Solving Integer Linear Programs with a Small Number of Global Variables and Constraints

Pavel Dvořák[1], **Eduard Eiben**[2], **Robert Ganian**[2], **Dušan Knop**[1], and **Sebastian Ordyniak**[2]
[1] Charles University, Prague, Czech Republic
[2] TU Wien, Vienna, Austria
koblich@iuuk.mff.cuni.cz, knop@kam.mff.cuni.cz, {eiben,ganian,ordyniak}@ac.tuwien.ac.at

Abstract

Integer Linear Programming (ILP) has a broad range of applications in various areas of artificial intelligence. Yet in spite of recent advances, we still lack a thorough understanding of which structural restrictions make ILP tractable. Here we study ILP instances consisting of a small number of "global" variables and/or constraints such that the remaining part of the instance consists of small and otherwise independent components; this is captured in terms of a structural measure we call *fracture backdoors* which generalizes, for instance, the well-studied class of N-fold ILP instances.

Our main contributions can be divided into three parts. First, we formally develop fracture backdoors and obtain exact and approximation algorithms for computing these. Second, we exploit these backdoors to develop several new parameterized algorithms for ILP; the performance of these algorithms will naturally scale based on the number of global variables or constraints in the instance. Finally, we complement the developed algorithms with matching lower bounds. Altogether, our results paint a near-complete complexity landscape of ILP with respect to fracture backdoors.

1 Introduction

Integer Linear Programming (ILP) is the archetypical representative of an NP-complete optimization problem and has a broad range of applications in various areas of artificial intelligence. In particular, a wide variety of problems in artificial intelligence are efficiently solved in practice via a translation into an ILP, including problems from areas such as planning [van den Briel *et al.*, 2005; Vossen *et al.*, 1999], process scheduling [Floudas and Lin, 2005], packing [Lodi *et al.*, 2002], vehicle routing [Toth and Vigo, 2001], and network hub location [Alumur and Kara, 2008].

In spite of recent advances [Ganian and Ordyniak, 2016; Ganian *et al.*, 2017; Jansen and Kratsch, 2015], we still lack a deep understanding of which structural restrictions make ILP tractable. The goal of this line of research is to identify structural properties (formally captured by a numerical *structural parameter* k) which allow us to solve ILP efficiently. In particular, one seeks to either solve an ILP instance $\mathbf{I}$ in time $f(k) \cdot |\mathbf{I}|^{\mathcal{O}(1)}$ (a so-called *fixed-parameter algorithm*), or at least in time $|\mathbf{I}|^{f(k)}$ (a so-called *XP algorithm*), where f is a computable function. This approach lies at the core of the now well-established *parameterized complexity* paradigm [Downey and Fellows, 2013; Cygan *et al.*, 2015] and has yielded deep results capturing the tractability and intractability of numerous prominent problems in diverse areas of computer science—such as Constraint Satisfaction, SAT, and a plethora of problems on directed and undirected graphs.

In general, structural parameters can be divided into two groups based on the way they are designed. *Decompositional parameters* capture the structure of instances by abstract tools called decompositions; treewidth is undoubtedly the most prominent example of such a parameter, and previous work has obtained a detailed complexity map of ILP with respect to the treewidth of natural graph representations of instances [Ganian and Ordyniak, 2016; Ganian *et al.*, 2017]. On the other hand, *backdoors* directly measure the "distance to triviality" of an instance: the number of simple operations required to put the instance into a well-defined, polynomially tractable class. While the backdoor approach has led to highly interesting results for problems such as Constraint Satisfaction [Gaspers *et al.*, 2017] and SAT [Gaspers and Szeider, 2012], it has so far been left mostly unexplored in the arena of ILP.

1.1 Our Contribution

Here, we initiate the study of backdoors to triviality for ILP by analyzing backdoors which fracture the instance into small, easy-to-handle components. Such *fracture backdoors* can equivalently be viewed as measuring the number of global variables or global constraints in an otherwise "compact" instance; in fact, we identify and analyze three separate cases depending on whether we allow global variables only, global constraints only, or both. We obtain a near-complete complexity landscape for the considered parameters: in particular, we identify the circumstances under which they can be used to obtain fixed-parameter and XP algorithms for ILP, and otherwise prove that such algorithms would violate well-established complexity assumptions. Our results are summarized in the following Table 1 (formal definitions are given in Section 3).

As is evident from the table, backdoor size on its own is not sufficient to break the NP-hardness of ILP; this is far from

	Variable	Constraint	Mixed
param.	FPT (Cor. 9)	FPT (Cor. 9)	XP (Cor. 8)
unary	pNP-c (Th 13)	XP, W[1]-h (Th 12, 14)	pNP-c (Th 13)
arbitrary	pNP-c	pNP-c (Th 15)	pNP-c

Table 1: Complexity landscape for fracture backdoors. Columns distinguish whether we consider variable backdoors, constraint backdoors, or mixed backdoors. Rows correspond to restrictions placed on coefficients in the ILP instance.

surprising, and the same situation arose in previous work on treewidth. However, while positive results on treewidth (as well as other considered decompositional parameters such as *torso-width* [Ganian *et al.*, 2017]) required the imposition of domain restrictions on variables, in the case of backdoors one can also deal with instances with unrestricted variable domains—by instead restricting the values of coefficients which appear in the ILP instance. Here, we distinguish three separate cases (corresponding to three rows in Table 1): coefficients bounded by the parameter value, coefficients which are encoded in unary, and no restrictions. It is worth noting that in the case of treewidth, ILP remains NP-hard even when coefficients are restricted to ± 1 and 0.

Our results in row 1 represent a direct generalization of three extensively studied classes of ILP, specifically n-fold ILP, two-stage stochastic ILP and 4-block N-fold ILP [De Loera *et al.*, 2013; Onn, 2010]. The distinction lies in the fact that while in the case of all three previously mentioned special cases of ILP the ILP matrix must be completely uniform outside of its global part, here we impose no such restriction. The only part of our complexity landscape which remains incomplete, the case of mixed backdoors combined with bounded coefficients, then corresponds to resolving a challenging open problem in the area of N-folds: the fixed-parameter (in)tractability of 4-block N-fold ILP [Hemmecke *et al.*, 2010]. A fixed-parameter algorithm for 4-block N-fold would also provide significant algorithmic improvements for problems in areas such as social choice [Knop *et al.*, 2017].

In the intermediate case of coefficient values encoded in unary (row 2), we surprisingly show that ILP remains polynomially tractable when the number of global constraints is bounded by a constant, but becomes NP-hard if we use global variables instead. To be precise, we obtain an XP algorithm parameterized by constraint backdoors, rule out the existence of a fixed-parameter algorithm for this case, and also rule out XP algorithms for variable and mixed backdoors. These also represent our most technical results: especially the XP algorithm requires the combination of deep linear-algebraic techniques with tools from the parameterized complexity toolbox.

Last but not least, all our algorithmic results first require us to compute a fracture backdoor. It turns out that computing fracture backdoors in ILP is closely related to solving the VERTEX INTEGRITY problem [Drange *et al.*, 2016] on bipartite graphs; unfortunately, while the problem has been studied on numerous graph classes including cobipartite graphs, its complexity remained open on bipartite graphs. Here we obtain both an exact fixed-parameter algorithm as well as a polynomial time approximation algorithm for finding fracture backdoors. As an additional result, we also show that the problem is NP-complete using a novel reduction.

The paper is structured as follows. After introducing the necessary notions and notation in the preliminaries, we proceed to formally define our parameter and develop algorithms for computing the desired backdoors. We then present our results separated by the type of restrictions put on the size of the matrix coefficients in the remaining sections.

2 Preliminaries

We will use standard graph terminology, see for instance the textbook by Diestel [2012]. In the following let $\mathbf{A}$ be a $n \times m$ matrix and let C and R be a subset of columns and rows of $\mathbf{A}$, respectively. We denote by $\mathbf{A}_{(R,C)}$ the submatrix of $\mathbf{A}$ restricted to the columns in C and the rows in R. We also denote by $\mathbf{A}_{(*,C)}$ and $\mathbf{A}_{(R,*)}$ the submatrix of $\mathbf{A}$ restricted to the columns in C and the submatrix of $\mathbf{A}$ restricted to the rows in R, respectively. We denote by $c_{\mathbf{A}}$ the maximum absolute value of any entry of $\mathbf{A}$. For a vector $\mathbf{b}$ of size n, we will use $\mathbf{b}[i]$ to denote its i-th entry and we denote by $c_{\mathbf{b}}$ the maximum absolute value of any entry of $\mathbf{b}$.

2.1 Integer Linear Programming

For our purposes, it will be useful to consider ILP instances which are in *equation form*. Formally, let an ILP instance $\mathbf{I}$ be a tuple $(\mathbf{A}, \mathbf{x}, \mathbf{b}, \mathbf{l}, \mathbf{u}, \eta)$, where $\mathbf{A}$ is a $m \times n$ matrix of integers (the *constraint matrix*), $\mathbf{x}$ is a vector of *variables* of size n, $\mathbf{b}$ is an integer vector of size m (the *right-hand side*), $\mathbf{l}, \mathbf{u}$ are vectors of n elements of $\mathbb{Z} \cup \{\pm\infty\}$ (the *lower* and *upper bounds*, respectively), and η is an integer vector of size n (the *optimization function*). Let A be the i-th row of $\mathbf{A}$; then we will call $A\mathbf{x} = \mathbf{b}[i]$ a *constraint* of $\mathbf{I}$. We will use var($\mathbf{I}$) to denote the set of *variables* (i.e., the elements of $\mathbf{x}$), and $\mathcal{F}(\mathbf{I})$ (or just $\mathcal{F}$) to denote the set of constraints. For a subset U of var($\mathbf{I}$) $\cup$ $\mathcal{F}(\mathbf{I})$, we denote by $C(U)$ the columns of $\mathbf{A}$ corresponding to variables in U and by $R(U)$ the rows of $\mathbf{A}$ corresponding to constraints in U.

A (partial) assignment α is a mapping from some subset of var($\mathbf{I}$), denoted by var(α), to $\mathbb{Z}$. An assignment α is called *feasible* if (1) it satisfies every constraint in $\mathcal{F}$, i.e., if $A\alpha(\mathbf{x}) = \mathbf{b}[i]$ for each i-th row A of $\mathbf{A}$, and (2) it satisfies all the upper and lower bounds, i.e., $\mathbf{l}[i] \leq \alpha(\mathbf{x}[i]) \leq \mathbf{u}[i]$. Furthermore, α is called a *solution* if the value of $\eta\alpha(\mathbf{x})$ is maximized over all feasible assignments; observe that the existence of a feasible assignment does not guarantee the existence of a solution (there may exist an infinite sequence of feasible assignments α with increasing values of $\eta\alpha(\mathbf{x})$; in this case, we speak of *unbounded* instances). Given an instance $\mathbf{I}$, the task in the ILP problem is to compute a solution for $\mathbf{I}$ or correctly determine that no solution exists. We remark that other formulations of ILP exist (e.g., a set of inequalities over variables); it is well-known that these are equivalent and can be transformed into each other in polynomial time. Moreover, such transformations will only change our parameters (defined in Section 3) by a constant factor.

Aside from general integer linear programming, we will also be concerned with two subclasses of the problem. ILP-FEASIBILITY is formulated equivalently as ILP, with the restriction that η must be the 0-vector. UNARY ILP is the class

of all ILP instances which are supplied in a unary bit encoding; in other words, the input size of UNARY ILP upper-bounds not only the number of variables and constraints, but also the absolute values of all numbers in the input. UNARY ILP remains NP-complete in general, but in our setting there will be cases where its complexity will differ from general ILP. Combining both restrictions gives rise to UNARY ILP-FEASIBILITY.

There are several ways of naturally representing ILP instances as graphs. The representation that will be most useful for our purposes will be the so-called *incidence graph*: the incidence graph $G_{\mathbf{I}}$ of an ILP instance $\mathbf{I}$ is the graph whose vertex set is $\text{var}(\mathbf{I}) \cup \mathcal{F}(\mathbf{I})$ and two vertices s, t are adjacent iff $s \in \text{var}(\mathbf{I})$, $t \in \mathcal{F}$ and s occurs in t with a non-zero coefficient. An instance $\mathbf{I}'$ is a *connected component* of $\mathbf{I}$ if it is the subinstance of $\mathbf{I}$ corresponding to a connected component of $G_{\mathbf{I}}$; formally, $\mathcal{F}(\mathbf{I}') \subseteq \mathcal{F}(\mathbf{I})$ is the set of constraints that occur in a connected component of $G_{\mathbf{I}}$ and $\eta(\mathbf{I}')$ is the restriction of $\eta(\mathbf{I})$ to $\text{var}(\mathcal{F}(\mathbf{I}'))$. For a set $Z \subseteq \mathcal{F}(\mathbf{I}) \cup \text{var}(\mathbf{I})$, we will also use $\mathbf{I} \setminus Z$ to denote the ILP instance obtained by removing all constraints in Z from $\mathcal{F}(\mathbf{I})$ and removing all variables in Z from all constraints in $\mathcal{F}(\mathbf{I}) \setminus Z$ and from η.

2.2 Parameterized Complexity

In parameterized algorithmics [Flum and Grohe, 2006; Niedermeier, 2006; Downey and Fellows, 2013] the runtime of an algorithm is studied with respect to a parameter $k \in \mathbb{N}$ and input size n. The basic idea is to find a parameter that describes the structure of the instance such that the combinatorial explosion can be confined to this parameter. In this respect, the most favorable complexity class is FPT (*fixed-parameter tractable*) which contains all problems that can be decided by an algorithm running in time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is a computable function. Problems that can be solved in this time are called *fixed-parameter tractable* (fpt). Aside from the complexity class FPT, we will also make use of the complexity classes W[1] (proving hardness for W[1] is often used to show that a problem is unlikely to be in FPT), XP (containing problems that can be solved in polynomial time whenever the parameter is bounded) and pNP (problems complete for pNP remain NP-complete even when the parameter is bounded). To obtain our lower bounds, we will need the notion of *parameterized reduction*, see for instance Downey and Fellows (2013).

2.3 ILP with Structured Matrices

Our results build on and extend the classical variable-dimension ILP techniques detailed for instance in the work of De Loera *et al.*; Onn; Hemmecke *et al.*. Below, we provide a basic introduction to these techniques and related results.

Let $\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{pmatrix}$ be a 2×2 block integer matrix. The *N-fold 4-block product of* $\mathbf{A}$ (denoted by $\mathbf{A}^{(N)}$) is the following integer matrix

$$\mathbf{A}^{(N)} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \mathbf{A}_2 & \cdots & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 & 0 & \cdots & 0 \\ \mathbf{A}_3 & 0 & \mathbf{A}_4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_3 & 0 & 0 & \cdots & \mathbf{A}_4 \end{pmatrix}.$$

Here $\mathbf{A}_1$ is an $r \times s$ matrix, $\mathbf{A}_2$ is an $r \times t$ matrix, $\mathbf{A}_3$ is an $u \times s$ matrix, and $\mathbf{A}_4$ is an $u \times t$ matrix; for convenience, we let $b_{\mathbf{A}} = \max(r, s, t, u)$. We call an instance $(\mathbf{A}, \mathbf{x}, \mathbf{b}, \mathbf{l}, \mathbf{u}, \eta)$ of ILP an *N-fold 4-block* if $\mathbf{A}$ is an N-fold 4-block product of some 2×2 block integer matrix. Observe that in such instances the vector $\mathbf{x}$ is naturally partitioned into a global part (consisting of s variables) and a local part.

Theorem 1 (Hemmecke *et al.* 2010)**.** *Let a and z be constants and let $\mathbf{I}$ be an N-fold 4-block ILP instance with $c_{\mathbf{A}} \leq a$, $b_{\mathbf{A}} \leq z$, then $\mathbf{I}$ can be solved in polynomial time.*

In the parameterized complexity setting, the above theorem yields an XP algorithm solving ILP parameterized by $\max(b_{\mathbf{A}}, c_{\mathbf{A}})$ if the matrix is a N-fold 4-block product. We note that the existence of a fixed-parameter algorithm for this problem remains a challenging open problem [Hemmecke *et al.*, 2010]. However, the problem is known to be fixed-parameter tractable when either $\mathbf{A}_1$ and $\mathbf{A}_3$ or $\mathbf{A}_1$ and $\mathbf{A}_2$ are omitted; these variants are called the *N-fold ILP* problem and the *2-stage stochastic ILP* problem, respectively.

Theorem 2 (Hemmecke *et al.* 2013, De Loera *et al.* 2013)**.** *N-fold ILP and 2-stage stochastic ILP are fpt parameterized by $c_{\mathbf{A}}$ and $b_{\mathbf{A}}$.*

3 The Fracture Number

We are now ready to formally introduce the studied parameter and related notions. An ILP instance $\mathbf{I}$ is called *ℓ-compact* if each connected component of $\mathbf{I}$ contains at most ℓ variables and constraints; equivalently, each connected component of $G_{\mathbf{I}}$ contains at most ℓ vertices. It is not difficult to observe that any ℓ-compact ILP instance is fixed-parameter tractable parameterized by ℓ; indeed, using the algorithm of Lenstra, Jr. (1983) we can compute a solution for each component individually and then combine the solutions to solve $\mathbf{I}$.

A set $Z \subseteq \mathcal{F} \cup \text{var}(\mathbf{I})$ is called a *backdoor to ℓ-compactness* if $\mathbf{I} \setminus Z$ is ℓ-compact; moreover, if $Z \cap \mathcal{F} = \emptyset$ then Z is called a *variable-backdoor to ℓ-compactness*, and if $Z \cap \text{var}(\mathbf{I}) = \emptyset$ then Z is a *constraint-backdoor to ℓ-compactness*. We use $b_\ell(\mathbf{I})$ to denote the cardinality of a minimum backdoor to ℓ-compactness, and similarly $b_\ell^V(\mathbf{I})$ and $b_\ell^C(\mathbf{I})$ for variable-backdoors and constraint-backdoors to ℓ-compactness, respectively. It is easy to see that, depending on the instance, $b_\ell^V(\mathbf{I})$ can be arbitrarily larger or smaller than $b_\ell^C(\mathbf{I})$. On the other hand, $b_\ell(\mathbf{I}) \leq \min(b_\ell^V(\mathbf{I}), b_\ell^C(\mathbf{I}))$.

Clearly, the choice of ℓ has a major impact on the size of backdoors to ℓ-compactness; in particular, $b_\ell(\mathbf{I})$ could be arbitrarily larger than $b_{\ell+1}(\mathbf{I})$, and the same of course also holds for variable- and constraint-backdoors. Since we will be interested in dealing with cases where both ℓ and $b_\ell(\mathbf{I})$ are small, we will introduce the *fracture number* $\mathfrak{p}$ which provides bounds on both ℓ and b_ℓ; in particular, we let $\mathfrak{p}(\mathbf{I}) = \min_{\ell \in \mathbb{N}}(\max(\ell, b_\ell(\mathbf{I})))$. Furthermore, we say that a backdoor *witnesses* $\mathfrak{p}(\mathbf{I})$ if $|Z| \leq \mathfrak{p}(\mathbf{I})$ and $\mathbf{I} \setminus Z$ is $\mathfrak{p}(\mathbf{I})$-compact. We define $\mathfrak{p}^C(\mathbf{I})$ and $\mathfrak{p}^V(\mathbf{I})$ similarly, with $b_\ell(\mathbf{I})$ replaced by $b_\ell^C(\mathbf{I})$ and $b_\ell^V(\mathbf{I})$, respectively. If the instance $\mathbf{I}$ is clear from the context, we omit the reference to $\mathbf{I}$; see Figure 1 for an example.

We remark that the fracture number represents a strict generalization of the parameter $b_{\mathbf{A}}$ used in Theorems 1 and 2; in

$$\texttt{maximize} \sum_{\texttt{i}=1}^{7} \texttt{i} \cdot \texttt{x}_\texttt{i}, \quad \text{where}$$
$$\sum_{\texttt{i}=1}^{7} \texttt{x}_\texttt{i} = 32,$$
$$1\texttt{x}_1 + \texttt{y} = 6, \qquad 2\texttt{x}_2 + \texttt{y} = 9,$$
$$3\texttt{x}_3 + \texttt{y} = 14, \qquad 4\texttt{x}_4 + \texttt{y} = 21,$$
$$5\texttt{x}_5 + \texttt{y} = 30, \qquad 6\texttt{x}_6 + \texttt{y} = 41.$$

Figure 1: The constraints and optimization function of a simple ILP instance with $\mathfrak{p} = 2$, witnessed by a backdoor containing y and the first constraint.

particular, $\mathfrak{p} \leq 2b_{\mathbf{A}}$ (and similarly for $\mathfrak{p}^V$ and $\mathfrak{p}^C$ for the latter two theorems). Moreover, the fracture number is well-defined for all ILP instances, not only for N-fold 4-block products. In this respect, N-fold 4-block products with bounded $b_{\mathbf{A}}$ form the subclass of instances with bounded $\mathfrak{p}$ such that each component *must contain precisely the same submatrix*. It is not difficult to see that this is indeed a very strong restriction.

4 Computing the Fracture Number

Our evaluation algorithms for ILP require a backdoor set as part of their input. In this section we show how to efficiently compute small backdoor sets, i.e., we show how to solve the following problem.

FRACTURE BACKDOOR DETECTION (BD)	
Instance:	An ILP instance $\mathbf{I}$ and a natural number k.
Parameter:	k
Question:	Determine whether $\mathfrak{p}(\mathbf{I}) \leq k$ and if so output a backdoor set witnessing this.

We also define the variants V-BD and C-BD that are concerned with finding a variable or a constraint backdoor, respectively, in the natural way. We note that BD and its variants are closely related to the so-called VERTEX INTEGRITY problem on bipartite graphs [Drange *et al.*, 2016]. We use this connection to obtain our complexity result for BD and as a side result show that VERTEX INTEGRITY is NP-complete even on bipartite graphs, which has not been known so far.

Theorem 3. BD*,* V-BD*, and* C-BD *are* NP*-complete.*

Proof Sketch. The proof follows from a reduction from 3-SAT where every literal occurs in exactly two clauses; this variant is well-known to be NP-complete [Garey and Johnson, 1979]). At its core, the reduction utilizes variable gadgets as well as clause gadgets, and the main difficulty lies in designing these in order to ensure that the graph remains bipartite. □

Even though BD is NP-complete, here we provide two efficient algorithms for solving it: we show that the problem is fixed-parameter tractable parameterized by k and can be approximated in polynomial time within a factor of k. Both of these algorithms are based on the observation that any backdoor has to contain at least one vertex from every connected subgraph of the instance of size $k + 1$.

Theorem 4. BD*,* V-BD*, and* C-BD *can be solved in time* $\mathcal{O}((k+1)^k|E(G)|)$ *and are hence fpt.*

Theorem 5. BD*,* V-BD*, and* C-BD *can be approximated in polynomial time within a factor of* $k + 1$.

5 The Case of Bounded Coefficients

The goal of this section is to obtain the algorithmic results presented on the first row of Table 1. Recall that in this case we will be parameterizing also by $c_{\mathbf{A}}$, which is the maximum absolute coefficient occurring in $\mathbf{A}$. Before we proceed to the results themselves, we first need to introduce a natural notion of "equivalence" among the components of an ILP instance.

Let Z be a backdoor to ℓ-compactness for an ILP instance $\mathbf{I}$. We define the equivalence relation $\sim$ on the components of $\mathbf{I} \setminus Z$ as follows: two components C_1 and C_2 are equivalent iff there exists a bijection γ between $\text{var}(C_1)$ and $\text{var}(C_2)$ such that the ILP instance obtained from $\mathbf{I}$ after renaming the variables in $\text{var}(C_1)$ and $\text{var}(C_2)$ according to γ and γ^{-1}, respectively, is equal to $\mathbf{I}$. We say that components C_1 and C_2 have the same *type* if $C_1 \sim C_2$.

Lemma 6. *Let* $\mathbf{I}$ *be an ILP instance and* $k = \mathfrak{p}(\mathbf{I})$*. For any backdoor witnessing* $\mathfrak{p}(\mathbf{I})$*,* $\sim$ *has at most* $\big(2c_{\mathbf{A}}(\mathbf{I}) + 1\big)^{2k^2}$ *equivalence classes. Moreover, one can test whether two components have the same type in time* $\mathcal{O}(k!k^2)$*.*

We now proceed to the main tool used for our algorithms.

Theorem 7. *Let* $\overline{\mathbf{I}}$ *be an ILP instance with matrix* $\overline{\mathbf{A}}$*,* Z *be a backdoor set witnessing* $\mathfrak{p}(\overline{\mathbf{I}})$*, and let* n *be the number of components of* $\overline{\mathbf{I}} \setminus Z$*. There is an algorithm which runs in time* $\mathcal{O}(n^2(\mathfrak{p}(\mathbf{I})+1)! + |\mathbf{I}|)$ *and computes a* $(r+u) \times (s+t)$ *matrix* $\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{pmatrix}$*, a positive integer* $N \leq n$*, and a* 4*-block* N*-fold instance* $\mathbf{I} = (\mathbf{A}^{(N)}, \mathbf{x}, \mathbf{b}, \mathbf{l}, \mathbf{u}, \eta)$ *such that:*

(P1) any solution for $\mathbf{I}$ *can be transformed (in polynomial time) into a solution for* $\overline{\mathbf{I}}$ *(and vice versa), and*

(P2) $\max\{r, s\} \leq \mathfrak{p}(\overline{\mathbf{I}})$ *and* $\max\{t, u\} \leq f\big(c_{\overline{\mathbf{A}}}, \mathfrak{p}(\overline{\mathbf{I}})\big)$ *for some computable function* f*.*

Proof Sketch. For each component C of $\overline{\mathbf{I}} \setminus Z$, we define the following three matrices:

- The matrix $\mathbf{Q}_C^V$ is the part of the constraints in C dealing with variables in Z, that is, $\overline{\mathbf{A}}_{\mathcal{F}(C),\text{var}(Z)}$,
- the matrix $\mathbf{Q}_C^C$ is the part of the constraints in Z dealing with $\text{var}(C)$, that is, $\overline{\mathbf{A}}_{\mathcal{F}(Z),\text{var}(C)}$, and
- the matrix $\mathbf{Q}_C$ is the part of the constraints in C dealing with $\text{var}(C)$, that is, $\overline{\mathbf{A}}_{\mathcal{F}(C),\text{var}(C)}$.

The algorithm starts by computing the equivalence classes of $\sim$ in time at most $O(n^2(\mathfrak{p}(\mathbf{I})+1)!)$ using Lemma 6. Observe that if $C \sim D$ then $(\mathbf{Q}_C, \mathbf{Q}_C^V, \mathbf{Q}_C^C) = (\mathbf{Q}_D, \mathbf{Q}_D^V, \mathbf{Q}_D^C)$ (modulo rearrangement of columns and rows). The multiplicity $\text{mult}(T)$ of a type T is the number of components in $\overline{\mathbf{I}} \setminus Z$ of type T. Let N be the maximum multiplicity of any type of $\overline{\mathbf{I}}$.

Observe that if all types already have the same multiplicity then we can simply set $\mathbf{I} = \overline{\mathbf{I}}$. Indeed, it suffices to rearrange the matrices $\mathbf{Q}_C$, $\mathbf{Q}_C^C$, and $\mathbf{Q}_C^V$ of all components into the matrices $\mathbf{A}_4$, $\mathbf{A}_2$, and $\mathbf{A}_3$ as follows. We arrange the matrices $\mathbf{Q}_C$ of all components on a diagonal of the matrix $\mathbf{A}_4$, the matrices $\mathbf{Q}_C^C$ next to each other in the matrix $\mathbf{A}_2$, and the matrices $\mathbf{Q}_C^V$ under each other in the matrix $\mathbf{A}_3$. The bounds on r, s, t and u then follow from Lemma 6.

On the other hand, if the multiplicities of types vary then we apply the following procedure for each type T with multiplicity lower than N. We first pick one component C of type T and add $N - \mathrm{mult}(T)$ new copies of C to $\overline{\mathbf{I}}$. Let O be the set of all original components of type T and let S be the set of all newly created copies of C. For every variable of every component in S we set its lower and upper bound to 0; this ensures that components in S will have no impact on the global constraints and the optimization function.

Next, for every component in $O \cup S$ we introduce one new auxiliary copy of every variable in that component. These variable-copies have the same coefficients in $\overline{\mathbf{A}}$ as the originals, but they occur neither in the global constraints nor in the optimization function. Formally, we have extended the type T to $(\mathbf{Q}_C^V, [\mathbf{Q}_C^C \mid 0], [\mathbf{Q}_C \mid \mathbf{Q}_C])$.

Finally, for components in O we set the lower and upper bounds of the variable-copies to 0 (thereby nullifying the variable-copies in these components), and for components in S we set the lower and upper bounds of the variable copies to the same values as the corresponding original variable in C (allowing them to "replace" the original variables in S). The end result is that any solution for $\overline{\mathbf{I}}$ can be adapted to a solution for $\mathbf{I}$ by reusing its assignment in C for all components in S *without affecting η and constraints in Z*. It is straightforward to show that (P1) is satisfied, moreover (P2) follows as in the case that all components had the same multiplicity since we at most double the number of variables of any component. □

The algorithmic consequences of Theorem 7 together with Theorems 1, 4 and 2 are the following corollaries.

Corollary 8. *Let a and z be constants and let $\mathbf{I}$ be an ILP instance with $c_{\mathbf{A}}(\mathbf{I}) \leq a$ and $\mathfrak{p}(\mathbf{I}) \leq z$, then $\mathbf{I}$ can be solved in polynomial time.*

Corollary 9. *ILP is fpt when parameterized by $\max\{c_{\mathbf{A}}, \mathfrak{p}^V\}$ and also when parameterized by $\max\{c_{\mathbf{A}}, \mathfrak{p}^C\}$.*

6 Unary ILP

Here we will prove that UNARY ILP is polynomial-time solvable when $\mathfrak{p}^C$ is bounded by a constant; this contrasts the case of general ILP, which remains NP-hard in this case (see Theorem 15 later). In particular, we will give an XP algorithm for UNARY ILP parameterized by $\mathfrak{p}^C$. We will also present lower bounds showing that such an algorithm cannot exist for UNARY ILP parameterized by $\mathfrak{p}^V$ or $\mathfrak{p}$, and rule out the existence of a fixed-parameter algorithm for $\mathfrak{p}^C$.

6.1 The Algorithm

The crucial, and also most technically demanding, part of this result is showing that it suffices to restrict our search space to assignments over polynomially bounded variable domains. We will first need the following technical lemma.

Lemma 10. *Let $\mathbf{I}$ be an instance of* UNARY ILP *with matrix $\mathbf{A}$. Then for any set D of linearly dependent columns of $\mathbf{A}$, it holds that $\mathbf{A}_{(*,D)}$ contains a subset of at most $\mathfrak{p}^C(\mathbf{I})(\mathfrak{p}^C(\mathbf{I})+1)$ linearly dependent columns.*

We are now ready to show that we only need to consider solutions with polynomially bounded variable domain.

Lemma 11. *Let $\mathbf{I}$ be a feasible instance of* UNARY ILP-FEASIBILITY *of size n. Then, there exists a solution α with $|\alpha(v)| \leq m_L$ for every $v \in \mathrm{var}(\mathbf{I})$, where $m_L = 8\big(2(\mathfrak{p}^C(\mathbf{I})+2)^2\big)!(n)^{2(\mathfrak{p}^C(\mathbf{I})+2)^2}$.*

Proof Sketch. We start with a constraint backdoor Z witnessing $\mathfrak{p}^C$ and a solution α for $\mathbf{I} = (\mathbf{A}, \mathbf{x}, \mathbf{b}, \mathbf{l}, \mathbf{u}, \eta)$ that minimizes the number of variables that are assigned "large" values, values exceeding some bound $2m_{\mathrm{S}}$ with $2m_{\mathrm{S}} < m_{\mathrm{L}}$. We then consider the set V of all variables whose value assigned by α exceeds $2m_{\mathrm{S}}$ and distinguish two cases: either the columns $C(V)$ of $\mathbf{A}$ are linearly dependent or not. In the former case there is a non-zero vector $\mathbf{a}$ with $|V|$ entries such that $\mathbf{A}_{(*,C(V))}\mathbf{a} = 0$. This implies that $\alpha_\Delta(\mathbf{x}) = \alpha(\mathbf{x}) + \Delta\mathbf{a}'$ satisfies $\mathbf{A}\alpha_\Delta(\mathbf{x}) = \mathbf{b}$ for every Δ, where $\mathbf{a}'$ is the extension of $\mathbf{a}$ to all variables in $\mathbf{x}$ by zero entries. The aim is now to show that there exist $\mathbf{a}$ and Δ such that $\alpha_\Delta(\mathbf{x})$ is a feasible solution for $\mathbf{I}$ with less than $|V|$ variables exceeding $2m_{\mathrm{S}}$, thereby obtaining a contradiction to our choice of α. The most difficult part here is to show that there is a vector $\mathbf{a}$ whose absolute integer values do not exceed m_{S}. As a first step we employ Lemma 10 to obtain a subset Y of $C(V)$ that is minimally linearly dependent and has size at most $\mathfrak{p}^C(\mathbf{I})(\mathfrak{p}^C(\mathbf{I})+1)$. Because for every $y \in Y$ it holds that $Y \setminus \{y\}$ is a set of linearly independent columns, we can employ Cramer's Rule [Schrijver, 1998] on $\mathbf{A}_{R,Y\setminus\{y\}}\mathbf{a}'' = \mathbf{a}_y\mathbf{A}_{R,\{y\}}$, where R is a set of $|Y|-1$ linearly independent rows of $\mathbf{A}_{*,Y\setminus\{y\}}$, $\mathbf{a}''$ is the projection of $\mathbf{a}$ to $Y \setminus \{y\}$, and $\mathbf{a}_y$ is the entry of $\mathbf{a}$ corresponding to y. This allows us to obtain the required bounds for $\mathbf{a}$.

Hence it remains to deal with the case that the columns in $C(V)$ are linearly independent. Then there is a set R of $|V|$ linearly independent rows in $\mathbf{A}_{(*,C(V))}$ and the matrix $\mathbf{A}_{(R,C(V))}$ is non-singular. Consider the set S of components in $\mathbf{I} \setminus Z$ that have at least one variable or constraint corresponding to a column or row in $\mathbf{A}_{(R,C(V))}$. Let $C_1, \ldots, C_p$ be the restrictions of all components in S to variables and constraints corresponding to columns or rows in $\mathbf{A}_{(R,C(V))}$. Then because the rows in $R(C_i)$ are linearly independent, we obtain that C_i has at least as many variables as constraints. Moreover, because $\mathbf{A}_{(R,C(V))}$ is a square matrix there can be at most $|Z|$ components from $C_1, \ldots, C_p$ that have more variables than constraints. Let C_i be a component that has the same number of variables as constraints and let D_i be the unique component of $\mathbf{I} \setminus Z$ containing C_i. Observe that all non-zero entries of $\mathbf{A}_{(R(C_i),*)}$ correspond to variables in D_i. Hence α has to satisfy $\mathbf{A}_{(R(C_i),C(D_i))}\alpha(\mathbf{x}') = \mathbf{b}'$, where $\mathbf{x}'$ and $\mathbf{b}'$ are the restrictions of $\mathbf{x}$ and $\mathbf{b}$ respectively to the entries corresponding to rows in $R(C_i)$. Because $\mathbf{A}_{(R(C_i),C(C_i))}$ is non-singular, we obtain that the values of α for the variables of C_i are completely determined by the values of α for the variables of $D_i \setminus C_i$, which in turn are at most $2m_{\mathrm{S}}$ because $\mathrm{var}(D_i \setminus C_i) \subseteq \mathrm{var}(\mathbf{I}) \setminus V$. Using Cramer's Rule we calculate that $\alpha(v) \leq m_{\mathrm{M}}$ for every such variable v of C_i, where $2m_{\mathrm{S}} < m_{\mathrm{M}} < m_{\mathrm{L}}$. Thus we already obtained a bound on α for almost all variables of $\mathbf{I}$, the only variables that remain must be contained in some component C_i that has more variables than constraints. Let $C_1, \ldots, C_r$ be all such components from $C_1, \ldots, C_p$ and let $U = \bigcup_{1\leq i\leq r} \mathrm{var}(C_i)$.

As mentioned above there can be at most $|Z| \leq \mathfrak{p}^C(\mathbf{I})$ such components. Because $C(U)$ is a subset of linearly independent columns there is a set R' of $|U|$ linearly independent rows in $\mathbf{A}_{(*,C(U))}$. Then α has to satisfy $\mathbf{A}_{(R',*)}\alpha(\mathbf{x}) = \mathbf{b}$ and because $\mathbf{A}_{(R',C(U))}$ is non-singular, we obtain that the values of α for the variables in U are completely determined by the values of α for all remaining variables, which as shown above are at most m_{M}. Using Cramer's Rule we calculate that $\alpha(v) \leq m_{\mathrm{L}}$ for every such variable v in U. □

To complete the proof of the desired statement, we use a recent result of [Ganian *et al.*, 2017, Proposition 2 and Theorem 11] on solving ILP using treewidth (which is always at most $\mathfrak{p}$) and obtain:

Theorem 12. UNARY ILP *is polynomial-time solvable for any fixed value of* $\mathfrak{p}^C(\mathbf{I})$*, where* $\mathbf{I}$ *is the input instance.*

6.2 Lower Bounds

We complement our algorithm with matching lower bounds: strong NP-hardness for variable and mixed backdoors, W[1]-hardness in the case of constraint backdoors, and weak NP-hardness for constraint and mixed backdoors.

Theorem 13. UNARY ILP-FEASIBILITY *is* pNP*-hard parameterized by* $\mathfrak{p}^V(\mathbf{I})$.

Proof Sketch. We prove the theorem by a polynomial-time reduction from the well-known NP-hard 3-COLORABILITY problem [Garey and Johnson, 1979]: given a graph, decide whether the vertices of G can be colored with three colors such that no two adjacent vertices of G share the same color.

The main idea behind the reduction is to represent a 3-partition of the vertex set of G by the domain values of three "global" variables. The value of each of these global variables will represent a subset of vertices of G that will be colored using the same color. To represent a subset of the vertices of G in terms of domain values of the global variables, we will associate every vertex of G with a unique prime number and represent a subset by the value obtained from the multiplication of all prime numbers of vertices contained in the subset. To ensure that the subsets represented by the global variables correspond to a valid 3-partition of G we will introduce constraints which ensure that: (1) for every prime number representing some vertex of G exactly one of the global variables is divisible by that prime number (ensuring that every vertex of G is assigned to exactly one color class), and (2) for every edge $\{u, v\}$ of G, no global variable is divisible by the prime numbers representing u and v at the same time. □

Theorem 14. UNARY ILP-FEASIBILITY *is* W[1]*-hard parameterized by* $\mathfrak{p}^C(\mathbf{I})$.

Proof Sketch. We prove the theorem by a parameterized reduction from MULTICOLORED CLIQUE, which is well-known to be W[1]-complete [Pietrzak, 2003]. The main idea behind the reduction is to first guess one vertex from each part V_i and one edge between every two parts V_i and V_j and to then verify that the selected vertices and edges form a k-clique in G.

The first step is achieved by introducing one binary variable for every vertex and edge of G together with $2k + 2\binom{k}{2}$ global constraints that ensure that (1) exactly one of the variables representing the vertices in V_i is set to one and (2) exactly one of the variables representing the edges between V_i and V_j is set to one. The second step is achieved by identifying each vertex of G with a unique number such that the sum of any two numbers assigned to two vertices of G is unique. By identifying each edge of G with the sum of the numbers assigned to its endpoints, it is then possible to verify that the selected vertices and edges form a k-clique by checking whether the number assigned to the selected edge e is equal to the sum of the numbers assigned to the selected vertices in V_i and V_j. Sets of numbers for which the sum of every two numbers from the set is unique are also known as Sidon sequences. Using classical results by Erdös and Turán [1941] and Bertrand's postulate [Aigner *et al.*, 2010] we observe that a Sidon sequence with n elements whose largest element is at most $8n^2$ can be computed in polynomial time. □

Theorem 15. ILP *is* NP*-hard even if* $\mathfrak{p}^C = 1$.

7 Concluding Notes

In order to overcome the complexity barriers of ILP, a wide range of problems have been encoded in restricted variants of ILP such as 2-stage stochastic ILP and N-fold ILP; examples for the former include a range of transportation and logistic problems [Powell and Topaloglu, 2003; Hrabec *et al.*, 2015], while examples for the latter range from scheduling [Knop and Koutecký, 2016] to, e.g., computational social choice [Knop *et al.*, 2017]. Our framework provides a unified platform which generalizes 2-stage stochastic ILP, N-fold ILP and also 4-block N-fold ILP. Moreover, it represents a natural measure of the complexity of ILPs which can be applied to any ILP instance, including those which lie outside of the scope of all previously known algorithmic frameworks. In fact, one may view our algorithmic results as "algorithmic meta-theorems" for ILP, where previously known algorithms for 2-stage stochastic ILP, N-fold ILP and 4-block N-fold ILP only represent a simple base case.

Our algorithms are complemented with matching lower bounds showing that the considered restrictions are, in fact, necessary in order to obtain fixed-parameter or XP algorithms. The only remaining blank part in the presented complexity map is the question of whether mixed fracture backdoors admit a fixed-parameter algorithm in case of bounded coefficients; we consider this a major open problem in the area. A first step towards settling this question would be to resolve the fixed-parameter (in)tractability of 4-block N-fold ILP; progress in this direction seems to require new techniques and insights [Hemmecke *et al.*, 2010].

Acknowledgements

The authors acknowledge support by the Austrian Science Fund (FWF, project P26696), CE-ITI (project P202/12/G061 of GAČR). The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 616787. Robert Ganian is also affiliated with FI MU, Brno, Czech Republic.

References

[Aigner *et al.*, 2010] Martin Aigner, Günter M Ziegler, and Alfio Quarteroni. *Proofs from the Book*, volume 274. Springer, 2010.

[Alumur and Kara, 2008] 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.

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

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

[Diestel, 2012] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[Downey and Fellows, 2013] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[Drange *et al.*, 2016] Pål Grønås Drange, Markus S. Dregi, and Pim van 't Hof. On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76(4):1181–1202, 2016.

[Erdös and Turán, 1941] Paul Erdös and Pál Turán. On a problem of sidon in additive number theory, and on some related problems. *Journal of the London Mathematical Society*, 1(4):212–215, 1941.

[Floudas and Lin, 2005] ChristodoulosA. Floudas and Xiaoxia Lin. Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. *Annals of Operations Research*, 139(1):131–162, 2005.

[Flum and Grohe, 2006] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[Ganian and Ordyniak, 2016] Robert Ganian and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP. In *Proc. AAAI*, pages 710–716, 2016.

[Ganian *et al.*, 2017] Robert Ganian, Sebastian Ordyniak, and M. S. Ramanujan. Going beyond primal treewidth for (M)ILP. In *Proc. AAAI*, pages 815–821, 2017.

[Garey and Johnson, 1979] Michael R. Garey and David R. Johnson. *Computers and Intractability*. W. H. Freeman and Company, New York, San Francisco, 1979.

[Gaspers and Szeider, 2012] Serge Gaspers and Stefan Szeider. Backdoors to satisfaction. In *The Multivariate Algorithmic Revolution and Beyond*, volume 7370 of *LNCS*, pages 287–317. Springer Verlag, 2012.

[Gaspers *et al.*, 2017] Serge Gaspers, Neeldhara Misra, Sebastian Ordyniak, Stefan Szeider, and Stanislav Zivny. Backdoors into heterogeneous classes of SAT and CSP. *J. Comput. Syst. Sci.*, 85:38–56, 2017.

[Hemmecke *et al.*, 2010] Raymond Hemmecke, Matthias Köppe, and Robert Weismantel. A polynomial-time algorithm for optimizing over N-fold 4-block decomposable integer programs. In *Proc. IPCO*, pages 219–229, 2010.

[Hemmecke *et al.*, 2013] R. Hemmecke, S. Onn, and L. Romanchuk. n-fold integer programming in cubic time. *Math. Program*, 137(1-2):325–341, 2013.

[Hrabec *et al.*, 2015] Dušan Hrabec, Pavel Popela, Jan Roupec, Jan Mazal, and Petr Stodola. *Two-Stage Stochastic Programming for Transportation Network Design Problem*, page 1725. Springer International Publishing, Cham, 2015.

[Jansen and Kratsch, 2015] Bart M. P. Jansen and Stefan Kratsch. A Structural Approach to Kernels for ILPs: Treewidth and Total Unimodularity. In *Proc. ESA*, volume 9294 of *LNCS*, pages 779–791. Springer Verlag, 2015.

[Knop and Koutecký, 2016] Dušan Knop and Martin Koutecký. Scheduling meets n-fold integer programming. *CoRR*, abs/1603.02611, 2016.

[Knop *et al.*, 2017] Dušan Knop, Martin Koutecký, and Matthias Mnich. Voting and bribing in single-exponential time. In *Proc. STACS 2017*, volume 66 of *LIPIcs*, pages 46:1–46:14. Schloss Dagstuhl, 2017.

[Lenstra, Jr., 1983] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983.

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

[Niedermeier, 2006] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

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

[Pietrzak, 2003] Krzysztof Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. of Computer and System Sciences*, 67(4):757–771, 2003.

[Powell and Topaloglu, 2003] Warren B Powell and Huseyin Topaloglu. Stochastic programming in transportation and logistics. *Handbooks in operations research and management science*, 10:555–635, 2003.

[Schrijver, 1998] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

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

[van den Briel *et al.*, 2005] Menkes van den Briel, Thomas Vossen, and Subbarao Kambhampati. Reviving integer programming approaches for AI planning: A branch-and-cut framework. In *Proc. ICAPS*, pages 310–319, 2005.

[Vossen *et al.*, 1999] Thomas Vossen, Michael O. Ball, Amnon Lotem, and Dana S. Nau. On the use of integer programming models in AI planning. In *Proc. IJCAI*, pages 304–309. Morgan Kaufmann, 1999.