infty(A) \cdot n)$, and subsequently also reduce the right hand side $\mathbf{b}$ to some $\mathbf{b}'$ which is bounded by $g_\infty(A)\text{poly}(n)$.
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 [40] to obtain an equivalent objective $\mathbf{w}'$ with $\log \|\mathbf{w}'\|_\infty = \text{poly}(g_\infty(A) \cdot n)$.
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 a comparison 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 [40] 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 [32] 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 [31] over [73] 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 currently perhaps the best entry point into the area for any newcomer, including graduate and even undergraduate students. Still, we are (slowly) working on developing lecture notes, which will be even more approachable.

3 High-multiplicity n-fold IP

Much research following [73] 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 [19, 20]. Our interest now 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 [66] we present a fast algorithm for it:

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

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

A natural application of Theorem 3 are *high-multiplicity scheduling problems* [18, 55, 89].

Proof Ideas. To solve a high-multiplicity problem, one needs a succinct description of solutions. The fundamental and 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 [96]; 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 [92]. 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 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 $(\|A\|_\infty rs)^{\mathcal{O}(rs)}$, and
3. it solves the residual instance by an existing n-fold IP algorithm such as Theorem 1.

The proximity theorem 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. [19] 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. Because our proximity theorem holds also for separable convex objectives, it allowed us in a subsequent work [68] 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 [75] 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:

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

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 [8] 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 can too be extended to the MILP case. This is the subject of [10]. In [9], we explore the more general setting of Mixed IP with separable convex objectives.

4.1 Linear Optimization

The main result of [10] is the following:

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

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

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. The key is considering the *fractionality* of an MILP instance, which is the minimum of the maxima of the denominators in optimal solutions. For example, it is well-known that the natural LP for the VERTEX COVER problem has half-integral optima, that is, there exists an optimum with all values in $\{0, \frac{1}{2}, 1\}$.

The usual way to go about proving fractionality bounds is via Cramer's rule and a sufficiently good bound on the determinant. However, the 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.) 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. To the best of our knowledge, this is the first time an algorithm for ILP has been lifted to MILP in this way.

We also show the limits of this approach by lower-bounding the fractionality of inverses for other classes of matrices where efficient ILP algorithms are known.

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. [35] 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 algorithm [75] can also be extended, e.g. using [49, 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

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

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 [6] on the notorious subset selection problem in statistical learning that, over the past decades, algorithmic and hardware advances have elevated *convex* (and therefore, non-linear) mixed-integer optimization to a comparable level of relevance in applications. Given the encouraging results of [10], one may 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 [56] 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 [16] reduces continuous separable convex optimization to a polynomial number of linear optimization problems.

In [9], 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 with the regime of few rows and small coefficients:

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

This improves the current state-of-the-art, double-exponential bound for MIP 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 [75].

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 [84] recently highlighted by Paat et al. [85]. Using this bound we show 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. Thus, an integer or continuous optimum is always "almost correct" in the integer part. Once the integer part is correctly assigned, we get a purely continuous residual problem, which is polynomial-time solvable.

Set $\mathbb{X} = \mathbb{Z}^z \times \mathbb{R}^q$. Consider the problem where we allow the bounds $\mathbf{l}, \mathbf{u} \in \mathbb{X}$ and right-hand side $\mathbf{b} \in \mathbb{R}^m$ to be fractional:

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

Already deciding feasibility of this variant is NP-hard for totally unimodular matrices [17]. 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. For the case of 2-stage constraints, we give an XP algorithm:

▶ **Theorem 6.** *The problem MILP$_{frac}$ where A is a 2-stage matrix, 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 matrix with blocks of size bounded by a parameter and $\|A\|_\infty = 1$ already for linear objective functions. Moreover, we show that 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 $\|A\|_\infty = 1$.

5 Beyond Small Coefficients and Graver Bases – Coming Full Circle

Intuitively, 2-stage matrices have few "global" columns, and n-fold matrices have few "global" rows, whose deletion makes the matrix block-diagonal, respectively. The combination of both is a 4-block n-fold matrix, which has few rows and few columns which are "global". Hemmecke et al. [51] 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 question about block-structured IP is whether this algorithm is qualitatively optimal, that is, whether 4-block n-fold IP is W[1]-hard or FPT.

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 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 and n-fold IP are FPT as well.

In [21], we prove that both the feasibility of 2-stage ILP and the optimization of uniform n-fold ILP (i.e., with all global blocks identical) are FPT 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:

▶ **Theorem 7.** *Let L be the encoding length of the ILP instances mentioned below.*

1. *The feasibility of a generalized (i.e., blocks may differ) 2-stage 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 $g(k, \max \|A_2^i\|_\infty) \cdot \text{poly}(L)$ 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 is necessary (unless P = NP), as one can reduce Subset Sum to a feasibility n-fold ILP with $k = 2$ and the diagonal blocks being $\{0, 1\}$-matrices. Regarding 2-stage ILP, Eisenbrand and Rothvoss have recently shown an elegant extension to the optimization variant [34].

Proof Ideas. The two parts of Theorem 7 each rely on different techniques, and, not surprisingly, they significantly depart from the approach through Graver bases. In both cases, the problem is ultimately reduced to (mixed) integer programming with a bounded number of (integral) variables, which is then solved using Lenstra's algorithm [75] (or any of its newer strengthenings [62, 90]), and this allows it to cope with large entries on input.

This connection to fixed-dimension ILP is fascinating: we have come full circle, from the old tractable class of small dimension programs, through the newer and seemingly unrelated class of block-structured programs, back to programs with small dimension. Still, the present techniques do not seem to extend to the case of *separable convex* objectives. We believe that extending the results above to separable convex objectives, or ruling out that such an extension exists, is an important open problem.

The first part of Theorem 7 is based on a new structural result about integer cones, which allows us to process the (many) input constraints and create a fixed-dimension ILP describing the feasible set of global variables. Once those are found and fixed, the problem decomposes into independent and small subproblems.

The second part of Theorem 7 relies on the following Brick Decomposition Lemma: 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, this instance can be massaged to a form solvable by Lenstra's algorithm.

Recall the n-fold model of the makespan minimization on uniformly related machines ($Q||C_{\max}$) problem introduced in Section 1.2. 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 [67]), or by the argument above and then solving the resulting fixed-dimension ILP instance using, e.g., the algorithm of Reis and Rothvoss [90]. This second approach gives a worse complexity bound, but provides a useful new perspective.

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. doi:10.1016/J.EJOR.2007.06.008.

2 Gautam Appa, Balázs Kotnyek, Konstantinos Papalamprou, and Leonidas Pitsoulis. Optimization with binet matrices. *Operations research letters*, 35(3):345–352, 2007. doi:10.1016/J.ORL.2006.04.003.

3 Stephan Artmann, Robert Weismantel, and Rico Zenklusen. A strongly polynomial algorithm for bimodular integer linear programming. In *STOC 2017*, pages 1206–1219. ACM, 2017. doi:10.1145/3055399.3055473.

4 Matthias Aschenbrenner and Raymond Hemmecke. Finiteness theorems in stochastic integer programming. *Foundations of Computational Mathematics*, 7(2):183–227, 2007. doi:10.1007/S10208-005-0174-1.

5 Jacques F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Comput. Manag. Sci.*, 2(1):3–19, 2005. doi:10.1007/S10287-004-0020-Y.

6 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.

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

8 Robert E Bixby. Solving real-world linear programs: A decade and more of progress. *Operations research*, 50(1):3–15, 2002. doi:10.1287/OPRE.50.1.3.17780.

9 Cornelius Brand, Martin Koutecký, Alexandra Lassota, and Sebastian Ordyniak. Separable Convex Mixed-Integer Optimization: Improved Algorithms and Lower Bounds. In *ESA 2024*, volume 308 of *LIPIcs*, pages 32:1–32:18, Dagstuhl, Germany, 2024. doi:10.4230/LIPIcs.ESA.2024.32.

10 Cornelius Brand, Martin Koutecký, and Sebastian Ordyniak. Parameterized algorithms for MILPs with small treedepth. In *AAAI 2021*, pages 12249–12257. AAAI Press, 2021. doi:10.1609/AAAI.V35I14.17454.

11 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.

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

13 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.

14 Lin Chen and Dániel Marx. Covering a tree with rooted subtrees–parameterized and approximation algorithms. In *SODA 2018*, pages 2801–2820. SIAM, 2018. doi:10.1137/1.9781611975031.178.

15 Lin Chen, Dániel Marx, Deshi Ye, and Guochuan Zhang. Parameterized and approximation results for scheduling with a low rank processing time matrix. In *STACS 2017*, volume 66 of *LIPIcs*, pages 22:1–22:14, 2017. doi:10.4230/LIPICS.STACS.2017.22.

16 Sergei Chubanov. A polynomial-time descent method for separable convex optimization problems with linear constraints. *SIAM Journal on Optimization*, 26(1):856–889, 2016. doi:10.1137/14098524X.

17 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.

18 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. doi:10.1137/0213007.

19 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 *SODA 2021*, pages 1666–1681. SIAM, 2021. doi:10.1137/1.9781611976465.101.

20 Jana Cslovjecsek, Friedrich Eisenbrand, Michał Pilipczuk, Moritz Venzin, and Robert Weismantel. Efficient sequential and parallel algorithms for multistage stochastic integer programming using proximity. In *ESA 2021*, volume 204 of *LIPIcs*, pages 33:1–33:14, 2021. doi:10.4230/LIPIcs.ESA.2021.33.

21 Jana Cslovjecsek, Martin Koutecký, Alexandra Lassota, Michał Pilipczuk, and Adam Polak. Parameterized algorithms for block-structured integer programs with large entries. In *SODA 2024*, pages 740–751. SIAM, 2024. doi:10.1137/1.9781611977912.29.

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

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

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

25 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. doi:10.1137/151002915.

26 Jesús A. De Loera, Raymond Hemmecke, Shmuel Onn, and Robert Weismantel. N-fold integer programming. *Discrete Optimization*, 5(2):231–241, 2008. doi:10.1016/J.DISOPT.2006.06.006.

27 Jesús A. De Loera and Shmuel Onn. The complexity of three-way statistical tables. *SIAM J. Comput*, 33(4):819–836, 2004. doi:10.1137/S0097539702403803.

28 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. doi:10.1137/040610623.

29 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.

30 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.

31 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.

32 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. doi:10.1016/J.ORL.2023.10.001.

33 Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. An algorithmic theory of integer programming, 2019. arXiv:1904.01361.

34 Friedrich Eisenbrand and Thomas Rothvoss. A parameterized linear formulation of the integer hull. *CoRR*, abs/2501.02347, 2025. doi:10.48550/arXiv.2501.02347.

35 Farbod Ekbatani, Bento Natura, and László A. Végh. Circuit imbalance measures and linear programming. *CoRR*, abs/2108.03616, 2021. arXiv:2108.03616.

36 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.

37 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.

38 Samuel Fiorini, Gwenaël Joret, Stefan Weltge, and Yelena Yuditsky. Integer programs with bounded subdeterminants and two nonzeros per row. In *FOCS 2021*, pages 13–24. IEEE, 2021. doi:10.1109/FOCS52979.2021.00011.

39 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. doi:10.1007/S10479-005-3446-X.

40 András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987. doi:10.1007/BF02579200.

41 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. URL: http://www.aaai.org/Library/AAAI/1990/aaai90-001.php.

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

43 Robert Ganian, Sebastian Ordyniak, and M. S. Ramanujan. Going beyond primal treewidth for (M)ILP. In *AAAI*, pages 815–821, 2017. doi:10.1609/AAAI.V31I1.10644.

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

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

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

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

48 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. doi:10.1007/S43069-024-00310-2.

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

50 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. doi:10.1007/S10951-023-00788-4.

51 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. doi:10.1007/S10107-013-0638-Z.

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

53 Raymond Hemmecke and Rüdiger Schultz. Decomposition of test sets in stochastic integer programming. *Mathematical Programming*, 94:323–341, 2003. doi:10.1007/S10107-002-0322-1.

54 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.

55 Dorit S. Hochbaum and Ron Shamir. Strongly polynomial algorithms for the high multiplicity scheduling problem. *Oper. Res.*, 39(4):648–653, 1991. doi:10.1287/OPRE.39.4.648.

56 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. doi:10.1145/96559.96597.

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

58 Serkan Hoşten and Seth Sullivant. A finiteness theorem for markov bases of hierarchical models. *J. Comb. Theory A*, 114(2):311–321, 2007. doi:10.1016/J.JCTA.2006.06.001.

59 Bart M. P. Jansen and Stefan Kratsch. A structural approach to kernels for ILPs: Treewidth and total unimodularity. In *ESA*, volume 9294 of *Lecture Notes in Computer Science*, pages 779–791, 2015. doi:10.1007/978-3-662-48350-3_65.

60 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. doi:10.1007/S10107-021-01694-3.

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

62 Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, August 1987. doi:10.1287/MOOR.12.3.415.

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

64 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations, The IBM Research Symposia Series*, pages 85–103. Springer, 1972. doi:10.1007/978-1-4684-2001-2_9.

65 Kim-Manuel Klein, Adam Polak, and Lars Rohwedder. On minimizing tardy processing time, max-min skewed convolution, and triangular structured ILPs. In *SODA 2023*, pages 2947–2960, 2023. doi:10.1137/1.9781611977554.CH112.

66 Dušan Knop, Martin Koutecký, Asaf Levin, Matthias Mnich, and Shmuel Onn. High-multiplicity N-fold IP via configuration LP. *Mathematical Programming*, 200(1):199–227, June 2023. doi:10.1007/s10107-022-01882-9.

67 Dušan Knop and Martin Koutecký. Scheduling meets n-fold integer programming. *J. Scheduling*, 21(5):493–503, 2018. doi:10.1007/S10951-017-0550-0.

68 Dušan Knop and Martin Koutecký. Scheduling kernels via configuration LP. In *ESA 2022*, volume 244 of *LIPIcs*, pages 73:1–73:15, 2022. doi:10.4230/LIPICS.ESA.2022.73.

69 Dušan Knop, Martin Koutecký, Asaf Levin, Matthias Mnich, and Shmuel Onn. Multitype integer monoid optimization and applications. *CoRR*, abs/1909.07326, 2019. arXiv:1909.07326.

70 Dušan Knop, Martin Koutecký, and Matthias Mnich. A unifying framework for manipulation problems. In *AAMAS 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.

71 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.

72 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.

73 Martin Koutecký, Asaf Levin, and Shmuel Onn. A parameterized strongly polynomial algorithm for block structured integer programs. In *Proc. ICALP 2018*, volume 107 of *LIPIcs*, pages 85:1–85:14, 2018. doi:10.4230/LIPICS.ICALP.2018.85.

74 Martin Koutecký and Nimrod Talmon. Multi-party campaigning. In *AAAI 2021*, pages 5506–5513. AAAI Press, 2021. doi:10.1609/AAAI.V35I6.16693.

75 Hendrik W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983. doi:10.1287/MOOR.8.4.538.

76 Andrea Lodi, Silvano Martello, and Michele Monaci. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252, 2002. doi:10.1016/S0377-2217(02)00123-6.

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

78 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.

79 Alexander Martin. *Integer programs with block structure*. PhD thesis, Zuse Institute Berlin, 1999.

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

81 Martin Nägele, Richard Santiago, and Rico Zenklusen. Congruency-constrained TU problems beyond the bimodular case. In *SODA 2022*, pages 2743–2790. SIAM, 2022. doi:10.1137/1.9781611977073.108.

82 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

83 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.

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

85 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.

86 Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, 1981. doi:10.1145/322276.322287.

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

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

89 Harilaos N. Psaraftis. A dynamic programming approach for sequencing groups of identical jobs. *Oper. Res.*, 28(6):1347–1359, 1980. `doi:10.1287/OPRE.28.6.1347`.

90 Victor Reis and Thomas Rothvoss. The subspace flatness conjecture and faster integer programming. In *FOCS 2023*, pages 974–988. IEEE, 2023. `doi:10.1109/FOCS57990.2023.00060`.

91 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`.

92 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.

93 Éva Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986. `doi:10.1287/OPRE.34.2.250`.

94 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. `doi:10.1016/J.ORL.2015.11.009`.

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

96 J. M. van den Akker, J. A. Hoogeveen, and Steef L. van de Velde. Parallel machine scheduling by column generation. *Oper. Res.*, 47(6):862–872, 1999. `doi:10.1287/OPRE.47.6.862`.

97 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. URL: `http://www.aaai.org/Library/ICAPS/2005/icaps05-032.php`.

98 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. `doi:10.1016/J.ORL.2005.05.009`.

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