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

Pavel Dvořák¹, Eduard Eiben², Robert Ganian², Dušan Knop¹, and Sebastian Ordyniak²
¹ Charles University, Prague, Czech Republic
² TU Wien, Vienna, Austria

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 $f(k) \cdot |I|^{O(1)}$ (a so-called fixed-parameter algorithm), or at least in time $|I|^{f(k)}$ (a so-called XP algorithm), where $f$ is a computable function. This approach lies at the core of the now well-established parameterized complexity paradigm [Downey and Fellows, 2013; Cygan et al., 2015] and has yielded deep results capturing the tractability and intractability of numerous prominent problems in diverse areas of computer science—such as Constraint Satisfaction, SAT, and a plethora of problems on directed and undirected graphs.

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

1.1 Our Contribution

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

As is evident from the table, backdoor size on its own is not sufficient to break the NP-hardness of ILP; this is far from...
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 backbone. It turns out that computing fracture backdoors in ILP is closely related to solving the VERTEX INTEGRITY problem [Drange et al., 2016] on bipartite graphs; unfortunately, while the problem has been studied on numerous graph classes including cobiclique 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 \(A\) be a \(n \times m\) matrix and let \(C\) and \(R\) be a subset of columns and rows of \(A\), respectively. We denote by \(A_{(R,C)}\) the submatrix of \(A\) restricted to the columns in \(C\) and the rows in \(R\). We also denote by \(A_{(\ast ,C)}\) and \(A_{(R,\ast )}\) the submatrix of \(A\) restricted to the columns in \(C\) and the submatrix of \(A\) restricted to the rows in \(R\), respectively. We denote by \(c_{\text{UB}}\) the maximum absolute value of any entry of \(A\). For a vector \(b\) of size \(n\), we will use \(b[i]\) to denote its \(i\)-th entry and we denote by \(c_{\text{UB}}\) the maximum absolute value of any entry of \(b\).

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

A (partial) assignment \(\alpha\) is a mapping from some subset of \(\text{var}(I)\), denoted by \(\text{var}(\alpha)\), to \(\mathbb{Z}\). An assignment \(\alpha\) is called feasible if \((I)\) satisfies every constraint in \(\mathcal{F}\), i.e., if \(\alpha(x) = b[i]\) for each \(i\)-th row \(A\) of \(A\), and (2) it satisfies all the upper and lower bounds, i.e., \(l[i] \leq \alpha(x[i]) \leq u[i]\). Furthermore, \(\alpha\) is called a solution if the value of \(\eta(x)\) 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 \(\alpha\) with increasing values of \(\eta(x)\); in this case, we speak of unbounded instances). Given an instance \(I\), the task in the ILP problem is to compute a solution for \(I\) or correctly determine that no solution exists. We remark that other formulations of ILP exist (e.g., a set of inequalities over variables); it is well-known that these are equivalent and can be transformed into each other in polynomial time. Moreover, such transformations will only change our parameters (defined in Section 3) by a constant factor.

Aside from general integer linear programming, we will also be concerned with two subclasses of the problem. ILP-feasibility is formulated equivalently as ILP, with the restriction that \(\eta\) must be the 0-vector. Unary ILP is the class...
of all ILP instances which are supplied in a unary bit encoding; in other words, the input size of UNARY ILP upper-bounds not only the number of variables and constraints, but also the absolute values of all numbers in the input. UNARY ILP remains NP-complete in general, but in our setting there will be cases where its complexity will differ from general ILP. Combining both restrictions gives rise to UNARY ILP-FEASIBILITY.

There are several ways of naturally representing ILP instances as graphs. The representation that will be most useful for our purposes will be the so-called incidence graph: the incidence graph \( G_1 \) of an ILP instance \( I \) is the graph whose vertex set is \( \text{var}(I) \cup \mathcal{F}(I) \) and two vertices \( s, t \) are adjacent iff \( s \in \text{var}(I), t \in \mathcal{F} \) and \( s \) occurs in \( t \) with a non-zero coefficient. An instance \( I' \) is a connected component of \( I \) if it is the subinstance of \( I \) corresponding to a connected component of \( G_1 \); formally, \( \mathcal{F}(I') \subseteq \mathcal{F}(I) \) is the set of constraints that occur in a connected component of \( G_1 \) and \( \eta(I') \) is the restriction of \( \eta(I) \) to \( \mathcal{F}(I') \). For a set \( Z \subseteq \mathcal{F}(I) \cup \text{var}(I) \), we will also use \( I \setminus Z \) to denote the ILP instance obtained by removing all constraints in \( Z \) from \( \mathcal{F}(I) \) and removing all variables in \( Z \) from all constraints in \( \mathcal{F}(I) \setminus Z \) and from \( \eta \).

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 \( f(k) \cdot n^{O(1)} \), where \( f \) is a computable function. Problems that can be solved in this time are called fixed-parameter tractable (fpt). Aside from the complexity class FPT, we will also make use of the complexity classes \( W[1] \) (proving hardness for \( W[1] \) is often used to show that a problem is unlikely to be in FPT), XP (containing problems that can be solved in polynomial time whenever the parameter is bounded) and \( \text{pNP} \) (problems complete for \( \text{pNP} \) remain NP-complete even when the parameter is bounded). To obtain our lower bounds, we will need the notion of parameterized reduction, see for instance Downey and Fellows (2013).

2.3 ILP with Structured Matrices

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

Let \( A = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix} \) be a \( 2 \times 2 \) block integer matrix. The \( N \)-fold 4-block product of \( A \) (denoted by \( A^{(N)} \)) is the following integer matrix

\[
A^{(N)} = \begin{pmatrix} A_1 & A_2 & \cdots & A_2 \\ A_3 & 0 & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ A_3 & 0 & 0 & \cdots & A_4 \end{pmatrix}.
\]

Here \( A_1 \) is an \( r \times s \) matrix, \( A_2 \) is an \( r \times t \) matrix, \( A_3 \) is an \( u \times s \) matrix, and \( A_4 \) is an \( u \times t \) matrix; for convenience, we let \( b_A = \max(r, s, t, u) \). We call an instance \((A, x, b, l, u, \eta)\) of ILP an \( N\)-fold 4-block if \( A \) is an \( N\)-fold 4-block product of some \( 2 \times 2 \) block integer matrix. Observe that in such instances the vector \( x \) is naturally partitioned into a global part (consisting of \( s \) variables) and a local part.

Theorem 1 (Hemmecke et al., 2010). Let \( a \) and \( z \) be constants and let \( I \) be an \( N\)-fold 4-block ILP instance with \( c_A \leq a, b_A \leq z \), then \( I \) can be solved in polynomial time.

In the parameterized complexity setting, the above theorem yields an XP algorithm solving ILP parameterized by \( \max(b_A, c_A) \) 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 \( A_1 \) and \( A_3 \) or \( A_1 \) and \( A_2 \) are omitted; these variants are called the \( N\)-fold ILP problem and the 2-stage stochastic ILP problem, respectively.

Theorem 2 (Hemmecke et al., 2013; De Loera et al., 2013). \( N\)-fold ILP and 2-stage stochastic ILP are \( \text{fpt} \) parameterized by \( c_A \) and \( b_A \).

3 The Fracture Number

We are now ready to formally introduce the studied parameter and related notions. An ILP instance \( I \) is called \( \ell\)-compact if each connected component of \( I \) contains at most \( \ell \) variables and constraints; equivalently, each connected component of \( G_1 \) contains at most \( \ell \) vertices. It is not difficult to observe that any \( \ell\)-compact ILP instance is fixed-parameter tractable parameterized by \( \ell \); indeed, using the algorithm of Lenstra, Jr. (1983) we can compute a solution for each component individually and then combine the solutions to solve \( I \).

A set \( Z \subseteq \mathcal{F} \cup \text{var}(I) \) is called a backdoor to \( \ell\)-compactness if \( I \setminus Z \) is \( \ell\)-compact; moreover, if \( Z \cap \mathcal{F} = \emptyset \) then \( Z \) is called a variable-backdoor to \( \ell\)-compactness, and if \( Z \cap \text{var}(I) = \emptyset \) then \( Z \) is a constraint-backdoor to \( \ell\)-compactness. We use \( b_{\ell}(I) \) to denote the cardinality of a minimum backdoor to \( \ell\)-compactness, and similarly \( b_{\ell}^1(I) \) and \( b_{\ell}^2(I) \) for variable-backdoors and constraint-backdoors to \( \ell\)-compactness, respectively. It is easy to see that, depending on the instance, \( b_{\ell}^1(I) \) can be arbitrarily larger or smaller than \( b_{\ell}^2(I) \). On the other hand, \( b_{\ell}(I) \leq \min(b_{\ell}^1(I), b_{\ell}^2(I)) \).

Clearly, the choice of \( \ell \) has a major impact on the size of backdoors to \( \ell\)-compactness: in particular, \( b_{\ell}(I) \) could be arbitrarily larger than \( b_{\ell+1}(I) \), and the same of course also holds for variable- and constraint-backdoors. Since we will be interested in dealing with cases where both \( \ell \) and \( b_{\ell}(I) \) are small, we will introduce the fracture number \( p \) which provides bounds on both \( \ell \) and \( b_{\ell} \); in particular, we let \( p(I) = \min_{\ell \in \mathbb{N}} (\max(\ell, b_{\ell}(I))) \). Furthermore, we say that a backdoor witnesses \( p(I) \) if \( |Z| \leq p(I) \) and \( I \setminus Z \) is \( p(I)\)-compact. We define \( p^1(I) \) and \( p^2(I) \) similarly, with \( b_{\ell}(I) \) replaced by \( b_{\ell}^1(I) \) and \( b_{\ell}^2(I) \), respectively. If the instance \( I \) is clear from the context, we omit the reference to \( I \); see Figure 1 for an example.

We remark that the fracture number represents a strict generalization of the parameter \( b_A \) used in Theorems 1 and 2; in
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 $c_A$, which is the maximum absolute coefficient occurring in $A$. 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 $\ell$-compactness for an ILP instance $I$. We define the equivalence relation $\sim$ on the components of $I \setminus Z$ as follows: two components $C_1$ and $C_2$ are equivalent iff there exists a bijection $\gamma$ between $\operatorname{var}(C_1)$ and $\operatorname{var}(C_2)$ such that the ILP instance obtained from $I$ after renaming the variables in $\operatorname{var}(C_1)$ and $\operatorname{var}(C_2)$ according to $\gamma$ and $\gamma^{-1}$, respectively, is equal to $I$. We say that components $C_1$ and $C_2$ have the same type if $C_1 \sim C_2$.

**Lemma 6.** Let $I$ be an ILP instance and $k = p(I)$. For any backdoor witnessing $p(I)$, $\sim$ has at most $(2c_A(I) + 1)^{2k^2}$ equivalence classes. Moreover, one can test whether two components have the same type in time $O(k!)$. We now proceed to the main tool used for our algorithms.

**Theorem 7.** Let $I$ be an ILP instance with matrix $A$, $Z$ be a backdoor set witnessing $p(I)$, and let $n$ be the number of components of $I \setminus Z$. There is an algorithm which runs in time $O(n^2(p(I) + 1)! + |I|)$ and computes a $(r + u) \times (s + t)$ matrix $A = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}$, a positive integer $N \leq n$, and a $4$-block $N$-fold instance $I = (A^{(N)}, x, b, l, u, n)$ such that:

1. (P1) any solution for $I$ can be transformed (in polynomial time) into a solution for $I$ (and vice versa), and
2. (P2) $\max\{r, s\} \leq p(I)$ and $\max\{t, u\} \leq f(c_A, p(I))$ for some computable function $f$.

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

- the matrix $Q_C^Z$ is the part of the constraints in $C$ dealing with variables in $Z$, that is, $A_{\mathcal{F}(C), \operatorname{var}(Z)}$,
- the matrix $Q_C^Z$ is the part of the constraints in $C$ dealing with variables in $Z$, that is, $A_{\mathcal{F}(Z), \operatorname{var}(C)}$, and
- the matrix $Q_C$ is the part of the constraints in $C$ dealing with variables in $Z$, that is, $A_{\mathcal{F}(C), \operatorname{var}(C)}$.

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

Observe that if all types already have the same multiplicity then we can simply set $I = I$. Indeed, it suffices to rearrange the matrices $Q_C, Q_C^Z$, and $Q_D$ of all components into the matrices $A_1, A_2,$ and $A_3$ as follows. We arrange the matrices $Q_C$ of all components on a diagonal of the matrix $A_1$, the matrices $Q_C^Z$ next to each other in the matrix $A_2$, and the matrices $Q_D^Z$ under each other in the matrix $A_3$. The bounds on $r, s, t$ and $u$ then follow from Lemma 6.
On the other hand, if the multiplicities of types vary then we apply the following procedure for each type $T$ with multiplicity lower than $N$. We first pick one component $C$ of type $T$ and add $N - \text{mult}(T)$ new copies of $C$ to $I$. Let $O$ be the set of all original components of type $T$ and let $S$ be the set of all newly created copies of $C$. For every variable of every component in $S$ we set its lower and upper bound to 0; this ensures that components in $S$ will have no impact on the global constraints and the optimization function.

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

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

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

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

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

### 6 Unary ILP

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

#### 6.1 The Algorithm

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

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

We are now ready to show that we only need to consider solutions with polynomially bounded variable domain.
As mentioned above there can be at most $|Z| \leq p^C(I)$ such components. Because $C(U)$ is a subset of linearly independent columns there is a set $R'$ of $|U|$ linearly independent rows in $A_{x;C(U)}$. Then $\alpha$ has to satisfy $A_{(R',x)}\alpha(x) = b$ and because $A_{(R',C(U))}$ is non-singular, we obtain that the values of $\alpha$ for the variables in $U$ are completely determined by the values of $\alpha$ for all remaining variables, which as shown above are at most $m_{iM}$. Using Cramer’s Rule we calculate that $\alpha(v) \leq ml$ for every such variable $v$ in $U$.

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

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

### 6.2 Lower Bounds

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

**Theorem 13.** **UNARY ILP-FEASIBILITY** is $\#P$-hard parameterized by $p^C(I)$.

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

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

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

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

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

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

### 7 Concluding Notes

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

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

### Acknowledgements

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


