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

(koblich@iuuk.mff.cuni.cz, knop@kam.mff.cuni.cz)

[2] TU Wien, Vienna, Austria

({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 **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).

	Variable	Constraint	Mixed
param.	FPT (Cor. 12)	FPT (Cor. 12)	XP (Cor. 11)
unary	pNP-c (Th 20)	XP, W[1]-h (Th 19, 21)	pNP-c (Th 20)
arbitrary	pNP-c	pNP-c (Th 22)	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.

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 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 bipar-

tite 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 $\mathbf{A}$ and by $\det(\mathbf{A})$ the determinant 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 $\mathbf{b}$. We will also use the two following well-known facts [Schrijver, 1998].

Proposition 1. *Let $\mathbf{A}$ be an integer $k \times k$ matrix. Then $\det(\mathbf{A})$ is integer and $|\det(\mathbf{A})| \leq k! \Pi_{1 \leq i \leq k} c_{\mathbf{A}_{(*,\{i\})}}$.*

Proposition 2 (Cramer's rule)**.** *Let $\mathbf{A}$ be a $k \times k$ non-singular (i.e., with non-zero determinant) matrix and $\mathbf{b}$ a vector. Then the equation $\mathbf{A}\mathbf{x} = \mathbf{b}$ has a unique solution such that $\mathbf{x}[i] = \frac{\det(\mathbf{A}(i))}{\det(\mathbf{A})}$, where $\mathbf{A}(i)$ is the matrix formed by replacing the i-th column of $\mathbf{A}$ with the vector $\mathbf{b}$.*

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 $n \times m$ matrix of integers (the *constraint matrix*),
- $\mathbf{x}$ is a vector of *variables* of size m,
- $\mathbf{b}$ is an integer vector of size m (the *right-hand side*),
- $\mathbf{l}, \mathbf{u}$ are vectors of elements of $\mathbb{Z} \cup \{\pm\infty\}$ (the *lower* and *upper bounds*, respectively), and
- η is an integer vector of size m (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 $\mathrm{var}(\mathbf{I})$ 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 $\mathrm{var}(\mathbf{I}) \cup \mathcal{F}(\mathbf{I})$, 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 $\mathrm{var}(\mathbf{I})$, denoted by $\mathrm{var}(\alpha)$, to $\mathbb{Z}$. An assignment α is called *feasible* if

1. 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. 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 [Schrijver, 1998]. 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.

1. ILP-FEASIBILITY is formulated equivalently as ILP, with the restriction that η must be the 0-vector. All hardness results for ILP-FEASIBILITY immediately carry over to ILP.

2. 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 $\mathrm{var}(\mathbf{I}) \cup \mathcal{F}(\mathbf{I})$ and two vertices s, t are adjacent iff $s \in \mathrm{var}(\mathbf{I})$, $t \in \mathcal{F}$ 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 $\mathrm{var}(\mathcal{F}(\mathbf{I}'))$. For a set $Z \subseteq \mathcal{F}(\mathbf{I}) \cup \mathrm{var}(\mathbf{I})$, 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).

To obtain our lower bounds, we will need the notion of a parameterized reduction. Formally, a *parameterized problem* is a subset of $\Sigma^* \times \mathbb{N}$, where Σ is the input alphabet. Let $L_1 \subseteq \Sigma_1^* \times \mathbb{N}$ and $L_2 \subseteq \Sigma_2^* \times \mathbb{N}$ be parameterized problems. A *parameterized reduction* (or fpt-reduction) from L_1 to L_2 is a mapping $P : \Sigma_1^* \times \mathbb{N} \to \Sigma_2^* \times \mathbb{N}$ such that (i) $(x, k) \in L_1$ iff $P(x, k) \in L_2$, (ii) the mapping can be computed by an fpt-algorithm w.r.t. parameter k, and (iii) there is a computable function g such that $k' \leq g(k)$, where $(x', k') = P(x, k)$.

Next, we will define the complexity classes needed to describe our lower bounds. The class $\mathsf{W}[1]$ captures parameterized intractability and contains all problems that are fpt-reducible to INDEPENDENT SET when parameterized by the size of the solution. The following relations between the parameterized complexity classes hold: $\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{XP}$, where the class XP contains all problems solvable in time $\mathcal{O}(n^{f(k)})$ for a computable function f. Showing $\mathsf{W}[1]$-hardness for a problem rules out the existence of an fpt-algorithm under standard complexity assumptions.

The class pNP is defined as the class of problems that are solvable by a non-deterministic Turing machine in fpt time. In our pNP-hardness proofs, we will make use of the following characterization of pNP-hardness given in the book by Flum and Grohe [2006], Theorem 2.14: any parameterized problem that remains NP-hard when the parameter is set to some constant is pNP-hard. For problems in NP, we have $\mathsf{W}[1] \subseteq \mathsf{pNP}$ and in particular showing pNP-hardness rules out the existence of algorithms with a running time of $\mathcal{O}(n^{f(k)})$. For our algorithms, we will use the following result as a subroutine. Note that this is a streamlined version of the original statement of the theorem, as used in the area of parameterized algorithms [Fellows *et al.*, 2008].

Proposition 3 (Lenstra, Jr.; Kannan; Frank and Tardos [1983; 1987; 1987])**.** *There is an algorithm that solves an input ILP instance* $\mathbf{I} = (\mathcal{F}, \eta)$ *in time* $p^{\mathcal{O}(p)} \cdot |\mathbf{I}|$, *where* $p = |\mathrm{var}(\mathbf{I})|$.

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 4 (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 5 (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 can be solved in time at most $\ell^{\mathcal{O}(\ell)} \cdot |\mathbf{I}|$ due to Proposition 3; indeed, we can compute a solution for $\mathbf{I}$ by combining solutions for each connected component of $\mathbf{I}$, and hence it suffices to apply Proposition 3 independently on each component.

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 4 and 5; in

```
maximize ∑⁷ᵢ₌₁ i · xᵢ,        where
∑⁷ᵢ₌₁ xᵢ = 32,
1x₁ + y = 6,              2x₂ + y = 9,
3x₃ + y = 14,             4x₄ + y = 21,
5x₅ + y = 30,             6x₆ + 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 a 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. Observe that at its core the above problem and its variants are really a problem on the incidence graph of the ILP instance. Namely, the problems can be equivalently stated as the following graph problem.

FRACTURE VERTEX DELETION (FVD)

Instance: An undirected bipartite graph G with bipartition $\{U, W\}$, a set $D \in \{U, V(G)\}$, and an integer k.
Parameter: k
Question: Is there a set $B \subseteq D$ of at most k vertices such that every connected component of $G \setminus B$ has size at most k?

It is worth noting that this graph problem is closely related to the so-called VERTEX INTEGRITY problem, which has been studied on a variety of graph classes, including co-bipartite graphs [Drange *et al.*, 2016]. Unfortunately, to the best of our knowledge nothing is known about its complexity on bipartite graphs.

To see that each variant of BD is equivalent to a specific subcase of the FVD problem (in particular depending on the choice of D in the instance), consider the following polynomial time reductions in both directions. Given an instance $(\mathbf{I}, k)$ of BD, then the instance $(G_{\mathbf{I}}, V(G_{\mathbf{I}}), k)$ of FVD is easily seen to be equivalent. Similarly, if $(\mathbf{I}, k)$ is an instance of V-BD or C-BD, then $(G_{\mathbf{I}}, \text{var}(\mathbf{I}), k)$ and $(G_{\mathbf{I}}, \mathcal{F}(\mathbf{I}), k)$ are

equivalent instances of FVD. Moreover, if $I = (G, V(G), k)$ is an instance of FVD, then $(\mathbf{I}, k)$, where $\mathbf{I}$ is any ILP instance such that $G_{\mathbf{I}}$ is isomorphic with G is an equivalent instance of BD. Similarly, if $I = (G, U, k)$ is an instance of FVD, then $(\mathbf{I}, k)$, where $\mathbf{I}$ is any ILP instance such that $G_{\mathbf{I}}$ is isomorphic with G and $\mathrm{var}(\mathbf{I}) = U$, is an equivalent instance of V-BD. Note that such an instance $\mathbf{I}$ can for instance be obtained as follows:

- for every vertex $v \in U$, $\mathbf{I}$ has one variable v with arbitrary domain,
- for every vertex $v \in W$, $\mathbf{I}$ has one constraint with arbitrary non-zero coefficients on the variables in $N_G(v)$,

To justify a parameterized complexity analysis of our detection problems, we first show NP-completeness of our problems. It is worth noting that the NP-completeness of FRACTURE VERTEX DELETION was far from obvious at first glance due to the restriction to bipartite graphs; indeed, for instance the related problem of deleting at most k vertices such that the remaining graph only contains isolated vertices (VERTEX COVER) is well-known to be polynomial on bipartite graphs.

Theorem 6. BD, V-BD, and C-BD are NP-complete.

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.

Proof of Theorem 6. Because of the equivalence between BD, V-BD, C-BD and the FVD problem, it is sufficient to show that FVD is NP-complete for both choices of D. Because any solution to FVD can be verified in polynomial time, it holds that FVD is in NP. Towards showing NP-hardness of FVD we give a polynomial time reduction from a known variant of the 3-SATISFIABILITY problem. Given a 3-CNF formula Φ with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$ such that every literal occurs in exactly two clauses (this variant of 3-SATISFIABILITY is known to be NP-complete [Garey and Johnson, 1979]), we construct the instance $\langle G, D, k \rangle$ of FVD as follows. We set $k = n + 2m$ and the graph G will be the disjoint union of certain variable and clause gadgets introduced below plus connections between these variable and clauses gadgets. Namely, for every variable x_i, the graph G contains the variable gadget $G(x_i)$ with the following vertices and edges:

- two vertices x_i and $\overline{x_i}$,
- $k - 5$ vertices $c_i^1, \ldots, c_i^{k-5}$,
- for every j with $1 \leq j \leq k - 5$ the two edges $\{x_i, c_i^j\}$ and $\{\overline{x_i}, c_i^j\}$.

Moreover for every clause C_j of Φ with literals l_j^1, l_j^2, l_j^3, the graph G contains a clause gadget $G(C_J)$ with the following vertices and edges:

- three vertices l_j^1, l_j^2, and l_j^3,
- $k - 3$ vertices $b_j^1, \ldots, b_j^{k-3}$,

- for every i with $1 \leq i \leq k - 3$ the three edges $\{b_j^i, l_j^1\}$, $\{b_j^i, l_j^2\}$, and $\{b_j^i, l_j^3\}$.

Note that $G(C_j)$ is simple a complete bipartite graph with bipartition $\{\{l_j^1, l_j^2, l_j^3\}, \{b_j^i : 1 \leq i \leq k - 3\}$. Now G consists of the disjoint union of $G(x_1), \ldots, G(x_n), G(C_1), \ldots, G(C_m)$ plus the following vertices and edges, which ensure the required connections between the variable and clause gadgets:

- For every clause C_j (for some j with $1 \leq j \leq m$) with literals l_j^1, l_j^2, and l_j^3 and every $a \in \{1, 2, 3\}$ we add the vertices d_j^a and e_j^a and the edges $\{l_j^a, d_j^a\}$ and $\{l_j^a, e_j^a\}$ to G. Moreover, if $l_j^a = x_i$ for some i with $1 \leq i \leq n$, we additionally add the edges $\{x_i, d_j^a\}$ and $\{x_i, e_j^a\}$ to G and if on the other hand $l_j^a = \overline{x_i}$ for some i as above, then we add the edges $\{\overline{x_i}, d_j^a\}$ and $\{\overline{x_i}, e_j^a\}$ to G.

This completes the construction of G, which is clearly bipartite as for instance witnessed by the bipartition $\{U, V(G) \setminus U\}$, where $U = \{x_i, \overline{x_i}, l_j^1, l_j^2, l_j^3 : 1 \leq i \leq n \land 1 \leq j \leq m\}$. We will show below that there is always a solution that is entirely contained in U, which implies that the hardness result holds for $D \in \{U, V(G)\}$, and hence all versions of the fracture backdoor set problem, i.e., BD, V-BD, and C-BD, are NP-complete. Note that the reduction can be computed in polynomial time and it remains to show the equivalence between the two instances.

Towards showing the forward direction, assume that $\alpha : \{x_1, \ldots, x_n\} \to \{0, 1\}$ is a satisfying assignment for Φ. Because α satisfies Φ it follows that for every clause C_j with literals l_j^1, l_j^2, and l_j^3 there is at least one index $a(C_j) \in \{1, 2, 3\}$ such that the literal $l_j^{a(C_j)}$ is satisfied by α. We claim that the set B defined by:

- for every i with $1 \leq i \leq n$, B contains x_i if $\alpha(x_i) = 1$ and $\overline{x_i}$, otherwise,
- for every j with $1 \leq j \leq m$, B contains the vertices in $\{l_j^b : b \in \{1, 2, 3\} \setminus \{a(C_j)\}\}$.

is a solution for (G, U, k). Because B contains exactly one vertex for every variable of Φ and exactly two vertices for every clause of Φ, it holds that $|B| = k = n + 2m$, as required. Moreover, $B \subseteq U$. It hence only remains to show that every component of $G \setminus B$ has size at most k. Towards showing this first consider a component C of $G \setminus B$ that contains at least one vertex from a variable gadget $G(x_i)$ for some i with $1 \leq i \leq n$. Then $G(x_i) \cap B \in \{\{x_i, \overline{x_i}\}\}$ and hence $G(x_i) \setminus B$ is connected, which implies that $G(x_i) \setminus B \subseteq C$. W.l.o.g. assume that $G(x_i) \cap B = \{x_i\}$. Then $\alpha(x_i) = 1$ and it follows that all literal vertices of clause gadgets that correspond to the literal $\overline{x_i}$ are contained in B. Since moreover $\overline{x_i}$ is contained in exactly two clauses, we obtain that C consists of exactly $k - 4$ vertices in $G(x_i) \setminus B$ plus the four vertices $d_{j_1}^{a_1}$, $e_{j_1}^{a_1}, d_{j_2}^{a_2}$ and $e_{j_2}^{a_2}$ defined by $l_{j_1}^{a_1} = \overline{x_i}$ and $l_{j_2}^{a_2} = \overline{x_i}$. Hence in total C contains exactly k vertices as required. Now consider a component C that contains at least one vertex from a clause gadget $G(C_j)$ for some j with $1 \leq j \leq m$. Then $|G(C_j) \cap B| = 2$ and moreover B contains all but exactly

one literal vertex say l_j^a for some $a \in \{1, 2, 3\}$ from $G(C_j)$. W.l.o.g. let x_i be the literal of C_j corresponding to l_j^a. Then $\alpha(x_i) = 1$ and hence $x_i \in B$. It follows that C consists of the exactly $k - 2$ vertices in $G(C_j) \setminus B$ plus the two vertices d_j^a and e_j^a. Hence in total C contains exactly k vertices, as required. Because every component of $G \setminus B$ that neither contains a vertex from a vertex gadget nor from a clause gadget has size exactly one, this shows that B is indeed a solution for (G, U, k) and hence also for $(G, V(G), k)$.

Towards showing the reverse direction, let B be a solution for $(G, V(G), k)$. We first show that w.l.o.g. we can assume that $B \subseteq U$. So assume that $B \nsubseteq U$. We distinguish three cases: B contains a vertex d_j^a or e_j^a for some j and a with $1 \le j \le m$ and $1 \le a \le 3$. Let u and v be the two vertices adjacent to d_j^a and e_j^a. If B contains both d_j^a and e_j^a, then it is straightforward to verify that $B \setminus \{d_j^a, e_j^a\} \cup \{u, v\}$ is also a solution. So assume that B contains only d_j^a (the case that B contains only e_j^a is analogous). If $\{u, v\} \subseteq B$, then $B \setminus \{d_j^a\}$ is still a solution. Hence assume that w.l.o.g. $u \notin B$. But then $(B \setminus \{d_j^a\}) \cup \{u\}$ is a solution. Hence in all cases we could transform B into a solution that does not contain a vertex d_j^a or e_j^a. Next consider the case that B contains some vertex c_i^j for some i and j with $1 \le i \le n$ and $1 \le j \le k - 5$. In this case one can use an argumentation very similar to the previous case to transform B into a solution not containing such a vertex. Hence there only remains the case that B contains some vertex b_j^i for some i and j with $1 \le i \le k - 3$ and $1 \le j \le m$. In this case it is straightforward to verify that removing all vertices from $B \cap \{b_j^1, \ldots, b_j^{k-3}\}$ and replacing those with an equal (or less) amount of vertices in $\{l_j^1, l_j^2, l_j^3\}$ will again give a solution. Hence we can assume that $B \subseteq U$.

We show next that B contains at least one of x_i and $\overline{x_i}$ from every variable gadget $G(x_i)$. Suppose not and consider the component C of $G \setminus B$ containing x_i. Because $B \subseteq U$, we obtain that C contains all $k - 3$ vertices in $G(x_i)$ and additionally at least the 8 vertices adjacent to x_i and $\overline{x_i}$. Hence $|C| \ge k - 3 + 8 > k$ a contradiction to our assumption that B is a solution.

We show next that B contains at least two of $\{l_j^1, l_j^2, l_j^3\}$ from every clause gadget $G(C_j)$. Suppose not and consider a component C of $G \setminus B$ containing at least one vertex from $G(C_j)$. Because $B \subseteq U$, we obtain that C contains all of the at least $k - 3 + 2 = k - 1$ vertices in $G(C_j) \setminus B$ and additionally the at least four vertices adjacent to the (at least two) literal vertices in $\{l_j^1, l_j^2, l_j^3\} \setminus B$. Hence $|C| \ge k - 1 + 4 > k$ a contradiction to our assumption that B is a solution.

Hence B contains at least one vertex for every variable of Φ and at least two vertices for every clause of Φ. Moreover, because B is a solution it holds that $|B| \le k = n + 2m$. Hence $|B| = n + 2m$ and B contains exactly one vertex from every variable gadget and exactly two vertices from every clause gadget. We claim that the assignment α with $\alpha(x_i) = 1$ if and only if $x_i \in B$ is a satisfying assignment for Φ. Suppose not and let C_j be a clause of Φ that is not satisfied by Φ and let l_j^a be the (unique) literal vertex of $G(C_j)$ that is not in B. Consider the component C of $G \setminus B$ that contains l_j^a

and assume w.l.o.g. that $l_j^a = x_i$ for some i with $1 \le i \le n$. Because α does not satisfy C_j, we obtain that $x_i \notin B$. Because furthermore $B \subseteq U$ we obtain that C contains all of the $k - 3 + 1 = k - 2$ vertices in $G(C_j) \setminus B$ and additionally at least the two vertices adjacent to l_j^a as well as the vertex x_i. Hence in total C contains at least $k - 2 + 3 > k$ vertices, a contradiction to our assumption that B is a solution. □

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

Proof. Because of the equivalence of the problems BD, V-BD, and C-BD with the FVD problem, it is sufficient to show the result for FVD.

We will show the lemma by providing a depth-bounded search tree algorithm for any instance $I = \langle G, D, k \rangle$ of FVD, which is based on the following observations.

O1 If G is not connected then a solution for I can be obtained as the disjoint union of solutions for every component of G.

O2 If G is connected and C is any set of $k + 1$ vertices of G such that $G[C]$ is connected, then any solution for I has to contain at least one vertex from C.

These observations lead directly to the following recursive algorithm that given an instance $I = \langle G, D, k \rangle$ of FVD either determines that the instance is a NO-instance or outputs a solution $B \subseteq D$ of minimal size for I. The algorithm also remembers the maximum size of any component in a global constant c, which is set to k for the whole duration of the algorithm. The algorithm first checks whether G is connected. If G is not connected the algorithm calls itself recursively on the instance $(C, D \cap C, k)$ for each component C of G. If one of the recursive calls returns NO or if the size of the union of the solutions returned for each component exceeds k, the algorithm returns that I is a NO-instance. Otherwise the algorithm returns the union of the solutions returned for each component of G.

If G is connected and $|V(G)| \le c$, the algorithm returns the empty set as a solution. Otherwise, i.e., if G is connected but $|V(G)| > c$ the algorithm first computes a set C of $c + 1$ vertices of G such that $G[C]$ is connected. This can for instance be achieved by a depth-first search that starts at any vertex of G and stops as soon as $c + 1$ vertices have been visited. If $C \cap D = \emptyset$ then the algorithm returns NO. Otherwise the algorithm branches on the vertices in $C \cap D$, i.e., for every $v \in C \cap D$ the algorithm recursively computes a solution for the instance $(G \setminus \{v\}, k - 1)$. It then returns the solution of minimum size returned by any of those recursive calls, or NO-if none of those calls return a solution. This completes the description of the algorithm. The correctness of the algorithm follows immediately from the above observations.

6

Moreover the running time of the algorithm is easily seen to be dominated by the maximum time required for the case that at each step of the algorithm G is connected. In this case the running time can be obtained as the product of the number of branching steps times the time spent on each of those. Because at each recursive call the parameter k is decreased by at least one and the number of branching choices is at most $c + 1$, we obtain that there are at most $(c+1)^k = (k+1)^k$ branching steps. Furthermore, the time at each branching step is dominated by the time required to check whether G is connected, which is linear in the number of edges of G. Putting everything together, we obtain $\mathcal{O}((k+1)^k|E(G)|)$ as the total time required by the algorithm, which completes the proof of the lemma. □

We note that the depth-first search algorithm in the above proof can be easily transformed into a polynomial time approximation algorithm for BD and its variants that exhibits an approximation ratio of $k + 1$. In particular, instead of branching on the vertices of a connected subgraph C of G with $k+1$ vertices, this algorithm would simply add all the vertices of C into the current solution. This way we obtain:

Theorem 8. 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 value of a 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 $\mathrm{var}(C_1)$ and $\mathrm{var}(C_2)$ such that the ILP instance obtained from $\mathbf{I}$ after renaming the variables in $\mathrm{var}(C_1)$ and $\mathrm{var}(C_2)$ 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 9. *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 $\left(2c_\mathbf{A}(\mathbf{I}) + 1\right)^{2k^2}$ equivalence classes. Moreover, one can test whether two components have the same type in time $\mathcal{O}(k!k^2)$.*

Proof. Let Z be the backdoor witnessing $\mathfrak{p}(\mathbf{I})$ and fix a component C of $\mathbf{I} \setminus Z$. First observe that there are only 3 submatrices of the constraint matrix of $\mathbf{I}$ that can contain nonzero coefficients and containing an element of C; we refer to Fig. 2, where we denote these matrices $\mathbf{Q}_C, \mathbf{Q}_C^V$, and $\mathbf{Q}_C^C$. We will now bound the number of these possibly nonzero coefficients. In order to do this we denote by $g_v = |\mathrm{var}(Z)|, g_c = |Z| - g_v, c_v = |\mathrm{var}(C)|, c_c = |C| - c_v$, and $c = \max c_c, c_v$. Observe that $c \leq k$. Now the number of possibly nonzero coefficients is bounded by $g_v c_c + g_c c_v + c_c c_v \leq (g_v + g_c)c + c^2 \leq 2k^2$. We finish

Figure 2: A situation for a component C.

the proof of the first part by observing that the number of possible coefficients is bounded by $2c_\mathbf{A}(\mathbf{I}) + 1$.

Observe that two components C_1 and C_2 have the same type if their number of constraints and variables is the same and there exist a permutation of variables of C_1 and a permutation of constraints of C_1 such that the three submatrices of $\mathbf{I}$ containing nonzero elements are exactly the same. Again as $|C| \leq k$ one can check all pairs of permutations in time $k!$ and for each pair we are checking $\mathcal{O}(k^2)$ entries. □

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

Theorem 10. *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 an $(r+u) \times (s+t)$ 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 $\max\{t, u\} \leq f\left(c_{\overline{\mathbf{A}}}, \mathfrak{p}(\overline{\mathbf{I}})\right)$ for some computable function f.

Proof. Let $\mathcal{C}$ be the set of connected components of $\overline{\mathbf{I}} \setminus Z$. We define a triple of matrices $(\mathbf{Q}_C^V, \mathbf{Q}_C^C, \mathbf{Q}_C)$ for a component C. Please refer to Figure 2.

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

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

- the matrix $\mathbf{Q}_C$ is the part of constraints in C dealing with $\mathrm{var}(C)$, that is, $\overline{\mathbf{A}}_{cF(C), \mathrm{var}(C)}$.

Observe that this totally decomposes all constraints and variables contained in C as all coefficient for other variables are 0 and variables of C cannot appear in other components. For a triple of matrices $T = (\mathbf{Q}^V, \mathbf{Q}^C, \mathbf{Q})$ a component C has type T if $\mathbf{Q}^V = \mathbf{Q}_C^V$, $\mathbf{Q}_C = \mathbf{Q}_C^C$, and $\mathbf{Q} = \mathbf{Q}_C$ holds. The set of all possible types is the set

$$\mathcal{T} = \left\{ T = (\mathbf{Q}^V, \mathbf{Q}^C, \mathbf{Q}) : \exists C \in \mathcal{C} \text{ with type } T \right\}.$$

7

The multiplicity $\operatorname{mult}(T)$ of type $T \in \mathcal{T}$ is the number of components in $\mathcal{C}$ having type T. We set $N = \max_{T \in \mathcal{T}} \operatorname{mult}(T)$.

The idea of the proof is to build the matrix $\mathbf{A}_1$ from Z and matrices $\mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4$ as representatives of the types in such a way that the resulting N-fold 4 block ILP is equivalent to the given ILP instance $\bar{\mathbf{I}}$.

The matrix $\mathbf{A}_1$ is simply the submatrix of Z that is the part of global constraints of $\overline{\mathbf{A}}$ containing $\operatorname{var}(Z)$ only.

Claim 1. There is an ILP instance $\hat{\mathbf{I}}$ that is equivalent to ILP instance $\bar{\mathbf{I}}$ with $\operatorname{mult}_{\hat{\mathbf{I}}}(T) = N$ for all $T \in \mathcal{T}_{\bar{\mathbf{I}}}$. Moreover, $c_{\hat{\mathbf{I}}} = c_{\bar{\mathbf{I}}}$ and the sizes of the matrices $\mathbf{Q}$ can only double.

In this case we put all possible matrices on a diagonal of the relevant matrix $\mathbf{A}_4$, next to each other in the matrix $\mathbf{A}_2$, and under each other in the matrix $\mathbf{A}_3$. That is we set $\mathbf{A}_2$ to horizontal concatenation of all $(\mathbf{Q}_T^C)_{T \in \mathcal{T}}$, $\mathbf{A}_3$ to vertical concatenation of $(\mathbf{Q}_T^V)_{T \in \mathcal{T}}$, and finally $\mathbf{A}_4$ has matrices $(\mathbf{Q}_T)_{T \in \mathcal{T}}$ on its diagonal. The bound on size of the matrix $\mathbf{A}$ follows from Lemma 9 and Claim 1. □

Proof of Claim 1. The idea here is to take a type with less representatives and add a new one as a copy of a previous one. But this has to be done carefully in order to maintain equivalence of intermediate ILPs. For the local part we start by observing that if we add a copy of some previous component, then the set of solutions for these two components is the same. However, as these components also interact with the global constraints we would like to have to restrict the set of solutions of the newly added component to all 0 solution only. Note that this cannot be done using lower and upper bounds only as the former set of solutions does not have to contain such a solution. That is, the (optimal) setting of global variables together with setting all component local variables to 0 can violate the right-hand side. In order to achieve the claim, we extend the matrices we have obtained from the component C in the following way. Let C be of type $T = (\mathbf{Q}_C^V, \mathbf{Q}_C^C, \mathbf{Q}_C)$ then the *extension of type T* if

$$\hat{T} = \left(\mathbf{Q}_C^V, [\mathbf{Q}_C^C \mid \mathbf{0}], [\mathbf{Q}_C \mid \mathbf{Q}_C]\right).$$

We denote the former C-variables as x_C and the new C-variables as $\hat{x}_C$. We say that the extension is of

- *first kind* if $\ell_C \le x_C \le u_C$ and $0 \le \hat{x}_C \le 0$, and
- *second kind* if $0 \le x_C \le 0$ and $\ell_C \le \hat{x}_C \le u_C$.

Note that with this we have only doubled the number of local variable of component C.

Claim 2. Let $\mathbf{I}$ be an ILP instance and let T be a type of $\mathbf{I}$. Denote $\mathbf{I}_{T \to \hat{T}}$ the ILP instance $\mathbf{I}$ where components of type T are replaced with components of $\hat{T}$ of the first kind. Then, there is a bijection between solutions of ILP instances $\mathbf{I}$ and $\mathbf{I}_{T \to \hat{T}}$.

Proof. Note that it holds that $\hat{x}_C = 0$ for every component C of type $\hat{T}$. Now a solution for $\mathbf{I}_{T \to \hat{T}}$ has a natural projection to a solution of $\mathbf{I}$ (forget all $\hat{x}_C$ variables). Furthermore, a solution for $\mathbf{I}$ can be extended to a solution of $\mathbf{I}_{T \to \hat{T}}$ by setting $\hat{x}_C = 0$ for each component C of type T. This yields a bijection between the solution sets. □

We say that a component C is *extended* if it has been created by the extension of the first kind. We transform all components with multiplicity less than N to extended components and denote $\mathbf{I}_E$ the resulting ILP instance. Note that by Claim 2 the ILP instances $\mathbf{I}$ and $\mathbf{I}_E$ are in equivalent.

Claim 3. Let $\mathbf{I}$ be an ILP instance, let C be a component of $\mathbf{I}$, and let C' be an extension of C of the second kind. Denote $\mathbf{I}'$ the ILP instance $\mathbf{I}$ with C' added (i.e., it has one more component) then instances $\mathbf{I}$ and $\mathbf{I}'$ are equivalent.

Proof. First we argue that $\mathbf{I}$ does have a solution if and only if $\mathbf{I}'$ does. To see this take a solution $\mathbf{x}$ of $\mathbf{I}$ and let x_C be the part of $\mathbf{x}$ corresponding to C-variables. We build a solution to $\mathbf{I}'$ follows. We copy the solution of every variable but the variables of C'. We set variables $x_{C'} = 0$ and $\hat{x}_{C'} = x_C$.

Note that by this we have actually build a natural correspondence between the set of solutions to $\mathbf{I}$ and the set of solutions to $\mathbf{I}'$. Observe that this correspondence is not one-to-one as in general there can be more possibilities how to extend the solution to variables $\hat{x}_{C'}$. We say that all these solutions project to the same solution $\mathbf{x}$ to instance $\mathbf{I}$. However, as all the C'-variables do not occur in the objective function the value of the objective function of all solutions that project to $\mathbf{x}$ is the same. □

By combining the two claims it is possible to transform ILP instance $\mathbf{I}$ to $\hat{\mathbf{I}}$ with the following properties.

- all components of $\hat{\mathbf{I}}$ are either extended or for their type T it holds that $\operatorname{mult}_{\mathbf{I}}(T) = N$,
- for each type $\hat{T}$ of $\hat{\mathbf{I}}$ it holds that $\operatorname{mult}_{\hat{\mathbf{I}}}(\hat{T}) = N$,
- $b_\ell(\hat{\mathbf{I}}) = b_\ell(\mathbf{I})$,
- number of variables in $\hat{\mathbf{I}}$ is at most twice the number of variables in $\mathbf{I}$.

□

Let us now discuss algorithmic consequences of Theorem 10 for all types of backdoors. Together with Theorems 4 and 7, we obtain the following corollary.

Corollary 11. *Let a and z be constants and let $\mathbf{I}$ be an ILP instance with $c_{\mathbf{A}}(\mathbf{I}) \le a$ and $\mathfrak{p}(\mathbf{I}) \le z$, then $\mathbf{I}$ can be solved in polynomial time.*

For variable and constraint backdoors, using Theorem 5 instead of Theorem 4 yields the following results.

Corollary 12. *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 22 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.

Before showing this we need some preparation.

Proposition 13. *Let $\mathbf{A}$ be an integer $k \times k$ non-singular matrix and $\mathbf{b}$ an integer vector. Then $|\mathbf{x}[i]| \leq k! c_{\mathbf{b}}(c_{\mathbf{A}})^{k-1}$ for the unique $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} = \mathbf{b}$.*

Proof. Because of Proposition 2 it holds that

$$\mathbf{x}[i] = \frac{\det(\mathbf{A}(i))}{\det(\mathbf{A})}.$$

Moreover, since $\mathbf{A}$ is a non-singular integer matrix, we have that $|\det(\mathbf{A})| \geq 1$ and thus $|\mathbf{x}[i]| \leq |\det(\mathbf{A}(i))|$, which together with Proposition 1 implies $|\mathbf{x}[i]| \leq |\det(\mathbf{A}(i))| \leq k! c_{\mathbf{b}}(c_{\mathbf{A}})^{k-1}$, as required. $\square$

Lemma 14. *Let $\mathbf{Q}$ be a $k \times n$ matrix of rank k, $\mathbf{y}$ be a vector of n variables, $\mathbf{d}$ be a vector of size k, I be a set of k linearly independent columns of $\mathbf{Q}$, V be their corresponding variables in $\mathrm{var}(\mathbf{y})$, and let β be an assignment of the variables in $\mathrm{var}(\mathbf{y})$ such that $\mathbf{Q}\beta(\mathbf{y}) = \mathbf{d}$. Then for every $v \in V$, it holds that*

$$|\beta(v)| \leq k! \Big(c_{\mathbf{d}} + c_{\mathbf{Q}} \sum_{u \in \mathrm{var}(\mathbf{y}) \setminus V} \beta(u)\Big)(c_{\mathbf{Q}})^{k-1}.$$

Proof. Let $\mathbf{y}'$ be $\mathbf{y}$ restricted to the variables in $\mathrm{var}(\mathbf{y}) \setminus V$ and let J be the set of all columns of $\mathbf{Q}$ that are not in I. We will now apply the assignment β for the variables in $\mathbf{y}'$ to $\mathbf{Q}$. This will give us a set of equations that need to be satisfied for the variables in V allowing us to obtain a bound on β for these variables. Namely, the right-hand side denoted by $\mathbf{d}'$ of our equations is obtained from $\mathbf{d}$ by subtracting the application of β to $\mathbf{Q}_{(*,J)}$, i.e., $\mathbf{d}' = \mathbf{d} - \mathbf{Q}_{(*,J)}\beta(\mathbf{y}')$, which after restricting $\mathbf{Q}$ to the columns I and using the restriction $\mathbf{y}''$ of $\mathbf{y}$ to the variables in V gives us the following equations that are satisfied by β:

$$\mathbf{Q}_{(*,I)}\beta(\mathbf{y}'') = \mathbf{d}' \tag{1}$$

Note that because I is a set of k linearly independent columns the matrix $\mathbf{Q}_{(*,I)}$ is non-singular. Moreover, observe that $\mathbf{d}'[i] \leq c_{\mathbf{d}} + c_{\mathbf{Q}} \sum_{u \in \mathrm{var}(\mathbf{y}) \setminus V} \beta(u)$ for every i with $1 \leq i \leq k$. Because β satisfies 1 we obtain from Proposition 13 that

$$|\beta(v)| \leq k! \Big(c_{\mathbf{d}} + c_{\mathbf{Q}} \sum_{u \in \mathrm{var}(\mathbf{y}) \setminus V} \beta(u)\Big)(c_{\mathbf{Q}})^{k-1},$$

for every variable $v \in V$. $\square$

The following lemma provides an important ingredient for Lemma 17 below. Its proof crucially makes use of the specific structure of our ILP instance.

Lemma 15. *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.*

Proof. Let $Z \subseteq \mathcal{F}(\mathbf{I})$ be a constraint backdoor for $\mathbf{I}$ of size at most $\mathfrak{p}^C(\mathbf{I})$ and let $\mathbf{s}$ be a non-zero vector satisfying $\mathbf{A}_{(*,D)}\mathbf{s} = \mathbf{0}$. Let $C_1, \ldots, C_p$ be all components of $\mathbf{I} \setminus Z$ that contain at least one variable corresponding to a column in D and let D_i be the set of all columns in D that correspond to variables in C_i. Moreover, let $\mathbf{s}_{C_i}$ be the restriction of $\mathbf{s}$ to the entries corresponding to variables in C_i. Note that if $p \leq \mathfrak{p}^C(\mathbf{I}) + 1$, then D already contains at most $\mathfrak{p}^C(\mathbf{I})(\mathfrak{p}^C(\mathbf{I}) + 1)$ linearly dependent columns and the lemma follows. So we can assume in the following that $p > \mathfrak{p}^C(\mathbf{I}) + 1$. Denote by $\mathbf{w}_{C_i}$ the vector $\mathbf{A}_{(*,D_i)}\mathbf{s}_{C_i}$. If $\mathbf{w}_{C_i} = \mathbf{0}$, then the variables in C_i that $\mathbf{s}$ does not assign to 0 correspond to at most $\mathfrak{p}^C(\mathbf{I})$ linearly dependent columns and the lemma follows. Otherwise, it is easy to observe that if $\mathbf{w}_{C_i}[j] \neq 0$ then j corresponds to a constraint in Z. Hence for every C_i all non-zero entries of the vector $\mathbf{w}_{C_i}$ correspond to constraints in Z. Consequently any subset of $\mathfrak{p}^C(\mathbf{I}) + 1$ vectors from $\mathbf{w}_{C_1}, \ldots, \mathbf{w}_{C_p}$ in particular the vectors $\mathbf{w}_{C_1}, \ldots, \mathbf{w}_{C_{\mathfrak{p}^C(\mathbf{I})+1}}$ are linearly dependent (since all their non-zero entries correspond to constraints in Z and $|Z| \leq \mathfrak{p}^C(\mathbf{I})$), which implies that the set $\bigcup_{1 \leq i \leq \mathfrak{p}^C(\mathbf{I})+1} D_i$ is the required subset of at most $\mathfrak{p}^C(\mathbf{I})(\mathfrak{p}^C(\mathbf{I}) + 1)$ linearly dependent columns of $\mathbf{A}_{(*,D)}$. $\square$

Lemma 16. *Let $\mathbf{I} = (\mathbf{A}, \mathbf{x}, \mathbf{b}, \mathbf{l}, \mathbf{u}, \eta)$ be an ILP instance, α a solution for $\mathbf{I}$, and δ a non-zero integer vector such that $\alpha + \delta$ and $\alpha - \delta$ are feasible assignments for $\mathbf{I}$. Then $\eta\delta = 0$ and moreover $\alpha + \delta$ and $\alpha - \delta$ are also solutions for $\mathbf{I}$.*

Proof. Assume for a contradiction that $\eta\delta \neq 0$, then either $\eta(\alpha + \delta) > \eta(\alpha)$ or $\eta(\alpha - \delta) > \eta(\alpha)$, contradicting that α is a solution. $\square$

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

Lemma 17. *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. Let $\mathbf{I} = (\mathbf{A}, \mathbf{x}, \mathbf{b}, \mathbf{l}, \mathbf{u}, \eta)$ be the provided instance of UNARY ILP and let $Z \subseteq \mathcal{F}(\mathbf{I})$ be a constraint backdoor witnessing $\mathfrak{p}^C(\mathbf{I})$.

Let

$$m_S = \big((\mathfrak{p}^C(\mathbf{I}) + 1)^2\big)!(n)^{(\mathfrak{p}^C(\mathbf{I})+1)^2},$$

$$m_M = 4\big((\mathfrak{p}^C(\mathbf{I}) + 2)^2\big)!(n)^{(\mathfrak{p}^C(\mathbf{I})+2)^2}.$$

For a solution α of $\mathbf{I}$, let $V(\alpha)$ be the set of all variables v of $\mathbf{I}$ such that $|\alpha(v)| \geq 2m_S$. Let us now consider a solution α which minimizes the size of $V(\alpha)$. Observe that because $m_L \geq 2m_S$ it holds that if $|V(\alpha)| = 0$ then the lemma holds, and so we may assume that $V(\alpha)$ is non-empty.

In the following we consider the submatrix $\mathbf{B} = \mathbf{A}_{(*,V(\alpha))}$. Let us first consider the case where the columns of $\mathbf{B}$ are

9

linearly dependent. We show that in this case, we can find a solution α' such that $|V(\alpha')| < |V(\alpha)|$, which contradicts the choice of α.

Because of Lemma 15 there is a non-empty set O of linearly dependent columns of $\mathbf{B}$ of size at most $\mathfrak{p}^C(\mathbf{I})(\mathfrak{p}^C(\mathbf{I}) + 1)$. Consider a subset $Y = \{\mathbf{v}^1, \dots, \mathbf{v}^{|Y|}\}$ of linearly dependent columns of O such that the columns of each proper subset of Y are linearly independent and let $X = Y \setminus \{\mathbf{v}^{|Y|}\}$. Because Y is a minimal set of linearly dependent columns, it holds that there is a vector $\mathbf{a}$ without any zero entries such that $\mathbf{B}_{(*,Y)}\mathbf{a} = \mathbf{0}$, which implies the existence of a vector $\mathbf{a}_X$, again without zero entries, such that $\mathbf{B}_{(*,X)}\mathbf{a}_X = \mathbf{v}^{|Y|}$. We will show that there is a vector $\mathbf{a}$ that is integer and satisfies $|\mathbf{a}[i]| \le m_S$ for every $1 \le i \le |Y|$. We start by solving $\mathbf{B}_{(*,X)}\mathbf{a}_X = \mathbf{v}^{|Y|}$ using Cramer's rule. Because the columns in X are linearly independent, it follows that $\mathbf{B}_{(*,X)}$ has a set R of linearly independent rows with $|R| = |X|$. Then because the matrix $\mathbf{B}_{(R,X)}$ is non-singular, we have that there is a unique $\mathbf{a}_X$ such that $\mathbf{B}_{(R,X)}\mathbf{a}_X = \mathbf{v}_R^{|Y|}$, where $\mathbf{v}_R^{|Y|}$ denotes the restriction of the vector $\mathbf{v}^{|Y|}$ to the entries associated with the columns in R. Moreover, because there is a non-zero vector $\mathbf{a}_X$ with $\mathbf{B}_{(*,X)}\mathbf{a}_X = \mathbf{v}^{|Y|}$, it follows that the unique vector $\mathbf{a}_X$ satisfying $\mathbf{B}_{(R,X)}\mathbf{a}_X = \mathbf{v}_R^{|Y|}$ also satisfies $\mathbf{B}_{(*,X)}\mathbf{a}_X = \mathbf{v}^{|Y|}$. Using Cramer's Rule, we obtain $\mathbf{a}_X[i] = \frac{\det(\mathbf{B}_{(R,X)}(i))}{\det(\mathbf{B}_{(R,X)})}$ for every i with $1 \le i \le |X|$ as the unique vector satisfying $\mathbf{B}_{(R,X)}\mathbf{a}_X = \mathbf{v}_R^{|Y|}$.

Hence the vector $\mathbf{d}$ with $\mathbf{d}[i] = \mathbf{a}_X[i]\det(\mathbf{B}_{(R,X)}) = \det(\mathbf{B}_{(R,X)}(i))$ for every i with $1 \le i \le |X|$ and $\mathbf{d}[|Y|] = -\det(\mathbf{B}_{(R,X)})$ is a non-zero integer vector that satisfies $\mathbf{B}_{(*,Y)}\mathbf{d} = \mathbf{0}$. From Proposition 1, we obtain that

$$
\begin{aligned}
|\mathbf{d}[i]| \\
&\le \left(\mathfrak{p}^C(\mathbf{I})(\mathfrak{p}^C(\mathbf{I}) + 1)\right)!(c_{\mathbf{A}})^{\mathfrak{p}^C(\mathbf{I})(\mathfrak{p}^C(\mathbf{I})+1)} \\
&\le \left((\mathfrak{p}^C(\mathbf{I}) + 1)^2\right)!(n)^{(\mathfrak{p}^C(\mathbf{I})+1)^2} \\
&= m_S,
\end{aligned}
$$

as required.

For notational convenience we will in the following assume that $\mathbf{A}$ starts with the columns $\mathbf{v}^1, \dots, \mathbf{v}^{|Y|}$ from Y. Let $\mathbf{w}$ be the vector defined by:

- $\mathbf{w}[i] = \mathbf{d}[i]$, if $i \le |Y|$, and
- $\mathbf{w}[i] = 0$ otherwise

Note that $\mathbf{A}\mathbf{w} = \mathbf{0}$. For an integer Δ, let $\alpha_\Delta : \mathrm{var}(\mathbf{I}) \to \mathbb{Z}$ denote the assignment $\alpha_\Delta = \alpha + \Delta\mathbf{w}$. Note that α_Δ is an integral assignment, moreover because

$$
\mathbf{A}\alpha_\Delta(\mathbf{x}) = \mathbf{A}\alpha(\mathbf{x}) + \Delta\mathbf{A}\mathbf{w} = \mathbf{A}\alpha(\mathbf{x})
$$

it follows that α_Δ is a feasible integral assignment for $\mathbf{A}\mathbf{x} = \mathbf{b}$ for every $\Delta \in \mathbb{Z}$. Let Δ be the integer with smallest absolute value such that there is at least one variable $v \in V(\alpha)$ with $|\alpha_\Delta(v)| \le 2m_S$. We claim that for every $|\delta| \le |\Delta|$, α_δ is a solution for $\mathbf{I}$. We first show that $\mathbf{l}[i] \le \alpha_\delta(\mathbf{x}[i]) \le \mathbf{u}[i]$ for every i with $1 \le i \le |\mathrm{var}(\mathbf{I})|$. If $\mathbf{x}[i]$ corresponds to a column that is not in Y, then $\alpha_\delta(\mathbf{x}[i]) = \alpha(\mathbf{x}[i])$, which implies $\mathbf{l}[i] \le \alpha_\delta(\mathbf{x}[i]) \le \mathbf{u}[i]$. Otherwise, assume w.l.o.g.

that $\alpha(\mathbf{x}[i]) \ge 0$ (the case that $\alpha(\mathbf{x}[i]) < 0$ is symmetric). Because $\alpha(\mathbf{x}[i]) \ge 2m_S$ and $|\mathbf{d}[j]| \le m_S$ for every j with $1 \le j \le |Y|$ together with the choice of Δ, we obtain that $m_S \le \alpha_\delta(\mathbf{x}[i])$. Because $m_S > n$ and since α is a feasible solution it follows that $\mathbf{u}[i] = \infty$ and $\mathbf{l}[i] \le m_S$, which shows that $\mathbf{l}[i] \le \alpha_\delta(\mathbf{x}[i]) \le \mathbf{u}[i]$. Hence in particular α_Δ and also $\alpha_1 = \alpha + \mathbf{w}$ and $\alpha_{-1} = \alpha - \mathbf{w}$ are feasible assignments, which together with Lemma 16 (after setting δ to $\mathbf{w}$) implies that $\eta\mathbf{w} = 0$ and hence $\eta\alpha = \eta\alpha_\Delta$. Consequently α_Δ is a solution for $\mathbf{I}$ with $|V(\alpha_\Delta)| < |V(\alpha)|$, contradicting our choice of α.

We conclude that the columns of $\mathbf{B}$ must be linearly independent, which implies that there is a set R of $|V(\alpha)|$ linearly independent rows in $\mathbf{B}$. Consider the set S of all components of $\mathbf{I} \setminus Z$ that have a non-empty intersection with either $V(\alpha)$ or the constraints corresponding to the rows in R. Let $C_1, \dots, C_p$ be the restrictions of the components in S to the variables in $V(\alpha)$ and the constraints in R.

Observe that for every component C_i, it holds that the rows in R that correspond to constraints in C_i are zero everywhere but at the entries corresponding to variables in C_i. Because the rows in R are independent it follows that every component must have at least as many variables as constraints. Moreover, because $\mathbf{B}_{(R,*)}$ is a square matrix and the only rows in R that do not correspond to constraints in components, correspond to the constraints in Z, we obtain that there are at most $|Z| \le \mathfrak{p}^C(\mathbf{I})$ components that have strictly more variables than constraints, all other components have the same number of rows and columns. Let C_i be a component with the same number of rows as columns and let C_i' be the unique component of $\mathbf{I} \setminus Z$ containing C_i. Let $Q = \mathbf{A}_{(C(C_i'),R(C_i))}$ and $\mathbf{y}$ be the subvector of $\mathbf{x}$ restricted to the variables of C_i', $\mathbf{d}$ be the subvector of $\mathbf{b}$ restricted to entries that correspond to the constraints of C_i, $V = \mathrm{var}(C_i)$, I the set of columns of Q corresponding to the variables in V, and β the assignment α restricted to the variables in $\mathbf{y}$. Because the rows in Q are independent its rank is $|\mathcal{F}(C_i)|$, because α satisfies $\mathbf{A}\alpha(\mathbf{x}) = \mathbf{b}$ and all but the columns corresponding to the variables in $\mathrm{var}(C_i')$ of $\mathbf{A}_{*,\mathcal{F}(C_i)}$ are zero everywhere, it holds that $\mathbf{Q}\beta(\mathbf{y}) = \mathbf{d}$. Hence we can apply Lemma 14 for $\mathbf{Q}$, $\mathbf{y}$, $\mathbf{d}$, V, I, and β and obtain:

$$
\begin{aligned}
|\alpha(v)| \\
&\le \mathfrak{p}^C(\mathbf{I})!\left(c_{\mathbf{b}} + c_{\mathbf{A}}\sum_{u \in \mathrm{var}(C_i') \setminus \mathrm{var}(C_i)} \alpha(u)\right)(c_{\mathbf{A}})^{\mathfrak{p}^C(\mathbf{I})-1} \\
&\le \mathfrak{p}^C(\mathbf{I})!\left(c_{\mathbf{b}} + c_{\mathbf{A}}\mathfrak{p}^C(\mathbf{I})2m_S\right)(c_{\mathbf{A}})^{\mathfrak{p}^C(\mathbf{I})-1} \\
&\le \mathfrak{p}^C(\mathbf{I})!4m_S\mathfrak{p}^C(\mathbf{I})(n)^{\mathfrak{p}^C(\mathbf{I})} \\
&\le 4\left((\mathfrak{p}^C(\mathbf{I})+2)^2\right)!(n)^{(\mathfrak{p}^C(\mathbf{I})+2)^2} \\
&= m_M
\end{aligned}
$$

for every variable $v \in V$. The second to last inequality follows because $|\alpha(v)| \le 2m_S$ for every v in $\mathrm{var}(C_i') \setminus \mathrm{var}(C_i)$, which is because $(\mathrm{var}(C_i') \setminus \mathrm{var}(C_i)) \subseteq (\mathrm{var}(\mathbf{I}) \setminus V(\alpha))$. This shows that the assignment α is bounded by m_M for all variables contained in components C_i that have the same number of variables and constraints. Consider the remaining components $D_1, \dots, D_s$ among $C_1, \dots, C_p$, i.e., the components among $C_1, \dots, C_p$ that have more variables than constraints.

Recall that $s \leq |Z| \leq \mathfrak{p}^C(\mathbf{I})$. Let $V = \bigcup_{1 \leq i \leq s} \mathrm{var}(D_i)$ and let J be the corresponding columns of V in $\mathbf{A}$. Note that $|J| \leq (\mathfrak{p}^C(\mathbf{I}))^2$. Because $V \subseteq V(\alpha)$ it holds that J is a set of linearly independent columns. Hence there is a set R' of $|J|$ linearly independent rows in $\mathbf{A}_{(*,J)}$.

Let $\mathbf{Q} = \mathbf{A}_{(R',*)}$, $\mathbf{y} = \mathbf{x}$, $\mathbf{d}$ be the subvector of $\mathbf{b}$ restricted to entries that correspond to the rows in R', I be the columns in J restricted to the rows in R', and $\beta = \alpha$. Because the rows in $\mathbf{Q}$ are independent its rank is $|I|$, because $\mathbf{Q}$ is a submatrix of $\mathbf{A}$ only restricted in rows, we have $\mathbf{Q}\beta(\mathbf{y}) = \mathbf{d}$. Hence we can apply Lemma 14 for $\mathbf{Q}$, $\mathbf{y}$, $\mathbf{d}$, V, I, and β and obtain:

$$
\begin{aligned}
|\alpha(v)| \\
&\leq (\mathfrak{p}^C(\mathbf{I})^2)! \Big(c_{\mathbf{b}} + c_{\mathbf{A}} \sum_{u \in \mathrm{var}(\mathbf{I}) \setminus V} \alpha(u) \Big) (c_{\mathbf{A}})^{(\mathfrak{p}^C(\mathbf{I}))^2 - 1} \\
&\leq (\mathfrak{p}^C(\mathbf{I})^2)! (c_{\mathbf{b}} + c_{\mathbf{A}} |\mathrm{var}(\mathbf{I})| m_{\mathbf{M}}) (c_{\mathbf{A}})^{(\mathfrak{p}^C(\mathbf{I}))^2 - 1} \\
&\leq 8 \big(2(\mathfrak{p}^C(\mathbf{I}) + 2)^2 \big)! (n)^{2(\mathfrak{p}^C(\mathbf{I}) + 2)^2} \\
&= m_{\mathrm{L}}
\end{aligned}
$$

for every variable $v \in V$. The second to last inequality follows follows because $|\alpha(v)| \leq m_{\mathbf{M}}$ for every v in $\mathrm{var}(\mathbf{I}) \setminus V$, as shown previously. This concludes the proof of the lemma. $\square$

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:

Proposition 18 (Proposition 2 and Theorem 11 in Ganian *et al.* 2017). *Let* $\mathbf{I} = (\mathbf{A}, \mathbf{x}, \mathbf{b}, \mathbf{l}, \mathbf{u}, \eta)$ *be an ILP with incidence treewidth* ω *and such that* $\mathbf{l}[i] \neq -\infty$ *and* $\mathbf{u}[i] \neq \infty$ *for every entry* i. *Then* $\mathbf{I}$ *can be solved in time* $\mathcal{O}((c_{\mathbf{A}} \cdot \Delta \cdot |var(\mathbf{I})|)^\omega)(|\mathrm{var}(\mathbf{I})| + |\mathcal{F}(\mathbf{I})|)$, *where* $\Delta = \max_i\{|\mathbf{l}[i]|, |\mathbf{u}[i]|\}$.

Theorem 19. UNARY ILP *is polynomial time solvable for any fixed value of* $\mathfrak{p}^C(\mathbf{I})$, *where* $\mathbf{I}$ *is the input instance.*

Proof. Let $\mathbf{I}$ be an input instance of UNARY ILP encoded in n bits and let $\mathbf{I}'$ be the instance obtained from $\mathbf{I}$ by replacing $-\infty$ and ∞ entries in $\mathbf{l}$ and $\mathbf{u}$ with $-m_{\mathrm{L}}$ and m_{L}, respectively (for the definition of m_{L} see the statement of Lemma 17). It follows from Lemma 17 that $\mathbf{I}$ and $\mathbf{I}'$ are equivalent ILP instances. Now let ω be the incidence treewidth of $\mathbf{I}'$ (which is equal to the incidence treewidth of $\mathbf{I}$). Observe that $\omega \leq \mathfrak{p}^C(\mathbf{I})$ and hence it follows from Proposition 18 that $\mathbf{I}'$ (and thus also $\mathbf{I}$) can be solved in time $\mathcal{O}((c_{\mathbf{A}} \cdot m_{\mathrm{L}} \cdot |var(\mathbf{I})|)^{\mathfrak{p}^C(\mathbf{I})})(|\mathrm{var}(\mathbf{I})| + |\mathcal{F}(\mathbf{I})|)$. $\square$

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

Proof. 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 (which in turn represents a 3-coloring 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:

C1 For every prime number representing some vertex of G exactly one of the global variables is divisible by that prime number. This ensures that every vertex of G is assigned to exactly one color class.

C2 For every edge $\{u, v\}$ of G it holds that no global variable is divisible by the prime numbers representing u and v at the same time. This ensures that no two adjacent vertices of G are assigned to the same color class.

Thus let G be the given instance of 3-COLORING and assume that the vertices of G are uniquely identified as elements of $\{1, \ldots, |V(G)|\}$. In the following we denote by $p(i)$ the i-th prime number for any positive integer i, where $p(1) = 2$. We construct an instance $\mathbf{I}$ of ILP-FEASIBILITY in polynomial time with $\mathfrak{p}^V(\mathbf{I}) \leq 25$, and coefficients bounded by a polynomial in $V(G)$ such that G has a 3-coloring if and only if $\mathbf{I}$ has a feasible assignment. This instance $\mathbf{I}$ has the following variables:

- The *global variables* c_1, c_2, and c_3 with an arbitrary positive domain, whose values will represent a valid 3-Partitioning of $V(G)$.

- For every i and j with $1 \leq i \leq |V(G)|$ and $1 \leq j \leq 3$, the variables $m_{i,j}$, $\mathrm{sl}^1_{i,j}$, and $\mathrm{sl}^2_{i,j}$ (with an arbitrary non-negative domain), $r_{i,j}$ (with domain between 0 and $p(i) - 1$), and $u_{i,j}$ (with binary domain). These variables are used to secure condition C1.

- For every $e \in E(G)$, $i \in e$, and j with $1 \leq j \leq 3$, the variables $m_{e,i,j}$, $\mathrm{sl}^3_{e,i,j}$, $\mathrm{sl}^4_{e,i,j}$, and $\mathrm{sl}^5_{e,j}$ (with an arbitrary non-negative domain), $r_{e,i,j}$ (with domain between 0 and $p(i) - 1$), and $u_{e,i,j}$ (with binary domain). These variables are used to secure condition C2.

Note that the variables $\mathrm{sl}^1_{i,j}$, $\mathrm{sl}^2_{i,j}$, $\mathrm{sl}^3_{e,i,j}$, $\mathrm{sl}^4_{e,i,j}$, and $\mathrm{sl}^5_{e,i}$ are so-called "Slack" variables, whose sole purpose is to obtain an ILP instance that is in equation normal form. The instance $\mathbf{I}$ has the following constraints (in the following let α be any feasible assignment of $\mathbf{I}$):

- domain restrictions for all variables as given above, i.e.:

  - for every i and j with $1 \leq i \leq |V(G)|$ and $1 \leq j \leq 3$, the constraints $c_j \geq 0$, $m_{i,j} \geq 0$, $\mathrm{sl}^1_{i,j} \geq 0$, $\mathrm{sl}^2_{i,j} \geq 0$, $0 \leq r_{i,j} \leq p(i) - 1$, and $0 \leq u_{i,j} \leq 1$.

- for every $e \in E(G)$, $i \in e$, and j with $1 \le j \le 3$, the constraints $m_{e,i,j} \ge 0$, $\text{sl}^3_{e,i,j} \ge 0$, $\text{sl}^4_{e,i,j} \ge 0$, $\text{sl}^5_{e,j} \ge 0$, $0 \le r_{e,i,j} \le p(i) - 1$, and $0 \le u_{e,i,j} \le 1$.

- The following constraints, introduced for each $1 \le i \le |V(G)|$ and $1 \le j \le 3$, together guarantee that condition C1 holds:

  - Constraints that ensure that $\alpha(r_{i,j})$ is equal to the remainder of $\alpha(c_j)$ divided by $p(i)$, i.e., the constraint $c_j = p(i)m_{i,j} + r_{i,j}$.
  - Constraints that ensure that $\alpha(u_{i,j}) = 0$ if and only if $\alpha(r_{i,j}) = 0$, i.e., the constraints $u_{i,j} + \text{sl}^1_{i,j} = r_{i,j}$ and $r_{i,j} + \text{sl}^2_{i,j} = (p(i) - 1)u_{i,j}$. Note that together the above constraints now ensure that $\alpha(u_{i,j}) = 0$ if and only if c_j is divisible by $p(i)$.
  - Constraints that ensure that exactly one of $\alpha(u_{i,1})$, $\alpha(u_{i,2})$, and $\alpha(u_{i,3})$ is equal to 0, i.e., the constraints $u_{i,1} + u_{i,2} + u_{i,3} = 2$. Note that together all the above constraints now ensure condition C1 holds.

- The following constraints, introduced for each $1 \le j \le 3$, together guarantee that condition C2 holds:

  - Constraints that ensure that for every $e \in E(G)$ and $i \in e$, it holds that $\alpha(r_{e,i,j})$ is equal to the remainder of c_j divided by $p(i)$, i.e., the constraint $c_j = p(i)m_{e,i,j} + r_{e,i,j}$.
  - Constraints that ensure that for every $e \in E(G)$, $i \in e$, and j with $1 \le j \le 3$ it holds that $\alpha(u_{e,i,j}) = 0$ if and only if $\alpha(r_{e,i,j}) = 0$, i.e., the constraints $u_{e,i,j} + \text{sl}^3_{e,i,j} = r_{e,i,j}$ and $r_{e,i,j} + \text{sl}^4_{e,i,j} = (p(i) - 1)u_{e,i,j}$. Note that together the above constraints now ensure that $\alpha(u_{e,i,j}) = 0$ if and only if c_j is divisible by $p(i)$.
  - Constraints that ensure that for every $e = \{i, k\} \in E(G)$ and j with $1 \le j \le 3$ it holds that at least one of $\alpha(u_{e,i,j})$ and $\alpha(u_{e,k,j})$ is non-zero, i.e., the constraint $u_{e,i,j} + u_{e,k,j} - \text{sl}^5_{e,j} = 1$. Note that together with all of the above constraints this now ensures condition C2.

This completes the construction of $\mathbf{I}$. Clearly $\mathbf{I}$ can be constructed in polynomial time, and the largest coefficient used by $\mathbf{I}$ is equal to $p(|V(G)|)$. It is well-known that $p(i)$ is upper-bounded by $O(i \log i)$ due to the Prime Number Theorem, and so this in particular implies that the numbers which occur in $\mathbf{I}$ are bounded by a polynomial in $|V(G)|$.

Following the construction and explanations provided above, it is not difficult to see that $\mathbf{I}$ has a feasible assignment if and only if G has a 3-coloring. Indeed, for any 3-coloring of G, one can construct a feasible assignment of $\mathbf{I}$ by computing the prime-number encoding for vertices that receive colors $1, 2, 3$ and assign these three numbers to c_1, c_2, c_3, respectively. Such an assignment allows us to straightforwardly satisfy the constraints ensuring C1 holds (since each prime occurs in exactly one global constraint), the constraints ensuring C2 holds (since each edge is incident to at most one of each color) while maintaining the domain bounds.

On the other hand, for any feasible assignment α, clearly each of $\alpha(c_1), \alpha(c_2), \alpha(c_3)$ will be divisible by some subset of prime numbers between 2 and $p(|V(G)|)$. In particular, since α is feasible it follows from the construction of our first group of constraints that each prime between 2 and $p(|V(G)|)$ divides precisely one of $\alpha(c_1), \alpha(c_2), \alpha(c_3)$, and so this uniquely encodes a corresponding candidate 3-coloring for the vertices of the graph. Finally, since α also satisfies the second group of constraints, this candidate 3-coloring must have the property that each edge is incident to at exactly 2 colors, and so it is in fact a valid 3-coloring.

It remains to show that $\mathfrak{p}^V(\mathbf{I}) \le 25$. We show this by showing that the set $B = \{c_1, c_2, c_3\}$ is a variable backdoor set to 25-compactness. Note that the graph $G_\mathbf{I} \setminus \{c_1, \dots, c_3\}$ has only two types of components (all other components are isomorphic to one of the two types):

- for every i with $1 \le i \le |V(G)|$, one component containing the variables $m_{i,1}, \dots, m_{i,3}$, $\text{sl}^1_{i,1}, \dots, \text{sl}^1_{i,3}$, $\text{sl}^2_{i,1}, \dots, \text{sl}^2_{i,3}$, $r_{i,1}, \dots, r_{i,3}$, $u_{i,1}, \dots, u_{i,3}$. Moreover, these 15 variables occur in exactly 10 constraints together; these are the constraints introduced above to ensure condition C1. Hence the total size of these components is 25.

- for every $e = \{w, v\} \in E(G)$ and j with $1 \le j \le 3$, one component on the vertices $m_{e,w,j}, m_{e,v,j}, \text{sl}^3_{e,w,j}, \text{sl}^4_{e,v,j}$, $r_{e,w,j}, r_{e,v,j}, u_{e,w,j}, u_{e,v,j}$, and $\text{sl}^5_{e,j}$. Moreover, these 9 variables occur in exactly 7 constraints together; these are the constraints introduced above to ensure condition C2. Hence the total size of these components is 16.

This show that B is a variable backdoor to 25-compactness, as required. □

Theorem 21. UNARY ILP-FEASIBILITY *is* W[1]-*hard parameterized by* $\mathfrak{p}^C(\mathbf{I})$.

Proof. We prove the theorem by a parameterized reduction from MULTICOLORED CLIQUE, which is well-known to be W[1]-complete [Pietrzak, 2003]. Given an integer k and a k-partite graph G with partition $V_1, \dots, V_k$, the MULTICOLORED CLIQUE problem ask whether G contains a k-clique. In the following we denote by $E_{i,j}$ the set of all edges in G with one endpoint in V_i and the other endpoint in V_j, for every i and j with $1 \le i < j \le k$. To show the lemma, we will construct an instance $\mathbf{I}$ of ILP-FEASIBILITY in polynomial time that has a constraint backdoor set of size $2k + 2\binom{k}{2}$ to 3-compactness and coefficients bounded by a polynomial in $|V(G)|$ such that G has a k-clique if and only if $\mathbf{I}$ has a feasible assignment.

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, i.e., verifying that

the chosen vertices and edges indeed form a k-clique of G, 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. Indeed a *Sidon sequence* is a sequence of natural numbers such that the sum of every two distinct numbers in the sequence is unique. For our reduction we will need a Sidon sequence of $|V(G)|$ natural numbers, i.e., containing one number for each vertex of G. Since the numbers in the Sidon sequence will be used as coefficients of **I**, we need to ensure that the largest of these numbers is bounded by a polynomial in G. Indeed [Erdös and Turán, 1941] shows that a Sidon sequence containing n elements and whose largest element is at most $2p^2$, where p is the smallest prime number larger or equal to n can be constructed in polynomial time. Together with Bertrand's postulate [Aigner *et al.*, 2010], which states that for every natural number n there is a prime number between n and $2n$, we obtain that a Sidon sequence containing $|V(G)|$ numbers and whose largest element is at most $8|V(G)|^2$ can be found in polynomial time. In the following we will assume that we are given such a Sidon sequence $\mathcal{S}$ and we denote by $\mathcal{S}(i)$ the i-th element of $\mathcal{S}$ for any i with $1 \leq i \leq |V(G)|$. Moreover, we denote by $\max(\mathcal{S})$ and $\max_2(\mathcal{S})$ the largest element of $\mathcal{S}$ respectively the maximum sum of any two numbers in $\mathcal{S}$.

We are now ready to construct the instance **I** of ILP-FEA-SIBILITY such that G has a k-clique if and only if **I** has a feasible assignment. This instance **I** has the following variables:

- For every $v \in V(G)$ a binary variable v (with domain $\{0, 1\}$) that is 1 if v is selected to be in the k-clique and 0 otherwise.

- For every $e \in E(G)$ a binary variable e (with domain $\{0, 1\}$) that is 1 if e is selected to be in the k-clique and 0 otherwise.

- For every i with $1 \leq i \leq k$, a variable v_i (with unrestricted domain), which will be set to $\mathcal{S}(v)$ if the vertex $v \in V_i$ was selected to be in the k-clique.

- For every i and j with $1 \leq i < j \leq k$, a variable $e_{i,j}$ (with unrestricted domain), which will be set to $\mathcal{S}(v) + \mathcal{S}(u)$ if the edge $e \in E_{i,j}$ with $e = \{u, v\}$ was selected to be in the k-clique.

I has the following constraints:

- Constraints that restrict the domains of all variables as specified above, i.e.:
  - for every $v \in V(G)$, the constraints $0 \leq v \leq 1$.
  - for every $e \in E(G)$, the constraints $0 \leq e \leq 1$.

  We will denote by D the set of all these constraints.

- for every i with $1 \leq i \leq k$, the constraint $\sum_{v \in V_i} v = 1$, which ensures that from every part V_i exactly one vertex is selected to be in the k-clique. We will denote by V_{SEL} the set of all these constraints.

- for every i and j with $1 \leq i < j \leq k$, the constraint $\sum_{e \in E_{i,j}} e = 1$, which ensures that between any two parts V_i and V_j exactly one edge is selected to be in the k-clique. We will denote by E_{SEL} the set of all these constraints.

- for every i with $1 \leq i \leq k$, the constraint $\sum_{v \in V_i} \mathcal{S}(v)v = v_i$, which ensures that v_i is equal to $\mathcal{S}(v)$ whenever v is selected for the k-clique. We will denote by V_{ASS} the set of all these constraints.

- for every i and j with $1 \leq i < j \leq k$, the constraint $\sum_{e = \{u,v\} \in E_{i,j}} (\mathcal{S}(u) + \mathcal{S}(v))e = e_{i,j}$, which ensures that $e_{i,j}$ is equal to $\mathcal{S}(u) + \mathcal{S}(v)$ whenever the edge $e \in E_{i,j}$ with endpoints u and v is selected for the k-clique. We will denote by E_{ASS} the set of all these constraints.

- for every i and j with $1 \leq i < j \leq k$, the constraint $v_i + v_j = e_{i,j}$, which ensures that between any two parts V_i and V_j the vertices selected for the clique are equal to the endpoints of the edge chosen between the two parts. We will denote by VE_{CHECK} the set of all these constraints.

This completes the construction of **I**. Clearly **I** can be constructed in polynomial time, and the largest coefficient used by I is equal to $\max_2(\mathcal{S})$, which is at most $2\max(\mathcal{S}) \leq 16|V(G)|^2$. We first show that **I** has a small constraint backdoor to 3-compactness, and hence our parameter can bounded in terms of k. Namely, we claim that the set $B = V_{SEL} \cup E_{SEL} \cup V_{ASS} \cup E_{ASS} \cup VE_{CHECK}$ of constraints of **I** is a constraint backdoor of size at most $2k + 3\binom{k}{2}$ to 3-compactness. Clearly, the components of $G_\mathbf{I} \setminus B$ have size at most 3, i.e., $G_\mathbf{I}$ has one component of size one for every variable in $\{v_1, \ldots, v_k, e_{1,2}, \ldots, e_{k-1,k}\}$ as well as one component of size 3 for every $a \in V(G) \cup E(G)$, containing the variable a together with the two constrains $0 \leq a$ and $a \leq 1$. Moreover, the sets $V_{SEL}, E_{SEL}, V_{ASS}, E_{ASS}$, and VE_{CHECK} have sizes at most k, $\binom{k}{2}$, k, $\binom{k}{2}$, and $\binom{k}{2}$, respectively, which implies that $|B| \leq 2k + 3\binom{k}{2}$.

It remains to show that G has k-clique if and only if **I** is feasible. For the forward direction suppose that G has a k-clique on the vertices $c_1, \ldots, c_k$, where $c_i \in V_i$ for every i with $1 \leq i \leq k$. Then it is straightforward to verify that the assignment α with:

- $\alpha(c_i) = 1$ for every i with $1 \leq i \leq k$ and $\alpha(v) = 0$ for every $v \in V(G) \setminus \{c_1, \ldots, c_k\}$,

- $\alpha(\{c_i, c_j\}) = 1$ for every i and j with $1 \leq i < j \leq k$ and $\alpha(e) = 0$ for every $e \in E(G) \setminus \{\{c_i, c_j\} : 1 \leq i < j \leq k\}$,

- $\alpha(v_i) = \mathcal{S}(c_i)$ for every i with $1 \leq i \leq k$, and

- $\alpha(e_{i,j}) = \mathcal{S}(c_i) + \mathcal{S}(c_j)$ for every i and j with $1 \leq i < j \leq k$

is a feasible assignment for **I**.

For the reverse direction suppose that we are given a feasible assignment α for $\mathbf{I}$. Then because α satisfies the constraints in $D \cup V_{SEL} \cup E_{SEL}$ we obtain that for every i and j with $1 \leq i < j \leq k$ it holds that exactly one of the variables in V_i and exactly one of the variables in $E_{i,j}$ is set to one. Let c_i denote the unique vertex in V_i with $\alpha(c_i) = 1$ and similarly let $d_{i,j}$ denote the unique edge in $E_{i,j}$ with $\alpha(d_{i,j}) = 1$. It follows from the constraints in V_{ASS} that $\alpha(v_i) = \mathcal{S}(c_i)$ and similarly using the constraints in E_{ASS} we obtain that $\alpha(e_{i,j}) = \mathcal{S}(u) + \mathcal{S}(v)$, where u and v are the endpoints of the edge $d_{i,j}$ in G. Moreover, we obtain from the constraints in VE_{CHECK} that $v_i + v_j = e_{i,j}$ and hence $\mathcal{S}(c_i) + \mathcal{S}(c_j) = \mathcal{S}(u) + \mathcal{S}(v)$, where again u and v are the endpoints of the edge $d_{i,j}$ in G. Because $\mathcal{S}$ is a Sidon sequence, it follows that this can only hold if $\mathcal{S}(c_i) = \mathcal{S}(u)$ and $\mathcal{S}(c_j) = \mathcal{S}(v)$, which implies that $c_i = u$ and $c_j = v$. This shows that the endpoints of the selected edges $d_{1,2}, \ldots, d_{k-1,k}$ are the vertices in $c_1, \ldots, c_k$ and hence $G[\{c_1, \ldots, c_k\}]$ is a k-clique of G. $\square$

We conclude with a simple reduction ruling out an extension of our XP algorithm to general ILPs parameterized by $\mathfrak{p}^c$.

Theorem 22. ILP *is* NP-*hard even if* $\mathfrak{p}^C = 1$.

Proof. We show the result by a polynomial reduction from the SUBSET SUM problem, which is well-known to be weakly NP-complete. Given a set $S := \{s_1, \ldots, s_n\}$ of integers and an integer s, the SUBSET SUM problem asks whether there is a subset $S' \subseteq S$ such that $\sum_{s \in S'} s' = s$. Let $I := (S, s)$ with $S := \{s_1, \ldots, s_n\}$ be an instance of SUBSET SUM. We will construct an equivalent ILP instance $\mathbf{I}$ with $\mathfrak{p}^C(\mathbf{I}) = 1$ in polynomial time as follows. The instance $\mathbf{I}$ has n binary variables $x_1, \ldots, x_n$ and apart from the domain constraints for these variables only one global constraint defined by $\sum_{1 \leq i \leq n} s_i x_i = s$. Because $\mathbf{I}$ has only one constraint, it holds that $\mathfrak{p}^C(\mathbf{I}) = 1$ and moreover it is straightforward to verify that $\mathbf{I}$ is equivalent to (S, s) (this has also for instance been shown in [Jansen and Kratsch, 2015, Theorem 1]). $\square$

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 based on *fracture backdoors* provides a unified platform which generalizes 2-stage stochastic ILP, N-fold ILP and also 4-block N-fold ILP. More importantly though, 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 believe that this is in fact 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, which was also left open in previous work; progress in this direction seems to require new techniques and insights [Hemmecke *et al.*, 2010].

Acknowledgements

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.

[Fellows *et al.*, 2008] Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Frances A. Rosamond, and Saket Saurabh. Graph layout problems parameterized by vertex

cover. In *Proc. ISAAC*, Lecture Notes in Computer Science, pages 294–305. Springer, 2008.

[Floudas and Lin, 2005] Christodoulos A. 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.

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

[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 *to appear in Proc. AAAI*, 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 Hans L. Bodlaender, Rod Downey, Fedor V. Fomin, and Dániel Marx, editors, *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*, pages 287–317. Springer Verlag, 2012.

[Gaspers et al., 2017] Serge Gaspers, Neeldhara Misra, Sebastian Ordyniak, Stefan Szeider, and Stanislav Živný. 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 *Lecture Notes in Computer Science*, pages 779–791. Springer Verlag, 2015.

[Kannan, 1987] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.

[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 *to appear in Proc. STACS*, 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, Philadelphia, PA, USA, 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. AAAI, 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.