Designing Compact ILPs via Fast Witness Verification

Michał Włodarczyk ✉ ⓘ
University of Warsaw, Poland

Abstract

The standard formalization of preprocessing in parameterized complexity is given by kernelization. In this work, we depart from this paradigm and study a different type of preprocessing for problems without polynomial kernels, still aiming at producing instances that are easily solvable in practice. Specifically, we ask for which parameterized problems an instance (I, k) can be reduced in polynomial time to an integer linear program (ILP) with $\mathsf{poly}(k)$ constraints.

We show that this property coincides with the parameterized complexity class WK[1], previously studied in the context of Turing kernelization lower bounds. In turn, the class WK[1] enjoys an elegant characterization in terms of witness verification protocols: a yes-instance should admit a witness of size $\mathsf{poly}(k)$ that can be verified in time $\mathsf{poly}(k)$. By combining known data structures with new ideas, we design such protocols for several problems, such as r-WAY CUT, VERTEX MULTIWAY CUT, STEINER TREE, and MINIMUM COMMON STRING PARTITION, thus showing that they can be modeled by compact ILPs. We also present explicit ILP and MILP formulations for WEIGHTED VERTEX COVER on graphs with small (unweighted) vertex cover number. We believe that these results will provide a background for a systematic study of ILP-oriented preprocessing procedures for parameterized problems.

1 Introduction

Kernelization [38] provides a rigorous theory for data preprocessing in the following sense: given an instance I with parameter k we would like to transform it in polynomial time into an equivalent instance (I', k') so that $|I'| + k' \le f(k)$ for a slowly growing function f. Historically, kernelization was meant as just another tool for establishing fixed-parameterized tractability [25] but later it grew into a mature theory on its own, especially after a hardness framework has been established [6]. The main question in this paradigm is: which parameterized problems admit polynomial kernels, that is, when can the function f be estimated as polynomial? From the practical point of view, once the instance size is reduced, it may become amenable to brute-force or heuristic methods.

In recent years, empirical observations have provided evidence that analyzing preprocessing through the lens of kernelization might be too restrictive. In the annual PACE challenge [29, 43, 53, 54], co-organized with the IPEC conference, competitors solve a collection of instances of a certain NP-hard problem related to parameterized complexity. It turned out that the many successful submissions [3, 4, 9, 23, 24] do not follow the algorithms from FPT handbooks but instead reduce the problem to (mixed) integer linear programming (ILP) and then employ a specialized solver. Even though the contestants are not allowed to use

the state-of-the-art industrial solvers, already the open source solvers often outperform sophisticated FPT algorithms with provable worst-case guarantees. However, it is usually challenging to model the problem in question as a concise ILP.

In our opinion, parameterized complexity should seek an explanation of this phenomenon and understand which problems can be handled with ILP-driven preprocessing. In this work, we make a first step in this direction and investigate one of the possible ways of formalizing such a notion of preprocessing. Even though modeling NP-hard problems as succinct (mixed) ILPs has been a subject of operation research since the 1950s [14, 22, 37, 42, 65, 66, 71], no systematical study has addressed this question from the perspective of parameterized complexity so far.

What kind of preprocessing?   Following the design of kernelization theory, we consider polynomial-time procedures that output an equivalent instance (of a possibly different decision problem) that should be well-structured as long as the initial parameter is not too large. While there are many ways to measure the difficulty of an ILP [1, 11, 17, 18, 28, 30, 40, 48, 36, 50], the most natural are the number of constraints and the number of variables. In this work, we focus on the first one.

Specifically, we are interested in processing an instance (I, k) in polynomial time to rewrite it as an ILP of the form $(Ax \leq b, x \in \mathbb{Z}_{\geq 0}^n)$, where $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$, and m is polynomially bounded in k. In other words, we would like to reduce solving the problem in question to feasibility check of an ILP with only $\mathsf{poly}(k)$ constraints. This strategy matches the assumption guiding the kernelization paradigm: when the parameter k is small, then heuristic algorithms should work fast in practice when the search space has "dimension"[1] $\mathsf{poly}(k)$. Naturally, it would be desirable to obtain ILP formulations with $m = \mathcal{O}(k)$ or $m = \mathcal{O}(k^2)$ but we believe that a new framework should first focus on establishing general bounds and more fine-grained results will follow. For comparison, the first work on polynomial kernelization for FEEDBACK VERTEX SET only guaranteed a kernel of size $\mathcal{O}(k^{11})$ [13] but it was soon improved to $\mathcal{O}(k^2)$ [69].

What kind of problems?   Before we delve further into technicalities, let us reflect on for which parameterized problems the question above makes sense. First of all, we are interested in problems without (known) polynomial kernels. Secondly, any problem that can be reduced as described above must (1) belong to NP and (2) admit an FPT algorithm with running time of the form $2^{\mathsf{poly}(k)} \cdot n^{\mathcal{O}(1)}$ [51]. Restricting to only such problems allows us to assume $\log n \leq \mathsf{poly}(k)$ because otherwise the FPT running time above can be estimated as polynomial in n. Later we will also see another property that must be satisfied: an instance (I, k) should admit an NP-witness of size $\mathsf{poly}(k, \log |I|)$. Therefore, we should not hope to handle structural parameterizations like treewidth due to known barriers [27].

Fast witness verification.   The question outlined above can be formalized in the following way: we ask which parameterized problems admit a *polynomial parameter transformation* (PPT) to ILP feasibility checking parameterized by the number of constraints. It is easy to see that SET COVER parameterized by the universe size admits such a reduction and we will present a reduction in the opposite direction as well. In turn, the class of problems reducible

---

[1]   Usually the "dimension" of an ILP refers to the number of variables but the known FPT algorithms [32, 51] for solving ILP parameterized by the number of constraints m perform computations in a lattice of dimension m.

by PPT to Set Cover is known as WK[1]. The WK-hierarchy has been introduced by Hermelin, Kratsch, Sołtys, Wahlström, and Wu [45] in the context of Turing kernelization. What is interesting for us, the class WK[1] comprises those parameterized problems that admit an NP-witness of size $\mathsf{poly}(k)$ whose correctness can be verified in time $\mathsf{poly}(k)$. This elegant characterization has remained merely a curiosity because the main focus of the work [45] was to prove problem hardness with respect to the WK-hierarchy[2]. We shall exploit this characterization in a different context.

Related work. A more expressive variant of ILP allows one to write lower and upper bounds $\ell_i \le x_i \le u_i$ that do not count to the constraint limit (see Section 2). Rohwedder and Węgrzycki [68] considered such ILP parameterized by the number of constraints and studied fine-grained equivalences with problems such as Closest String, Discrepancy Minimization, and Set Multi-Cover. Kernelization of ILP Feasibility has been considered in [56, 59]. For more applications of ILP in parameterized complexity, see, e.g., [15, 41, 44, 55].

Apart from witness verification protocols studied here, there are other models of non-deterministic computation considered in parameterized complexity, mainly in the context of space limitations [7, 33, 67, 72].

Our contribution

We show that the class WK[1] provides an exact characterization of parameterized problems that can be reduced to ILP with few constraints as long as the entries in the corresponding matrix cannot be too large. The latter restriction is necessary because unbounded entries allow one to express Unbounded Subset Sum (which is NP-hard [62]) with just one equality constraint. More formally, we consider the feasibility problem of the form $(Ax \le b,\, x \in \mathbb{Z}_{\ge 0}^n)$, where $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$, and define $\Delta(A)$ to be the maximal absolute value of an entry in the matrix A.

▶ **Theorem 1.** Integer Linear Programming Feasibility *is* WK[1]*-complete when:*
- *parameterized by* $m + \log(\Delta(A))$,
- *parameterized by* m *only, under the assumption that* $\Delta(A) = 1$ *and the vector* b *is given in unary.*

Consequently, a parameterized problem admits a polynomial parameter transformation to ILP Feasibility (under the considered parameterizations) if and only if it belongs to the class WK[1]. Notably, it follows that allowing for polynomially large numbers in an ILP does not lead to any qualitative advantage. A similar equivalence is known for the variant with bounded variables [68] but therein the reduction involves exponentially many variables. That variant is easily seen to be WK[1]-hard (Observation 13).

Hermelin et al. [45] already showed that WK[1] includes problems of local nature, such as k-Path, Connected Vertex Cover, Min Ones d-SAT, or Steiner Tree, all parameterized by the solution size. Motivated by Theorem 1, we show that this class captures many more problems, also those of seemingly non-local nature. To this end, we employ various data structures to design fast witness verification protocols. The definitions of the listed problems and the related results are discussed in the corresponding sections.

---

[2] This characterization is implicit in [45] so for completeness we provide a proof in the full version of the article.

▶ **Theorem 2.** *The following parameterized problems belong to the class* WK[1]*:*
- *r*-WAY CUT / *solution size,*
- VERTEX MULTIWAY CUT / *solution size,*
- MINIMUM COMMON STRING PARTITION / *partition size,*
- LONG PATH / *feedback vertex set number,*
- STEINER TREE / *number of terminals,*
- OPTIMAL DISCRETIZATION / *solution size.*

Theorems 1 and 2 imply that all the listed problems can be reduced to ILP with $\mathsf{poly}(k)$ constraints. In fact, we provide parameterized reductions to BINARY NDTM HALTING, which then reduces to SET COVER, which in turn can be readily modeled by a compact ILP. We remark that the proofs of containment in WK[1] are not technically involved but they exploit dynamic data structures in a novel context. In our opinion, their value lies in creating a new link between the fields of integer programming, proof protocols, and data structures.

The results above only provide implicit reductions to ILP where the number constraints is an unspecified polynomial in the parameter. We complement those with an explicit reduction for WEIGHTED VERTEX COVER / VC, where the parameter is the vertex cover number of the input graph. We attain only linearly many constraints, however with a caveat that we require all the variables to be binary. In addition, we present a mixed integer linear program where the number of integral variables is also linear in the parameter.

▶ **Theorem 3.** WEIGHTED VERTEX COVER / VC *admits a linear parameter transformation to* BINARY ILP FEASIBILITY *parameterized by* $m + \log(\Delta(A))$*. Furthermore, the problem admits a linear parameter transformation to* MILP FEASIBILITY *parametrized by the number of integral variables.*

Organization of the paper. In Section 2 we provide background on ILPs and MILPs, together with a discussion on the class WK[1]. Section 3 is devoted to proving Theorem 1 while the proof of Theorem 2 is given in Section 4. In Section 5 we give a closer look at WEIGHTED VERTEX COVER.

2   Preliminaries

On default, we measure the running time of algorithms in the standard RAM model with logarithmic word length. All the summoned algorithmic results are deterministic. Throughout the paper, we use the variable n to denote the input size during the reductions or the number of variables when analyzing integer programs. The number of constraints is denoted by m.

Integer linear programming. For $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$, we consider ILP in the *standard form without upper bounds*: $(Ax \leq b, \ x \in \mathbb{Z}_{\geq 0}^n)$. Note that it is allowed to write inequalities "$\geq$" as well since one can multiply coefficients by -1. The problem INTEGER LINEAR PROGRAMMING (ILP) FEASIBILITY asks, given A, b, whether there exists a vector x satisfying the system above. One can also consider ILP in the *equality form*: $(Ax = b, \ x \in \mathbb{Z}_{\geq 0}^n)$ where $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$. It is easy to transform a system of inequalities $(Ax \leq b, \ x \in \mathbb{Z}_{\geq 0}^n)$ into the equality form by introducing m new non-negative variables and turning each inequality $\sum a_i x_i \leq b$ into equality $\sum a_i x_i + z = b$ where z is a fresh variable. This transformation does not affect the number of constraints and results in matrix A' obtained from A by inserting m columns with only 0 and 1 entries. We make note of this observation in the following terms. Recall that $\Delta(A)$ is the maximal absolute value of an entry in the matrix A.

▶ **Observation 4.** *A system of inequalities* $(Ax \leq b, x \in \mathbb{Z}_{\geq 0}^n)$, $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$, *can be transformed in polynomial time into an equivalent system of equalities* $(A'y = b, y \in \mathbb{Z}_{\geq 0}^{n+m})$ *where* $A' \in \mathbb{Z}^{m \times (n+m)}$ *and* $\Delta(A') = \max(1, \Delta(A))$.

Another common variant of ILP is the *standard form with upper bounds*: $(Ax \leq b, x \leq u, x \in \mathbb{Z}_{\geq 0}^n)$ where $u \in \mathbb{Z}_{\geq 0}^n$. This form effectively expresses n additional inequalities which are not counted as constraints. When u is the all-1s-vector, we call the corresponding decision problem BINARY ILP FEASIBILITY. All variants admit FPT algorithms parameterized by the number of constraints m and $\Delta(A)$. On the other hand, parameterization by m alone is unlikely to be FPT even when the input is given in unary [39, Theorem 2].

▶ **Proposition 5** ([51, Theorem 5]). *Feasibility of an ILP in the standard form* **without** *upper bounds can be decided in time* $\mathcal{O}(m \cdot \Delta(A))^{2m} \cdot \log(||b||_\infty)$.

▶ **Proposition 6** ([32, Theorem 4.1]). *Feasibility of an ILP in the standard form* **with** *upper bounds can be decided in time* $(m \cdot \Delta(A))^{\mathcal{O}(m^2)} \cdot n + s^{\mathcal{O}(1)}$ *where* s *is the input encoding size.*

In *mixed integer linear programming* (MILP) we have two groups of variables: n integral ones and ℓ fractional ones. Let $x[a, b]$ denote the truncation of the vector x to coordinates $[a, b]$. In a variant with upper bounds we look for a vector x satisfying the system $(Ax \leq b, x \leq u, x[1, n] \in \mathbb{Z}_{\geq 0}^n, x[n+1, n+\ell] \in \mathbb{R}_{\geq 0}^\ell)$ where $A \in \mathbb{Z}^{m \times (n+\ell)}$, $u \in \mathbb{Z}^{n+\ell}$, and $b \in \mathbb{Z}^m$. When the number n of integral variables is small, one can solve a MILP using an extension of Lenstra's algorithm. Note that in contrast to parameterization by m, now $\Delta(A)$ can be unbounded.

▶ **Proposition 7** ([52]). *Feasibility of a MILP with* n *integral variables can be determined in time* $n^{\mathcal{O}(n)} \cdot s^{\mathcal{O}(1)}$, *where* s *is the input encoding size.*

Class WK[1]. A *polynomial parameter transformation* (PPT) is a reduction between parameterized problems P, Q that runs in polynomial time and maps instance (I, k) of P to an equivalent instance (I', k') of Q satisfying $k' \leq \mathsf{poly}(k)$. The WK-hierarchy has been introduced as a hardness hierarchy for problems unlikely to admit a polynomial Turing kernel [45]. For each $t \geq 1$ the class WK[t] comprises problems that are PPT-reducible to WEIGHTED SAT over a certain class of formulas, parameterized by the number of 1s in an assignment times $\log n$. Since we are only interested in the class WK[1], it is more convenient to rely on its equivalent characterization as the PPT-closure of the following WK[1]-complete problem [45].

BINARY NDTM HALTING   **Parameter:** k
Input: nondeterministic Turing machine M over a binary alphabet, integer k
Question: can M halt on the empty string in k steps?

Another canonical WK[1]-complete problem is SET COVER.

SET COVER   **Parameter:** $|U|$
Input: set U, family of sets $\mathcal{F} \subseteq 2^U$, integer ℓ
Question: is there a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ of size ℓ such that the union of $\mathcal{F}'$ is U?

WK[1]-completeness of BINARY NDTM HALTING gives rise to a particularly elegant characterization of WK[1] in terms of witness verification protocols.

▶ **Definition 8.** *A* witness verification protocol *for a parameterized problem L is a pair of algorithms* $(\mathcal{A}, \mathcal{B})$ *with the following specification.*

Given an instance (I, k) *the algorithm* $\mathcal{A}$ *runs in polynomial time and outputs a string* $s(I, k) \in \{0, 1\}^*$ *and integer* $\ell(I, k)$*. The algorithm* $\mathcal{B}$ *is given the concatenation of* $s(I, k)$ *and a string* $w \in \{0, 1\}^{\ell(I,k)}$ *and outputs a yes/no answer. We require that* $(I, k) \in L$ *if and only if there exists* w *for which* $\mathcal{B}$ *answers "yes".*

For a function $f: \mathbb{N}^2 \to \mathbb{N}$ *we say that the protocol has complexity* f *if (a)* $\ell(I, k) \leq f(|I|, k)$ *and (b) the running time of* $\mathcal{B}$ *on* $s(I, k) + w$ *is bounded by* $f(|I|, k)$*.*

The following characterization was stated implicitly in [45, §1]. We provide a proof in the full version of the article for completeness.

▶ **Theorem 9.** *A parameterized problem L belongs to the class WK[1] if and only if:*
1. $(I, k) \in L$ *can be decided in time* $2^{\mathsf{poly}(k)} \cdot |I|^{\mathcal{O}(1)}$*, and*
2. *L admits a witness verification protocol of complexity* $\mathsf{poly}(k, \log |I|)$*.*

3   WK[1]-completeness of ILP

We summon several facts about the structure of solutions to ILP in the equality form. First we bound the support of a solution using the integral Carathéodory's Theorem.

▶ **Proposition 10** ([31, Corollary 5]). *Consider an optimization ILP of the form* $\min\{c^T x: Ax = b, x \in \mathbb{Z}_{\geq 0}^n\}$ *where* $A \in \mathbb{Z}^{m \times n}, b \in \mathbb{Z}^m, c \in \mathbb{Z}^n$*. If this program has a finite optimum, then it has an optimal solution* x^* *in which the number of non-zero entries is* $\mathcal{O}(m(\log m + \log(\Delta))$ *where* Δ *is the maximum absolute value of an entry in A and c.*

Next, we bound the ℓ_1-norm of a solution.

▶ **Proposition 11** ([51, Lemma 3]). *If the system* $(Ax = b, x \in \mathbb{Z}_{\geq 0}^n)$ *is feasible, it has a solution* x^* *satisfying* $||x^*||_1 \leq \mathcal{O}(m \cdot \Delta(A))^m \cdot (||b||_\infty + 1)$*.*

At the expense of solving an LP relaxation, we can assume that the entries in the vector b are bounded. The following proposition is a corollary from the proximity bound by Eisenbrand and Weismantel [32].

▶ **Proposition 12** ([51, §4.2]). *There is a polynomial-time algorithm that, given* $A \in \mathbb{Z}^{m \times n}$ *and* $b \in \mathbb{Z}^m$*, returns a vector* $b' \in \mathbb{Z}^m$ *such that:*
1. $||b'||_\infty = \mathcal{O}(m \cdot \Delta(A))^{m+1}$*,*
2. $(Ax = b, x \in \mathbb{Z}_{\geq 0}^n)$ *is feasible if and only if* $(Ax = b', x \in \mathbb{Z}_{\geq 0}^n)$ *is feasible.*

We are ready to pin down the complexity status of ILP FEASIBILITY.

▶ **Theorem 1.** INTEGER LINEAR PROGRAMMING FEASIBILITY *is* WK[1]*-complete when:*
- *parameterized by* $m + \log(\Delta(A))$*,*
- *parameterized by* m *only, under the assumption that* $\Delta(A) = 1$ *and the vector* b *is given in unary.*

Proof. We first show that the latter variant is WK[1]-hard. It admits a trivial PPT to the first variant, so then it remains to show that the first one belongs to WK[1].

We prove WK[1]-hardness by reducing from SET COVER parameterized by the universe size. This problem is known to be WK[1]-complete [45]. For each set $F \in \mathcal{F}$ we create a variable $x_F \in \mathbb{Z}_{\geq 0}$. For each $u \in U$ we write a constraint $\sum_{F \ni u} x_F \geq 1$, enforcing that u

must be covered. Then we add a constraint $\sum_{F \in \mathcal{F}} x_F \leq \ell$. It is easy to see that any integral solution corresponds to a vertex cover of size at most ℓ. This yields a polynomial parameter transformation: the number of constraints is $|U| + 1$, $\Delta(A) = 1$, and $||b||_\infty \leq |U|$.

Now we show that the first problem belongs to WK[1]. First we rewrite an ILP into the equality form (Observation 4). By Theorem 9 we have to check that the problem is solvable in time $2^{\mathsf{poly}(k)} \cdot s^{\mathcal{O}(1)}$ and that it admits a witness verification protocol with complexity $\mathsf{poly}(k, \log s)$, where s stands for the encoding size of the instance. The running time from Proposition 5 can be estimated as $2^{\mathcal{O}(k \log k)} \cdot s^{\mathcal{O}(1)}$ where $k = m + \Delta(A)$ so it suffices to design a witness verification protocol $(\mathcal{A}, \mathcal{B})$. The algorithm $\mathcal{A}$ preprocesses the ILP using Proposition 12 to ensure that $||b||_\infty = \mathcal{O}(m \cdot \Delta(A))^{m+1}$.

We apply Proposition 10 for the objective vector $c = 0$ so Δ equals $\Delta(A)$. We infer that if the ILP is feasible, then it admits a solution x^* with at most $\mathcal{O}(k^2)$ non-zero variables. By applying Proposition 12 to the ILP restricted to only these variables we can further assume that $||x^*||_1 \leq \mathcal{O}(m \cdot \Delta(A))^{2m+1} = 2^{\mathcal{O}(k^2 \log k)}$. Hence x^* can be encoded using $\mathcal{O}(k^2 \log s)$ bits for the choice of the non-zero variables and $\mathcal{O}(k^4 \log k)$ bits for their values. This gives an encoding of the witness supplied to the algorithm $\mathcal{B}$. Checking each constraint can be done in time polynomial with respect to the number of non-zero variables and the number of relevant bits, which is $\mathsf{poly}(k, \log s)$. This concludes the description of the witness verification protocol. ◀

By inspecting the reduction from Set Cover to ILP Feasibility, one can easily see that it also works with additional constraints $x_u \in \{0, 1\}$ for every $u \in U$. It follows that the variant with bounded variables is also WK[1]-hard, however its containment in WK[1] is not known and we find it unlikely (see Section 6).

▶ **Observation 13.** Binary ILP Feasibility *is* WK[1]*-hard when parameterized by* m, *under the assumption that* $\Delta(A) = 1$ *and the vector* b *is given in unary.*

4 Fast Witness Verification

We show containment in WK[1] for the problems listed in Theorem 2 using the criterion from Theorem 9. The first condition will be always satisfied by summoning a known FPT algorithm and we will focus on designing witness verification protocols.

4.1 Cut problems

In this section we treat r-Way Cut and Vertex Multiway Cut. We will employ connectivity oracles to quickly verify that a solution candidate forms a cut.

▶ **Definition 14.** *A* vertex (resp. edge) failure connectivity oracle *for a graph G and integer d supports two types of operations. An* update *operation is performed only once and it processes a set D of vertices (resp. edges) from G of size at most d. In a* query *operation, for given vertices $u, v \in G$ the oracle determines whether u, v are connected in $G - D$.*

▶ **Proposition 15** ([61]). *There is a deterministic vertex (resp. edge) failure connectivity oracle for undirected graphs with polynomial preprocessing time, update time $\mathcal{O}(d^2 \cdot \log^4 n \cdot \log^4 d)$, and worst-case query time $\mathcal{O}(d)$.*

The statement in [61] concerns vertex failure connectivity only but this setting generalizes edge failure connectivity. To see this, subdivide each edge in G and mimic removal of an edge by removing the corresponding vertex (see also [46]).

---

r-WAY CUT **Parameter:** k

Input: undirected graph G, integers r and k

Question: is there a set $X \subseteq E(G)$ such that $G - X$ has at least r connected components and $|X| \leq k$?

The vertex-deletion variant of the problem is W[1]-hard already under the parameterization by $k + r$ [63]. Edge deletion is easier and the above stated variant is FPT [19]. However, it does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly [20]. We will show that r-WAY CUT belongs to WK[1] which, in the light of Theorem 1, makes the problem amenable to ILP-based preprocessing.

▶ **Lemma 16.** r-WAY CUT $\in$ WK[1].

Proof. By Theorem 9 we need to check that the problem is solvable in time $2^{\mathsf{poly}(k)}n^{\mathcal{O}(1)}$ and that it admits a witness verification protocol with complexity $\mathsf{poly}(k, \log n)$. The first condition is satisfied due to known $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$-time algorithm [19].

Observe that removal of one edge can increase the number of connected component by at most one. Let $\mathcal{C}(G)$ denote the set of connected components of G and $\ell = |\mathcal{C}(G)|$. If $\ell + k < r$ then clearly there can be no solution, so let us assume $r - \ell \leq k$. Suppose that $X \subseteq E(G)$ is a solution. For every $C \in \mathcal{C}(G)$ that is being split into $r_C > 1$ parts, there exists vertices $v_1, v_2, \ldots, v_{r_C} \in V(C)$ that are pairwise disconnected in $G - X$. The number of connected component of $G - X$ equals $\sum_{C \in \mathcal{C}(G)} r_C$. Since it suffices to split $r - \ell \leq k$ components, we may assume that there at most k components $C \in \mathcal{C}(G)$ for which $r_C > 1$. Therefore, when X is a solution, there must exist $\leq 2k$ vertices that can be divided into $t \leq k$ groups $V_1, V_2, \ldots, V_t$ so that (i) each group V_i is pairwise connected in G, (ii) $v \in V_i$ and $u \in V_j$ are not connected in G whenever $i \neq j$, (iii) distinct $u, v \in V_i$ are not connected in $G - X$, and (iv) $\sum_{i=1}^{t}(|V_i| - 1) = r - \ell$. This constitutes a certificate that X is a solution.

We now design the witness verification protocol $(\mathcal{A}, \mathcal{B})$. The algorithm $\mathcal{A}$ computes the connected components of G, assigns vertices in each component a unique label, and initializes the data structure from Proposition 15. The algorithm $\mathcal{B}$ inherits the state of the data structure together with labels marking the connected components and is provided the following witness: a set X of at most k edges and $t \leq k$ vertex sets $V_1, V_2, \ldots, V_t$ with $\leq 2k$ vertices in total. First we check that all the given vertices are pairwise distinct, in time $\mathcal{O}(k^2)$. Then we check the conditions (i)-(iv). The conditions (i)-(ii) can be verified using labels in time $\mathcal{O}(k^2)$. To handle condition (iii), we first perform the update operation on the edge failure connectivity oracle for set X in time $\mathcal{O}(k^3 \cdot \log^4 n)$. Then we query the oracle for each pair $u, v \in V_i$ for each $i \in [1, t]$ to check that they are disconnected in $G - X$. This takes total time $\mathcal{O}(k^3)$. Finally, condition (iv) boils down to number additions, which are performed in time $\mathcal{O}(k)$. This concludes the description of the protocol. ◀

We move on to another cut-based problem.

VERTEX MULTIWAY CUT **Parameter:** k

Input: undirected graph G, set $T \subseteq V(G)$, integer k

Question: is there a set $X \subseteq V(G) \setminus T$ such that each connected component of $G - X$ has at most one vertex from T and $|X| \leq k$?

The existence of a polynomial kernel for VERTEX MULTIWAY CUT remains a major open problem [60]. Positive results include special cases where $|T| = \mathcal{O}(1)$ [60] or where the input graph is planar [49], and a quasi-polynomial kernel for the edge-deletion variant [70]. In addition, there exist reduction rules that compress the terminal set.

▶ **Proposition 17** ([21, Lemma 2.7]). *There is a polynomial-time algorithm that, given an instance (G, T, k) of* Vertex Multiway Cut*, outputs an equivalent instance (G', T', k') with $k' \leq k$ and $|T'| \leq 2k'$.*

Our witness verification protocol combines this procedure with the vertex failure connectivity oracle.

▶ **Lemma 18.** Vertex Multiway Cut $\in$ WK[1].

Proof. Vertex Multiway Cut can be solved in time $2^k \cdot n^{\mathcal{O}(1)}$ [21] and it remains to construct a witness verification protocol. The algorithm $\mathcal{A}$ executes the reduction from Proposition 17 and so we can assume $|T| \leq 2k$. Then it initializes the vertex failure connectivity oracle from Proposition 15. The algorithm $\mathcal{B}$ is given the reduced instance, the state of the oracle, and the witness as simply a solution candidate $X \subseteq V(G)$, $|X| \leq k$. First it checks if $X \cap T = \emptyset$ in time $\mathcal{O}(k^2)$. Then it performs the update operation on the oracle for set X, in time $\mathcal{O}(k^3 \cdot \log^4 n)$. Finally, it asks oracle for each pair $u, v \in T$ whether u, v are disconnected in $G - X$, in total time $\mathcal{O}(k^3)$. Hence the complexity of the protocol is $\mathsf{poly}(k, \log n)$. ◀

4.2 Minimum Common String Partition

For two strings x, y of equal length, their *common string partition* of size k is given by partitions $x = x_1 x_2 \ldots x_k$, $y = y_1 y_2 \ldots y_k$, such that there exists a permutation $f \colon [k] \to [k]$ satisfying $x_i = y_{f(i)}$ for each $i \in [1, k]$.

Minimum Common String Partition **Parameter:** k
Input: strings x, y of equal length, integer k
Question: is there a common string partition of x, y of size at most k?

The study of this problem is motivated by applications in comparative genomics [12]. To the best of our knowledge, nothing is known about its kernelization status. In order to design a witness verification protocol, we will employ the following data structure for maintaining a dynamic collection of strings.

▶ **Proposition 19** ([64]). *There is a deterministic data structure that supports the following operations in worst-case time $\mathcal{O}(\log^2 m \cdot \log^* m)$ where m is the total number of operations.*
1. *Create a singleton string with symbol z.*
2. *Create a new string by concatenating two strings from the collection.*
3. *Create two new strings by splitting a string from the collection at the given index.*
4. *Check if two strings in the collection are equal.*
Moreover, operations $(2, 3)$ keep the host strings in the collection.

▶ **Lemma 20.** Minimum Common String Partition $\in$ WK[1].

Proof. The problem can be solved in time $2^{\mathcal{O}(k^2 \log k)} \cdot n^{\mathcal{O}(1)}$ [12] and so it suffices to give a witness verification protocol of complexity $\mathsf{poly}(k, \log n)$. The algorithm $\mathcal{A}$ builds strings x, y using operations $(1, 3)$ of data structure from Proposition 19, then it outputs the computed data structure.

A witness is a partition $x = x_1 x_2 \ldots x_k$, specified by k indices from the range $[1, n]$, together with a permutation $f \colon [k] \to [k]$. The algorithm $\mathcal{B}$ inherits the state of the data structure from $\mathcal{A}$ and creates strings $x_1, x_2, \ldots, x_k$ using the split operation. Then it concatenates them in the order given by f and checks if the resulting string is equal to y.

The witness can be encoded using $k \log n + k \log k$ bits. The verification process requires $2k + 1$ operations on the data structure, each taking time $\mathcal{O}(\log^2 n \cdot \log^* n)$. ◀

4.3   Long Path

A *feedback vertex set* (FVS) in a graph G is a set $X \subseteq V(G)$ for which $G - X$ is acyclic. The FVS number of G is the size of its smallest feedback vertex set.

LONG PATH / FVS **Parameter:** $k =$ FVS number of G
Input: undirected graph G, integer ℓ
Question: is there a path on ℓ vertices in G?

It is known that LONG PATH admits a polynomial kernel when parameterized by the vertex cover number [8] and in the same paper it was asked whether the same holds for the FVS number parameterization. To the best of our knowledge, this question remains open since 2013. An FPT algorithm for LONG PATH / FVS is implied by the following result because treewidth is never larger that the FVS number.

▶ **Proposition 21** ([5, Theorem 3.20]). LONG PATH *can be solved in time* $2^{\mathcal{O}(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ *where* tw *denotes the treewidth of the input graph.*

We remark that this result has been stated under the assumption that a tree decomposition of width tw is provided in the input but this assumption can be dropped due to known approximation algorithms for treewidth [5, §5].

For a tree T and $u, v \in V(T)$ let $P_T(u, v)$ denote the unique (u, v)-path in T. The length of a path P is defined as $|V(P)|$.

▶ **Lemma 22.** *The problem* LONG PATH / FVS *belongs to* WK[1].

Proof. The FPT algorithm follows from Proposition 21 since $\mathsf{tw}(G) \leq \mathsf{fvs}(G)$. We construct a witness verification protocol $(\mathcal{A}, \mathcal{B})$. The algorithm $\mathcal{A}$ first runs 2-approximation for FEEDBACK VERTEX SET [2] so we can assume that a feedback vertex set X of size at most $2k$ is known. Next, it computes the connected components of $G - X$ by assigning vertices in each component a unique label. Then, on each tree T we compute the following information. For each $v_1, v_2 \in V(T)$ we store the length of the path $P_T(v_1, v_2)$. For each $v_1, v_2, u_1, u_2 \in V(T)$ we store a bit indicating whether $V(P_T(v_1, v_2)) \cap V(P_T(u_1, u_2)) = \emptyset$. The algorithm $\mathcal{A}$ outputs all these structures, together with the adjacency matrix of G.

Observe that when P is a path in G, then it has at most $2k + 1$ maximal subpaths $P_1, P_2, \ldots, P_t$ in $G - X$. The witness is a description of a path P as a sequence of vertices from X and pairs of vertices from $G - X$ that form the endpoints of subpaths P_i. To verify the witness we need to check (i) that the total length of path is ℓ, (ii) that the consecutive vertices in the description are adjacent in G, and (iii) that the given subpaths are pairwise disjoint. To check condition (i) we read the length of each path P_i specified by endpoints v_1, v_2. This quantity can be retrieved from the precomputed table and the total length is then computed using $\mathcal{O}(k)$ addition. Condition (ii) is tested with the adjacency matrix. To check condition (iii), we consider all pairs $i, j \in [1, t]$ and if the endpoints of P_i, P_j belong to the same tree, we determine disjointedness of P_i, P_j using the precomputed table. Then we also compare the listed vertices from X to ensure that they are pairwise distinct. The witness can be encoded with $\mathcal{O}(k \log n)$ bits while the verification takes time $\mathcal{O}(k^2)$. This yields a witness verification protocol of complexity $\mathsf{poly}(k, \log n)$.   ◀

4.4   Steiner Tree

STEINER TREE / T **Parameter:** $k = |T|$
Input: graph G, set $T \subseteq V(G)$, integer ℓ
Question: is there a tree $X \subseteq E(G)$ that spans T and $|X| \leq \ell$?

Hermelin et al. [45] showed that STEINER TREE is WK[1]-complete when parameterized by ℓ. Since $k \leq \ell$ in any non-trivial instance, this implies that the problem is WK[1]-hard when parameterized by k and, furthermore, that it does not admit a polynomial kernel unless $NP \subseteq coNP/poly$.

▶ **Lemma 23.** *The problem* STEINER TREE / T *belongs to WK[1].*

Proof. STEINER TREE can be solved in time $3^k \cdot n^{\mathcal{O}(1)}$ [26] and we present a witness verification protocol. The algorithm $\mathcal{A}$ computes all the pairwise distances in G and outputs them together with the set T. The witness supplied to algorithm $\mathcal{B}$ is given by a set $Y \subseteq V(G) \setminus T$ of at most $k-1$ vertices and a tree F on the vertex set $T \cup Y$. The verification protocol reads the distances in G between each pair $u, v \in T \cup Y$ where $uv \in E(F)$ and checks if their sum is at most ℓ.

If the verification succeeds then clearly G contains a connected subgraph spanning T on $\leq \ell$ edges, so it also contains a tree with this property. In the other direction, suppose that there exists a solution tree X. This tree has at most k leaves so the number of vertices of degree greater than 2 is at most $k-1$. The witness guesses such vertices that are not in T and the tree F obtained from X by contracting each vertex $V(X) \setminus T$ of degree two to one of its neighbors. The total number of edges lying on paths corresponding to $E(F)$ is clearly $|X|$ which bounds the sum of distances read by the algorithm $\mathcal{B}$. This concludes the correctness proof of the protocol. The witness can be encoded using $\mathcal{O}(k \log n)$ bits while verification requires $\mathcal{O}(k)$ additions of numbers from $[1, n]$. ◄

4.5 Optimal Discretization

Consider sets $W_1, W_2 \subseteq \mathbb{Q}^2$. A pair (X, Y) of sets $X, Y \subseteq \mathbb{Q}$ is called a separation of (W_1, W_2) if for every $(x_1, y_1) \in W_1$ and $(x_2, y_2) \in W_2$ there exists an element of X strictly between x_1 and x_2 or an element of Y strictly between y_1 and y_2. A geometric interpretation of (X, Y) is a family of axis-parallel lines (vertical line ℓ_x for each $x \in X$ and horizontal line ℓ_y for each $y \in Y$) so that for every component C of $\mathbb{R}^2 \setminus \bigcup_{z \in X \cup Y} \ell_z$ the closure of C contains either only points from W_1 or only points from W_2.

OPTIMAL DISCRETIZATION   **Parameter:** k
Input: sets $W_1, W_2 \subseteq \mathbb{Q}^2$, integer k
Question: is there a separation (X, Y) of (W_1, W_2) with $|X| + |Y| \leq k$?

The geometric interpretation makes the problem attractive from the perspective of learning theory. This motivated the study of parameterized algorithms for OPTIMAL DISCRETIZATION: it turns out to be FPT [58] but only when we consider axis-parallel separating lines [10]. To the best of our knowledge, nothing is known about its kernelization status.

For $F \subseteq \mathbb{Q}$ we define $h(W) = \{ \frac{x_1 + x_2}{2} : x_1, x_2 \in F \}$. For $W \subseteq \mathbb{Q}^2$ we define sets $h_X(W), h_Y(W) \subseteq \mathbb{Q}$ as respectively $h(W_X), g(W_Y)$, where W_X, W_Y are projections of W into X/Y-coordinates. It is easy to modify any separation (X, Y) of (W_1, W_2) to only use coordinates that lie exactly in between two coordinates from the input, without affecting the set of pairs that are separated.

▶ **Observation 24.** *There exists a minimum-size separation* (X, Y) *of* (W_1, W_2) *satisfying* $X \subseteq h_X(W_1 \cup W_2), Y \subseteq h_Y(W_1 \cup W_2)$.

▶ **Lemma 25.** OPTIMAL DISCRETIZATION $\in$ WK[1].

Proof. First we check that OPTIMAL DISCRETIZATION can be solved in time $2^{\mathcal{O}(k^2 \log k)} n^{\mathcal{O}(1)}$ [58]. The algorithm $\mathcal{A}$ of the witness verification protocol considers all tuples (x_1, x_2, y_1, y_2) where $x_1, x_2 \in h_X(W_1 \cup W_2) \cup \{-\infty, \infty\}$, $y_1, y_2 \in h_Y(W_1 \cup W_2) \cup \{-\infty, \infty\}$, $x_1 < x_2$, and $y_1 < y_2$. Their number is polynomial in $|W_1| + |W_2|$. For a tuple (x_1, x_2, y_1, y_2) it checks whether $[x_1, x_2] \times [y_1, y_2]$ contains at least one point from W_1 and from W_2. If it contains both, then such tuple is marked as bad. The algorithm $\mathcal{A}$ outputs the set of bad tuples, sorted lexicographically.

The witness supplied to the algorithm $\mathcal{B}$ is a pair (X, Y) where $X \subseteq h_X(W_1 \cup W_2)$, $Y \subseteq h_Y(W_1 \cup W_2)$, and $|X| + |Y| \leq k$. By Observation 24 we can assume that there exists an optimal solution of this form. Moreover, we require the sets X, Y to be listed in the increasing order of elements. The verification mechanism considers all tuples (x_1, x_2, y_1, y_2) where x_1, x_2 are consecutive elements in $X \cup \{-\infty, \infty\}$ and likewise for y_1, y_2. There are $\mathcal{O}(k^2)$ such tuples, each corresponding to a connected component of $\mathbb{R}^2 \setminus \bigcup_{z \in X \cup Y} \ell_z$. Then (X, Y) forms a separation for (W_1, W_2) if and only if none of these tuples has been marked as bad. One can check if (x_1, x_2, y_1, y_2) is bad by performing binary search in the data structure provided by the algorithm $\mathcal{A}$. In summary, the witness can be encoded with $\mathcal{O}(k \cdot \log n)$ bits whereas verification takes time $\mathcal{O}(k^2 \cdot \log n)$. ◀

5　Weighted Vertex Cover

The *vertex cover number* (VC number) of G is the size of the smallest vertex cover in G.

WEIGHTED VERTEX COVER / VC　　　　　　　　　　　**Parameter:** VC number of G
Input: graph G, weight function $w \colon V(G) \to \mathbb{Z}_{\geq 0}$ given in unary, integer ℓ
Question: is there a vertex cover in G of total weight at most ℓ?

The unweighted variant of VERTEX COVER is probably the most heavily studied problem in kernelization [16, 35, 38, 57]. The weighted variant has a polynomial kernel when parameterized by the number of vertices in the sought solution [34] but it is unlikely to admit a polynomial kernel when parameterized by the VC number, even when the weights are given in unary [47]. We remark that if the weights were provided in binary, the theorem below would also hold but only when treating the logarithm of the maximal weight as an additional parameter.

▶ **Theorem 3.** WEIGHTED VERTEX COVER / VC *admits a linear parameter transformation to* BINARY ILP FEASIBILITY *parameterized by* $m + \log(\Delta(A))$. *Furthermore, the problem admits a linear parameter transformation to* MILP FEASIBILITY *parametrized by the number of integral variables.*

Proof. We can assume that a vertex cover Y of G of size at most $2k$ is known, due to the classic 2-approximation algorithm. The problem can be readily solved in time $2^{|Y|} \cdot |V(G)|$ so by the standard argument we can assume that $\log n \leq k$, where n is the total input size. Consider the following mixed integer linear program.

$$\min \sum_{v \in V(G)} w(v) \cdot x_v$$

$$\forall_{u \in Y} \quad \sum_{v \in N_G(u)} x_v + \deg_G(u) \cdot x_u \geq \deg_G(u)$$

$$\forall_{v \in V(G)} \quad 0 \leq x_v \leq 1$$

$$\forall_{u \in Y} \quad x_u \in \mathbb{Z}$$

The number of integral variables, as well as the number of constrains, is $|Y| \leq 2k$. Recall that the bounds $0 \leq x_v \leq 1$ are not counted as constraints in our setup.

We claim that the optimum of this program equals the minimum weight vertex cover in G. First, suppose that $S \subseteq V(G)$ is a vertex cover. We claim that the binary characteristic vector x of S satisfies all the constraints. Consider $u \in Y$. If $u \in S$ then clearly $\sum_{v \in N_G(u)} x_v + \deg_G(u) \cdot x_u \geq \deg_G(u)$. If $u \notin S$ then $N_G(u) \subseteq S$ and again the inequality holds.

In the other direction, consider a feasible solution x. We claim that set $S = \{v \in V(G) : x_v = 1\}$ forms a vertex cover. We need to show that every edge $uw \in E(G)$ is covered by S. Since Y is also a vertex cover, we can assume w.l.o.g. that $u \in Y$. Suppose that $x_u < 1$. Because x_u is an integral variable, this implies $x_u = 0$, and so it must hold that $\sum_{v \in N_G(u)} x_v \geq \deg_G(u)$. All the variables are upper bounded by 1, which enforces that $x_v = 1$ for every $v \in N_G(u)$, in particular for $v = w$. Hence the presented MILP indeed models minimum weight vertex cover.

In order to obtain a feasibility problem we replace the objective function with a constraint $\sum_{v \in V(G)} w(v) \cdot x_v \leq \ell$. This proves the second part of the theorem. Observe that the entries in the obtained ILP matrix are vertex degrees and vertex weights, which are at most 2^k because we assumed $\log n \leq k$. This gives the bound on $\Delta(A)$ and proves the first part of the theorem, by simply treating all the variables as integral. ◀

It is tempting to try to get rid of the upper bounds $x_v \leq 1$ and attain an ILP in the form considered in Theorem 1. If we allow for not $\mathcal{O}(k)$ but $\mathsf{poly}(k)$ constraints, this is equivalent to proving that WEIGHTED VERTEX COVER / VC is in WK[1]. A potential witness verification protocol could guess the intersection of a solution S with the vertex cover Y but it is unclear how to verify quickly that the remaining part of the solution has bounded weight.

6 Conclusion

We have presented a characterization of parameterized problems reducible to ILP with few constraints and classified several new problems into this category through witness verification protocols. We find this connection surprising and inspiring for exploring new directions in parameterized complexity. Can we further extend this classification? In particular, we ask whether the following problems belong to WK[1]: WEIGHTED VERTEX COVER / VC, CONNECTED FVS, DIRECTED FVS, and VERTEX PLANARIZATION (the last three parameterized by the solution size).

Our classification concerns ILPs in the standard form without upper bounds. Allowing for upper bounds on variables gives a greater expressive power when the number of constraints is a parameter (see Observation 13 and also [68]). It would be interesting to also characterize parameterized problems reducible to upper-bounded ILP with few constraints. However, the location of such a feasibility problem in the parameterized complexity landscape remains unclear because we do not know whether it admits short NP-witnesses [72]. Yet another direction is to describe problems that are reducible to ILP with few variables.

Containment in WK[1] only ensures that a problem can be modeled by ILP with $\mathsf{poly}(k)$ constraints. Even though the degree of this polynomial can be retraced from the chain of reductions, it is unlikely to be optimal. As the natural next step towards bridging the gap between theory and practice, we would like to optimize the polynomial degree, by designing explicit ILP formulation with $\mathcal{O}(k^c)$ constraints, where c is as small as possible, ideally one. A good starting point would be to consider problems with simple proofs of containment in WK[1], such as k-PATH or CONNECTED VERTEX COVER [45].

────── **References** ──────

1   Stephan Artmann, Robert Weismantel, and Rico Zenklusen. A strongly polynomial algorithm for bimodular integer linear programming. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 1206–1219, New York, NY, USA, 2017. Association for Computing Machinery. `doi:10.1145/3055399.3055473`.

2   Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999. `doi:10.1137/S0895480196305124`.

3   Max Bannach, Florian Chudigiewitsch, Kim-Manuel Klein, and Marcel Wienöbst. PACE Solver Description: UzL Exact Solver for One-Sided Crossing Minimization. In Édouard Bonnet and Paweł Rzążewski, editors, *19th International Symposium on Parameterized and Exact Computation (IPEC 2024)*, volume 321 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:4, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.IPEC.2024.28`.

4   Moritz Bergenthal, Jona Dirks, Thorben Freese, Jakob Gahde, Enna Gerhard, Mario Grobler, and Sebastian Siebertz. PACE Solver Description: GraPA-JAVA. In Holger Dell and Jesper Nederlof, editors, *17th International Symposium on Parameterized and Exact Computation (IPEC 2022)*, volume 249 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:4, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.IPEC.2022.30`.

5   Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation*, 243:86–111, 2015. 40th International Colloquium on Automata, Languages and Programming (ICALP 2013). `doi:10.1016/j.ic.2014.12.008`.

6   Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009. `doi:10.1016/j.jcss.2009.04.001`.

7   Hans L. Bodlaender, Carla Groenland, Jesper Nederlof, and Céline M. F. Swennenhuis. Parameterized problems complete for nondeterministic FPT time and logarithmic space. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 193–204. IEEE, 2021. `doi:10.1109/FOCS52979.2021.00027`.

8   Hans L Bodlaender, Bart MP Jansen, and Stefan Kratsch. Kernel bounds for path and cycle problems. *Theoretical Computer Science*, 511:117–136, 2013. `doi:10.1016/j.tcs.2012.09.006`.

9   Kimon Boehmer, Lukas Lee George, Fanny Hauser, and Jesse Palarus. PACE Solver Description: Arcee. In *19th International Symposium on Parameterized and Exact Computation (IPEC 2024)*, volume 321 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 33:1–33:4, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.IPEC.2024.33`.

10   Édouard Bonnet, Panos Giannopoulos, and Michael Lampis. On the Parameterized Complexity of Red-Blue Points Separation. In Daniel Lokshtanov and Naomi Nishimura, editors, *12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*, volume 89 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:13, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.IPEC.2017.8`.

11   Cornelius Brand, Martin Koutecký, and Sebastian Ordyniak. Parameterized algorithms for MILPs with small treedepth. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12249–12257. AAAI Press, 2021. `doi:10.1609/AAAI.V35I14.17454`.

12 Laurent Bulteau and Christian Komusiewicz. Minimum common string partition parameterized by partition size is fixed-parameter tractable. In Chandra Chekuri, editor, *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 102–121. SIAM, 2014. `doi:10.1137/1.9781611973402.8`.

13 Kevin Burrage, Vladimir Estivill-Castro, Michael Fellows, Michael Langston, Shev Mac, and Frances Rosamond. The undirected feedback vertex set problem has a poly(k) kernel. In Hans L. Bodlaender and Michael A. Langston, editors, *Parameterized and Exact Computation*, pages 192–202, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. `doi:10.1007/11847250_18`.

14 Der-San Chen, Robert G Batson, and Yu Dang. *Applied Integer Programming: Modeling and Solution*. Wiley Online Library, 2009. `doi:10.1002/9781118166000`.

15 Hua Chen, Lin Chen, and Guochuan Zhang. FPT algorithms for a special block-structured integer program with applications in scheduling. *Mathematical Programming*, 208(1):463–496, 2024. `doi:10.1007/s10107-023-02046-z`.

16 Benny Chor, Mike Fellows, and David Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In *Graph-Theoretic Concepts in Computer Science: 30th International Workshop, WG 2004, Bad Honnef, Germany, June 21-23, 2004. Revised Papers 30*, pages 257–269. Springer, 2005. `doi:10.1007/978-3-540-30559-0_22`.

17 Jana Cslovjecsek, Martin Koutecký, Alexandra Lassota, Michał Pilipczuk, and Adam Polak. Parameterized algorithms for block-structured integer programs with large entries. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 740–751. SIAM, 2024. `doi:10.1137/1.9781611977912.29`.

18 William H. Cunningham and Jim Geelen. On integer programming and the branch-width of the constraint matrix. In Matteo Fischetti and David P. Williamson, editors, *Integer Programming and Combinatorial Optimization*, pages 158–166, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. `doi:10.1007/978-3-540-72792-7_13`.

19 Marek Cygan, Paweł Komosa, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, Saket Saurabh, and Magnus Wahlström. Randomized contractions meet lean decompositions. *ACM Transactions on Algorithms (TALG)*, 17(1):1–30, 2020. `doi:10.1145/3426738`.

20 Marek Cygan, Stefan Kratsch, Marcin Pilipczuk, Michał Pilipczuk, and Magnus Wahlström. Clique cover and graph separation: New incompressibility results. *ACM Trans. Comput. Theory*, 6(2), May 2014. `doi:10.1145/2594439`.

21 Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. On multiway cut parameterized above lower bounds. *ACM Trans. Comput. Theory*, 5(1):3:1–3:11, 2013. `doi:10.1145/2462896.2462899`.

22 George Bernard Dantzig, Delbert R Fulkerson, and Selmer Martin Johnson. On a linear-programming, combinatorial approach to the traveling-salesman problem. *Operations Research*, 7(1):58–66, 1959. `doi:10.1287/opre.7.1.58`.

23 Jona Dirks, Mario Grobler, Roman Rabinovich, Yannik Schnaubelt, Sebastian Siebertz, and Maximilian Sonneborn. PACE solver description: PACA-JAVA. In Petr A. Golovach and Meirav Zehavi, editors, *16th International Symposium on Parameterized and Exact Computation, IPEC 2021, September 8-10, 2021, Lisbon, Portugal*, volume 214 of *LIPIcs*, pages 30:1–30:4. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPICS.IPEC.2021.30`.

24 Alexander Dobler. PACE Solver Description: CRGone. In Édouard Bonnet and Paweł Rzążewski, editors, *19th International Symposium on Parameterized and Exact Computation (IPEC 2024)*, volume 321 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:4, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.IPEC.2024.29`.

25 Rodney G. Downey and Michael R. Fellows. Parameterized computational feasibility. In Peter Clote and Jeffrey B. Remmel, editors, *Feasible Mathematics II*, pages 219–244, Boston, MA, 1995. Birkhäuser Boston. `doi:10.1007/978-1-4612-2566-9_7`.

26   S. E. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971. `doi:10.1002/net.3230010302`.

27   Andrew Drucker, Jesper Nederlof, and Rahul Santhanam. Exponential Time Paradigms Through the Polynomial Time Lens. In Piotr Sankowski and Christos Zaroliagis, editors, *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.ESA.2016.36`.

28   Pavel Dvořák, Eduard Eiben, Robert Ganian, Dušan Knop, and Sebastian Ordyniak. Solving integer linear programs with a small number of global variables and constraints. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 607–613. AAAI Press, 2017. `doi:10.5555/3171642.3171730`.

29   M. Ayaz Dzulfikar, Johannes K. Fichte, and Markus Hecher. The PACE 2019 Parameterized Algorithms and Computational Experiments Challenge: The Fourth Iteration. In Bart M. P. Jansen and Jan Arne Telle, editors, *14th International Symposium on Parameterized and Exact Computation (IPEC 2019)*, volume 148 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:23, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.IPEC.2019.25`.

30   Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. Sparse integer programming is fixed-parameter tractable. *Math. Oper. Res.*, 50(3):2141–2156, 2025. `doi:10.1287/MOOR.2023.0162`.

31   Friedrich Eisenbrand and Gennady Shmonin. Carathéodory bounds for integer cones. *Oper. Res. Lett.*, 34(5):564–568, 2006. `doi:10.1016/J.ORL.2005.09.008`.

32   Friedrich Eisenbrand and Robert Weismantel. Proximity results and faster algorithms for integer programming using the Steinitz lemma. *ACM Trans. Algorithms*, 16(1):5:1–5:14, 2020. `doi:10.1145/3340322`.

33   Michael Elberfeld, Christoph Stockhusen, and Till Tantau. On the space and circuit complexity of parameterized problems: Classes and completeness. *Algorithmica*, 71(3):661–701, 2015. `doi:10.1007/S00453-014-9944-Y`.

34   Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. Polynomial kernels for weighted problems. *J. Comput. Syst. Sci.*, 84:1–10, 2017. `doi:10.1016/j.jcss.2016.06.004`.

35   Michael R. Fellows, Bart M.P. Jansen, and Frances Rosamond. Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *European Journal of Combinatorics*, 34(3):541–566, 2013. Combinatorial Algorithms and Complexity. `doi:10.1016/j.ejc.2012.04.008`.

36   Samuel Fiorini, Gwenaël Joret, Stefan Weltge, and Yelena Yuditsky. Integer programs with bounded subdeterminants and two nonzeros per row. *J. ACM*, 72(1), January 2025. `doi:10.1145/3695985`.

37   Christodoulos A Floudas and Xiaoxia Lin. Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. *Annals of Operations Research*, 139:131–162, 2005. `doi:10.1007/s10479-005-3446-x`.

38   Fedor Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019. `doi:10.1017/9781107415157`.

39   Fedor V Fomin, Fahad Panolan, Maadapuzhi Sridharan Ramanujan, and Saket Saurabh. On the optimality of pseudo-polynomial algorithms for integer programming. *Mathematical Programming*, 198(1):561–593, 2023. `doi:10.1007/s10107-022-01783-x`.

40   Robert Ganian and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ilp. *Artificial Intelligence*, 257:61–71, 2018. `doi:10.1016/j.artint.2017.12.006`.

41   Tomáš Gavenčiak, Martin Koutecký, and Dušan Knop. Integer programming in parameterized complexity: Five miniatures. *Discrete Optimization*, 44:100596, 2022. Optimization and Discrete Geometry. `doi:10.1016/j.disopt.2020.100596`.

42 Michel X Goemans and Young-Soo Myung. A catalog of steiner tree formulations. *Networks*, 23(1):19–28, 1993. `doi:10.1002/net.3230230104`.

43 Ernestine Großmann, Tobias Heuer, Christian Schulz, and Darren Strash. The PACE 2022 Parameterized Algorithms and Computational Experiments Challenge: Directed Feedback Vertex Set. In Holger Dell and Jesper Nederlof, editors, *17th International Symposium on Parameterized and Exact Computation (IPEC 2022)*, volume 249 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:18, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.IPEC.2022.26`.

44 Danny Hermelin, Shlomo Karhi, Michael Pinedo, and Dvir Shabtay. New algorithms for minimizing the weighted number of tardy jobs on a single machine. *Annals of Operations Research*, 298:271–287, 2021. `doi:10.1007/s10479-018-2852-9`.

45 Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. A Completeness Theory for Polynomial (Turing) Kernelization. *Algorithmica*, 71(3):702–730, 2015. `doi:10.1007/S00453-014-9910-8`.

46 Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM*, 48(4):723–760, July 2001. `doi:10.1145/502090.502095`.

47 Bart M. P. Jansen and Hans L. Bodlaender. Vertex cover kernelization revisited - upper and lower bounds for a refined parameter. *Theory Comput. Syst.*, 53(2):263–299, 2013. `doi:10.1007/S00224-012-9393-4`.

48 Bart M. P. Jansen and Stefan Kratsch. A structural approach to kernels for ilps: Treewidth and total unimodularity. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 779–791. Springer, 2015. `doi:10.1007/978-3-662-48350-3_65`.

49 Bart M. P. Jansen, Marcin Pilipczuk, and Erik Jan van Leeuwen. A deterministic polynomial kernel for odd cycle transversal and vertex multiway cut in planar graphs. *SIAM J. Discret. Math.*, 35(4):2387–2429, 2021. `doi:10.1137/20M1353782`.

50 Klaus Jansen, Alexandra Lassota, and Lars Rohwedder. Near-linear time algorithm for n-fold ILPs via color coding. *SIAM Journal on Discrete Mathematics*, 34(4):2282–2299, 2020. `doi:10.1137/19M1303873`.

51 Klaus Jansen and Lars Rohwedder. On Integer Programming and Convolution. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, volume 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:17, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.ITCS.2019.43`.

52 Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, 12(3):415–440, 1987. `doi:10.1287/moor.12.3.415`.

53 Leon Kellerhals, Tomohiro Koana, André Nichterlein, and Philipp Zschoche. The PACE 2021 Parameterized Algorithms and Computational Experiments Challenge: Cluster Editing. In Petr A. Golovach and Meirav Zehavi, editors, *16th International Symposium on Parameterized and Exact Computation (IPEC 2021)*, volume 214 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.IPEC.2021.26`.

54 Philipp Kindermann, Fabian Klute, and Soeren Terziadis. The PACE 2024 Parameterized Algorithms and Computational Experiments Challenge: One-Sided Crossing Minimization. In Édouard Bonnet and Paweł Rzążewski, editors, *19th International Symposium on Parameterized and Exact Computation (IPEC 2024)*, volume 321 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:20, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.IPEC.2024.26`.

55 Dušan Knop and Martin Koutecký. Scheduling meets n-fold integer programming. *Journal of Scheduling*, 21:493–503, 2018. `doi:10.1007/S10951-017-0550-0`.

56   Stefan Kratsch. On polynomial kernels for integer linear programs: Covering, packing and feasibility. In Hans L. Bodlaender and Giuseppe F. Italiano, editors, *Algorithms – ESA 2013*, pages 647–658, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. `doi:10.1007/978-3-642-40450-4_55`.

57   Stefan Kratsch. A randomized polynomial kernelization for vertex cover with a smaller parameter. *SIAM Journal on Discrete Mathematics*, 32(3):1806–1839, 2018. `doi:10.1137/16M1104585`.

58   Stefan Kratsch, Tomás Masařík, Irene Muzi, Marcin Pilipczuk, and Manuel Sorge. Optimal discretization is fixed-parameter tractable. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1702–1719. SIAM, 2021. `doi:10.1137/1.9781611976465.103`.

59   Stefan Kratsch and Vuong Anh Quyen. On kernels for covering and packing ilps with small coefficients. In Marek Cygan and Pinar Heggernes, editors, *Parameterized and Exact Computation*, pages 307–318, Cham, 2014. Springer International Publishing. `doi:10.1007/978-3-319-13524-3_26`.

60   Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. *J. ACM*, 67(3):16:1–16:50, 2020. `doi:10.1145/3390887`.

61   Yaowei Long and Thatchaphol Saranurak. Near-optimal deterministic vertex-failure connectivity oracles. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 1002–1010. IEEE, 2022. `doi:10.1109/FOCS54457.2022.00098`.

62   George S Lueker. *Two NP-complete problems in nonnegative integer programming*. Princeton University. Department of Electrical Engineering, 1975.

63   Dániel Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, 2006. Parameterized and Exact Computation. `doi:10.1016/j.tcs.2005.10.007`.

64   Kurt Mehlhorn, R. Sundar, and Christian Uhrig. Maintaining dynamic sequences under equality tests in polylogarithmic time. *Algorithmica*, 17(2):183–198, 1997. `doi:10.1007/BF02522825`.

65   C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, October 1960. `doi:10.1145/321043.321046`.

66   George Nemhauser and Laurence Wolsey. *The Scope of Integer and Combinatorial Optimization*, chapter I.1, pages 1–26. John Wiley and Sons, Ltd, 1988. `doi:10.1002/9781118627372.ch1`.

67   Michał Pilipczuk and Marcin Wrochna. On space efficiency of algorithms working on structural decompositions of graphs. *ACM Trans. Comput. Theory*, 9(4):18:1–18:36, 2018. `doi:10.1145/3154856`.

68   Lars Rohwedder and Karol Węgrzycki. Fine-Grained Equivalence for Problems Related to Integer Linear Programming. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference (ITCS 2025)*, volume 325 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 83:1–83:18, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.ITCS.2025.83`.

69   Stéphan Thomassé. A 4k2 kernel for feedback vertex set. *ACM Trans. Algorithms*, 6(2), April 2010. `doi:10.1145/1721837.1721848`.

70   Magnus Wahlström. Quasipolynomial multicut-mimicking networks and kernels for multiway cut problems. *ACM Trans. Algorithms*, 18(2):15:1–15:19, 2022. `doi:10.1145/3501304`.

71   Justin C Williams. A linear-size zero – One programming model for the minimum spanning tree problem in planar graphs. *Networks: An International Journal*, 39(1):53–60, 2002. `doi:10.1002/net.10010`.

72   Michal Wlodarczyk. Does subset sum admit short proofs? In Julián Mestre and Anthony Wirth, editors, *35th International Symposium on Algorithms and Computation, ISAAC 2024, December 8-11, 2024, Sydney, Australia*, volume 322 of *LIPIcs*, pages 58:1–58:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. `doi:10.4230/LIPICS.ISAAC.2024.58`.