The Complexity Landscape of Decompositional Parameters for ILP

Robert Ganian and **Sebastian Ordyniak**
TU Wien, Vienna, Austria

Abstract

Integer Linear Programming (ILP) can be seen as the archetypical problem for NP-complete optimization problems, and a wide range of problems in artificial intelligence are solved in practice via a translation to ILP. Despite its huge range of applications, only few tractable fragments of ILP are known, probably the most prominent of which is based on the notion of total unimodularity. Using entirely different techniques, we identify new tractable fragments of ILP by studying structural parameterizations of the constraint matrix within the framework of parameterized complexity.

In particular, we show that ILP is fixed-parameter tractable when parameterized by the treedepth of the constraint matrix and the maximum absolute value of any coefficient occurring in the ILP instance. Together with matching hardness results for the more general parameter treewidth, we draw a detailed complexity landscape of ILP w.r.t. decompositional parameters defined on the constraint matrix.

Introduction

Integer Linear Programming (ILP) is among the most successful and general paradigms for solving computationally intractable optimization problems in computer science. In particular, a wide variety of problems in artificial intelligence are efficiently solved in practice via a translation into an Integer Linear Program, including problems from areas such as process scheduling (Floudas and Lin 2005), planning (van den Briel, Vossen, and Kambhampati 2005; Vossen et al. 1999), vehicle routing (Toth and Vigo 2001), packing (Lodi, Martello, and Monaci 2002), and network hub location (Alumur and Kara 2008). In its most general form ILP can be formalized as follows:

INTEGER LINEAR PROGRAM	
Input:	A matrix $A \in \mathbb{Z}^{m \times n}$ and two vectors $b \in \mathbb{Z}^m$ and $c \in \mathbb{Z}^n$.
Question:	Maximize cx for every $x \in \mathbb{Z}^n$ with $Ax \leq b$.

Closely related to ILP is the ILP-FEASIBILITY problem, where given A and b as above, the problem is to decide whether there is an $x \in \mathbb{Z}^n$ such that $Ax \leq b$. ILP, ILP-FEASIBILITY and various other highly restricted variants are well-known to be NP-complete (Papadimitriou 1981).

Despite the importance of the problem, an understanding of the influence of structural restrictions on the complexity of ILP is still in its infancy. This is in stark contrast to another well-known and general paradigm for the solution of problems in Computer Science, the Satisfiability problem (SAT). There, the parameterized complexity framework (Downey and Fellows 2013) has yielded deep results capturing the tractability and intractability of SAT with respect to a plethora of structural restrictions. In the context of SAT, one often considers structural restrictions on a graphical representation of the formula (such as the primal graph), and the aim is to design efficient fixed-parameter algorithms for SAT, i.e., algorithms running in time $\mathcal{O}(f(k)n^{\mathcal{O}(1)})$ where k is the value of the considered structural parameter for the given SAT instance and n is its input size. It is known that SAT is fixed-parameter tractable w.r.t. a variety of structural parameters, such as treewidth (Szeider 2003) or the directed variants of clique-width (Fischer, Makowsky, and Ravve 2008) and rank-width (Ganian, Hliněný, and Obdržálek 2010), but is not fixed-parameter tractable (under standard assumptions) for others, such as undirected clique-width (Ordyniak, Paulusma, and Szeider 2013).

Our contribution In this work, we initiate a similar line of research for ILP by studying the parameterized complexity of ILP w.r.t. various structural parameterizations. In particular, we consider parameterizations of the *primal graph* of the ILP instance, i.e., the undirected graph whose vertex set is the set of variables of the ILP instance and whose edges represent the occurrence of two variables in a common expression. We obtain a complete picture of the parameterized complexity of ILP w.r.t. well-known decompositional parameters of the primal graph, specifically treedepth, treewidth, and cliquewidth; our results are summarized in Table 1.

Our main algorithmic result (Theorem 5) shows that ILP is fixed-parameter tractable parameterized by the *treedepth* of the primal graph and the maximum absolute value ℓ of any coefficient occurring in A or b. Together with the classical results for *totally unimodular* matrices (Papadimitriou and Steiglitz 1982, Section 13.2.) and fixed number of variables (Lenstra and Jr. 1983), which use entirely different techniques, our result is one of the surprisingly few tractability results for ILP in its full generality.

We complete our complexity landscape (given in Table 1) with matching lower bounds, provided in terms of W[1]-hardness and paraNP-hardness results (see the

	ℓ	without ℓ
TD	FPT (Thm 5)	W[1]-h (Thm 11)
TW/CW	paraNP-h (Thm 12)	paraNP-h (Thm 12)
no	paraNP-h (Obs 1)	n.a.

Table 1: The complexity landscape of ILP obtained in this paper. The table shows the parameterized complexity of ILP parameterized by the treedepth (TD), treewidth (TW), or cliquewidth (CW) of the primal graph with (second column "ℓ") and without (third column "without ℓ") the additional parameterization by the maximum absolute value ℓ of any coefficient in A or b.

Preliminaries). Namely, we show that already ILP-FEASIBILITY is unlikely to be fixed-parameter tractable when parameterized by either only treedepth or only ℓ. Interestingly, our W[1]-hardness result for treedepth also holds in the strong sense, i.e., even for ILP instances whose size is bounded by a polynomial of n and m.

One might be tempted to think that, as is the case for SAT and numerous other problems, the fixed-parameter tractability result for treedepth carries over to the more general structural parameter treewidth. We show that this is not the case for ILP; along with very recent results for the Mixed Chinese Postman Problem (Gutin, Jones, and Wahlström 2015), this is only the second known case of a natural problem where using treedepth instead of treewidth actually "helps" in terms of fixed parameter tractability. Even more surprisingly, we show that already ILP-FEASIBILITY remains NP-hard for ILP instances of treewidth at most three and whose maximum coefficient is at most two. Observe that this also implies the same intractability results for the more general parameter clique-width.

Related Work We are not the first to consider decompositional parameterizations of the primal graph for ILP. However, previous results in this area either considered an ad-hoc bound on the variable domains (Jansen and Kratsch 2015) or required much stronger restrictions on the coefficients of A (non-negativity) (Cunningham and Geelen 2007).

Preliminaries

We will use standard graph terminology, see for instance the handbook by Diestel (2012). All our graphs are simple and loopless.

Integer Linear Programming

For our purposes, it will be useful to view an ILP instance as a set of linear inequalities rather than using the constraint matrix. Formally, let an ILP instance I be a tuple $(\mathcal{F}, \eta)$ where $\mathcal{F}$ is a set of linear inequalities over variables $X = x_1, \dots, x_n$ and η is a linear function over X of the form $\eta(X) = s_{x_1}x_1 + \cdots + s_{x_n n}x_n$. Each inequality $A \in \mathcal{F}$ is assumed to be of the form $c_{A,1}x + c_{A,2}x_{A,2} + \cdots \leq b_A$; the set of variables which occur in A is denoted var(A), and we let var$(I) = X$. The *arity* of A is $|\text{var}(A)|$. For a set of variables Y, let $\mathcal{F}(Y)$ denote the subset of $\mathcal{F}$ containing all inequalities $A \in \mathcal{F}$ such that $Y \cap \text{var}(A) \neq \emptyset$.

An assignment α is a mapping from X to $\mathbb{Z}$. For an assignment α and an inequality A, we denote by $A(\alpha)$ the left-side value of A obtained by applying α, i.e., $A(\alpha) = c_{A,1}\alpha(x_{A,1}) + c_{A,2}\alpha(x_{A,2}) + \dots$. An assignment α is called *feasible* if it satisfies every $A \in \mathcal{F}$, i.e., if $A(\alpha) \leq b_A$ for each $A \in \mathcal{F}$. Furthermore, α is called a *solution* if the value of $\eta(\alpha)$ 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 $\eta(\alpha)$). Given an instance I, the task in the ILP problem is to compute a solution for I if one exists, and otherwise to decide whether there exists a feasible assignment.

Given an ILP instance $I = (\mathcal{F}, \eta)$, the primal graph G_I of I is the graph whose vertex set is the set X of variables in I, and two vertices a, b are adjacent iff either there exists some $A \in \mathcal{F}$ containing both a and b or a, b both occur in η with non-zero coefficients.

Parameterized Complexity

In parameterized algorithmics (Downey and Fellows 1999; 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. Algorithms with this running time are called *fpt-algorithms*.

To obtain our lower bounds, we will need the notion of a *parameterized reduction* and the complexity classes W[1] and paraNP (Downey and Fellows 2013). Generally speaking, a parameterized reduction is a variant of the standard polynomial reduction which retains bounds on the parameter, and both W[1] and paraNP rule out the existence of fpt-algorithms; the latter additionally rules out the existence of algorithms running in time $n^{f(k)}$.

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; Ganian, Kim, and Szeider 2015).

Theorem 1 (Lenstra and Jr.; Kannan; Frank and Tardos (1983; 1987; 1987)). *An ILP instance $I = (\mathcal{F}, \eta)$ can be solved in time $\mathcal{O}(p^{2.5p+o(p)} \cdot |I|)$, where $p = |\text{var}(I)|$.*

Treewidth and Treedepth

Treewidth is the most prominent structural parameter and has been extensively studied in a number of fields. We refer to, e.g., Downey and Fellows (2013) for a definition of treewidth and the related notions of *tree-decompositions* and *bags*.

Another important notion that we make use of extensively is that of treedepth. Treedepth is a structural parameter closely related to treewidth, and the structure of graphs of bounded treedepth is well understood (Nešetřil and Ossona de Mendez 2012). A useful way of thinking about graphs of bounded treedepth is that they are (sparse) graphs with no long paths.

We formalize a few notions needed to define treedepth. A *rooted forest* is a disjoint union of rooted trees. For a vertex x in a tree T of a rooted forest, the *height* (or *depth*) of x in the forest is the number of vertices in the path from the root of T to x. The *height of a rooted forest* is the maximum height of a vertex of the forest.

Definition 2 (Treedepth)**.** Let the *closure* of a rooted forest $\mathcal{F}$ be the graph $\text{clos}(\mathcal{F}) = (V_c, E_c)$ with the vertex set $V_c = \bigcup_{T \in \mathcal{F}} V(T)$ and the edge set $E_c = \{xy\colon x \text{ is an ancestor of } y \text{ in some } T \in \mathcal{F}\}$. A *treedepth decomposition* of a graph G is a rooted forest $\mathcal{F}$ such that $G \subseteq \text{clos}(\mathcal{F})$. The *treedepth* $td(G)$ of a graph G is the minimum height of any treedepth decomposition of G.

We will later use T_x to denote the vertex set of the subtree of T rooted at a vertex x of T. Similarly to treewidth, it is possible to determine the treedepth of a graph in FPT time.

Proposition 3 (Nešetřil and Ossona de Mendez (2012))**.** *Given a graph G with n nodes and a constant w, it is possible to decide whether G has treedepth at most w, and if so, to compute an optimal treedepth decomposition of G in time $\mathcal{O}(n)$.*

We list a few useful facts about treedepth.

Proposition 4 (Nešetřil and Ossona de Mendez (2012))**.**

1. *If a graph G has no path of length d, then $td(G) \leq d$.*
2. *If $td(G) \leq d$, then G has no path of length 2^d.*
3. *$tw(G) \leq td(G)$.*
4. *If $td(G) \leq d$, then $td(G') \leq d + 1$ for any graph G' obtained by adding one vertex into G.*

Exploiting Treedepth to Solve ILP

Our goal in this section is to show that ILP is fixed parameter tractable when parameterized by the treedepth of the primal graph and the maximum coefficient in any constraint. We begin by formalizing our parameters. Given an ILP instance I, let $td(I)$ be the treedepth of G_I and let $\ell(I)$ be the maximum absolute coefficient which occurs in any inequality in I; to be more precise, $\ell(I) = \max\{\, |c_{A,j}|, |b_A| : A \in \mathcal{F}, j \in \mathbb{N}\,\}$. When the instance I is clear from the context, we will simply write ℓ and $k = td(I)$ for brevity. We will now state our main algorithmic result of this section.

Theorem 5. *ILP is fixed-parameter tractable parameterized by ℓ and k*

The main idea behind our fixed-parameter algorithm for ILP is to show that we can reduce the instance into an "equivalent instance" such that the number of variables of the reduced instance can be bounded by our parameters ℓ and k. We then apply Theorem 1 to solve the reduced instance.

For the following considerations, we fix an ILP instance $I = (\mathcal{F}, \eta)$ of size n along with a treedepth decomposition T of G_I with depth k. Given a variable set Y, the operation of *omitting* consists of deleting all inequalities containing at least one variable in Y and all variables in Y; formally, omitting Y from I results in the instance $I' = (\mathcal{F}', \eta')$ where $\mathcal{F}' = \mathcal{F} \setminus \mathcal{F}(Y)$ and η' is obtained by removing all variables in Y from η.

The following notion of equivalence will be crucial for the proof of Theorem 5. Let x, y be two variables that share a common parent in T. We say that x are y are *equivalent*, denoted $x \sim y$, if there exists a bijective function $\delta_{x,y} : T_x \to T_y$ (called the *renaming function*) such that $\delta_{x,y}(\mathcal{F}(T_x)) = \mathcal{F}(T_y)$; here $\delta_{x,y}(\mathcal{F}(T_x))$ denotes the set of inequalities in $\mathcal{F}(T_x)$ after the application of $\delta_{x,y}$ on each variable in T_x. It is easy to verify that $\sim$ is indeed an equivalence. Intuitively, the following lemma shows that if $x \sim y$ for two variables x and y of I, then (up to renaming) the set of all feasible assignments of the variables in T_x is equal to the set of all feasible assignments of the variables in T_y.

Lemma 6. *Let x, y be two variables of I such that $x \sim y$ and $s_a = 0$ for each $a \in T_x \cup T_y$. Let $I' = (\mathcal{F}', \eta')$ be the instance obtained from I by omitting T_y. Then there exists a solution α of $\text{var}(I)$ of value $w = \eta(\alpha)$ if and only if there exists a solution α' of $\text{var}(I')$ of value $w = \eta'(\alpha')$. Moreover, a solution α can be computed from any solution α' in linear time if the renaming function $\delta_{x,y}$ is known.*

Proof. Let α be a solution of $\text{var}(I)$ of value $w = \eta(\alpha)$. Since $\mathcal{F}' \subseteq \mathcal{F}$, it follows that setting α' to be a restriction of α to $\text{var}(I) \setminus T_y$ satisfies every inequality in $\mathcal{F}'$. Since variables in T_y do not contribute to η, it also follows that $\eta(\alpha) = \eta(\alpha')$.

On the other hand, let α' be a solution of $\text{var}(I')$ of value $w = \eta'(\alpha')$. Consider the assignment α obtained by extending α' to T_y by reusing the assignments of T_x on T_y. Formally, for each $y' \in T_y$ we set $\alpha(y') = \alpha'(\delta_{y,x}(y'))$ and for all other variables $a \in \text{var}(I')$ we set $\alpha(a) = \alpha'(a)$. By assumption, α and α' must assign the same values to any variable a such that $s_a \neq 0$, and hence $\eta(\alpha) = \eta(\alpha')$. To argue feasibility, first observe that any $A \in \mathcal{F}'$ must be satisfied by α since α and α' only differ on variables which do not occur in I'. Moreover, by definition of $\sim$ for each $A \in \mathcal{F} \setminus \mathcal{F}' = \mathcal{F}(T_y)$ there exists an inequality $A' \in \mathcal{F}'$ such that $\delta_{x,y}(A') = A$. In particular, this implies that $A(\alpha) = A'(\alpha) = A'(\alpha')$, and since $A'(\alpha') \leq b_{A'} = b_A$ we conclude that $A(\alpha) \leq b_A$. Consequently, α satisfies A.

The final claim of the lemma follows from the construction of α described above. □

In the following let z be a variable of I at depth $k - i$ in T for every i with $1 \leq i < k$ and let Z be the set of all children of z in T. Moreover, let m be the maximum size of any subtree rooted at a child of z in T, i.e., $m := \max_{z' \in Z} |T_{z'}|$. We will show next that the number of equivalent classes among the children of z can be bounded by the function $\#\text{C}(\ell, k, i, m) := 2^{(2\ell+1)^{k+1} \cdot m^i}$. Observe that this bound depends only on ℓ, k, m, and i and not on the size of I.

Lemma 7. *The equivalence relation $\sim$ has at most $\#\text{C}(\ell, k, i, m)$ equivalence classes over Z.*

Proof. Consider an element $a \in Z$. By construction of G_I, each inequality $A \in \mathcal{F}(T_a)$ only contains at most $k-i$ variables outside of T_a (specifically, the ancestors of a) and at most i variables in T_a. Furthermore, b_A and each coefficient of a variable in A is an integer whose absolute value does not exceed ℓ. From this it follows that there exists a finite number of inequalities which can occur in $\mathcal{F}(T_a)$. Specifically, the number of distinct combinations of coefficients for all the variables in A and for b_A is $(2\ell+1)^{k+1}$, and the number of distinct choices of variables in var$(A) \cap T_a$ is upper-bounded by $\binom{m}{i}$, and so we arrive at $|\mathcal{F}(T_a)| \leq (2\ell+1)^{k+1} \cdot \binom{m}{i} \leq (2\ell+1)^{k+1} \cdot m^i$.

Consequently, the set of all inequalities for individual children $y \in Z$ of z (up to renaming of T_y) has bounded cardinality. To formalize this bound, we first need a formal way of renaming all variables in the individual subtrees rooted in Z; without renaming, each $\mathcal{F}(T_y)$ would span a distinct set of variables and hence it would not be possible to bound the set of all such inequalities. So, for each y let δ_{y,x_0} be a bijective renaming function which renames all of the variables in T_y to the variable set $\{x_0^1, x_0^2, \ldots, x_0^{|T_y|}\}$ (in an arbitrary way). Now we can formally define $\Gamma_z = \{\, \mathcal{F}(T_{x_0}) : \delta_{y,x_0}(\mathcal{F}(T_y)), y \in Z \,\}$, and observe that Γ_z has cardinality at most $2^{(2\ell+1)^{k+1} \cdot m^i} = \#\mathrm{C}(\ell,k,i,m)$. To conclude the proof, recall that if two variables a, b satisfy $\mathcal{F}(T_a) = \delta_{b,a}(\mathcal{F}(T_b))$ for a bijective renaming function $\delta_{b,a}$, then $b \sim a$. Hence, the absolute bound on the cardinality of Γ_z implies that $\sim$ has at most $\#\mathrm{C}(\ell,k,i,m)$ equivalence classes over Z. ☐

It follows from the above Lemma that if z has more than $\#\mathrm{C}(\ell,k,i,m)$ children, then two of those must be equivalent. The next lemma shows that it is also possible to find such a pair of equivalent children efficiently.

Lemma 8. *Given a subset Z' of Z with $|Z'| = \#\mathrm{C}(\ell,k,i,m)+1$, then in time $\mathcal{O}(\#\mathrm{C}(\ell,k,i,m)^2 \cdot m!m)$ one can find two children x and y of Z such that $x \sim y$ together with a renaming function $\delta_{x,y}$ which certifies this.*

Proof. The algorithm finds x, y among the elements of Z' by brute-force, i.e., it checks for all distinct (of the at most $(\#\mathrm{C}(\ell,k,i,m)+1)^2$) pairs x and y of elements in Z' and all (of the at most $m!$) distinct bijective renaming functions $\delta_{x,y}$, whether $\delta_{x,y}(\mathcal{F}(T_x))$ is equal to $\mathcal{F}(T_y)$. If this is the case for some x, y, and $\delta_{x,y}$ as above it outputs x, y and $\delta_{x,y}$. Because of Lemma 7 and due to the cardinality of Z', there must exist $x, y \in Z'$ such that $x \sim y$. In particular, there must exist a renaming function $\delta_{x,y}$ such that $\delta_{x,y}(\mathcal{F}(T_x)) = \mathcal{F}(T_y)$. But then the above algorithm is guaranteed to find such $x, y, \delta_{x,y}$ since it performs an exhaustive search. ☐

Combining Lemma 6 and Lemma 8, we arrive at the following corollary.

Corollary 9. *If $|Z| > \#\mathrm{C}(\ell,k,i,m)+1$, then in time $\mathcal{O}(\#\mathrm{C}(\ell,k,i,m)^2 \cdot m!m)$ one can compute a subinstance $I' = (\mathcal{F}', \eta)$ of I with strictly less variables and the following property: there exists a solution α of I of value $w = \eta(\alpha)$ if and only if there exists a solution α' of I' of value w. Moreover, a solution α can be computed from any solution α' in linear time.*

Proof. In order to avoid having to consider all children of z, the algorithm first computes (an arbitrary) subset Z' of Z such that $|Z'| = \#\mathrm{C}(\ell,k,i,m)+2$. Then to be able to apply Lemma 8 without changing the set of solutions of I, the algorithm computes a subset Z'' of Z' such that $|Z''| = \#\mathrm{C}(\ell,k,i,m)+1$ and for every $z' \in Z''$ it holds that $s_{z''} = 0$ for every $z'' \in T_{z'}$. Note that since there are at most k variables of I with non-zero coefficients in η and these variables form a clique in G_I, all of them occur only in a single branch of T_z. It follows that Z'' as specified above exists and it can be obtained from Z' by removing the (at most one) element z' in Z' with $s_{z''} \neq 0$ for some $z'' \in T_{z'}$. Observe that this step of the algorithm takes time at most $\mathcal{O}(m \cdot (\#\mathrm{C}(\ell,k,i,m)+1))$.

The algorithm then proceeds as follows. It uses Lemma 8 to find two variables $x, y \in Z''$ such that $x \sim y$ and computes I' from I by omitting T_y from I. The running time of the algorithm follows from Lemma 8 since the running times of the other steps of the algorithm are dominated by the application of Lemma 8. Correctness and the claimed properties of I' follow from Lemma 8 and Lemma 6. ☐

Let $\tilde{s}_i$ and $\tilde{c}_i$ for every i with $1 \leq i \leq k$ be defined inductively by setting $\tilde{s}_k = 1$, $\tilde{c}_k = 0$, $\tilde{c}_i = \#\mathrm{C}(\ell,k,i,s_{i+1})+1$, and $\tilde{s}_i = \tilde{c}_i\tilde{s}_{i+1}+1$. The following Lemma shows that in time $\mathcal{O}(|I|\tilde{c}_1^2 \cdot \tilde{s}_1!\tilde{s}_1)$ one can compute an "equivalent" subinstance I' of I containing at most $\tilde{s}_1$ variables. Informally, $\tilde{s}_i$ is an upper bound on the number of nodes in a subtree rooted at depth i and $\tilde{c}_i$ is an upper bound on the number of children of a node at level i in I'.

Lemma 10. *There exists an algorithm that takes as input I and T, runs in time $\mathcal{O}(|I|\tilde{c}_1^2 \cdot \tilde{s}_1!\tilde{s}_1)$ and outputs an ILP instance I' containing at most $\tilde{s}_1$ variables with the following property: there exists a solution α of I of value $w = \eta(\alpha)$ if and only if there exists a solution α' of I' of value $w = \eta'(\alpha')$. Moreover, a solution α can be computed from any solution α' in linear time.*

Proof. The algorithm exhaustively applies Corollary 9 to every variable of T in a bottom-up manner, i.e., it starts by applying the corollary exhaustively to all variables at depth $k-1$ and then proceeds up the levels of T until it reaches depth 1. Let T' be the subtree of T obtained after the exhaustive application of Corollary 9 to T.

We will first show that if x is a variable at depth i of T', then x has at most $\tilde{c}_i$ children and $|T'_x| \leq \tilde{s}_i$. We will show the claim by induction on the depth i starting from depth k. Because all variables x of T at level k are leaves, it holds that x has $0 = \tilde{c}_k$ children in T' and $|T'_x| = 1 \leq \tilde{s}_k$, showing the start of the induction. Now let x be a variable at depth i of T' and let y be a child of x in T'. It follows from the induction hypothesis that $|T'_y| \leq \tilde{s}_{i+1}$. Moreover, using Corollary 9, we obtain that x has at most $\tilde{\#\mathrm{C}}(\ell,k,i,s_{i+1})+1 = \tilde{c}_i$ children in T' and thus $|T'_x| \leq \tilde{c}_i\tilde{s}_{i+1}+1 = \tilde{s}_i$, as required.

The running time of the algorithm now follows from the observation that (because every application of Corollary 9 removes at least one variable of I) Corollary 9 is applied at most $|I|$ times and moreover the maximum running time of any call to Corollary 9 is at most $\mathcal{O}(\tilde{c}_1^2 \cdot \tilde{s}_1!\tilde{s}_1)$. Correctness and the fact that α can be computed from α' follow from Corollary 9. □

Proof of Theorem 5. The algorithm proceeds in three steps. First, it applies Lemma 10 to reduce the instance I into an "equivalent" instance I' containing at most $\tilde{s}_1$ variables in time $\mathcal{O}(|I|\tilde{c}_1^2 \cdot \tilde{s}_1!\tilde{s}_1)$; in particular, a solution α of I can be computed in linear time from a solution α' of I'. Second, it uses Theorem 1 to compute a solution α' of I' in time at most $\mathcal{O}(\tilde{s}_1^{2.5\tilde{s}_1+o(\tilde{s}_1)} \cdot |I'|)$; because $\tilde{s}_1$ and $\tilde{c}_1$ are bounded by our parameters, the whole algorithm runs in FPT time. Third, it transforms the solution α' into a solution α of I. Correctness follows from Lemma 10 and Theorem 1. □

Lower Bounds and Hardness

In this section we will complete the complexity landscape by providing the required hardness results. Namely, we will show that already the ILP-FEASIBILITY problem is W[1]-hard parameterized by treedepth alone and paraNP-hard parameterized by both treewidth and the maximum absolute value of any coefficient.

It is well-known that already ILP-FEASIBILITY remains NP-hard even if the maximum absolute value of any coefficient is at most one (as follows, e.g., from the standard reduction from VERTEX COVER to ILP-FEASIBILITY).

Observation 1. ILP-feasibility is paraNP-hard parameterized by the maximum absolute value of any coefficient.

To simplify the constructions in the hardness proofs, we will often talk about constrains as equalities instead of inequalities. Clearly, every equality can be written in terms of two inequalities.

Theorem 11. ILP-FEASIBILITY *is strongly* W[1]*-hard parameterized by treedepth.*

Proof Sketch. We will show 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. We construct an instance I of ILP-FEASIBILITY in polynomial-time with treedepth at most $k+3$ and coefficients bounded by $\mathcal{O}(n)$ such that G has a k-clique if and only if I has a feasible assignment.

The main idea of the construction is to represent the choice of the vertices in a k-clique by the k variables $v_1, \dots, v_k$ (each with domain $\{0, \dots, n-1\}$) and to employ additional auxiliary variables and constraints that ensure that for every i and j with $1 \leq i < j \leq k$, the variables v_i and v_j cannot be assigned to the endpoints of a non-edge of G simultaneously.

I has the following variables:

- For every i with $1 \leq i \leq k$ one variable v_i with domain $\{0, \dots, n-1\}$.
- For every non-edge $e = \{v_i^l, v_j^r\} \in E(\bar{G})$, where $1 \leq i < j \leq k$ and $0 \leq l, r \leq n-1$, the variables u_1^e, u_2^e with domain $\{0, \dots, n-1\}$ and the variables v_1^e, v_2^e with domain $\{0, 1\}$.

I has the following constrains:

C1 The constrains that bound the domains of all the variables as specified above. More formally, for every i with $1 \leq i \leq k$ the constrains $0 \leq v_i$ and $v_i \leq n-1$, for every non-edge $e \in E(\bar{G})$ and every $i \in \{0,1\}$ the constrains $0 \leq u_i^e$, $u_i^e \leq n-1$, $0 \leq v_i^e$, and $v_i^e \leq 1$.

C2 For every non-edge $e = \{v_i^l, v_j^r\} \in E(\bar{G})$, where $1 \leq i < j \leq k$ and $0 \leq l, r \leq n-1$, the constraints $v_i = l + u_1^e - nv_1^e$, $v_j = r + u_2^e - nv_2^e$, and $u_1^e + u_2^e \geq 1$. Note that these constrains ensure that the partial assignment that sets v_i to l and v_j to r cannot be extended to a feasible assignment of I. The form of these constraints is inspired by a similar construction given by Jansen and Kratsch (2015).

This completes the construction of I. Clearly, I can be constructed from G and k in polynomial-time and $\ell(I) \leq n$.

We show next that G has a k-clique if and only if I has a feasible assignment. Towards the forward direction, suppose that G has k-clique, say defined on the vertices $v_1^{c_1}, \dots, v_k^{c_k}$ for some $c_1, \dots, c_k$ with $0 \leq c_1, \dots, c_k \leq n-1$. Let α be the assignment defined by $\alpha(v_i) = c_i$ for every i with $1 \leq i \leq k$, and for every non-edge $e = \{v_i^l, v_j^r\} \in E(\bar{G})$, where $1 \leq i < j \leq k$ and $0 \leq l, r \leq n-1$, we set:

- $\alpha(v_1^e) = 0$ and $\alpha(u_1^e) = c_i - l$, if $c_i - l \geq 0$. Otherwise, i.e., if $c_i - l < 0$, we set $\alpha(v_1^e) = 1$ and $\alpha(u_1^e) = c_i - l + n$.
- $\alpha(v_2^e) = 0$ and $\alpha(u_2^e) = c_j - r$, if $c_j - r \geq 0$. Otherwise, i.e., if $c_j - r < 0$, we set $\alpha(v_2^e) = 1$ and $\alpha(u_2^e) = c_j - r + n$.

We claim that α is a feasible assignment of I. It is straightforward to check that α satisfies all the domain restricting constrains given in C1. Hence, let $e = \{v_i^l, v_j^r\} \in E(\bar{G})$ with $1 \leq i < j \leq k$ and $0 \leq l, r \leq n-1$ be a non-edge of G. Then, the constrains $v_i = l + u_1^e - nv_1^e$ and $v_j = r + u_2^e - nv_2^e$ (defined in C2) are satisfied due to the definition of α. Moreover, because $v_1^{c_1}, \dots, v_k^{c_k}$ are the vertices of a k-clique of G, we obtain that either $c_i \neq l$ or $c_j \neq r$. It follows that $u_1^e + u_2^e \geq 1$. This shows that α satisfies all constrains of I and completes the proof of the forward direction.

For the reverse direction, suppose that I has a feasible assignment, say α. Because of the domain restrictions of the variables $v_1, \dots, v_k$, enforced by the constraints defined in C1, we obtain that $v_i^{\alpha(v_i)}$ corresponds to a vertex of G for every i with $1 \leq i \leq k$. Hence, $\{v_1^{\alpha(v_1)}, \dots, v_k^{\alpha(v_k)}\}$ is a set of k vertices of G, we claim that it is also a k-clique of G. Suppose for a contradiction that this is not the case, i.e., there are i and j with $1 \leq i < j \leq k$ such that $\{v_i^{\alpha(v_i)}, v_j^{\alpha(v_j)}\} \in E(\bar{G})$. It follows that I contains the constrains $v_i = \alpha(v_i) + u_1^e - nv_1^e$, $v_j = \alpha(v_j) + u_2^e - nv_2^e$ and $u_1^e + u_2^e \geq 1$, where $e = \{v_i^{\alpha(v_i)}, v_j^{\alpha(v_j)}\}$. Because $v_i = \alpha(v_i) + u_1^e - nv_1^e$, we obtain that $u_1^e - nv_1^e = 0$

and because the domain of u_1^e is $\{0,\ldots,n-1\}$, we obtain that $\alpha(v_1^e) = 0$ and $\alpha(u_1^e) = 0$. Similarly, because $v_j = \alpha(v_j) + u_2^e - nv_2^e$, we obtain that $u_2^e - nv_2^e = 0$ and because the domain of u_2^e is $\{0,\ldots,n-1\}$, we obtain that $\alpha(v_2^e) = 0$ and $\alpha(u_2^e) = 0$. But this contradicts the constraint $u_1^e + u_2^e \geq 1$ and hence also our assumption that α is a feasible assignment of I.

It remains to show that the treedepth of I is at most $k+3$, the proof of which can be found in the full version of the paper. □

The next theorem shows that ILP-FEASIBILITY is paraNP-hard parameterized by both treewidth and the maximum absolute value of any coefficient.

Theorem 12. ILP-FEASIBILITY *is* NP*-hard even on instances with treewidth at most three and where the maximum absolute value of any coefficient is at most two.*

Proof Sketch. 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 give the reduction in two steps: The first step consists of the reduction given in Theorem 1, Jansen and Kratsch (2015), which shows that ILP-FEASIBILITY is weakly NP-hard even on instances of treewidth at most two and results in an instance I' of ILP-FEASIBILITY of treewidth at most two such that I is a YES instance of SUBSET SUM if and only if I' is a YES instance of ILP-FEASIBILITY. The second step shows how to replace the possibly large coefficients in I' by auxiliary variables and constraints without increasing the treewidth by more than one and results in the final instance I'' of ILP-FEASIBILITY with treewidth at most three and $\ell(I'') \leq 2$ such that I'' is equivalent to I'.

We start by constructing the instance I'. The variables of I' are $\{\, x_i, y_i : 1 \leq i \leq n \,\}$. Moreover, I' has the following constraints: for every i with $1 \leq i \leq n$, the constraints $0 \leq x_i$ and $x_i \leq 1$ and the constraints (given as equalities) $y_1 = s_1x_1$, $y_{i+1} = s_{i+1}x_{i+1} + y_i$ for every i with $1 \leq i < n$, and $y_n = s$. It is straightforward to verify (and has also been shown by Jansen and Kratsch (2015)) that I' is equivalent to I and has treewidth at most two. Observe that because SUBSET SUM is only weakly NP-hard, we cannot assume that the integers in S and therefore the coefficients of I' are bounded (not even by a polynomial in the input size of I). This is why we need the second step to complete the reduction.

We now show how to obtain I'' from I'. First, we replace the constraint $y_1 = s_1x_1$ with the constraint $y_1 = z_1$ and for every i with $1 \leq i < n$, we replace the constraint $y_{i+1} = s_{i+1}x_{i+1} + y_i$ with the constraint $y_{i+1} = z_{i+1} + y_i$, where for every i with $1 \leq i \leq n$, z_i is a new auxiliary variable such that $\alpha(z_i)$ is equal to $s_i\alpha(x_i)$ for every feasible assignment α of I''. The following two paragraphs describe the auxiliary variables and constraints used to ensure that for i with $1 \leq i \leq n$, $\alpha(z_i)$ is equal to $s_i\alpha(x_i)$ for every feasible assignment α of I''.

For an integer o, let $B(o)$ be the set of indices of all bits that are equal to one in the binary representation of o, i.e., we have $o = \sum_{j \in B(o)} 2^j$. Moreover, let $b_{\max}(o)$ be the largest index in $B(o)$.

We introduce the new auxiliary variables $b_{i,0},\ldots,b_{i,b_{\max}(s_i)}$ and the constraint $b_{i,0} = x_i$ and for every p with $0 \leq p < b_{\max}(s_i)$ the constraint $b_{i,p+1} = 2 \cdot b_{i,p}$. Observe that the above constraints ensure that $\alpha(b_{i,j})$ is equal to $2^j\alpha(x_i)$ for every j with $0 \leq j \leq b_{\max}(s_i)$ and every feasible assignment α of I''. We also introduce the new auxiliary variables $z_{i,0},\ldots,z_{i,b_{\max}(s_i)}$ together with the following constraints: if $0 \in B(s_i)$ then we add the constraint $z_{i,0} = b_{i,0}$, and otherwise we add the constraint $z_{i,0} = 0$. Furthermore, for every p with $0 \leq p < b_{\max}(s_i)$, if $p+1 \in B(s_i)$ then the constraint $z_{i,p+1} = b_{i,p+1} + z_{i,p}$ and otherwise the constraint $z_{i,p+1} = z_{i,p}$. Observe that these constraints ensure that $\alpha(z_{i,j})$ is equal to $\sum_{p \in B(s_i)} \alpha(b_{i,p})$ for every j with $0 \leq j \leq b_{\max}(s_i)$ and any feasible assignment α of I''. Finally, we add the constraint $z_i = z_{i,b_{\max}(s_i)}$. Observe that now $\alpha(z_i)$ is equal to $s_i\alpha(x_i)$ for any feasible assignment α of I'', as required.

Finally, we replace the constraint $y_n = s$ with the constraint $y_n = z$, where z is a new auxiliary variable such that $\alpha(z)$ is equal to s for every feasible assignment α of I''. The auxiliary variables and constraints used to ensure that $\alpha(z)$ is equal to s for every feasible assignment α of I'', are very similar to the ones used to ensure that $\alpha(z_i)$ is equal to $s_i\alpha(x_i)$ and are described in detail only in the full version of this paper.

This completes the construction of I''. Because $b_{\max}(s)$ is at most $\lceil \log s \rceil$ and for every i with $1 \leq i \leq n$, $b_{\max}(s_i)$ is at most $\lceil \log s_i \rceil$ and hence polynomial in the input size of I, the construction of I'' from I' and hence also from I can be done in time polynomial in the input size of I. Moreover, I'' is equivalent to I' and $\ell(I'') \leq 2$. It remains to show that the treewidth of I' is at most three, the proof of which can be found in the full version of this paper. □

Concluding Notes

We presented the first results exploring the tractability of ILP w.r.t. structural parameterizations of the constraint matrix. Our main algorithmic result significantly pushes the frontiers of tractability for ILP instances and will hopefully serve as a precursor for the study of further structural parameterizations for ILP.

The provided results draw a detailed complexity landscape for ILP w.r.t. the most prominent decompositional width parameters. However, other approaches exploiting the structural properties of ILP instances still remain unexplored and represent interesting directions for future research. For instance, an adaptation of *backdoors* (Gaspers and Szeider 2012) to the ILP setting could lead to highly relevant algorithmic results.

Acknowledgments The authors wish to thank the anonymous reviewers for their helpful comments. The authors acknowledge support by the Austrian Science Fund (FWF, project P26696). Robert Ganian is also affiliated with FI MU, Brno, Czech Republic.

References

Alumur, S. A., and Kara, B. Y. 2008. Network hub location problems: The state of the art. *European Journal of Operational Research* 190(1):1–21.

Cunningham, W. H., and Geelen, J. 2007. On integer programming and the branch-width of the constraint matrix. In Fischetti, M., and Williamson, D. P., eds., *Integer Programming and Combinatorial Optimization, 12th International IPCO Conference, Ithaca, NY, USA, June 25-27, 2007, Proceedings*, volume 4513 of *Lecture Notes in Computer Science*, 158–166. Springer.

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

Downey, R. G., and Fellows, M. R. 1999. *Parameterized Complexity*. Springer.

Downey, R. G., and Fellows, M. R. 2013. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer.

Fellows, M. R.; Lokshtanov, D.; Misra, N.; Rosamond, F. A.; and Saurabh, S. 2008. Graph layout problems parameterized by vertex cover. In *ISAAC*, Lecture Notes in Computer Science, 294–305. Springer.

Fischer, E.; Makowsky, J. A.; and Ravve, E. R. 2008. Counting truth assignments of formulas of bounded tree-width or clique-width. *Discr. Appl. Math.* 156(4):511–529.

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

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*. Springer.

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

Ganian, R.; Hliněný, P.; and Obdržálek, J. 2010. Better algorithms for satisfiability problems for formulas of bounded rank-width. Technical Report arXiv:1006.5621v1, CoRR.

Ganian, R.; Kim, E. J.; and Szeider, S. 2015. Algorithmic applications of tree-cut width. In *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part II*, 348–360.

Gaspers, S., and Szeider, S. 2012. Backdoors to satisfaction. In Bodlaender, H. L.; Downey, R.; Fomin, F. V.; and Marx, D., eds., *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*, 287–317. Springer Verlag.

Gutin, G.; Jones, M.; and Wahlström, M. 2015. Structural parameterizations of the mixed chinese postman problem. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, 668–679.

Jansen, B. M. P., and Kratsch, S. 2015. A structural approach to kernels for ilps: Treewidth and total unimodularity. In Bansal, N., and Finocchi, I., eds., *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, 779–791. Springer.

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

Lenstra, H. W., and Jr. 1983. Integer programming with a fixed number of variables. *MATH. OPER. RES* 8(4):538–548.

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

Nešetřil, J., and Ossona de Mendez, P. 2012. *Sparsity: Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and Combinatorics*. Springer.

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

Ordyniak, S.; Paulusma, D.; and Szeider, S. 2013. Satisfiability of acyclic and almost acyclic cnf formulas. *Theoret. Comput. Sci.* 481:85–99.

Papadimitriou, C. H., and Steiglitz, K. 1982. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall.

Papadimitriou, C. H. 1981. On the complexity of integer programming. *J. ACM* 28(4):765–768.

Pietrzak, K. 2003. 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.

Szeider, S. 2003. Finding paths in graphs avoiding forbidden transitions. *Discr. Appl. Math.* 126(2-3):239–251.

Toth, P., and Vigo, D., eds. 2001. *The Vehicle Routing Problem*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

van den Briel, M.; Vossen, T.; and Kambhampati, S. 2005. Reviving integer programming approaches for AI planning: A branch-and-cut framework. In Biundo, S.; Myers, K. L.; and Rajan, K., eds., *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005), June 5-10 2005, Monterey, California, USA*, 310–319. AAAI.

Vossen, T.; Ball, M. O.; Lotem, A.; and Nau, D. S. 1999. On the use of integer programming models in AI planning. In Dean, T., ed., *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, 304–309. Morgan Kaufmann.