On the Descriptive Complexity of Vertex Deletion Problems

Max Bannach
European Space Agency, Advanced Concepts Team, Noordwijk, The Netherlands

Florian Chudigiewitsch
Universität zu Lübeck, Germany

Till Tantau
Universität zu Lübeck, Germany

Abstract

Vertex deletion problems for graphs are studied intensely in classical and parameterized complexity theory. They ask whether we can delete at most k vertices from an input graph such that the resulting graph has a certain property. Regarding k as the parameter, a dichotomy was recently shown based on the number of quantifier alternations of first-order formulas that describe the property. In this paper, we refine this classification by moving from quantifier alternations to individual quantifier patterns and from a dichotomy to a trichotomy, resulting in a complete classification of the complexity of vertex deletion problems based on their quantifier pattern. The more fine-grained approach uncovers new tractable fragments, which we show to not only lie in FPT, but even in parameterized constant-depth circuit complexity classes. On the other hand, we show that vertex deletion becomes intractable already for just one quantifier per alternation, that is, there is a formula of the form $\forall x \exists y \forall z(\psi)$, with ψ quantifier-free, for which the vertex deletion problem is W[1]-hard. The fine-grained analysis also allows us to uncover differences in the complexity landscape when we consider different kinds of graphs and more general structures: While basic graphs (undirected graphs without self-loops), undirected graphs, and directed graphs each have a different frontier of tractability, the frontier for arbitrary logical structures coincides with that of directed graphs.

2012 ACM Subject Classification Theory of computation → Finite Model Theory; Theory of computation → Complexity theory and logic; Theory of computation → Fixed parameter tractability; Theory of computation → W hierarchy

Keywords and phrases graph problems, fixed-parameter tractability, descriptive complexity, vertex deletion

Digital Object Identifier 10.4230/LIPIcs.MFCS.2024.17

Related Version *Full Version*: `https://arxiv.org/abs/2406.18299` [4]

1 Introduction

A recent research topic in parametrized complexity are *distance to triviality problems.* We are asked how many modification steps (the "distance") we need to apply to a logical structure in order to transform it into a "trivial" one – which can mean anything from "no edges at all" to "no cycles" or even more exotic properties like "no cycles of odd length." Such problems have been found highly useful in modern algorithm design [1, 2, 11, 21] and are now an important test bed for new algorithmic ideas and data reduction procedures [14, 15, 22, 23].

Many problems that have been studied thoroughly in the literature turn out to be vertex deletion problems. The simplest example arises from *vertex covers,* which measure the "distance in terms of vertex deletions" of a graph from being edge-free: A graph has a vertex cover of size k iff it can be made edge-free by deleting at most k vertices. For a slightly more complex example, the *cluster deletion problem* asks whether we can delete at most k vertices from a graph so that it becomes a cluster graph, meaning that every connected component is

49th International Symposium on Mathematical Foundations of Computer Science (MFCS 2024).
Editors: Rastislav Královič and Antonín Kučera; Article No. 17; pp. 17:1–17:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

a clique or, equivalently, is P_3-free (meaning, there is no induced path on three vertices). The *feedback vertex set problem* asks if we can delete at most k vertices, such that the resulting graph has no cycles. The *odd cycle transversal problem* asks if there is a set of vertices of size at most k, such that removing it destroys every odd cycle. Equivalently, the problem asks if we can delete at most k vertices, such that the resulting graph is bipartite.

To investigate the complexity of vertex deletion problems in a systematic way, it makes sense to limit the graph properties to have some structure. An early result in this direction [25] is the NP-completeness of vertex deletion to hereditary graph properties that can be tested in polynomial time. Intuitively, vertex deletion problems should be easier to solve for graph properties that are simpler to express. Phrased in terms of descriptive complexity theory, if we can describe a graph property using, say, a simple first-order formula, the corresponding vertex deletion problem should also be simple. The intuition was proven to be correct in 2020, when Fomin et al. [17] established a dichotomy based on the number of quantifier alternations that characterizes the classes of first-order logic formulas for which the vertex deletion problem is fixed-parameter tractable.

The results of Fomin et al. directly apply to some of the above examples: Consider the problem p-VERTEX-COVER, whose "triviality" property is described by the formula $\phi_{\text{vc}} = \forall x \forall y (x \not\sim y)$, or the problem p-CLUSTER-DELETION, whose triviality property is described by $\phi_{\text{cd}} = \forall x \forall y \forall z \big((x \sim y \land y \sim z) \to x \sim z\big)$. Both first-order formulas use *no quantifier alternations,* which by [17] already implies that the problems lie in para-P = FPT. Naturally, not all problems can be characterized so easily: Properties like acyclicity (which underlies the feedback vertex set problem) cannot be expressed in first-order logic and, thus, the results of Fomin et al. do not apply to them. Fomin et al. also show that if there are enough quantifier alternations (three, to be precise) in the first-order formulas describing the property, then the resulting vertex deletion problem can be W[1]-hard. Nevertheless, the descriptive approach allows us to identify large fragments of logical formulas and hence large classes of vertex deletion problems that are (at least fixed-parameter) tractable.

A first central question addressed in the present paper is whether the number of quantifier *alternations* (the property studied in [17]) overshadows all other aspects in making problems hard, or whether the individual quantifier pattern of the formula plays a significant role as well. This question appears to be of particular importance given that formulas describing natural problems (like ϕ_{vc} and ϕ_{cd} above) tend to have short and simple quantifier patterns: We might hope that even though we describe a particular triviality property using, say, four alternations, the fact that we use only, say, two existential quantifiers in total still assures us that the resulting vertex deletion problem is easy.

A second central question is whether the *kind* of graphs that we allow as inputs has an influence on the complexity of the problem. Intuitively, allowing only, say, *basic graphs* (simple undirected graphs without self-loops) should result in simpler problems than allowing directed graphs or even arbitrary logical structures as input. This intuition is known to be correct in the closely related question of deciding graph properties described in existential second-order logic. As we will see, in the context of vertex deletion problems it makes a difference whether we consider basic graphs, undirected graphs, or directed graphs, but not whether we consider directed graphs or arbitrary logical structures.

Our Contributions. We completely classify the parameterized complexity of vertex deletion problems in dependence of the quantifier pattern of the formulas that are used to express the triviality property and also in dependence of the kind of graphs that we allow as inputs (basic, undirected, directed, or arbitrary logical structures). An overview of the results is given in Table 1, where the following notations are used (detailed definitions are given later): For a first-order formula ϕ over the vocabulary $\tau = \{\sim^2\}$ of (directed, simple) graphs, the parameterized problem p_k-VERTEX-DELETION$_{\text{dir}}(\phi)$ (abbreviated p-VD$_{\text{dir}}(\phi)$) asks us to tell on input of a directed graph G and a parameter $k \in \mathbb{N}$ whether we can delete at most k vertices from G, so that for the resulting graph G' we have $G' \models \phi$. The problems p-VD$_{\text{undir}}(\phi)$ and p-VD$_{\text{basic}}(\phi)$ are the restrictions where the input graphs are undirected or basic graphs (undirected graphs without self-loops), respectively. For instance, p-VERTEX-COVER = p-VD$_{\text{basic}}(\phi_{\text{vc}})$ = p-VD$_{\text{basic}}\big(\forall x \forall y (x \not\sim y)\big)$. In the other direction, let p-VD$_{\text{arb}}(\phi)$ denote the generalization where we allow an arbitrary logical vocabulary τ and arbitrary (finite) logical structures $\mathcal{A}$ instead of just graphs G (and where "vertex deletion" should better be called "element deletion," but we stick with the established name). For a *(first-order) quantifier pattern p*, which is just a string of a's and e's standing for the universal and existential quantifiers at the beginning of a formula ϕ, we write p-VD$_{\text{basic}}(p)$ for the class of all problems p-VD$_{\text{basic}}(\phi)$ where ϕ has all its quantifiers at the beginning and they form the pattern p. For instance, p-VERTEX-COVER $\in$ p-VD$_{\text{basic}}(aa)$ as ϕ_{vc} has two universal quantifiers. The same notation is used for undirected graphs, directed graphs, and arbitrary structures.

Table 1 Complete complexity classification of vertex deletion problems for first-order formulas in dependence of the quantifier pattern $p \in \{a, e\}^*$ (where $p \preceq q$ means that p is a subsequence of q). The four different considered restrictions on the allowed input structures lead to three distinct complexity landscapes. Note that para-AC^0 $\subsetneq$ para-AC$^{0\uparrow}$ $\subseteq$ para-P = FPT holds and that it is a standard assumption that FPT $\cap$ W[2]-hard = $\emptyset$ also holds.

p-VD$_{\text{basic}}(p)$	$\subseteq$ para-AC^0, when	$p \preceq e^*a^*$ or eae.
	$\not\subseteq$ para-AC^0 but $\subseteq$ para-AC$^{0\uparrow}$, when	$eeae$, aae or $aee \preceq p \preceq e^*a^*e^*$.
	$\cap$ W[2]-hard $\neq \emptyset$, when	$aea \preceq p$.
p-VD$_{\text{undir}}(p)$	$\subseteq$ para-AC^0, when	$p \preceq ae$ or e^*a^*.
	$\not\subseteq$ para-AC^0 but $\subseteq$ para-AC$^{0\uparrow}$, when	eae, aae or $aee \preceq p \preceq e^*a^*e^*$.
	$\cap$ W[2]-hard $\neq \emptyset$, when	$aea \preceq p$.
p-VD$_{\text{dir}}(p)$ and p-VD$_{\text{arb}}(p)$	$\subseteq$ para-AC^0, when	$p \preceq e^*a^*$.
	$\not\subseteq$ para-AC^0 but $\subseteq$ para-AC$^{0\uparrow}$, when	$ae \preceq p \preceq e^*a^*e^*$.
	$\cap$ W[2]-hard $\neq \emptyset$, when	$aea \preceq p$.

The results in Table 1 give an answer to the first central question formulated earlier, which asked whether it is the *number of alternations* of quantifiers in patterns (and not so much the actual number of quantifiers) that are responsible for the switch from tractable to intractable observed by Fomin et al. [17], or whether the frontier is formed by short patterns that "just happen" to have a certain number of alternations. As can be seen, the latter is true: All intractability results hold already for very short and simple patterns. Thus, while it was previously known that there is a formula in Π_3 (meaning it has a pattern of the form $\forall^*\exists^*\forall^*$ or $a^*e^*a^*$ in our notation) defining an intractable problem, we show that already one quantifier per alternation (the pattern aea) suffices. On the positive side, Table 1 shows that all vertex deletion problems that are (fixed-parameter) tractable at all already lie in the classes para-AC^0 or at least para-AC$^{0\uparrow}$. From an algorithmic point of view this means that all of the vertex deletion problems that we classify as fixed-parameter tractable admit efficient *parallel* fixed-parameter algorithms.

Concerning the second central question, which asked whether it makes a difference which kind of graphs or logical structures we consider, Table 1 also provides a comprehensive answer: First, the *frontier of tractability* (the patterns where we switch from membership in FPT to hardness for W[1]) *is the same for all kinds of inputs* (namely from "does not contain *aea* as a subsequence" to "contains *aea* as a subsequence"). Second, if we classify the tractable fragments further, a more complex complexity landscape arises: While p-$\mathrm{VD}_{\mathrm{dir}}(p)$ and p-$\mathrm{VD}_{\mathrm{arb}}(p)$ have the same classification for all p, the classes p-$\mathrm{VD}_{\mathrm{basic}}(p)$ and p-$\mathrm{VD}_{\mathrm{undir}}(p)$ each exhibit a different behavior. In other words: For simple patterns p, it makes a difference whether the inputs are basic, undirected, or directed graphs.

The just-discussed structural results are different from classifications in dependence of quantifier patterns p established in previous works: Starting with Eiter et al. [13] and subsequently Gottlob et al. [20], Tantau [27] and most recently Bannach et al. [3], different authors have classified the complexity of *weighted definability problems* by the quantifier patterns used to describe them. In these problems, formulas have a free set variable and we ask whether there is an assignment to the set variable with at most k elements such that the formula is true. Since it is easy to see that the vertex deletion problems we study are special cases of this question, upper bounds from earlier research also apply in our setting. However, our results show that (as one would hope) for vertex deletion problems for many patterns p we get better upper bounds than in the more general setting. Furthermore, there is an interesting structural insight related to our second central question: While the results in [3] for weighted definability show that, there, the complexities for undirected graphs, directed graphs, and arbitrary logical structures all coincide (but differ for basic graphs), for the vertex deletion setting, we get three different complexity characterizations for basic, undirected, and directed graphs – but the latter coincide with arbitrary structures once more.

Related Work. The complexity-theoretic investigation of vertex deletion problems has a long and fruitful history. Starting in classical complexity theory, results on vertex deletion problems were established as early as in the late 1970s [24, 25, 28]. The focus was mostly on deletion to commonly known graph properties, such as planarity, acyclicity or bipartiteness.

Since it is very natural to regard the number of allowed modifications as the parameter of the problem, the investigation of vertex deletion problems quickly gained traction in parameterized complexity, with continued research to this day [7, 19, 26]. Specifically for graphs, similar problems like the deletion or modification of edges [8] or alternative distance measures such as elimination distance [18] are also considered. Regarding first-order definable properties, a dichotomy is shown in [17].

The quantifier patterns we employ in this paper have also received a lot of attention, especially in the context of descriptive complexity. Early uses go as far back as the classification of decidable fragments of first-order logic [6]. They were then considered in the context of classical complexity [13, 20, 27] and later also in the context of parameterized complexity [3].

Organization of this Paper. Following a review of basic concepts and terminology in Section 2, we present the complexity-theoretic classification of the vertex deletion problems for basic, undirected and directed graphs in Sections 3, 4 and 5, respectively. For theorems and lemmas marked "▼ [4]", the proofs can be found in the full version [4].

2 Background in Descriptive and Parameterized Complexity

Terminology from Finite Model Theory. In this paper, we will use standard terminology from finite model theory, for a thorough introduction, see, for example [12]. A *relational vocabulary* τ (also known as a *signature*) is a set of *relation symbols* to each of which we assign a positive *arity,* denoted using a superscript. For example, $\tau = \{P^1, E^2\}$ is a relational vocabulary with a monadic relation symbol P and a dyadic relation symbol E. A *τ-structure* $\mathcal{A}$ consists of a *universe* A and for each relation symbol $R \in \tau$ of some arity r of a relation $R^{\mathcal{A}} \subseteq A^r$. We denote the set of *finite τ-structures* as $\text{STRUC}[\tau]$. For a first-order τ-sentence ϕ, we write $\text{MODELS}(\phi)$ for the class of finite models of ϕ. A *decision problem* P is a subset of $\text{STRUC}[\tau]$ which is closed under isomorphisms. A formula ϕ *describes* P if $\text{MODELS}(\phi) = P$.

For τ-structures $\mathcal{A}$ and $\mathcal{B}$ with universes A and B, respectively, we say that $\mathcal{A}$ is an *induced substructure* of $\mathcal{B}$ if $A \subseteq B$ and for all r-ary $R \in \tau$, we have $R^{\mathcal{A}} = R^{\mathcal{B}} \cap A^r$. For a set $S \subseteq B$, we denote by $\mathcal{B} \setminus S$ the substructure induced on $B \setminus S$.

We regard *directed* graphs $G = (V, E)$ (which are pairs of a nonempty vertex set V and an edge relation $E \subseteq V \times V$) as logical structures $\mathcal{G}$ over the vocabulary $\tau_{\text{digraph}} = \{\sim^2\}$ where V is the universe and $\sim^{\mathcal{G}} = E$. An *undirected* graph is a directed graph that additionally satisfies $\phi_{\text{undirected}} := \forall x \forall y (x \sim y \to y \sim x)$, while a *basic* graph satisfies $\phi_{\text{basic}} := \forall x \forall y \big(x \sim y \to (y \sim x \land x \neq y)\big)$.

For a first-order logic formula in prenex normal form (meaning all quantifiers are at the front), we can associate a *quantifier prefix pattern* (or *pattern* for short), which are words over the alphabet $\{e, a\}$.[1] For example, the formula ϕ_{basic} has the pattern aa, while the formula $\phi_{\text{degree-}\geq 2} := \forall x \exists y_1 \exists y_2 \big((x \sim y_1) \land (x \sim y_2) \land (y_1 \neq y_2)\big)$ has the pattern aee. As another example, the formulas in the class Π_2 (which start with a universal quantifier and have one alternation) are exactly the formulas with a pattern $p \in \{a\}^* \circ \{e\}^*$, which we write briefly as $p \in a^* e^*$. We write $p \preceq q$ if p is a subsequence of q.

Terminology from Parameterized Complexity. We use standard definitions from parameterized complexity, see for instance [9, 10, 16]. A *parameterized problem* is a set $Q \subseteq \Sigma^* \times \mathbb{N}$ for an alphabet Σ. In an *instance* $(x, k) \in \Sigma^* \times \mathbb{N}$ we call x the *input* and k the *parameter*. The central problem we consider in this paper is the following:

▶ **Problem 2.1** ($\text{p-VD}_{\text{arb}}(\phi)$, where ϕ is a first-order τ-formula).

Instance: *(An encoding of) a logical τ-structure $\mathcal{A}$ and an integer $k \in \mathbb{N}$.*
Parameter: *k.*
Question: *Is there a set $S \subseteq A$ with $|S| \leq k$ such that $\mathcal{A} \setminus S \models \phi$?*

As mentioned earlier, we also consider the problems $\text{p-VD}_{\text{basic}}(\phi)$, where the input structures are basic graphs (formally, $\text{p-VD}_{\text{basic}}(\phi) = \text{p-VD}_{\text{arb}}(\phi) \cap \big(\text{MODELS}(\phi_{\text{basic}}) \times \mathbb{N}\big)$), the problems $\text{p-VD}_{\text{undir}}(\phi)$, where the input structures are undirected graphs, and $\text{p-VD}_{\text{dir}}(\phi)$, where the input structures are directed graphs. For a pattern $p \in \{a, e\}^*$, the class $\text{p-VD}_{\text{arb}}(p)$ contains all problems $\text{p-VD}_{\text{arb}}(\phi)$ such that ϕ has pattern p. The classes with the subscripts "basic", "undir", and "dir" are defined similarly.

[1] One uses "a" and "e" in patterns rather than "$\forall$" and "$\exists$" since in the context of second-order logic one needs a way to differentiate between first-order and second-order quantifiers and, there, "E" refers to a "second-order $\exists$" while "e" refers to a "first-order $\exists$". In our paper, we only use first-order quantifiers so only lowercase letters are needed.

We will consider some parameterized circuit complexity classes. We define para-AC^0 as the class of parameterized problems that can be decided by a family of unbounded fan-in circuits $(C_{n,k})_{n,k\in\mathbb{N}}$ of constant depth and size $f(k)\cdot n^{O(1)}$ for some computable function f. Similarly, para-FAC^0 is the class of functions that can be computed by a family of unbounded fan-in circuits $(C_{n,k})_{n,k\in\mathbb{N}}$ of constant depth and size $f(k)\cdot n^{O(1)}$ for some computable function f. For para-$\mathrm{AC}^{0\uparrow}$, we allow the circuit to have depth $f(k)$. Questions of uniformity will not be important in the present paper. For these classes, we have the following inclusions: para-$\mathrm{AC}^0 \subsetneq$ para-$\mathrm{AC}^{0\uparrow} \subseteq$ para-P $=$ FPT.

A parameterized problem $Q \subseteq \Sigma^* \times \mathbb{N}$ is para-AC^0-many-one-reducible to a problem $Q' \subseteq \Gamma^* \times \mathbb{N}$, written $Q \leq_{\mathrm{m}}^{\text{para-AC}^0} Q'$, if there is a function $f\colon \Sigma^* \times \mathbb{N} \to \Gamma^* \times \mathbb{N}$, such that (1) for all $(x,k) \in \Sigma^* \times \mathbb{N}$ we have $(x,k) \in Q$ iff $f(x,k) \in Q'$, (2) there is a computable function $g\colon \mathbb{N} \to \mathbb{N}$ such that for all $(x,k) \in \Sigma^* \times \mathbb{N}$, we have $k' \leq g(k)$, where $f(x,k) = (x',k')$, and (3) $f \in$ para-FAC^0. The more general para-AC^0 disjunctive truth table reduction, written $Q \leq_{\mathrm{dtt}}^{\text{para-AC}^0} Q'$, is defined similarly, only f maps (x,k) to a sequence $(x_1,k_1),\dots,(x_\ell,k_\ell)$ of instances such that (1′) $(x,k) \in Q$ iff there is an $i \in \{1,\dots,\ell\}$ with $(x_i,k_i) \in Q'$ and (2′) $k_i \leq g(k)$ holds for all $i \in \{1,\dots,\ell\}$. Both para-AC^0 and para-$\mathrm{AC}^{0\uparrow}$ are closed under $\leq_{\mathrm{m}}^{\text{para-AC}^0}$- and $\leq_{\mathrm{dtt}}^{\text{para-AC}^0}$-reductions.

3 Basic Graphs

Basic graphs, that is, undirected graphs without self-loops, are one of the simplest non-trivial logical structures one can imagine. Despite that, many NP-hard problems on graphs, like vertex cover, clique or dominating set, are NP-hard even for basic graphs. This also transfers in some sense to our setting: The "tractability frontier", the dividing line between the fragments which are tractable and those where we can express intractable problems, is the same for all graph classes we consider. However, when we shift our attention to the complexity landscape inside the tractable fragments, we also see that the complexity of the logical structure has an impact on the complexity of the problems we can define: Basic, undirected, and directed graphs all have provably distinct complexity characterizations.

We begin by stating the main theorem of the section, the complexity classification for basic graphs. In the rest of the section, we show the upper and lower bounds that lead to this classification.

▶ **Theorem 3.1** (Complexity Trichotomy for p-$\mathrm{VD}_{\mathrm{basic}}(p)$). *Let $p \in \{a,e\}^*$ be a pattern.*

1. p-$\mathrm{VD}_{\mathrm{basic}}(p) \subseteq$ para-AC^0, *if $p \preceq eae$ or $p \preceq e^*a^*$.*
2. p-$\mathrm{VD}_{\mathrm{basic}}(p) \subseteq$ para-$\mathrm{AC}^{0\uparrow}$ *but* p-$\mathrm{VD}_{\mathrm{basic}}(p) \not\subseteq$ para-AC^0, *if $eeae \preceq p$, $aae \preceq p$ or $aee \preceq p$ holds, but also still $p \preceq e^*a^*e^*$.*
3. p-$\mathrm{VD}_{\mathrm{basic}}(p)$ *contains a* W[2]*-hard problem, if $aea \preceq p$.*

The theorem covers all possible patterns. It follows from the following lemma, where we state the individual complexity characterizations we will prove:

▶ **Lemma 3.2** (Detailed Bounds for p-$\mathrm{VD}_{\mathrm{basic}}(p)$).

1. p-$\mathrm{VD}_{\mathrm{basic}}(eae) \subseteq$ para-AC^0.
2. p-$\mathrm{VD}_{\mathrm{basic}}(e^*a^*) \subseteq$ p-$\mathrm{VD}_{\mathrm{arb}}(e^*a^*) \subseteq$ para-AC^0.
3. p-$\mathrm{VD}_{\mathrm{basic}}(e^*a^*e^*) \subseteq$ p-$\mathrm{VD}_{\mathrm{arb}}(e^*a^*e^*) \subseteq$ para-$\mathrm{AC}^{0\uparrow}$.
4. p-$\mathrm{VD}_{\mathrm{basic}}(eeae)$ *contains a problem not in* para-AC^0.
5. p-$\mathrm{VD}_{\mathrm{basic}}(aae)$ *contains a problem not in* para-AC^0.
6. p-$\mathrm{VD}_{\mathrm{basic}}(aee)$ *contains a problem not in* para-AC^0.
7. p-$\mathrm{VD}_{\mathrm{basic}}(aea)$ *contains a* W[2]*-hard problem.*

Notice that in particular, we know unconditionally that $\mathrm{W}[2] \not\subseteq \text{para-AC}^0$, and, hence, a W[2]-hard problem cannot lie in para-AC^0. It is furthermore widely conjectured that $\mathrm{W}[2] \not\subseteq \text{para-AC}^{0\uparrow}$, as $\text{para-AC}^{0\uparrow} \subseteq \mathrm{FPT}$. We devote the rest of this section to proving the individual items of the lemma.

Upper Bounds. Previous work by Bannach et al. [3] showed that in the weighted definability setting, formulas with the pattern ae already suffice to describe W[2]-hard problems. We now show that the situation is more favorable in the vertex deletion setting, which is a special case of weighted definability: All problems in $\text{p-VD}_{\text{basic}}(e^*a^*e^*)$ are tractable and the problems in $\text{p-VD}_{\text{basic}}(e^*a^*)$ and in $\text{p-VD}_{\text{basic}}(eae)$ are even in para-AC^0, the smallest class commonly considered in parameterized complexity. We start with the last claim:

▶ **Lemma 3.3** (▼ [4]). $\text{p-VD}_{\text{basic}}(eae) \subseteq \text{para-AC}^0$.

Proof idea. To check whether we can delete at most k vertices to satisfy a formula with prefix pattern eae, we first branch over the possible assignments to the first existentially quantified variable. Now, the neighborhood of this variable induces a 2-coloring on the rest of the graph. For the rest of the prefix, ae, we prove that a vertex has to be deleted if and only if there is no special set of constant size, called *stable set*. This can all be checked in para-AC^0. ◀

Since the algorithms used to prove the next two upper bounds do not make use of the fact that the input structure is a basic graph, we prove them for arbitrary input structures.

▶ **Lemma 3.4.** $\text{p-VD}_{\text{arb}}(e^*a^*) \subseteq \text{para-AC}^0$.

Proof. For a given formula ϕ of the form $\exists x_1 \cdots \exists x_f \forall y_1 \cdots \forall y_g(\psi)$ for a quantifier-free formula ψ, we show that $\text{p-VD}_{\text{arb}}(\phi) \leq_{\text{dtt}}^{\text{para-AC}^0}$ p-g-HITTING-SET, where the hitting set problem is defined as shown below. Since p-g-HITTING-SET is known [5] to lie in para-AC^0, we get the claim.

▶ **Problem 3.5** (p-d-HITTING-SET for fixed $d \in \mathbb{N}$).

Instance: *A universe U and a set E of subsets $e \subseteq U$ (called* hyperedges*) with $|e| \leq d$ for all $e \in E$, and a number k.*

Parameter: k.

Question: *Is there a* hitting set *$X \subseteq V$, meaning that $X \cap e \neq \emptyset$ holds for all $e \in E$, with $|X| \leq k$?*

For an arbitrary input structure $\mathcal{A}$ with universe A, we proceed as follows: For the existentially bound variables x_1 to x_f we consider all possible assignments to them in parallel. For each of these, we prepare a query to the hitting set problem, resulting in n^f queries in total. For a given assignment, which fixes each x_i to some constant c_i, replace each occurrence of x_i in ϕ by c_i. Build a hitting set instance H as follows: The universe is $A \setminus \{c_1, \dots, c_f\}$. For each assignment $(d_1, \dots, d_g)$ of to the g universally quantified variables, check if the formula ψ is true, that is, whether $\mathcal{A} \models \psi(c_1, \dots, c_f, d_1, \dots, d_g)$. If this is not the case, add the hyperedge $\{d_1, \dots, d_g\} \setminus \{c_1, \dots, c_f\}$ to make sure that at least one element is deleted from the universe of $\mathcal{A}$ that cause this particular violation. If $\{d_1, \dots, d_g\} \setminus \{c_1, \dots, c_f\}$ is empty, an empty hyperedge is generated and the hitting set solver correctly rejects the input.

We claim that $\mathcal{A} \in \text{p-VD}_{\text{arb}}(\phi)$ iff for at least one of the constructed H we have $(H, k) \in$ p-g-HITTING-SET: For the first direction, let S with $|S| \leq k$ be the elements of $\mathcal{A}$'s universe that we can delete, that is, for which $\mathcal{A} \setminus S \models \phi$. Then there are constants $(c_1, \dots, c_f)$ that we can assign to the existentially bound variables such that $\mathcal{A} \setminus S \models \forall y_1 \cdots \forall y_g \big(\psi(c_1, \dots, c_f, y_1, \dots, y_g)\big)$. But, then, S is a hitting set of the instance corresponding to these constants: If there were an edge $e \subseteq A$ with $e \cap S = \emptyset$ in the hitting set instance, there would be an assignment to the y_i to elements in $A \setminus S$ that makes ψ false, violating the assumption.

For the other direction, let X with $|X| \le k$ be the solution of one of the produced hitting set instances with $(H, k) \in$ p-g-HITTING-SET (at least one must exist). Then $\mathcal{A} \setminus X \models \phi$, since we can assign the existentially bound variables to the values that correspond to H (which will not be in X by construction) and there can be no assignment to the universally quantified variables that makes ψ false as any assignment where this would be case is hit by X by construction and, thus, at least one element of the tuple that causes the violation gets removed in $\mathcal{A} \setminus X$. ◀

▶ **Lemma 3.6.** p-$\mathrm{VD}_{\mathrm{arb}}(e^*a^*e^*) \subseteq$ para-$\mathrm{AC}^{0\uparrow}$.

Proof. Let ϕ be fixed and of the form $\exists x_1 \cdots \exists x_f \forall y_1 \cdots \forall y_g \exists z_1 \cdots \exists z_h(\psi)$ for a quantifier-free formula ψ. We describe a para-$\mathrm{AC}^{0\uparrow}$-algorithm that, given an arbitrary input structure $\mathcal{A}$ with universe A, decides whether there is a set S with $|S| \le k$ such that $\mathcal{A} \setminus S \models \phi$.

Now, we have for each assignment to the universally quantified variables a witness which is bound by the block of h existential quantifiers. The problem compared to the e^*a^*-fragment is that by the deletion of elements, we could potentially destroy witnesses needed to satisfy other assignments. Because of this, we use a direct search tree algorithm to resolve violations of the universal quantifiers.

In detail, we once more consider all possible assignments $(c_1, \dots, c_f)$ to the x_i in parallel. Then we use k layers to find and resolve violations: At the start of each layer, we will already have fixed a set D of vertices that we wish to delete, starting in the first layer with $D = \emptyset$. Then in the layer, we find the (for example, lexicographically) first assignment of the y_i to elements $(d_1, \dots, d_f)$ that all lie in $A \setminus D$ for which we cannot find an assignment of the z_i to elements $(e_1, \dots, e_h)$ in $A \setminus D$ such that $\mathcal{A} \setminus D \models \psi(c_1, \dots, c_f, d_1, \dots, d_g, e_1, \dots, e_h)$. When we cannot find such an assignment, we can accept since we have found a D for which $\mathcal{A} \setminus D \models \phi$ holds. Otherwise, we *have to* delete one of the elements in $\{d_1, \dots, d_g\} \setminus \{c_1, \dots, c_f\}$ to make the formula true, so we branch over these at most g possibilities, entering g copies of the next layers, where the ith copy starts with $D \cup \{d_i\}$.

Since the block of universal quantifiers has constant length, the number of branches in each level of the search tree is constant, so the total size of the search tree is at most g^k. The depth of the search tree is bounded by the number of vertices we can delete, which is our parameter. In total, we get a para-$\mathrm{AC}^{0\uparrow}$ circuit. ◀

Lower Bounds. We now go on to show the lower bounds claimed in Lemma 3.2. The next lemmas all follow the same rough strategy: To show that some problems that can be expressed in the given fragments are (unconditionally) not in para-AC^0, we reduce from a variant of the reachability problem. In contrast, the last lower bound is obtained via a reduction from p-SET-COVER, and improves a result from Fomin et al. [17]. They establish that there is a formula $\phi \in \Pi_3$, such that p-$\mathrm{VD}_{\mathrm{basic}}(\phi)$ is W[2]-hard. In terms of patterns, the formula they construct has the pattern $a^5e^{26}a$. We show that there is a formula with pattern aea for which this holds.

The reachability problem that will be central for the following lower bounds is:

▶ **Problem 3.7** (p-MATCHED-REACH).

Instance: *A directed layered graph G with vertex set $\{1, \dots, n\} \times \{1, \dots, k\}$, where the ith layer is $V_i \coloneqq \{1, \dots, n\} \times \{i\}$, such that for each $i \in \{1, \dots, k-1\}$ the edges point to the next layer and they form a perfect matching between V_i and V_{i+1}; and two designated vertices $s \in V_1$ and $t \in V_k$.*

Parameter: k.

Question: *Is t reachable from s in G?*

(We require that in the encoding of G the vertex "addresses" (i, l) are given explicitly as, say, pairs of binary numbers, so that even a AC^0 circuit will have no trouble determining which vertices belong to a layer V_i or what the number k of layers is.)

Observe that the input instance can be alternatively described as a collection of n directed paths, each of length k. We call the paths in this graph *original paths* with *original vertices and edges*. We call the vertices in the layers V_1 and V_k the *outer vertices* and the vertices in the layers V_i for $i \in \{2, \dots, k-1\}$ the *inner vertices*. The reductions add vertices and edges to the graphs, which will be referred to as the *new vertices and edges* (and will be indicated in yellow in figures).

▶ **Fact 3.8** ([3]). p-MATCHED-REACH $\notin$ para-AC^0 *and, thus, for any problem Q with* p-MATCHED-REACH $\leq_{\mathrm{m}}^{\text{para-AC}^0} Q$ *we have $Q \notin$ para-AC^0.*

The proof of every lemma using a reduction from the matched reachability problem will consist of four parts:

1. The construction of a formula ϕ with the quantifier pattern p given in the lemma.
2. The construction of the instance for the vertex deletion problem (G', k') from the input instance of the matched reachability problem (G, s, t) (typically by adding new vertices and edges).
3. Showing $(G, s, t) \in$ p-MATCHED-REACH implies $(G', k') \in \text{p-VD}_{\text{basic}}(\phi)$, called the *forward direction.*
4. Showing $(G', k') \in \text{p-VD}_{\text{basic}}(\phi)$ implies $(G, s, t) \in$ p-MATCHED-REACH, called the *backward direction.*

We present the application of the above steps in detail in the following lemma. In subsequent lemmas, which follow the same line of arguments, but with appropriate variations in the constructions and correctness proofs, we only highlight the differences.

▶ **Lemma 3.9.** $\text{p-VD}_{\text{basic}}(eeae) \not\subseteq \text{para-AC}^0$.

Proof. We want there to be a deletion strategy for (G', k') iff in the instance (G, s, t), the vertices s and t lie on the same original path. We take $k' = k$, the number of layers in G, and construct a graph G' from G by adding two special vertices c_1 and c_2, and regard the adjacency of every vertex on the original paths to the vertices c_1 and c_2 as a 3-coloring with colors $i \in \{0, 1, 2\}$. We then add appropriate gadgets at the start and the end of each original path, with special gadgets being added at s and at t (although, in this proof, their "special gadgets" are just the empty gadget).

The formula. Consider the following formulas, where ϕ_a specifies that every vertex that is neither c_1 nor c_2 should be connected in a certain way to them, and ϕ_b asks that every vertex of color i should have a neighbor of color $(i-1) \pmod 3$. We encode the color 0 with $(x \sim c_1 \land x \not\sim c_2)$, the color 1 with $(x \not\sim c_1 \land x \sim c_2)$, and the color 2 with $(x \sim c_1 \land x \sim c_2)$.

$$\begin{aligned}
\phi_a(c_1,c_2,x) &= (c_1 \neq c_2) \wedge (c_1 \sim x \vee c_2 \sim x)\\
\phi_b(c_1,c_2,x,y) &= x \sim y \wedge ((x \sim c_1 \wedge x \sim c_2) \to (y \not\sim c_1 \wedge y \sim c_2))\\
&\quad \wedge ((x \not\sim c_1 \wedge x \sim c_2) \to (y \sim c_1 \wedge y \not\sim c_2))\\
&\quad \wedge ((x \sim c_1 \wedge x \not\sim c_2) \to (y \sim c_1 \wedge y \sim c_2))\\
\phi_{3.9} &= \exists c_1 \exists c_2 \forall x \exists y \big(((x \neq c_1) \wedge (x \neq c_2)) \to\\
&\qquad \big((y \neq c_1) \wedge (y \neq c_2) \wedge\\
&\qquad \phi_a(c_1,c_2,x) \wedge \phi_b(c_1,c_2,x,y)\big)\big)
\end{aligned}$$

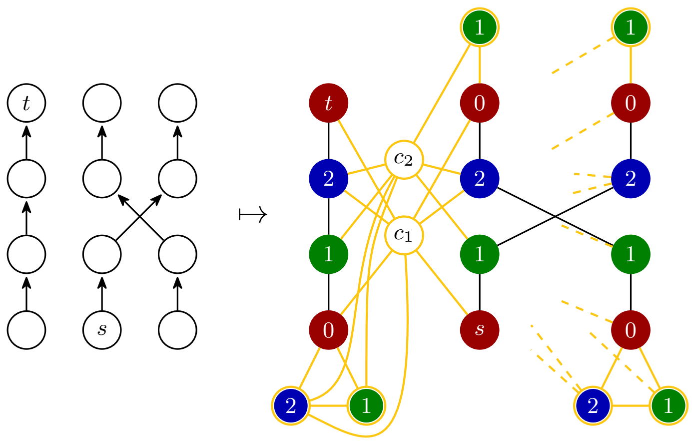

Figure 1 Example for the reduction from Lemma 3.9. The input graph on the left is a directed layered graph with perfect matchings between consecutive layers. The reduction maps it to the undirected graph shown right by forgetting about the direction of edges, by adding gadgets at the beginnings and ends of the paths (with special empty gadgets at s and t), and by adding two special vertices c_1 and c_2 that are connected in three different ways to the other vertices, corresponding to three different colors. Newly added vertices and edges are indicated in yellow. Note that the indicated colors, numbers, and labels are not part of the output, they are only for explaining how the formula interprets the connection of the vertices to c_1 and c_2.

The reduction. On input (G, s, t) the reduction first checks that the graph is, indeed, a layered graph with perfect matchings between consecutive levels (this can easily be done by an AC^0 circuit due to the way we encode G). Then, we let k' be the number k of layers in $G = (V, \sim)$ and construct $G' = (V', \sim')$ by first forgetting about the direction of the edges (making the graph undirected). We then add the following gadgets:

1. At each end $v \in V_k$ of a path, *except for* $v = t$, we add a vertex v' to V' and connect v to v', so $v \sim' v'$. Let V_{k+1} be the set of all new vertices added in this way. The gadget for $t \in V_k$ is empty: We do not add anything.
2. At each beginning $v \in V_1$ of a path, *except for* $v = s$, add two vertices v' and v'' to V' and connect the three vertices to a triangle, so $v \sim' v' \sim' v'' \sim' v$. Let V_0 contain all vertices v' added in this way and let V_{-1} contain all vertices v'' added in this way. Once more, the special gadget for $s \in V_1$ is just the empty gadget.
3. Finally, we add two further vertices c_1 and c_2 and connect them to the other vertices as follows: For $v \in V_i$ with $i \in \{-1, 0, 1, 2, \ldots, k+1\}$:

- If $i \equiv 0 \mod 3$, let $c_1 \sim' v$.
- If $i \equiv 1 \mod 3$, let $c_2 \sim' v$.
- If $i \equiv 2 \mod 3$, let $c_1 \sim' v$ and $c_2 \sim' v$.

An example for the reduction is depicted in Figure 1. We claim that through this construction, the instance (G', k') is in $\text{p-VD}_{\text{basic}}(\phi_{3.9})$ iff the input graph with vertices s and t is in p-MATCHED-REACH:

Forward direction. Suppose that $(G, s, t) \in$ p-MATCHED-REACH. We show that $(G', k') \in \text{p-VD}_{\text{basic}}(\phi_{3.9})$: In input G', just delete every vertex in the original s-t-path. Then every vertex $v \in V_i$ for $i \in \{2, \dots, k\}$ has its predecessor in the original path as a neighbor, and the predecessor has the previous color regarding the ordering. Furthermore, every vertex $v \in V_1$ is part of a triangle where the three vertices each have a different color, so every one of these three vertices has a neighbor of the previous color.

Backward direction. Suppose that $(G', k') \in \text{p-VD}_{\text{basic}}(\phi_{3.9})$. We show that $(G, s, t) \in$ p-MATCHED-REACH. By assumption, there is a set D of size $|D| \leq k = k'$ such that $G' \setminus D$ is a model of $\phi_{3.9}$. Observe that $c_1 \notin D$ and $c_2 \notin D$ must hold since they are the only vertices satisfying the formula part ϕ_a, which requires that there are two different vertices that are connect to everyone else. On the other hand, we *have to* delete s, since by construction, it has no neighbor with the previous color (s has color 0, the successor of s has color 1). But, now, the successor of s has no neighbor of the previous color, so we have to delete it as well. We have to continue for the whole original path of s, so D has to contain at least the vertices on the original path starting at s, which encompasses k vertices. If the last vertex $v \in V_k$ on the original path starting at s is not t (that is, if t is not reachable from s), then there is another vertex $v' \in V_{k+1}$ with $v \sim' v'$ and we also have to delete v', contradicting the assumption that we only have to delete k vertices. Thus, t must be reachable from s. ◀

▶ **Lemma 3.10** (▼ [4]). $\text{p-VD}_{\text{basic}}(aae) \not\subseteq \text{para-AC}^0$.

▶ **Lemma 3.11** (▼ [4]). $\text{p-VD}_{\text{basic}}(aee) \not\subseteq \text{para-AC}^0$.

▶ **Lemma 3.12** (▼ [4]). $\text{p-VD}_{\text{basic}}(aea)$ *contains a* W[2]*-hard problem.*

4 Undirected Graphs

Whether allowing self-loops has an impact on the complexity of the problems is hard to predict: While in the setting of Fomin et al. [17], the same dichotomy arises for basic and undirected graphs, in the setting of weighted definability considered by Bannach et al. [3], one class of problems jumps from being contained in para-AC^0 to containing para-NP-hard problems just by allowing self-loops. In our setting, we get an *intermediate* blow-up of the complexities by allowing self-loops: While the tractability frontier stays the same, the frontier of fragments that are solvable in para-AC^0 shifts.

Let us now classify the complexity of vertex deletion problems on undirected graphs. We can use some of the upper and lower bounds established in the section before, and only consider the differences.

▶ **Theorem 4.1** (Complexity Trichotomy for $\text{p-VD}_{\text{undir}}(p)$). *Let $p \in \{a, e\}^*$ be a pattern.*

1. $\text{p-VD}_{\text{undir}}(p) \subseteq \text{para-AC}^0$, *if $p \preceq ae$ or $p \preceq e^*a^*$.*
2. $\text{p-VD}_{\text{undir}}(p) \subseteq \text{para-AC}^{0\uparrow}$ *but* $\text{p-VD}_{\text{undir}}(p) \not\subseteq \text{para-AC}^0$, *if one of $eae \preceq p$, $aae \preceq p$ or $aee \preceq p$ holds, but still $p \preceq e^*a^*e^*$ holds.*
3. $\text{p-VD}_{\text{undir}}(p)$ *contains a* W[2]*-hard problem, if $aea \preceq p$.*

▶ **Lemma 4.2.**
1. $\text{p-VD}_{\text{undir}}(ae) \subseteq \text{para-AC}^0$.
2. $\text{p-VD}_{\text{undir}}(e^*a^*) \subseteq \text{para-AC}^0$.
3. $\text{p-VD}_{\text{undir}}(e^*a^*e^*) \subseteq \text{para-AC}^{0\uparrow}$.
4. $\text{p-VD}_{\text{undir}}(eae)$ *contains a problem not in* para-AC^0.
5. $\text{p-VD}_{\text{undir}}(aae)$ *contains a problem not in* para-AC^0.
6. $\text{p-VD}_{\text{undir}}(aee)$ *contains a problem not in* para-AC^0.
7. $\text{p-VD}_{\text{undir}}(aea)$ *contains a* $\text{W}[2]$*-hard problem.*

Proof. Item 1 is proven below in Lemma 4.3. Items 2 and 3 follow directly from Lemmas 3.4 and 3.6. Item 4 is proven below in Lemma 4.4, Item 5 follows from Lemma 3.10, Item 6 from Lemma 3.11 and Item 7 from Lemma 3.12. ◀

▶ **Lemma 4.3** (▼ [4]). $\text{p-VD}_{\text{undir}}(ae) \subseteq \text{para-AC}^0$.

▶ **Lemma 4.4** (▼ [4]). $\text{p-VD}_{\text{undir}}(eae) \not\subseteq \text{para-AC}^0$.

5 Directed Graphs and Arbitrary Structures

The final class of logical structures we investigate in this paper are directed graphs. Interestingly, from the viewpoint of quantifier patterns, this class of structures is as complex as arbitrary logical structures.

▶ **Theorem 5.1** (Complexity Trichotomy for $\text{p-VD}_{\text{dir}}(p)$). *Let $p \in \{a, e\}^*$ be a pattern.*
1. $\text{p-VD}_{\text{dir}}(p) \subseteq \text{para-AC}^0$, *if* $p \preceq e^*a^*$.
2. $\text{p-VD}_{\text{dir}}(p) \subseteq \text{para-AC}^{0\uparrow}$ *but* $\text{p-VD}_{\text{dir}}(p) \not\subseteq \text{para-AC}^0$, *if* $ae \preceq p \preceq e^*a^*e^*$.
3. $\text{p-VD}_{\text{dir}}(p)$ *contains a* $\text{W}[2]$*-hard problem, if* $aea \preceq p$.

▶ **Lemma 5.2.**
1. $\text{p-VD}_{\text{dir}}(e^*a^*) \subseteq \text{para-AC}^0$.
2. $\text{p-VD}_{\text{dir}}(e^*a^*e^*) \subseteq \text{para-AC}^{0\uparrow}$.
3. $\text{p-VD}_{\text{dir}}(ae)$ *contains a problem not in* para-AC^0.
4. $\text{p-VD}_{\text{dir}}(aea)$ *contains a* $\text{W}[2]$*-hard problem.*

Proof. Items 1 and 2 follow directly from Lemmas 3.4 and 3.6. Item 3 is shown in Lemma 5.3, and Item 4 follows from Lemma 3.12. ◀

▶ **Lemma 5.3** (▼ [4]). $\text{p-VD}_{\text{dir}}(ae) \not\subseteq \text{para-AC}^0$.

6 Conclusion

In this paper, we fully classified the parameterized complexity of vertex deletion problems where the target property is expressible by first-order formulas and where the inputs are basic graphs, undirected graphs, directed graphs, or arbitrary logical structures. The classification is based on the quantifier patterns of the formulas, and sheds additional light on the complexity properties that emerge from these patterns: We have seen that while the tractability barrier is the same for all logical structures, $\text{p-VD}_{\text{basic}}(e^*a^*e^*)$, $\text{p-VD}_{\text{undir}}(e^*a^*e^*)$, $\text{p-VD}_{\text{dir}}(e^*a^*e^*)$ and $\text{p-VD}_{\text{arb}}(e^*a^*e^*)$ all being tractable and $\text{p-VD}_{\text{basic}}(aea)$, $\text{p-VD}_{\text{undir}}(aea)$, $\text{p-VD}_{\text{dir}}(aea)$ as well as $\text{p-VD}_{\text{arb}}(aea)$ all containing intractable problems, in the tractable cases, basic, undirected and directed graphs have provably different complexities, the latter coinciding with arbitrary structures.

The granularity we gained with the viewpoint of quantifier patterns could be useful to examine the complexity of vertex deletions problems where the property is given by a formula of a more expressive logic: For both *monadic second-order logic* (MSO) and *existential second-order logic* (ESO), even the model checking problem becomes NP-hard. This would allow us to express many more natural problems such as feedback vertex set, that have no obvious formalization as a vertex deletion problem to plain FO-properties. Similarly, we could allow extensions such as transitive closure or fixed point operators.

Compared to previous work on weighted definability, where the objective is to instantiate a free set variable with *at most, exactly,* or *at least* k elements such that a formula holds, we only considered deleting *at most* k elements. How does the complexity of vertex deletion problems change, if we *have to* delete exactly k elements – or, for that matter, *at least* k elements?

References

1 Faisal N. Abu-Khzam, Rebecca L. Collins, Michael R. Fellows, Michael A. Langston, W. Henry Suters, and Christopher T. Symons. Kernelization algorithms for the vertex cover problem: Theory and experiments. In *Proceedings of the Sixth Workshop on Algorithm Engineering and Experiments and the First Workshop on Analytic Algorithmics and Combinatorics, New Orleans, LA, USA, January 10, 2004*, pages 62–69, 2004.

2 Akanksha Agrawal and M. S. Ramanujan. Distance from triviality 2.0: Hybrid parameterizations. In *Combinatorial Algorithms - 33rd International Workshop, IWOCA 2022, Trier, Germany, June 7–9, 2022, Proceedings*, pages 3–20, 2022. doi:10.1007/978-3-031-06678-8_1.

3 Max Bannach, Florian Chudigiewitsch, and Till Tantau. Existential second-order logic over graphs: Parameterized complexity. In Neeldhara Misra and Magnus Wahlström, editors, *18th International Symposium on Parameterized and Exact Computation, IPEC 2023, September 6-8, 2023, Amsterdam, The Netherlands*, volume 285 of *LIPIcs*, pages 3:1–3:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.IPEC.2023.3.

4 Max Bannach, Florian Chudigiewitsch, and Till Tantau. On the descriptive complexity of vertex deletion problems. Technical Report abs/2406.18299, Cornell University, 2023. arXiv:2406.18299.

5 Max Bannach and Till Tantau. Computing hitting set kernels by AC^0-circuits. *Theory of Computing Systems*, 64(3):374–399, 2020. doi:10.1007/s00224-019-09941-z.

6 Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem.* Perspectives in Mathematical Logic. Springer, 1997.

7 Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *Journal of the ACM*, 55(5), November 2008. doi:10.1145/1411509.1411511.

8 Christophe Crespelle, Pål Grønås Drange, Fedor V. Fomin, and Petr Golovach. A survey of parameterized algorithms and the complexity of edge modification. *Computer Science Review*, 48:100556, 2023. doi:10.1016/j.cosrev.2023.100556.

9 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015. doi:10.1007/978-3-319-21275-3.

10 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.

11 Maël Dumas and Anthony Perez. An improved kernelization algorithm for trivially perfect editing. In *18th International Symposium on Parameterized and Exact Computation, IPEC 2023, September 6–8, 2023, Amsterdam, The Netherlands*, pages 15:1–15:17, 2023. doi:10.4230/LIPICS.IPEC.2023.15.

12 Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory.* Springer, 2nd edition, 2005. doi:10.1007/3-540-28788-4.

13 Thomas Eiter, Yuri Gurevich, and Georg Gottlob. Existential second-order logic over strings. *Journal of the ACM*, 47(1):77–131, 2000. doi:10.1145/331605.331609.

14 Damir Ferizovic, Demian Hespe, Sebastian Lamm, Matthias Mnich, Christian Schulz, and Darren Strash. Engineering kernelization for maximum cut. In *Proceedings of the Symposium on Algorithm Engineering and Experiments, ALENEX 2020, Salt Lake City, UT, USA, January 5–6, 2020*, pages 27–41, 2020. doi:10.1137/1.9781611976007.3.

15 Aleksander Figiel, Vincent Froese, André Nichterlein, and Rolf Niedermeier. There and back again: On applying data reduction rules by undoing others. In *30th Annual European Symposium on Algorithms, ESA 2022, September 5–9, 2022, Berlin/Potsdam, Germany*, pages 53:1–53:15, 2022. doi:10.4230/LIPICS.ESA.2022.53.

16 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006. doi:10.1007/3-540-29953-X.

17 Fedor V. Fomin, Petr A. Golovach, and Dimitrios M. Thilikos. On the parameterized complexity of graph modification to first-order logic properties. *Theory of Computing Systems*, 64(2):251–271, 2020. doi:10.1007/s00224-019-09938-8.

18 Fedor V. Fomin, Petr A. Golovach, and Dimitrios M. Thilikos. Parameterized complexity of elimination distance to first-order logic properties. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 – July 2, 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470540.

19 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Hitting topological minors is FPT. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 1317–1326, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3357713.3384318.

20 Georg Gottlob, Phokion G. Kolaitis, and Thomas Schwentick. Existential second-order logic over graphs: Charting the tractability frontier. *Journal of the ACM*, 51(2):312–362, 2004. doi:10.1145/972639.972646.

21 Jiong Guo, Falk Hüffner, and Rolf Niedermeier. A structural view on parameterizing problems: Distance from triviality. In *Parameterized and Exact Computation, First International Workshop, IWPEC 2004, Bergen, Norway, September 14–17, 2004, Proceedings*, pages 162–173, 2004. doi:10.1007/978-3-540-28639-4_15.

22 Demian Hespe, Sebastian Lamm, Christian Schulz, and Darren Strash. Wegotyoucovered: The winning solver from the PACE 2019 challenge, vertex cover track. In *Proceedings of the SIAM Workshop on Combinatorial Scientific Computing, CSC 2020, Seattle, USA, February 11–13, 2020*, pages 1–11, 2020. doi:10.1137/1.9781611976229.1.

23 Demian Hespe, Christian Schulz, and Darren Strash. Scalable kernelization for maximum independent sets. *ACM Journal of Experimental Algorithmics*, 24(1):1.16:1–1.16:22, 2019. doi:10.1145/3355502.

24 M. S. Krishnamoorthy and Narsingh Deo. Node-deletion NP-complete problems. *SIAM Journal on Computing*, 8(4):619–625, 1979. doi:10.1137/0208049.

25 John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980. doi:10.1016/0022-0000(80)90060-4.

26 Jason Li and Jesper Nederlof. Detecting feedback vertex sets of size k in O*(2.7k) time. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 971–989, 2020. doi:10.1137/1.9781611975994.58.

27 Till Tantau. Existential second-order logic over graphs: A complete complexity-theoretic classification. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4–7, 2015, Garching, Germany*, pages 703–715, 2015. doi:10.4230/LIPIcs.STACS.2015.703.

28 Mihalis Yannakakis. Node-and edge-deletion NP-complete problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 253–264, New York, NY, USA, 1978. Association for Computing Machinery. doi:10.1145/800133.804355.