Parameterized Complexity of Elimination Distance to First-Order Logic Properties

FEDOR V. FOMIN and PETR A. GOLOVACH, Department of Informatics, University of Bergen, Norway
DIMITRIOS M. THILIKOS, LIRMM, Univ Montpellier, CNRS, Montpellier, France

The *elimination distance* to some target graph property $\mathcal{P}$ is a general graph modification parameter introduced by Bulian and Dawar. We initiate the study of elimination distances to graph properties expressible in first-order logic. We delimit the problem's fixed-parameter tractability by identifying sufficient and necessary conditions on the structure of prefixes of first-order logic formulas. Our main result is the following meta-theorem: For every graph property $\mathcal{P}$ expressible by a first order-logic formula $\varphi \in \Sigma_3$, that is, of the form

$$\varphi = \exists x_1 \exists x_2 \cdots \exists x_r \ \ \forall y_1 \forall y_2 \cdots \forall y_s \ \ \exists z_1 \exists z_2 \cdots \exists z_t \ \psi,$$

where ψ is a quantifier-free first-order formula, checking whether the elimination distance of a graph to $\mathcal{P}$ does not exceed k, is *fixed-parameter tractable* parameterized by k. Properties of graphs expressible by formulas from Σ_3 include being of bounded degree, excluding a forbidden subgraph, or containing a bounded dominating set. We complement this theorem by showing that such a general statement does not hold for formulas with even slightly more expressive prefix structure: There are formulas $\varphi \in \Pi_3$, for which computing elimination distance is W[2]-hard.

17

CCS Concepts: • **Theory of computation** → **Complexity theory and logic**; **Parameterized complexity and exact algorithms;**

Additional Key Words and Phrases: First-order logic, elimination distance, parameterized complexity, descriptive complexity

ACM Reference format:
Fedor V. Fomin, Petr A. Golovach, and Dimitrios M. Thilikos. 2022. Parameterized Complexity of Elimination Distance to First-Order Logic Properties. *ACM Trans. Comput. Logic* 23, 3, Article 17 (April 2022), 35 pages.
https://doi.org/10.1145/3517129

An extended abstract of this article appeared in the proceedings of LICS 2021. The two first authors have been supported by the Research Council of Norway via the project BWCA (314528). The third author has been supported by the ANR projects DEMOGRAPH (ANR-16-CE40-0028), ESIGMA (ANR-17-CE23-0010), and the French-German Collaboration ANR/DFG Project UTMA (ANR-20-CE92-0027).
Authors' addresses: F. V. Fomin and P. A. Golovach, Department of Informatics, University of Bergen, PB 7803, Bergen, 5020, Norway; emails: {fedor.fomin, petr.golovach}@uib.no; D. M. Thilikos, LIRMM, Univ Montpellier, CNRS, Montpellier, France, 161 rue Ada, Montpellier Cedex 5, 34095, France; email: sedthilk@thilikos.info.

1529-3785/2022/04-ART17 $15.00
https://doi.org/10.1145/3517129

1 INTRODUCTION

One of the successful concepts in parameterized complexity is the "distance from triviality" [20]. Roughly speaking, a parameter can measure the "distance" of the given instance from an instance that is solvable efficiently and then exploit such a distance algorithmically. In graph problems, a standard measure of distance from triviality is the *vertex deletion distance* to some specific graph property $\mathcal{P}$. That is, the minimum number of vertices whose deletion results in a graph in $\mathcal{P}$. An interesting alternative to vertex deletion distance, called *elimination distance* was introduced by Bulian and Dawar [5] in their study of the parameterized complexity of the graph isomorphism problem. The *elimination distance* of a graph G to graph property $\mathcal{P}$ is

$$\mathrm{ed}_{\mathcal{P}}(G) = \begin{cases} 0, & \text{if } G \in \mathcal{P}, \\ 1 + \min_{v \in V(G)} \mathrm{ed}_{\mathcal{P}}(G - v), & \text{if } G \notin \mathcal{P} \text{ and } G \text{ is connected}, \\ \max\{\mathrm{ed}_{\mathcal{P}}(C) \mid C \text{ is a component of } G\}, & \text{otherwise}. \end{cases}$$

Arguably, elimination distance can be seen as a non-deterministic version of vertex deletion distance, where the source of non-determinism is connectivity: Each vertex removal creates connected components and the elimination should be applied to each one of them as an independent vertex deletion scenario. In the most simple case where $\mathcal{P}$ is the property of being edgeless, vertex deletion distance to $\mathcal{P}$ generates *vertex cover*, while the elimination distance to $\mathcal{P}$ generates *tree-depth* [28].

In their follow-up work, Bulian and Dawar [6] proved that deciding whether a given n-vertex graph has elimination distance at most k to any minor-closed property of graphs can be done by an algorithm running in time $f(k) \cdot n^{O(1)}$ (that is an FPT-algorithm), and thus is fixed-parameter tractable parameterized by k. In the same paper, Bulian and Dawar [6] asked whether computing the elimination distance to graphs of bounded degree is fixed-parameter tractable.

The problem. The question of Bulian and Dawar is the departure point of our study. Every graph property characterized by a finite set of forbidden induced subgraphs (and thus the bounded degree property as well) is **first-order logic definable (FOL-definable)**, i.e., there is an FOL formula φ where $\mathcal{P} = \{G \mid G \models \varphi\}$. It is well-known that Model Checking for an FOL formula φ, that is deciding whether $G \models \varphi$, can be done in time $n^{O(|\varphi|)}$. It is also easy to design a backtracking algorithm following the definition of the elimination distance that, in time $n^{O(k)} \cdot n^{O(|\varphi|)}$, decides whether the elimination distance to a property expressible by an FOL formula φ is at most k. Thus, for every FOL formula φ, the problem asking, given as input a graph G and a non-negative integer k, whether the elimination distance from G to $\mathcal{P}_\varphi := \{G \mid G \models \varphi\}$ is k is in the parameterized complexity class XP (when parameterized by k). This brings us to the following question: What is the parameterized complexity of computing the elimination distance to FOL-definable properties?

Notice that the above general question could also be made for higher order logic-definable properties. In this direction, one may observe that there are formulas φ in **existential second-order logic (ESOL)** for which Model Checking for φ is already intractable: Such ESOL-definable problems are Hamiltonian Cycle or 3-Coloring that are NP-complete. This means that for the corresponding ESOL formulas φ the problem of checking whether $\mathrm{ed}_{\mathcal{P}_\varphi}(G) \leq k$, parameterized by k, is para-NP-complete. Motivated by this, we delimit our study to the framework of first-order logic where our parameterized problem is in XP for *every* FOL-formula. This permits us to set up the problem that we consider in this article, that is to completely determine the *prefix classes* of FOL that demark the parametric-tractability borders of elimination distance to FOL-definable properties (that is, FPT versus W-hardness).

The above question has been inspired by the study of Gottlob, Kolaitis, Schwentick in [19] who provided an analogous dichotomy result (P versus NP-complete) for ESOL-formulas, in several graph-theoretic contexts. They identified the set $\mathfrak{F}$ of prefix classes of ESOL such that, if $\varphi \in \mathfrak{F}$, then Model Checking for φ is polynomially solvable, while every prefix class not in $\mathfrak{F}$ contains some ESOL formula φ where Model Checking for φ is NP-complete.

Our results. We identify sufficient and necessary conditions on the structure of prefixes of first-order logic formulas demarcating tractability borders for computing the elimination distance.

Our main algorithmic contribution is the proof that computing the elimination distance to any graph property defined by a formula from Σ_3 is fixed-parameter tractable. We formally define prefix classes Π_i and Σ_i in the next section. For the purpose of this introduction, it is sufficient to know that every formula in $\varphi \in \Sigma_3$ can be written in the form

$$\varphi = \exists x_1 \exists x_2 \cdots \exists x_r \;\; \forall y_1 \forall y_2 \cdots \forall y_s \;\; \exists z_1 \exists z_2 \cdots \exists z_t \; \psi,$$

where ψ is a quantifier-free FOL-formula and r, s, t are non-negative integers.

Every graph property characterized by a finite set of forbidden subgraphs can be expressed by $\varphi \in \Sigma_3$. Actually, for this particular purpose, we can consider even more restricted formulas $\varphi \in \Pi_1 \subset \Sigma_3$ with only $\forall$ quantifications over variables. The property that the diameter of a graph is at most two cannot be expressed by using forbidden subgraphs but can easily be written as an FOL formula from $\Pi_2 \subset \Sigma_3$: $\forall u \forall v \exists w[(u = v) \vee (u \sim v) \vee ((u \sim w) \wedge (v \sim w))]$. Another interesting example of a property expressible in Σ_3 is the property of containing a universal vertex, and, more generally, having an r-dominating set of size at most d for constants r and d. Having a twin-pair, that is, a pair of vertices with equal neighborhoods, is also a property expressible in Σ_3.

Theorem 1 (Informal). *For every $\varphi \in \Sigma_3$, n-vertex graph G, and $k \geq 0$, deciding whether the elimination distance from G to property $\mathcal{P}_\varphi$, is at most k, can be done in time $f(k) \cdot n^{O(|\varphi|)}$ for some function f of k only.*

Our second theorem shows that the assumptions on the prefix of the formula are necessary. Let Π_3 be the class of first-order logic formulas of the form

$$\varphi = \forall x_1 \forall x_2 \cdots \forall x_r \;\; \exists y_1 \exists y_2 \cdots \exists y_s \;\; \forall z_1 \forall z_2 \cdots \forall z_t \; \psi,$$

where ψ is an FOL-formula without quantifiers and s, t, q are non-negative integers. We show that

Theorem 2 (Informal). *There is $\varphi \in \Pi_3$ such that deciding whether the elimination distance to $\mathcal{P}_\varphi$ is at most k, is* W[2]*-hard parameterized by k.*

Variants of elimination distance. The main reason why we give informal statements of our theorems in the introduction is due to the following issue: The definition of elimination distance is tailored to the graph properties $\mathcal{P}$ with the condition that $G \in \mathcal{P}$ if and only if $C \in \mathcal{P}$ for every component C of G. Graph properties defined by FOL do not necessarily satisfy such a condition. This leads to ambiguities. As an example, consider graph property $\mathcal{P} = \{G \mid G \models \forall x \forall y \; x = y\}$. Thus, $G \in \mathcal{P}$ if and only if G is a single-vertex graph. Let G be an edgeless graph with $n \geq 2$ vertices. Since $G \notin \mathcal{P}$ it would be a natural assumption that the elimination distance from G to $\mathcal{P}$ is positive. However, it is not: Every connected component of G is in $\mathcal{P}$ and, therefore,

$$\mathrm{ed}_{\mathcal{P}}(G) = \max\{\mathrm{ed}_{\mathcal{P}}(C) \mid C \text{ is a component of } G\} = 0.$$

To avoid such ambiguities, we refine the definition of the elimination distance.

Since we consider graph properties $\mathcal{P}_\varphi = \{G \mid G \models \varphi\}$ for formulas φ, we define the distances with respect to formulas. Notice that the notion of elimination distance combines "connectivity" and "inclusion" in a graph class. Depending on which of these two properties we want to prioritize, we give different definitions. Let φ be an FOL formula. The first definition prioritizes connectivity and the second prioritizes the graph property.

Definition 1 (Elimination Distances $\mathrm{ed}_\varphi^{\mathrm{conn}}$ *and* $\mathrm{ed}_\varphi^{\mathrm{prop}}$*).* For a graph G, we set

$$\mathrm{ed}_\varphi^{\mathrm{conn}}(G) = \begin{cases} 0, & \text{if } G \models \varphi \text{ or } G = (\emptyset, \emptyset), \\ 1 + \min_{v \in V(G)} \mathrm{ed}_\varphi^{\mathrm{conn}}(G - v), & \text{otherwise,} \end{cases}$$

if G is connected. We set

$$\mathrm{ed}_\varphi^{\mathrm{conn}}(G) = \max\{\mathrm{ed}_\varphi^{\mathrm{conn}}(C) \mid C \text{ is a component of } G\}$$

when G is not connected.

$$\mathrm{ed}_\varphi^{\mathrm{prop}}(G) = \begin{cases} 0, & \text{if } G \models \varphi \text{ or } G = (\emptyset, \emptyset), \\ 1 + \min_{v \in V(G)} \mathrm{ed}_\varphi^{\mathrm{prop}}(G - v), & \text{if } G \not\models \varphi \text{ and } G \text{ is connected,} \\ \max\{1, \max\{\mathrm{ed}_\varphi^{\mathrm{prop}}(C) \mid C \text{ is a component of } G\}\}, & \text{otherwise.} \end{cases}$$

We underline that, by our definition, $\mathrm{ed}_\varphi^{\mathrm{conn}}(G) = \mathrm{ed}_\varphi^{\mathrm{prop}}(G) = 0$ if $G = (\emptyset, \emptyset)$ for any formula φ. If φ is such that $G \in \mathcal{P}_\varphi$ if and only if $C \in \mathcal{P}_\varphi$ for every component C of G, then $\mathrm{ed}_{\mathcal{P}_\varphi}(G) = \mathrm{ed}_\varphi^{\mathrm{conn}}(G) = \mathrm{ed}_\varphi^{\mathrm{prop}}(G)$. However, in general $\mathrm{ed}_\varphi^{\mathrm{conn}}(G)$ and $\mathrm{ed}_\varphi^{\mathrm{prop}}(G)$ may differ significantly. Consider $\varphi = \exists u \exists v\ \neg(u = v) \wedge \neg(u \sim v)$ that defines the property that a graph has two nonadjacent vertices. Let G be the disjoint union of the complete n-vertex graph K_n and an isolated vertex. Then $G \models \varphi$ and, therefore, $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) = 0$. However, G is not connected and it is easy to see that $\mathrm{ed}_\varphi^{\mathrm{conn}}(K_n) = n$.

To analyze $\mathrm{ed}_\varphi^{\mathrm{conn}}(G)$ and $\mathrm{ed}_\varphi^{\mathrm{prop}}(G)$, we use equivalent definitions of these parameters via sets of vertices $X \subseteq V(G)$ of bounded "depth." We define this notion in Section 3 but informally the depth of X is the tree-depth of the torso of X (the *torso* of X is the graph obtained from $G[X]$ by making every two vertices $u, v \in X$ adjacent if there is a connected component C of $G - X$ whose neighborhood contains both u and v). This leads to another type of elimination distance $\mathrm{ed}_\varphi^{\mathrm{depth}}(G)$, which is defined as the minimum tree-depth of the torso of a *modulator* $X \subseteq V(G)$ such that $G - X \models \varphi$. The distances to graph properties, where the distance is measured as minimum "width" parameter of a modulator, recently got attention (see, e.g., [14, 22]) in Parameterized Complexity in the context of hybrid structural parameterizations.

Given the above, a more precise statement of Theorems 1 and 2 is that they hold for the distances $\mathrm{ed}_\varphi^{\mathrm{conn}}$, $\mathrm{ed}_\varphi^{\mathrm{prop}}$, and $\mathrm{ed}_\varphi^{\mathrm{depth}}$. Precise statements of both theorems and their proofs are given in Sections 4 and 5.

Related work. Results of this work fit into two popular trends in logic and parameterized complexity. A significant amount of research in descriptive complexity is devoted to the study of prefix classes of certain logics. We refer to the book of Börger, Grädel, and Gurevich [4], as well as the aforementioned work of Gottlob, Kolaitis, and Schwentick [19] for further references. The study of graph modification problems is a well-established trend in parameterized complexity. The books cited in [10, 13, 15, 29] provide a comprehensive overview of the area. In particular, Fomin, Golovach, and Thilikos [16] studied parameterized complexity of computing vertex deletion distance and edge editing to graph properties defined by first-order logic formulas; [16, Theorem 1] establishes fixed-parameter tractability for vertex removal to a graph property $\mathcal{P}_\varphi$ for $\varphi \in \Sigma_3$ and shows that the problem is W[2]-hard for some $\varphi \in \Pi_3$. While our Theorem 1 reaches the same tractability border for the elimination distance, the proof is significantly more complicated.

The general question on the parameterized complexity of elimination distance to graph properties was stated by Bulian and Dawar [5, 6]. Properties that have been considered so far are *minor-free graph classes* [6], *cluster graphs* [1, 2], *bounded degree graphs* [5, 24], and H-free graphs [1]. Moreover, Hols et al. [21] studied the existence of polynomial kernels for the Vertex Cover problem parameterized by the size of a deletion set to graphs of bounded elimination distance to certain graph classes. Lindermayr, Siebertz, and Vigny [24] proved that computing the elimination distance to graphs of bounded degree is fixed-parameter tractable when the input does not contain K_5 as a minor. While preparing our article, we have learned about the very recent work of Agrawal et al. [1]. Agrawal et al. established fixed-parameter tractability of computing an elimination distance to any graph property characterized by a finite set of graphs as forbidden induced subgraphs. Since graphs of bounded vertex degree can be characterized by a finite set of forbidden induced subgraphs, the work of Agrawal et al. answers the question of Bulian and Dawar [6] about the elimination distance to graphs of bounded degree.

Comparing with the result of Agrawal et al. [1], our Theorem 1 is more general. First, it provides the tractability of the elimination ordering to a strictly larger family of graph properties. Every graph property described by a finite set of forbidden induced subgraphs is also definable by a formula from Σ_3. However, properties like having a universal vertex or bounded diameter, which are expressible in Σ_3, cannot be described by forbidden subgraphs. Second, Theorem 1 holds for three variants of the elimination distance: $\mathrm{ed}_\varphi^{\mathrm{conn}}$, $\mathrm{ed}_\varphi^{\mathrm{prop}}$, and $\mathrm{ed}_\varphi^{\mathrm{depth}}$. With this terminology, the result of Agrawal et al. is only about computing $\mathrm{ed}_\varphi^{\mathrm{conn}}$. When it comes to the proof techniques, both Theorem 1 and the result of Agrawal et al. use recursive understanding, which seems to be a very natural technique for approaching problems about elimination distances. However, the details are quite different. To deal with various types of elimination distances and FOL formulas in a uniform way, we need different combinatorial characterizations of the distances via sets of bounded elimination depths. Furthermore, while solving our problems on unbreakable graphs is done by recursive branching algorithms, similarly to Agrawal et al., we do it in a different way that exploits the random separation technique to deal with our more general FOL framework. Moreover, the analysis of components of the graph obtained by the deletion of an elimination set for computing $\mathrm{ed}_\varphi^{\mathrm{depth}}(G)$, and especially, $\mathrm{ed}_\varphi^{\mathrm{prop}}(G)$, is a great deal more challenging. In particular, this is the reason why we apply the random separation technique contrary to the more straightforward tools used by Agrawal et al.

Overview of the approach. The first two variants of the elimination distance that we examine are defined recursively using the containment in the graph class $\mathcal{P}_\varphi$ as the base case. We start by providing equivalent formulations that are more suitable from the algorithmic perspective. For this, we introduce the notion of *elimination set of depth at most* d that is a set $X \subseteq V(G)$ that can be bijectively mapped to a rooted tree T of depth d expressing selection of elimination vertices in recursive calls. We next prove that $\mathrm{ed}_\varphi^{\mathrm{conn}}(G) \leq k$ if and only if G has an elimination set X of depth at most $k-1$ such that $C \models \varphi$ for every component C of $G - X$. Similar, however more technical, equivalent formulations is given for $\mathrm{ed}_\varphi^{\mathrm{prop}}$. All alternative definitions and the proofs of their equivalences to the recursive ones are gathered in Section 3.

The new definitions allow to certify a solution by a set X of bounded elimination depth. However, the size of X could be unbounded. Moreover, there could be many connected components of $G - X$ and the sizes of these components could be immense. We use the *recursive understanding technique*, introduced by Chitnis et al. in [8], to reduce the solution of the initial problem to a much more structured problem. In the reduced problem, we can safely assume that each yes-instance is certified by an elimination set X whose size, as well as the size of the union of all *but one* of the components of $G - X$, is also bounded by a function of k.

More precisely, by making use of recursive understanding, we can consider only instances that are $(p(k), k)$-unbreakable for some suitably chosen function p. Roughly speaking, a graph is $(p(k), k)$-unbreakable when it has no separator of size at most k that partitions the graph in two parts of size at least $p(k) + 1$ each. The application of recursive understanding uses the meta-algorithmic result of Lokshtanov et al. [26] and the fact that all variants of the elimination distance to $\mathcal{P}_\varphi$ are expressible in **monadic-second order logic (MSOL)** when ϕ is a formula in FOL (Lemma 5).

The $(p(k), k)$-unbreakability permits us to assume that $|X| \leq p(k) + k$. Moreover, exactly one connected component C_X of $G-X$, is *big*, that is, of size at least $p(k)+1$, and the size of $G-V(C_X)$ is bounded by some function of k (see Lemma 6). Given that C_X is the big component corresponding to a solution X, we also consider the set S_X of the neighbors of the vertices of C_X in G and we set $U_X = V(G)\setminus(V(C_X) \cup S_X)$. We show that $|S_X| \leq k$ and $|U_X| \leq p(k)$.

Our next step is to use the *random separation technique*, introduced by Cai, Chan, and Chan in [8]. We construct in FPT-time a family $\mathcal{F}$ of at most $f(k) \cdot \log n$ partitions (R, B) of $V(G)$ to "red" and "blue" vertices such that for every elimination set X corresponding to a potential solution, $\mathcal{F}$ contains some (R, B) where $U_X \subseteq R$ and $S_X \subseteq B$. In our algorithm, we go over all these blue-red partitions and, for each one of them, we check whether there exists an elimination set X (called *colorful elimination set*) where all vertices in S_X are blue and all vertices in U_X are red.

The correct "guess" of the above red-blue partition permits us to design a recursive procedure that solves the latter problem, i.e., finds a colorful elimination set X. This procedure is different for each of the three versions of the problem and its variants are presented in Section 4.2. The key task here is to identify the big component C_X. It runs in FPT-time and its correctness is based on the prefix structure of the formula ϕ.

Organization of the article. In Section 2, we provide the basic definitions of the concepts that we use in this article: complexity classes graphs and formulas. In Section 3, we prove some properties and relations between the elimination ordering variants that we consider. We also provide alternative definitions and we prove their equivalencies with the original ones. The main algorithmic result is in Section 4 where we explain how we apply the recursive understanding technique, the random separation technique, and we present the branching procedure for the "colorful version" of each variant. Section 5 gives the lower bound of the article. This uses a parameterized reduction from the Set Cover problem. Finally, in Section 6, we provide some discussion on the kernelization complexity of our problems as well as some directions on further research on elimination distance problems.

2 PRELIMINARIES

Sets. We use $\mathbb{N}$ to denote the set of all non-negative numbers. We denote by $\mathbf{a} = \langle a_1, \ldots, a_r \rangle$ a sequence of elements of a set A and call $\mathbf{a}$ an *r-tuple* of simply a *tuple*. Note that the elements of $\mathbf{a}$ are not necessarily distinct. We denote by $\mathbf{ab} = \langle a_1, \ldots, a_r, b_1, \ldots, b_s \rangle$ the *concatenation* of tuples $\mathbf{a} = \langle a_1, \ldots, a_r \rangle$ and $\mathbf{b} = \langle b_1 \ldots, b_s \rangle$.

Parameterized Complexity. We refer to the recent books of Cygan et al. [10] and Downey and Fellows [13] for a detailed introduction to the field. Formally, a *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ^* is a set of strings over a finite alphabet Σ. This means that an input of a parameterized problem is a pair (x, k), where x is a string over Σ and $k \in \mathbb{N}$ is a *parameter.* A parameterized problem is ***fixed-parameter tractable*** **(FPT)** if it can be solved in time $f(k) \cdot |x|^{O(1)}$ for some computable function f. Also, we say that a parameterized problem belongs to the class XP if it can be solved in time $|x|^{f(k)}$ for some computable function f. The complexity class FPT contains all fixed-parameter tractable problems. Parameterized complexity theory also provides tools to disprove the existence of an FPT-algorithm for a problem under plausible complexity-theoretic assumptions. The standard way is to show that the problem is W[1] or W[2]-hard using a *parameterized reduction* from a known W[1] or W[2]-hard problem; we refer to [10, 13] for the formal definitions of the classes W[1] and W[2] and parameterized reductions.

Graphs. We consider only undirected simple graphs and use the standard graph theoretic terminology (see, e.g., [12]). Throughout the article, we use n to denote $|V(G)|$ if it does not create confusion. For a set of vertices $S \subseteq V(G)$, we denote by $G[S]$ the subgraph of G induced by the vertices from S. We also define $G-S = G[V(G)\setminus S]$; we write $G-v$ instead of $G-\{v\}$ for a single vertex set. For a vertex v, $N_G(v)$ denotes the *open neighborhood* of v, that is, the set of vertices adjacent to v, and $N_G[v] = \{v\} \cup N_G(v)$ is the *closed neighborhood.* For $S \subseteq V(G)$, $N_G(S) = (\bigcup_{v\in S} N_G(v))\setminus S$ and $N_G[S] = \bigcup_{v\in S} N_G[v]$. For a vertex v, $d_G(v) = |N_G(v)|$ denotes the *degree* of v. For a path P with end-vertices u and v, we say that P is a (u,v)*-path*; the vertices of $V(P)\setminus\{u,v\}$ are *internal.* A graph G is *connected* if for every two vertices u and v, G contains a path whose end-vertices are u and v. For a positive integer k, G is k*-connected* if $|V(G)| \geq k$ and $G-S$ is connected for every $S \subseteq V(G)$ of size at most $k-1$. A *connected component* (or simply a *component*) is an inclusion maximal induced connected subgraph of G. For two distinct vertices u and v of a graph G, a set $S \subseteq V(G)$ is a (u,v)*-separator* if $G-S$ has no (u,v)-path.

A *rooted tree* is a tree T with a selected node r (we use the term "node" instead of "vertex" for such a tree) called a *root.* The selection of r defines the standard parent–child relation on $V(T)$. Nodes without children are called *leaves* and we use $L(T)$ to denote the set of leaves of T; note that a root is a leaf if $|V(T)| = 1$. The *depth* $\mathsf{depth}_T(v)$ of a node v is the distance between r and v, and the *depth* (or *height*) $\mathsf{depth}(T)$ of T is the maximum depth of a node. The nodes of the (r,v)-path are called *ancestors* of v. We use $A_T(v)$ to denote the set of ancestors of v in T. Note that v is its own ancestor; we say that an ancestor is *proper* if it is distinct from v. Two nodes u and v of T are *comparable* if either v is an ancestor of u or u is an ancestor of v. Otherwise, u and v are *incomparable.* A node w of T is the *lowest common ancestor* of nodes u and v if w is the ancestor of maximum depth of both u and v. Note that the lowest common ancestor is unique and if u and v are incomparable then the lowest common ancestor is distinct from u and v. A node v is a *descendant* of u if u is an ancestor of v. By $D_T(u)$, we denote the set of descendants of u in T. As with ancestors, a node is its own descendant and we say that a descendant v of u is *proper* if $u \neq v$. For a node v, the subtree induced by the descendants of v is the *subtree rooted in* v.

Logic. In this article, we deal with *first-order* and *monadic second-order* logic formulas on graphs.

The syntax of the **first-order logic (FOL)** formulas on graphs includes the logical connectives $\vee$, $\wedge$, $\neg$, variables for vertices, the quantifiers $\forall$, $\exists$ that are applied to these variables, the predicate $u \sim v$, where u and v are vertex variables and the interpretation is that u and v are adjacent, and the equality of variables representing vertices. It is also convenient to assume that we have the logical connectives $\rightarrow$ and $\leftrightarrow$. An FOL formula φ is in *prenex normal form* if it is written as $\varphi = \mathsf{Q}_1 x_1 \mathsf{Q}_2 x_2 \cdots \mathsf{Q}_t x_t \chi$ where each $\mathsf{Q}_i \in \{\forall, \exists\}$ is a quantifier, x_i is a variable, and χ is a quantifier-free part that may depend on the variables $x_1, \ldots, x_t$. Then $\mathsf{Q}_1 x_1 \mathsf{Q}_2 x_2 \cdots \mathsf{Q}_t x_t$ is referred to as the *prefix* of φ. From now on, when we write "FOL formula," we mean an FOL formula on graphs that is in prenex normal form. We assume that there is no nested requantification of a variable, that is, $x_1, \ldots, x_t$ are distinct. Also, we assume that a formula has no *free*, that is, non-quantified variables unless we explicitly say that free variables are permitted. For an FOL formula φ and a graph G, we write $G \models \varphi$ to denote that φ evaluates to *true* on G.

We use the arithmetic hierarchy (also known as Kleene-Mostowski hierarchy) for the classification of formulas in the first-order arithmetic language (see, e.g., [30]). For this, we define prefix classes according to alternations of quantifiers, that is, switchings from ∀ to ∃ or vice versa in the prefix string of the formula. Here, we allow a formula to have free variables. Let $\Sigma_0 = \Pi_0$ be the classes of FOL-formulas without quantifiers. For a positive integer ℓ, the class Σ_ℓ contains formulas that may be written in the form

$$\varphi = \exists x_1 \exists x_2 \cdots \exists x_s\ \psi,$$

where ψ is a $\Pi_{\ell-1}$-formula, s is some integer, and $x_1, \ldots, x_s$ are free variables of ψ. Respectively, Π_ℓ consists of formulas

$$\varphi = \forall x_1 \forall x_2 \cdots \forall x_s\ \psi,$$

where ψ is a $\Sigma_{\ell-1}$-formula and $x_1, \ldots, x_s$ are free variables of ψ. Note that for $\ell' > \ell$, $\Sigma_\ell \cup \Pi_\ell \subseteq \Sigma_{\ell'} \cap \Pi_{\ell'}$, that is, every Σ_ℓ or Π_ℓ formula is both a $\Sigma_{\ell'}$ and $\Pi_{\ell'}$-formula.

For technical reasons, we extend FOL formulas on graphs to structures of a special type. We say that a pair $(G, \mathbf{v})$, where $\mathbf{v} = \langle v_1, \ldots, v_r \rangle$ is an r-tuple of vertices of G, is an *r-structure*. Let φ be an FOL formula without free variables and let $\mathbf{x} = \langle x_1, \ldots, x_r \rangle$ be an r-tuple of distinct variables of φ. We denote by $\varphi[\mathbf{x}]$ the formula obtained from φ by the deletion of the quantification over $x_1, \ldots, x_r$, that is, these variables become the free variables of $\varphi[\mathbf{x}]$. For an r-structure $(G, \mathbf{v})$ with $\mathbf{v} = \langle v_1, \ldots, v_r \rangle$ and $\varphi[\mathbf{x}]$, we write $(G, \mathbf{v}) \models \varphi[\mathbf{x}]$ to denote that $\varphi[\mathbf{x}]$ evaluates to *true* on G if x_i is assigned v_i for $i \in \{1, \ldots, r\}$. If $r = 0$, that is, $\mathbf{v}$ and $\mathbf{x}$ are empty, then $(G, \mathbf{v}) \models \varphi[\mathbf{x}]$ is equivalent to $G \models \varphi$.

As a subroutine in our algorithms, we have to evaluate FOL formulas on graph, that is, solve the Model Checking problem. Let φ be an FOL formula. The task of Model Checking is, given a graph G, decide whether $G \models \varphi$. It was shown by Vardi [31] that Model Checking is PSPACE-complete. The problem is also hard from the parameterized complexity viewpoint when parameterized by the size of the formula. It was proved by Frick and Grohe in [18] that the problem is AW[∗]-complete for this parametrization (see, e.g., the book [15] for the definition of the class). Moreover, it can be noted that the problem is already W[1]-hard for formulas having only existential quantifiers, that is, for $\varphi \in \Sigma_1$, by observing that the existence of an independent set of size k can be easily expressed by such a formula and Independent Set is well-known to be W[1]-complete [13]. This implies that we cannot expect an FPT algorithm for the problem. However, it is easy to see that Model Checking is in XP when parameterized by the number of variables, because the problem for a formula with s variables can be solved in $O(n^s)$ time by the exhaustive search (the currently best algorithm is given by Williams in [32]). This explains the exponential dependence of the polynomials in running times in our algorithm on the number of variables. For referencing, we state the following observation:

Observation 1. *Model Checking for an FOL formula φ can be solved in $n^{O(|\varphi|)}$ time.*

In **monadic second-oder logic (MSOL)**, we additionally can quantify over sets of vertices and edges. Formally, we can use variables for sets of vertices and edges and have the predicate $x \in X$, where x is a vertex (an edge, respectively) variable and X a vertex set (an edge set, respectively) variable, denoting that x is an element of X. As with FOL formulas, we write $G \models \varphi$ to denote that an MSOL formula φ evaluates *true* on G. We refer to the book of Courcelle and Engelfriet [9] for the details of MSOL on graphs.

3 PROPERTIES OF ELIMINATION DISTANCE

In this section, we derive the properties of the elimination distances, $\mathrm{ed}_\varphi^{\mathrm{conn}}$ and $\mathrm{ed}_\varphi^{\mathrm{prop}}$, that will be used in the proof of the main theorem. We also define $\mathrm{ed}_\varphi^{\mathrm{depth}}$.

Observation 2. *For every FOL formula φ and every graph G, $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) \le \mathrm{ed}_\varphi^{\mathrm{conn}}(G) + 1$.*

Proof. The proof is by induction on the value of $\mathrm{ed}_\varphi^{\mathrm{conn}}(G)$.

Suppose that $\mathrm{ed}_\varphi^{\mathrm{conn}}(G) = 0$ for a graph G. If G is connected, then $G \models \varphi$ and $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) = 0$. Hence, the inequality holds. If G is disconnected, then $C \models \varphi$ for every component C of G. If $G \models \varphi$, then $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) = 0$. If $G \not\models \varphi$, then $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) = \max\{1, \max\{\mathrm{ed}_\varphi^{\mathrm{prop}}(C) \mid C \text{ is a component of } G\}\} = 1$. In both cases, $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) \leq \mathrm{ed}_\varphi^{\mathrm{conn}}(G) + 1$.

Assume that $\mathrm{ed}_\varphi^{\mathrm{conn}}(G) > 0$ and $\mathrm{ed}_\varphi^{\mathrm{prop}}(G') \leq \mathrm{ed}_\varphi^{\mathrm{conn}}(G') + 1$ for all G' with $\mathrm{ed}_\varphi^{\mathrm{conn}}(G') < \mathrm{ed}_\varphi^{\mathrm{conn}}(G)$. The claim is trivial if $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) = 0$. Suppose that $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) > 0$. We have two cases.

Case 1. G is connected. By definition, there is $u \in V(G)$ such that $\mathrm{ed}_\varphi^{\mathrm{conn}}(G) = 1 + \mathrm{ed}_\varphi^{\mathrm{conn}}(G - u)$. Because $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) > 0$,

$$\mathrm{ed}_\varphi^{\mathrm{prop}}(G) = 1 + \min_{v \in V(G)} \mathrm{ed}_\varphi^{\mathrm{prop}}(G - v) \leq 1 + \mathrm{ed}_\varphi^{\mathrm{prop}}(G - u).$$

Then, by induction,

$$\mathrm{ed}_\varphi^{\mathrm{prop}}(G) \leq 1 + \mathrm{ed}_\varphi^{\mathrm{prop}}(G - u) \leq 2 + \mathrm{ed}_\varphi^{\mathrm{conn}}(G - u) = 1 + \mathrm{ed}_\varphi^{\mathrm{conn}}(G).$$

Case 2. G is disconnected. Let $C_1, \ldots, C_s$ be the components of G. By definition, $\mathrm{ed}_\varphi^{\mathrm{conn}}(G) = \max_{1 \leq i \leq s} \mathrm{ed}_\varphi^{\mathrm{conn}}(C_i)$. In particular, we have that $\mathrm{ed}_\varphi^{\mathrm{conn}}(C_i) \leq \mathrm{ed}_\varphi^{\mathrm{conn}}(G)$ for every $i \in \{1, \ldots, s\}$. Notice that by the already proved claim for connected graphs in Case 1, $\mathrm{ed}_\varphi^{\mathrm{prop}}(C_i) \leq \mathrm{ed}_\varphi^{\mathrm{conn}}(C_i) + 1$ for every $i \in \{1, \ldots, s\}$. Because $\mathrm{ed}_\varphi^{\mathrm{conn}}(G) > 0$, $G \not\models \varphi$. Then

$$\begin{aligned}\mathrm{ed}_\varphi^{\mathrm{prop}}(G) &= \max\left\{1, \max_{1 \leq i \leq s} \mathrm{ed}_\varphi^{\mathrm{prop}}(C_i)\right\} \leq \max\left\{1, \max_{1 \leq i \leq s} (\mathrm{ed}_\varphi^{\mathrm{conn}}(C_i) + 1)\right\} \\ &= \max_{1 \leq i \leq s} \mathrm{ed}_\varphi^{\mathrm{conn}}(C_i) + 1 = \mathrm{ed}_\varphi^{\mathrm{conn}}(G) + 1,\end{aligned}$$

as required. This completes the proof. □

The example of $\varphi = \forall x \forall y\ x = y$ and an edgeless graph G with at least two vertices shows that the inequality in Observation 2 is tight. However, recall that the difference between $\mathrm{ed}_\varphi^{\mathrm{conn}}(G)$ and $\mathrm{ed}_\varphi^{\mathrm{prop}}(G)$ can be arbitrarily large, as demonstrated by the example given in the introduction with $\varphi = \exists u \exists v\ \neg(u = v) \wedge \neg(u \sim v)$ defining the property that a graph has two nonadjacent vertices.

For algorithmic purposes, it is convenient for us to define $\mathrm{ed}_\varphi^{\mathrm{conn}}(G)$ and $\mathrm{ed}_\varphi^{\mathrm{prop}}(G)$ via deletions of sets of vertices of G with a special structure. Similar approaches were exploited by Agrawal et al. [1] and Bulian and Dawar [5, 6] but we do it in a different way, because we consider two variants of elimination distances.

Let G be a graph and let $d \geq 0$ be an integer. We say that a set of vertices $X \subseteq V(G)$ is an *elimination set of depth at most* d if there is a rooted tree T of depth at most d and a bijective mapping $\alpha \colon V(T) \to X$ such that for every two distinct incomparable nodes x and y of T, $\alpha(A_T(v))$ is an $(\alpha(x), \alpha(y))$-separator in G, where v is the lowest common ancestor of x and y (recall that $A_T(v)$ denotes the set of ancestors of v). We also say that the pair (T, α) is a *representation* of X (or *represents* X). The *depth* of $X \subseteq V(G)$, denoted $\mathrm{depth}(X)$, is the minimum d such that X is an elimination set of depth at most d. We assume that the empty set is an elimination set of depth -1.

We call a representation (T, α) of an elimination set $X \subseteq V(G)$ *nice* if for every nonleaf node $v \in V(T)$ and its child x, the vertices of $\alpha(D_T(x))$ are in the same component of $G - A_T(v)$. The following property is useful for us:

Lemma 1. *Let G be a connected graph and let $d \geq 0$ be an integer. Then a nonempty $X \subseteq V(G)$ is an elimination set of depth at most d if and only if X has a nice representation (T, α) with* $\mathrm{depth}(T) \leq d$. *Moreover, if (T, α) is a representation of X, then there is a nice representation (T', α) of X with $V(T') = V(T)$ such that (i) $\alpha(L(T)) \subseteq \alpha(L(T'))$ and (ii) for each $v \in X$,* $\mathrm{depth}_T(\alpha^{-1}(v)) \geq \mathrm{depth}_{T'}(\alpha^{-1}(v))$.

Proof. Clearly, if $X \subseteq V(G)$ has a nice representation (T, α) with $\text{depth}(T) \leq d$, then X is an elimination set of depth at most d. For the opposite direction, it is sufficient to show the second claim. Let (T, α) be a representation of X and $\text{depth}(T) \leq d$. We show the existence of (T', α) satisfying (i) and (ii) by induction on d.

The claim is trivial if $|X| = 1$ as $T' = T$ in this case. Assume that $|X| \geq 2$ and $d \geq 1$. Denote by r the root of T and let $u = \alpha(r)$. Consider the components $C_1, \ldots, C_s$ of $G - u$ containing at least one vertex of X. For every $i \in \{1, \ldots, s\}$, let $X_i = V(C_i) \cap X$ and $U_i = \alpha^{-1}(X_i)$. For every $i \in \{1, \ldots, s\}$, we construct the tree T_i with the set of vertices $U_i \cup \{r\}$ as follows: For every $x \in U_i$ such that $x \neq r$, we find a proper ancestor $y \in U_i$ with respect to T of maximum depth and make y the parent of x, and if x has no ancestors in U_i, then we make r the parent of x. Because the choice of the parent is unique, T_i has no cycles, and because we assign the parent to every node distinct from r, we conclude that T_i is a tree. Denote by $\tilde{T}$ the union of $T_1, \ldots, T_s$ and set r be the root of $\tilde{T}$. Because every node of $\tilde{T}$ distinct from r got a parent from the set of its proper ancestors in T, (i) $\alpha(L(T)) \subseteq \alpha(L(\tilde{T}))$ and (ii) for each $v \in X$, $\text{depth}_T(\alpha^{-1}(v)) \geq \text{depth}_{\tilde{T}}(\alpha^{-1}(v))$.

We prove that $(\tilde{T}, \alpha)$ represents X. Consider incomparable nodes x and y of $\tilde{T}$ and denote by v their lowest common ancestor. We have to show that $\alpha(A_{\tilde{T}}(v))$ is an $(\alpha(x), \alpha(y))$-separator in G. This is trivial if $\alpha(x)$ and $\alpha(y)$ are in distinct components of $G - u$. Assume that $\alpha(x)$ and $\alpha(y)$ are in the same component C_i for some $i \in \{1, \ldots, s\}$, that is, $x, y \in U_i$. Note that by the construction of $\tilde{T}$, x and y are incomparable in T. Let v' be their lowest common ancestor in T. Clearly, v is an ancestor of v in T. By the construction of $\tilde{T}$, $A_{\tilde{T}}(v) \cap V(C_i) = A_T(v') \cap V(C_i)$. Because $A_T(v)$ separates $\alpha(x)$ and $\alpha(y)$, we have that $A_T(v) \cap V(C_i)$ is an $(\alpha(x), \alpha(y))$-separator. Therefore, $\alpha(A_{\tilde{T}}(v))$ is also an $(\alpha(x), \alpha(y))$-separator. This proves that $(\tilde{T}, \alpha)$ represents X.

Consider $i \in \{1, \ldots, s\}$. Observe that r has a unique child in U_i in $\tilde{T}$. Otherwise, if x and y are distinct children of r, then we have that x and y have no ancestors in U_i with respect to T. Let v be the lowest common ancestor of x and y in T. Note that $v \neq x, y$ and $\alpha(A_T(v))$ does not separate $\alpha(x)$ and $\alpha(y)$ contradicting that (T, α) represents X. Hence, r has the unique child r_i in U_i. Let $\tilde{T}_i$ be the subtree of $\tilde{T}$ rooted in r_i. We set $\alpha_i(x) = \alpha(x)$ for $x \in U_i$. Because $(\tilde{T}, \alpha)$ represents X, it is straightforward to verify that $(\tilde{T}_i, \alpha_i)$ represents X_j in the graph C_i. Because $\text{depth}(\tilde{T}_i) \leq d - 1$, we can apply the inductive assumption. We obtain that there is a nice representation (T'_i, α_i) of X_i in C_i with $V(T'_i) = V(\tilde{T}_i)$ such that (i) $\alpha(L(\tilde{T}_i)) \subseteq \alpha(L(T'_i))$ and (ii) for each $v \in X_i$, $\text{depth}_{\tilde{T}_i}(\alpha_i^{-1}(v)) \geq \text{depth}_{T'_i}(\alpha_i^{-1}(v))$. Notice that by the second condition, T'_i is rooted in r_i.

We construct the trees T'_i for all $i \in \{1, \ldots, s\}$ and then construct T' from their union by making $r_1, \ldots, r_s$ the children of r. Clearly, $\text{depth}(T') \leq d$ and (T', α) is a nice representation of X satisfying conditions (i) and (ii) of the lemma. □

It is also useful to characterize the depths of an elimination set in a disconnected graph.

Lemma 2. *Let G be a graph with components $C_1, \ldots, C_s$ and let $X \subseteq V(G)$ such that $X \neq \emptyset$. Then*

$$\text{depth}(X) = \min_{1 \leq i \leq s} \max\{\text{depth}(X_i), \max\{\text{depth}(X_j) \mid 1 \leq j \leq s,\ j \neq i\} + 1\}, \tag{1}$$

where $X_i = X \cap V(C_i)$ for $i \in \{1, \ldots, s\}$.

Proof. Recall that $\text{depth}(\emptyset) = -1$ by definition. This allows us to assume without loss of generality that $X_i \neq \emptyset$ for all $i \in \{1, \ldots, s\}$. Otherwise, we can delete each component C_i such that $X_i = \emptyset$ without violating the value of $\text{depth}(X)$ and the right part of (1).

To show that

$$\text{depth}(X) \leq \min_{1 \leq i \leq s} \max\{\text{depth}(X_i), \max\{\text{depth}(X_j) \mid 1 \leq j \leq s,\ j \neq i\} + 1\},$$

assume that the minimum value of the right part of (1) is achieved for $i \in \{1, \dots, s\}$. For every $j \in \{1, \dots, s\}$, let (T_j, α_j) be a representation of X_j in C_j, where T_j is rooted in r_j and $\text{depth}(T_j) = \text{depth}(X_j)$. We construct the tree T with the root $r = r_i$ from $T_1, \dots, T_s$ by making each r_j for $j \in \{1, \dots, s\} \setminus \{i\}$ a child of r. Clearly, $\text{depth}(T) = \max\{\text{depth}(T_i), \max\{\text{depth}(T_j) \mid 1 \leq j \leq s,\ j \neq i\} + 1\} = \max\{\text{depth}(X_i), \max\{\text{depth}(X_j) \mid 1 \leq j \leq s,\ j \neq i\} + 1\}$. We define $\alpha \colon V(T) \to X$ by setting $\alpha(x) = \alpha_i(x)$ whenever $x \in X_i$ for some $i \in \{1, \dots, s\}$. It is straightforward to verify that (T, α) represents X.

To show the opposite inequality

$$\text{depth}(X) \geq \min_{1 \leq i \leq s} \max\{\text{depth}(X_i), \max\{\text{depth}(X_j) \mid 1 \leq j \leq s,\ j \neq i\} + 1\},$$

let (T, α) be a representation of X, where T is rooted in r and $\text{depth}(T) = \text{depth}(X)$. By symmetry, we assume without loss of generality that $r \in V(C_1)$. For every $j \in \{1, \dots, s\}$, let $U_i = \alpha^{-1}(X_j)$.

The rest of the proof is done similarly to the proof of Lemma 1. For every $j \in \{1, \dots, s\}$, we construct the tree T_j with the set of vertices $U_j \cup \{r\}$ as follows: For every $x \in U_j$ such that $x \neq r$, we find a proper ancestor $y \in U_j$ of x with respect to T of maximum depth and make y the parent of x, and if x has no ancestors in U_j, then we make r the parent of x. Because the choice of the parent is unique, T_j has no cycles, and because we assign the parent to every node distinct from r, we conclude that T_j is a tree. Denote by T' the union of $T_1, \dots, T_s$ and set r be the root. Because every node of T' distinct from r got a parent from the set of its proper ancestors in T, $\text{depth}(T') \leq \text{depth}(T) = \text{depth}(X)$.

We claim that (T', α) represents X. To show this, let x and y be incomparable nodes of T' and let v be their lowest common ancestor. We show that $\alpha(A_{T'}(v))$ is an $(\alpha(x), \alpha(y))$-separator in G. This is trivial if $\alpha(x)$ and $\alpha(y)$ are in distinct components of G. Assume that $\alpha(x)$ and $\alpha(y)$ are in the same component C_j for some $j \in \{1, \dots, s\}$, that is, $x, y \in U_j$. Notice that by the construction of T', $A_{T'}(v) \cap C_j = A_T(v') \cap C_j$, where v' is the lowest common ancestor of x and y in T. Because $A_T(v')$ separates $\alpha(x)$ and $\alpha(y)$, we have that $A_T(v') \cap C_j$ is an $(\alpha(x), \alpha(y))$-separator. Therefore, $\alpha(A_{T'}(v))$ is an $(\alpha(x), \alpha(y))$-separator as well, as required.

Let $j \in \{2, \dots, s\}$. Observe that r has a unique child in U_j in T'. Otherwise, if x and y are distinct children of r, then we have that x and y have no ancestors in U_j. Let v be the lowest common ancestor of x and y in T. Note that $v \neq x, y$ and $\alpha(A_T(v))$ does not separate $\alpha(x)$ and $\alpha(y)$ contradicting that (T, α) represents X. Hence, r has the unique child r_j in U_i. Let T'_j be the subtree of T' rooted in r_j. Define $\alpha_j(x) = \alpha(x)$ for $x \in U_j$. Since (T', α) represents X, we obtain that (T'_j, α_j) represents X_j. Then $\text{depth}(X_j) \leq \text{depth}(T'_j) \leq \text{depth}(T) - 1 = \text{depth}(X) - 1$.

It is straightforward to verify that (T_1, α_1) represents X_1, where $\alpha_1(x) = \alpha(x)$ for $x \in U_1$. This means that $\text{depth}(X_1) \leq \text{depth}(T_1) \leq \text{depth}(X)$. Because $\text{depth}(X_j) + 1 \leq \text{depth}(X)$ for $j \in \{2, \dots, s\}$,

$$\begin{aligned}\text{depth}(X) &\geq \max\{\text{depth}(X_1), \text{depth}(X_2) + 1, \dots, \text{depth}(X_s) + 1\} \\ &\geq \min_{1 \leq i \leq s} \max\{\text{depth}(X_i), \max\{\text{depth}(X_j) \mid 1 \leq j \leq s,\ j \neq i\} + 1\}.\end{aligned}$$

This completes the proof. □

It is sufficient for our purposes to characterize $\text{ed}^{\text{conn}}_{\varphi}(G)$ for connected graphs and we do it in the following lemma:

Lemma 3. *Let φ be an FOL formula and let G be a connected graph. Let also $d \geq 0$ be an integer. Then $\text{ed}^{\text{conn}}_{\varphi}(G) \leq d$ if and only if G contains an elimination set X of depth at most $d - 1$ such that $C \models \varphi$ for every component C of $G - X$.*

Proof. First, we show that if $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(G) \leq d$, then G has an elimination set X of depth at most $d-1$ such that $C \models \varphi$ for every component C of $G-X$. The proof is by induction on d. The claim is trivial if $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(G) = 0$ as $\mathrm{depth}(\emptyset) = -1$ by definition. Let $d \geq \mathrm{ed}_{\varphi}^{\mathrm{conn}}(G) \geq 1$.

Because G is connected and $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(G) > 0$, there is $v \in V(G)$ such that $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(G) = 1 + \mathrm{ed}_{\varphi}^{\mathrm{conn}}(G - v)$. We construct a node r of T and set it be the root. If $C \models \varphi$ for every component C of $G-v$, then the construction of X and T is completed and we define $\alpha(r) = v$. Otherwise, let $C_1, \dots, C_s$ be the components of $G-v$ such that $C_i \not\models \varphi$ for $i \in \{1, \dots, s\}$. Clearly, $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(C_i) \leq d-1$ for $i \in \{1, \dots, s\}$. Let $i \in \{1, \dots, s\}$. By induction, there is an elimination set $X_i \subseteq V(C_i)$ of depth at most $d-2$ such that $H \models \varphi$ for every component H of $C_i - X_i$. Then there is a corresponding representation (T_i, α_i) of X_i in C_i. Let r_i be the root of T_i. We define $X = \{v\} \cup \bigcup_{i=1}^{s} X_i$, and construct T from $T_1, \dots, T_s$ by making $r_1, \dots, r_s$ the children of r. Finally, we define

$$\alpha(x) = \begin{cases} v, & \text{if } x = r, \\ \alpha_i(x), & \text{if } x \in V(C_i) \text{ for some } i \in \{1, \dots, s\}. \end{cases}$$

It is straightforward to verify that X is an elimination set of depth at most $d-1$ with respect to (T, α).

For the opposite direction, we assume that X is an elimination set of minimum depth such that $C \models \varphi$ for every component C of $G-X$. We assume that the depth of X is $d-1$ and prove that $\mathrm{ed}_{\varphi}^{\mathrm{conn}} G \leq d$. The proof is by induction on d.

The claim is trivial if $d = 0$, that is, if $X = \emptyset$. Suppose that $d = 1$, that is, the depth of an elimination set X is zero and, therefore, $X = \{u\}$ for some $u \in V(G)$. We have that $C \models \varphi$ for every component C of $G-u$. This means that $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(C) = 0$ for every component C and, therefore, $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(G-u) = 0$. Then because $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(G) > 0$, $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(G) = 1 + \min_{v \in V(G)} \mathrm{ed}_{\varphi}^{\mathrm{conn}}(G-v) = 1 + \mathrm{ed}_{\varphi}^{\mathrm{conn}}(G-u) = 1 \leq d$.

Suppose that $d \geq 2$ and the claim holds for the lesser values of d. Because G is connected, by Lemma 1, there is a nice representation (T, α) of X with $\mathrm{depth}(T) = d-1$. Let r be the root of T and $u = \alpha(r)$. Because $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(G) > 0$, $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(G) = 1 + \min_{v \in V(G)} \mathrm{ed}_{\varphi}^{\mathrm{conn}}(G-v) \leq 1 + \mathrm{ed}_{\varphi}^{\mathrm{conn}}(G-u)$ and it is sufficient to show that $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(G-u) \leq d-1$. For this, we have to prove that $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(C) \leq d-2$ for every component C of $G-u$.

If $V(C) \cap X = \emptyset$ for a component C, then C is a component of $G-X$ and we have that $C \models \varphi$. Then $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(C) = 0 \leq d-2$. Consider the components $C_1, \dots, C_s$ of $G-u$ such that $V(C_i) \cap X \neq \emptyset$. Because (T, α) is nice, r has s children $x_1, \dots, x_s$ such that for every $i \in \{1, \dots, s\}$, $\alpha(V(T_i)) \subseteq V(C_i)$, where T_i is the subtree of T rooted in x_i. Let $\alpha_i \colon V(T_i) \to V(C_i)$ be the restriction of α on $V(T_i)$ for $i \in \{1, \dots, s\}$. Consider $i \in \{1, \dots, s\}$. We have that (T_i, α_i) is a representation of $X_i = X \cap V(C_i)$. Notice that for each component C of $C_i - X_i$, $C \models \varphi$. Clearly, $\mathrm{depth}(T_i) < \mathrm{depth}(T)$. This implies that we can use the inductive assumption and conclude that $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(C_i) \leq d-1$. Therefore, $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(G-u) \leq d-1$ and this concludes the proof. □

To characterize $\mathrm{ed}_{\varphi}^{\mathrm{prop}}(G)$, we need additional definitions.

Let G be a connected graph and let $X \subseteq V(G)$ be an elimination set represented by (T, α). We say that a node $x \in V(T)$ is an *anchor* of a component C of $G-X$ if x is the node of maximum depth in T such that $\alpha(x) \in N_G(V(C))$. We also say that C is *anchored* in x. Notice that the definition of an elimination set immediately implies the following property:

Observation 3. *Let G be a connected graph and let $X \subseteq V(G)$ be an elimination set represented by (T, α). Then for every component C of $G-X$, $N_G(V(C)) \subseteq \alpha(A_T(x))$, where x is an anchor of C.*

In particular, Observation 3 implies that an anchor of each component of $G-X$ is unique. For a node $x \in V(T)$, we denote by $\mathcal{P}_x$ the set of components of $G-X$ anchored in x, and G_x denotes the subgraph of G induced by the vertices of the graphs of $\mathcal{P}_x$, that is, G_x is the union of the components of $G - X$ anchored in x. Clearly, $\mathcal{P}_x$ and G_x may be empty. Note that the anchors of the components of $G - X$ depend on the choice of a representation. Therefore, we use the above notation only when (T, α) is fixed and clear from the context.

Lemma 4. *Let φ be an FOL formula and let G be a connected graph with $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) > 0$. Let also d be a positive integer. Then $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) \le d$ if and only if G contains an elimination set X of depth at most $d - 1$ with a representation (T, α) such that the following is fulfilled:*

(i) *for every nonleaf node $x \in V(T)$, $C \models \varphi$ for every $C \in \mathcal{P}_x$,*
(ii) *for every leaf x of T with $\mathrm{depth}_T(x) \le d - 2$, either $G_x \models \varphi$ or $C \models \varphi$ for every $C \in \mathcal{P}_x$,*
(iii) *for every leaf x of T with $\mathrm{depth}_T(x) = d - 1$, $G_x \models \varphi$.*

Proof. The lemma is proved similarly to Lemma 3. We begin by showing that if $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) \le d$, then G has an elimination set X of depth at most $d - 1$ with a representation (T, α) such that conditions (i)–(iii) are fulfilled. For this, we inductively construct X and (T, α) with $\mathrm{depth}(T) \le d - 1$ using the definition of $\mathrm{ed}_\varphi^{\mathrm{prop}}(G)$.

Since G is connected and $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) > 0$, there is $v \in V(G)$ such that $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) = 1 + \mathrm{ed}_\varphi^{\mathrm{conn}}(G - v)$. We construct a node r of T and set it be the root. If either $G - v \models \varphi$ or $C \models \varphi$ for every component C of $G - v$, then the construction of X and T is completed and we define $\alpha(r) = v$. Note that $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) = 1$ in the first case and $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) = 2$ in the second. This implies that (i)–(iii) are fulfilled. Assume from now on that this is not the case.

Denote by $C_1, \ldots, C_s$ the components of $G - v$ such that $C_i \not\models \varphi$ for $i \in \{1, \ldots, s\}$. By definition, $\mathrm{ed}_\varphi^{\mathrm{prop}}(C_i) \le d - 1$ for $i \in \{1, \ldots, s\}$. Notice that for each component C of $G - v$ distinct from $C_1, \ldots, C_s$, $C \models \varphi$. Then, we can assume inductively that for every $i \in \{1, \ldots, s\}$, there is an elimination set $X_i \subseteq V(C_i)$ of depth at most $d - 2$ with respect to C_i with a representation (T_i, α_i) such that conditions (i)–(iii) are fulfilled (for $d := d - 1$). Let r_i be the root of T_i for $i \in \{1, \ldots, s\}$. We define $X = \{v\} \cup \bigcup_{i=1}^{s} X_i$, and construct T from $T_1, \ldots, T_s$ by making $r_1, \ldots, r_s$ the children of r. Then, we set

$$\alpha(x) = \begin{cases} v, & \text{if } x = r, \\ \alpha_i(x), & \text{if } x \in V(C_i) \text{ for some } i \in \{1, \ldots, s\}. \end{cases}$$

Using the inductive assumptions that (i)–(iii) are fulfilled for X_i with (T_i, α_i) for every $i \in \{1, \ldots, s\}$ and the observation that $\mathcal{P}_r$ consists of the components of $G - v$ distinct from $C_1, \ldots, C_s$, we show that (i)–(iii) are fulfilled for X and the representation (T, α).

For $x = r$, (i) holds, because $C \models \varphi$ for every $C \in \mathcal{P}_r$ by the construction of T. If $x \ne r$ is a nonleaf vertex of T, then x is a nonleaf vertex of T_i for some $i \in \{1, \ldots, s\}$. By the inductive assumption, $C \models \varphi$ for every component of $G_i - X_i$ anchored in x with respect to T_i. By the construction of T, every $C \in \mathcal{P}_x$ is a component of $G_i - X_i$ anchored in x implying that (i) is fulfilled. To see (ii), let x be a leaf of T with $\mathrm{depth}_T(x) \le d - 2$. Then x is a leaf of T_i for some $i \in \{1, \ldots, s\}$ and $\mathrm{depth}_{T_i}(x) \le d - 3$. By induction, either $(G_i)_x \models \varphi$ or $C \models \varphi$ for every component C of $G_i - X_i$ anchored in x with respect to T_i. Note that $(G_i)_x = G_x$ and every $C \in \mathcal{P}_x$ is a component of $G_i - X_i$ anchored in x with respect to T_i. Therefore, (ii) holds. For (iii), the arguments are almost the same. If x is a leaf of T with $\mathrm{depth}_T(x) \le d - 1$, then x is a leaf of T_i for some $i \in \{1, \ldots, s\}$ and $\mathrm{depth}_{T_i}(x) \le d - 2$. By the assumption, $(G_i)_x \models \varphi$ with respect to T_i. Since $(G_i)_x = G_x$, we have that (iii) holds.

To show the implication in the opposite direction, assume that X is an elimination set of depth at most $d - 1$ with a representation (T, α) satisfying (i)–(iii). By the second claim of Lemma 1, we can assume that T is nice. We show that $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) \le d$ by induction on $\mathrm{depth}(T)$.

Suppose that $\text{depth}(T) = 0$, that is, the depth of an elimination set X is zero and, therefore, $X = \{u\}$ for some $u \in V(G)$. If $d = 1$, then $G_u \models \varphi$ and $\text{ed}_\varphi^{\text{prop}}(G) = 1$. If $d \geq 2$, then either $G_u \models \varphi$ or $C \models \varphi$ for every component C of $G - u$. In both cases, $\text{ed}_\varphi^{\text{prop}}(G) \leq 2$ by the definition of $\text{ed}_\varphi^{\text{prop}}(G)$.

Assume that $\text{depth}(T) \geq 1$. In particular, $d \geq 2$. Since G is connected and $\text{ed}_\varphi^{\text{prop}}(G) > 0$, $\text{ed}_\varphi^{\text{prop}}(G) = 1 + \min_{v \in V(G)} \text{ed}_\varphi^{\text{prop}}(G - v) \leq 1 + \text{ed}_\varphi^{\text{prop}}(G - u)$ and it is sufficient to show that $\text{ed}_\varphi^{\text{prop}}(G - u) \leq d - 1$. If $G - u \models \varphi$, then $\text{ed}_\varphi^{\text{prop}}(G) = 1 \leq d$. Assume from now on that $G - u \not\models \varphi$. Then, by the definition of $\text{ed}_\varphi^{\text{conn}}(G)$, it is sufficient to show that $\text{ed}_\varphi^{\text{prop}}(C) \leq d - 2$ for every component C of $G - u$.

If $V(C) \cap X = \emptyset$ for a component C of $G - u$, then $C \in \mathcal{P}_r$ and $C \models \varphi$. Then $\text{ed}_\varphi^{\text{prop}}(C) = 0 \leq d - 2$. Consider the components $C_1, \dots, C_s$ of $G - u$ such that $V(C_i) \cap X \neq \emptyset$. Because (T, α) is nice, r has s children $x_1, \dots, x_s$ such that for every $i \in \{1, \dots, s\}$, $\alpha(V(T_i)) \subseteq V(C_i)$, where T_i is the subtree of T rooted in x_i. Let $\alpha_i : V(T_i) \to V(C_i)$ be the restriction of α on $V(T_i)$ for $i \in \{1, \dots, s\}$. Consider $i \in \{1, \dots, s\}$. We have that (T_i, α_i) is a representation of $X_i = X \cap V(C_i)$ satisfying (i)–(iii). Notice that $\text{depth}(T_i) < \text{depth}(T)$. Then by the inductive assumption $\text{ed}_\varphi^{\text{prop}}(C_i) \leq d - 1$. Therefore, $\text{ed}_\varphi^{\text{prop}}(G - u) \leq d - 1$ and this concludes the proof. □

Lemmas 3 and 4 demonstrate that $\text{ed}_\varphi^{\text{conn}}(G)$ and $\text{ed}_\varphi^{\text{prop}}(G)$, respectively, can be defined via the deletion of an elimination set. We also use these results to define a third variant of the elimination distance.

Definition 2 (Elimination Distance $\text{ed}_\varphi^{\text{depth}}$). Let φ be an FOL formula. For a graph G, $\text{ed}_\varphi^{\text{depth}}(G)$ is the minimum d such that G has an elimination set $X \subseteq V(G)$ of depth $d - 1$ such that $G - X \models \varphi$.

Notice that if the trees in the considered representations of elimination sets are constrained to be paths, then we obtain the classical *deletion distance*, that is, the minimum size of a set $X \subseteq V(G)$ such that $G - X \models \varphi$. Thus, $\text{ed}_\varphi^{\text{depth}}$ is a direct generalization of this classical distance, where the measure of the distance is not the size of X but the depth of X. We defined the depth of a set $X \subseteq V(G)$ using a representation. However, there is an equivalent definition that uses the notion of *tree-depth* (see, e.g., [28] for the definition); in fact, the tree-depth of G equals $\text{ed}_{\mathcal{P}}(G)$, where $\mathcal{P}$ is the class of empty graphs. Let G be a graph and let $X \subseteq V(G)$. Recall that the torso of X is the graph H obtained from $G[X]$ by making every two vertices $u, v \in X$ adjacent if three is a component C of $G - X$ such that $u, v \in N_G(V(C))$. Then the following property can be seen from [5, 6] and also can be easily shown by the definitions of tree-depth and the depth of an elimination set.

Observation 4. *For a set $X \subseteq V(G)$ and an integer k,* $\text{depth}(X) \leq k - 1$ *if and only if the tree-depth of the torso of X is at most k.*

Thus, $\text{ed}_\varphi^{\text{depth}}$ is minimum k such that there is $X \subseteq V(G)$ whose torso has the tree-depth at most k and $G - X \models \varphi$.

Observe that $\text{ed}_\varphi^{\text{depth}}(G)$ is incomparable with $\text{ed}_\varphi^{\text{conn}}(G)$ and $\text{ed}_\varphi^{\text{prop}}(G)$. To see this, consider first $\varphi = \forall x \forall y\, [(x = y) \vee (x \sim y))]$ defining the property that a graph is complete. Consider the graph G_n constructed from two disjoint copies H_1 and H_2 of K_n for $n \geq 1$ by adding a vertex w and making it adjacent to every vertex of H_1 and H_2. Notice that to obtain a complete graph from G, we have to delete either the vertices of H_1 or H_2. Because these graphs are complete, $\text{ed}_\varphi^{\text{depth}}(G_n) = n$. However, the deletion of w results in the graph whose components are complete graphs. Hence, $\text{ed}_\varphi^{\text{conn}}(G_n) = 1$ and it can be seen that $\text{ed}_\varphi^{\text{prop}}(G_n) = 2$. For the opposite direction, consider

$$\varphi = \forall x \forall y \forall z \exists u \exists v\, [((x \sim y) \wedge (y \sim z)) \to ((x = z) \vee (x \sim z))] \wedge \neg(u = v) \wedge \neg(u \sim v)].$$

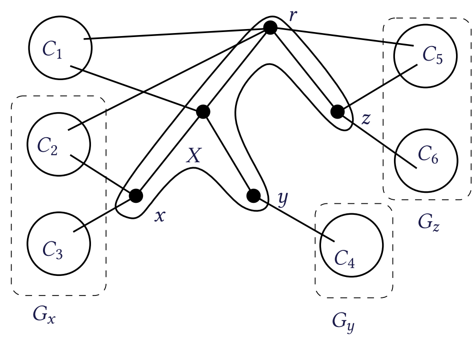

Fig. 1. Conditions for components $C_1, \ldots, C_6$ of $G - X$; $d = 3$.

This formula defines the property that a graph does not contain an induced path on three vertices and has at least two nonadjacent vertices, that is, a graph is disjoint union of at least two complete graphs. Let G_n be the graph obtained from two copies H_1 and H_2 of K_n by joining a vertex w_1 of H_1 with a vertex w_4 of H_2 by a path $w_1w_2w_3w_4$. Since $G_n - \{w_2, w_3\} \models \varphi$ and for every $w \in V(G_n)$, $G_n - w \not\models \varphi$, we have that $\mathrm{ed}_\varphi^{\mathrm{depth}}(G_n) = 2$. We claim that $\mathrm{ed}_\varphi^{\mathrm{conn}}(G_n), \mathrm{ed}_\varphi^{\mathrm{prop}}(G_n) \geq n$. The claim is trivial for $n = 1$. Assume that $n \geq 2$ and consider $w \in V(G_n)$. If $w \notin \{w_1, \ldots, w_4\}$, then $\mathrm{ed}_\varphi^{\mathrm{conn}}(G_n - w) \geq \mathrm{ed}_\varphi^{\mathrm{conn}}(G_{n-1}) \geq n - 1$. Similarly, $\mathrm{ed}_\varphi^{\mathrm{prop}}(G_n - w) \geq \mathrm{ed}_\varphi^{\mathrm{prop}}(G_{n-1}) \geq n - 1$. If $w \in \{w_1, \ldots, w_4\}$, then neither $(G_n - w) \models \varphi$ nor $C \models \varphi$ for every component C of $G_n - w$. Notice that $G_n - w$ has a component that is a complete graph with either $n - 1$ or n vertices. For this component C, we have that $\mathrm{ed}_\varphi^{\mathrm{conn}}(C) = \mathrm{ed}_\varphi^{\mathrm{prop}}(C) \geq n-1$. We conclude that $\mathrm{ed}_\varphi^{\mathrm{conn}}(G_n - w) \geq n-1$ and $\mathrm{ed}_\varphi^{\mathrm{prop}}(G_n - w) \geq n - 1$ for every choice $w \in V(G_n)$. Therefore, $\mathrm{ed}_\varphi^{\mathrm{conn}}(G_n), \mathrm{ed}_\varphi^{\mathrm{prop}}(G_n) \geq n$ (in fact, $\mathrm{ed}_\varphi^{\mathrm{conn}}(G_n) = \mathrm{ed}_\varphi^{\mathrm{prop}}(G_n) = n$).

Summarizing, now we have three definitions of eliminations distances via elimination sets. We use Figure 1 to illustrate the differences for the requirements for $G - X$ for a given elimination set X. For $\mathrm{ed}_\varphi^{\mathrm{conn}}$, it should hold that $C_i \models \varphi$ for every $i \in \{1, \ldots, 6\}$. For $\mathrm{ed}_\varphi^{\mathrm{prop}}$, we require that (i) $C_1 \models \varphi$, (ii) $C_5 \models \varphi$ and $C_6 \models \varphi$ or $G_z \models \varphi$ and (ii) $G_x \models \varphi$ and $G_y \models \varphi$. For $\mathrm{ed}_\varphi^{\mathrm{depth}}$, $H \models \varphi$, where $H = G - X$ is the union of $C_1 \ldots, C_6$.

Given an FOL formula φ, we define the following three variants of the Elimination Distance problems for $\star \in \{\mathrm{conn}, \mathrm{prop}, \mathrm{depth}\}$:

Elimination Distance–($\star$) to φ parameterized by k

Input:	A graph G and a nonnegative integer k.
Task:	Decide whether $\mathrm{ed}_\varphi^{\star}(G) \leq k$.

These problems are closely connected to the Deletion to φ problem for a formula φ that asks, given a graph G and a nonnegative integer k, whether there is a set S of size at most k such that $G - S \models \varphi$. In particular, Observation 3 implies the following:

Observation 5. *Deletion to φ and Elimination Distance–($\star$) to φ for $\star \in \{\mathrm{conn}, \mathrm{prop}, \mathrm{depth}\}$ are equivalent on instances (G, k), where G is a $(k + 1)$-connected graph.*

4 AN FPT ALGORITHM FOR Σ_3-FORMULAS

In this section, we show the main algorithmic result, Theorem 1, that Elimination Distance–($\star$) to φ is FPT for formulas from Σ_3. Now, we state this theorem formally.

Theorem 1. *For every FOL formula* $\varphi \in \Sigma_3$, *Elimination Distance–*($\star$) *to* φ *can be solved in* $f(k) \cdot n^{O(|\varphi|)}$ *time for each* $\star \in \{\mathrm{conn}, \mathrm{prop}, \mathrm{depth}\}$.

We prove the theorem using the *recursive understanding* technique introduced by Chitnis et al. [8]. It was recently demonstrated by Agrawal et al. [1] that this approach is useful for elimination problems. As we are interested in the quality result, we apply the meta theorem of Lokshtanov et al. [26] (see the arXiv version [27] for more details). This considerably simplifies the arguments, but makes the proof nonconstructive. Moreover, we only show the existence of nonuniform FPT algorithms. However, we conjecture that it may be possible (however, very technical) to show the theorem in a constructive way by giving uniform algorithm by either using the original approach of Chitnis et al. [8] or the dynamic programming scheme proposed by Cygan et al. [11].

The remaining part of the section contains the proof of Theorem 1. In Section 4.1, we introduce the notation and provide auxiliary results needed to apply the recursive understanding technique, and in Section 4.2, we prove that the Elimination Distance–($\star$) to φ is FPT for the key case when the input graphs cannot be partitioned in big parts by separators of bounded size.

4.1 Recursive Understanding

Let G be a graph. A pair (A, B), where $A, B \subseteq V(G)$ and $A \cup B = V(G)$, is called a *separation* of G if there is no edge uv with $u \in A \backslash B$ and $v \in B \backslash A$. In other words, $A \cap B$ is a (u, v)-separator for every $u \in A \backslash B$ and $v \in B \backslash A$. The *order* of (A, B) is $|A \cap B|$.

Let p, q be positive integers. A graph G is said to be (p, q)-*unbreakable* if for every separation (A, B) of G of order at most q, either $|A \backslash B| \leq p$ or $|B \backslash A| \leq p$, that is, G has no separator of size at most q that partitions the graph into two parts of size at least $p + 1$ each.

We state the restricted variant of the meta theorem of Lokshtanov et al. [26]. Lokshtanov et al. proved the theorem for structures and counting monadic second-order logic. For us, it is sufficient to state the theorem for graphs and MSOL.

Theorem 3 ([26, Theorem 1]). *Let* ψ *be an MSOL formula. For all* $q \in \mathbb{N}$, *there exists* $p \in \mathbb{N}$ *such that if there exists an algorithm that solves Model Checking for* ψ *on* (p, q)*-unbreakable graphs in* $O(n^d)$ *time for some* $d \geq 4$, *then Model Checking for* ψ *can be solved on general graphs in* $O(n^d)$ *time.*

It is crucial that the considered problems may be expressed in MSOL. The fact that the property that $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(G) \leq k$ can be expressed in MSOL was already used by Lindermayr et al. [25]. We provide a short unifying proof for $\mathrm{ed}_{\varphi}^{\mathrm{conn}}(G) \leq k$ and $\mathrm{ed}_{\varphi}^{\mathrm{prop}}(G) \leq k$ in our terms. For $\mathrm{ed}_{\varphi}^{\mathrm{depth}}(G) \leq k$, the arguments are different.

Lemma 5. *For every FOL formula* φ, *every* $\star \in \{\mathrm{conn}, \mathrm{prop}, \mathrm{depth}\}$, *and every integer* $k \geq 0$, *there is an MSOL formula* $\psi_k^{\star}$ *such that for each graph* G, $G \models \psi_k^{\star}$ *if and only if* $\mathrm{ed}_{\varphi}^{\star}(G) \leq k$.

Proof. We use capital letters to denote vertex set variables and small letters are used for vertex variables. To simplify notation, we introduce some auxiliary formulas. Notice that we can express that $Z = X \cap Y$ in MSOL and we write $X \cap Y$ for such an expression. Similarly, we write $X - Y$ to express that $Z = X \backslash Y$, and we write $X - y$ for $X \backslash \{y\}$. Also $\overline{X}$ is used for the complement of X. It is well-known that the connectivity property can be expressed in MSOL, because of the following observation: A set $X \subseteq V(G)$ induces a connected subgraph of G if and only if for every partition (U, W) of X, there is an edge $uw \in E(G)$ such that $u \in U$ and $w \in W$. Then, we can observe that for every $X \subseteq V(G)$, $G[X]$ is a component of G if and only if X induces a connected subgraph but for every $v \in V(G) \backslash X$, $G[X \cup \{v\}]$ is not a connected graph. This allows us to use the MSOL formula $\mathrm{comp}(X)$ with a free variable X expressing the property that X induces a component.

Clearly, every FOL formula is an MSOL formula. In particular, this means that we can construct the MSOL formula $\varphi(X)$ for a free variable X expressing the property that the subgraph induced by X models φ.

First, we show the lemma for $\star \in \{\text{conn}, \text{prop}\}$ using the definitions. For this, we inductively construct ψ_k^{conn} and ψ_k^{prop}.

It is easy to see that for $k = 0$, $\psi_0^{\text{conn}} = \forall X\ \text{comp}(X) \to \varphi(X)$.

Now let $k \geq 1$ and assume that ψ_{k-1}^{conn} is constructed. Then, we can define the MSOL formula $\psi_{k-1}^{\text{conn}}(X)$ for a free variable X expressing the property that the subgraph induced by X models ψ_{k-1}^{conn}. Then it is straightforward to verify that

$$\psi_k^{\text{conn}} = \psi_{k-1}^{\text{conn}} \vee (\forall X\ \text{comp}(X) \to (\exists x (x \in X) \wedge \psi_{k-1}^{\text{conn}}(X - x))).$$

Next, we construct ψ_k^{prop} for $k \geq 0$. It is straightforward to see that $\psi_0^{\text{prop}} = \varphi$ and

$$\psi_1^{\text{prop}} = \psi_0^{\text{prop}} \vee \Big(\forall X\ \text{comp}(X) \to (\psi_0^{\text{prop}}(X) \vee (\exists x\ (x \in X) \wedge \psi_0^{\text{prop}}(X - x)))\Big).$$

Then for $k \geq 2$,

$$\psi_k^{\text{prop}} = \psi_{k-1}^{\text{prop}} \vee \Big(\forall X\ \text{comp}(X) \to (\exists x\ (x \in X) \wedge \psi_{k-1}^{\text{prop}}(X - x))\Big),$$

where $\psi_{k-1}^{\text{prop}}(X)$ for a free variable X expresses the property that the subgraph induced by X models ψ_{k-1}^{prop}.

Finally, we prove the claim for ψ_k^{depth}. Here, the proof is more complicated and uses Lemmas 1 and 2. We express the property that X is an elimination set of set at most d.

By Lemma 1, if G is a connected graph and $d \geq 0$, then $\text{depth}(X) \leq d$ if and only if X has a nice representation of depth at most d. For a free variable X and an integer $d \geq -1$, we define the formula $\xi_d(X)$ expressing that X has a nice representation (T, α) of depth at most d. For $d = -1$, $\xi_d(X) = (X = \emptyset)$, and for $d = 0$, $\xi_d(X) = (|X| = 1)$ by the definition (clearly, the property $|X| = 1$ can be expressed in MSOL). Assume that $d \geq 1$ and $\xi_{d-1}(X)$ is already constructed. Additionally, we assume that we are given the formula $\xi_{d-1}(X, Y)$ that expresses the property that X has a nice representation of depth at most $d - 1$ in the subgraph induced by Y. For this, we observe that $\xi_{d-1}(X, Y)$ can be constructed from $\xi_{d-1}(X)$ in a straightforward way. Also, we use $\text{comp}(Y, x)$ to denote the formula expressing that Y induces a component of the subgraph obtained by the deletion of x. Then

$$\xi_d(X) = \xi_{d-1}(X) \vee \Big(\exists x\ (x \in X) \wedge (\forall Y\ (\text{comp}(Y, x) \wedge (X \cap Y \neq \emptyset)) \to \xi_{d-1}(X \cap Y, Y))\Big).$$

To see this, it is sufficient to observe that $\exists x\ (x \in X) \wedge (\forall Y\ (\text{comp}(Y, x) \wedge (X \cap Y \neq \emptyset)) \to \xi_{d-1}(X \cap Y, Y))$ expresses that G has a vertex $x \in X$ such that for the root r of T, $x = \alpha(r)$, and in each component C of $G - x$ containing some vertices of X, there is a subtree of T of depth at most $d - 1$ that can be used to represent $V(C) \cap X$ in C.

Now, we construct the formula $\tilde{\xi}_d$ that expresses that X is an elimination set of depth at most d using Lemma 2. It is easy to see that $\tilde{\xi}_{-1}(X) = (X = \emptyset)$ and $\tilde{\xi}_d(X) = (|X| = 1)$. Assume that $d \geq 1$, $\tilde{\xi}_{d-1}(X)$ is already constructed, and we have a formula $\tilde{\xi}_{d-1}(X, Y)$ expressing that the depth of X is at most $d - 1$ in the subgraph induced by Y. Then by Lemma 2,

$$\begin{aligned}\tilde{\xi}_d(X) &= \tilde{\xi}_{d-1}(X)\\ &\quad \vee(\exists Y\ \text{comp}(Y) \wedge \tilde{\xi}_d(X \cap Y, Y) \wedge (\forall Z\ (\text{comp}(Z) \wedge (Z \neq Y)) \to \tilde{\xi}_{d-1}(X \cap Z, Z))).\end{aligned}$$

Using $\tilde{\xi}_d$ for $d \geq -1$, we can write ψ_k^{depth} for $k \geq 0$ as follows:

$$\psi_k^{\text{depth}} = \exists X\ \tilde{\xi}_{k-1} \wedge \varphi(\overline{X}).$$

This completes the proof. □

Theorem 3 and Lemma 5 allow us to reduce the proof of Theorem 1 to solving Elimination Distance−($\star$) to φ for $\star \in \{\text{conn}, \text{prop}, \text{depth}\}$ on unbreakable graphs. For this, we show that any elimination set in an unbreakable graph has bounded size.

Lemma 6. *Let G be a (p, q)-unbreakable graph for positive integers p and q with $|V(G)| > (3p + 2q)(p + 1)$. Let also $X \subseteq V(G)$ be an elimination set of depth at most $d \leq q - 1$. Then $|X| \leq p + q$. Furthermore, there is a unique component C of $G - X$ with at least $p + 1$ vertices and $|V(G)\setminus N_G[V(C)]| \leq p$.*

Proof. Let (T, α) be a representation of X with $\text{depth}(T) \leq d$. Denote by r the root of T.

First, we show the weaker bound $|X| \leq 3p + 2q$.

For the sake of contradiction, assume that $|X| \geq 3p + 2q + 1$. Because T is a tree, it has a node x such that every component of $T - x$ has at most $\frac{1}{2}|V(T)|$ nodes. Let $S = A_T(x)$ and $S' = \alpha(A_T(x))$. Since $\text{depth}(T) \leq d \leq q - 1$, $|S| = |S'| \leq q$. By the definition of a representation, for every two distinct components C and C' of $T - S$, and every $x \in \alpha(V(C))$ and $y \in \alpha(V(C'))$, S' is an (x, y)-separator in G.

We now claim that every component C of $T-S$ has at most p nodes. Suppose to the contrary that there is a component C of $T - S$ with at least $p + 1$ nodes. Consider the components $C_1, \ldots, C_s$ of $G-S'$ such that $V(C_i) \cap \alpha(V(C)) \neq \emptyset$. Define $A = S \cup \bigcup_{i=1}^{s} V(C_i)$. Note that $|A\setminus S| \geq |V(C)| \geq p+1$. Let $Y = V(T)\setminus(S \cup V(C))$. By the choice of x, $|V(C)| \leq \frac{1}{2}|V(T)|$. Then $|Y| \geq \frac{1}{2}|V(T)| - q \geq (\frac{3}{2}p + q + \frac{1}{2}) - q = \frac{3}{2}p + \frac{1}{2} \geq p + 1$. Observe that for every node $y \in Y$, $\alpha(y) \notin V(C_i)$ for $i \in \{1, \ldots, s\}$. Then $\alpha(Y) \subseteq V(G)\setminus A$ and $|V(G)\setminus A| \geq p + 1$. For $B = (V(G)\setminus A) \cup S'$, we have that (A, B) is a separation of G with $S' = A \cap B$. In particular, (A, B) is a separation of order at most q. However, $|A\setminus B| \geq p + 1$ and $|B\setminus A|$. This contradicts the unbreakability condition and the claim follows.

Denote by $C_1, \ldots, C_s$ the components of $T - S$. Consider a set of indices $I \subseteq \{1, \ldots, s\}$ such that $|\bigcup_{i\in I} V(C_i)| \geq p + 1$ and for every proper $I' \subset I$, $|\bigcup_{i\in I'} V(C_i)| \leq p$. Such a set I exists, because $|\bigcup_{i=1}^{s} V(C_i)| \geq |V(T)| - q \geq 3p + q + 1$. Since each component has at most p nodes, we have that $|\bigcup_{i\in I} V(C_i)| \leq 2p$. Then, because $|V(T)\setminus S| \geq 3p + 1$, $|\bigcup_{i\in\{1,\ldots,s\}\setminus I} V(C_i)| \geq p + 1$.

Consider the components $C'_1, \ldots, C'_t$ of $G - S'$ that contain at least one vertex of $\alpha(V(C_i))$ for some $i \in I$. Define $A = S' \cup \bigcup_{i=1}^{t} V(C'_i)$. Note that $|A\setminus S'| \geq p+1$, because $|\bigcup_{i\in I} V(C_i)| \geq p+1$. Let $B = (V(G)\setminus A) \cup S'$. Since $\bigcup_{i\in\{1,\ldots,s\}\setminus I} \alpha(V(C_i)) \subseteq B\setminus S$, $|B\setminus S| \geq p + 1$. Then, we obtain that (A, B) is a separation of G of order at most q with $|A\setminus B| \geq p + 1$ and $|B\setminus A| \geq p + 1$; a contradiction. This concludes the proof of our claim that $|X| \leq 3p + 2q$.

Now, we improve the obtained upper bound. Because $|X| \leq 3p+2q$ and $|V(G)| > (3p+2q)(p+1)$, $|V(G)\setminus X| > (3p + 2q)p$. Observe that for the set of leaves $L(T)$, we have that $|L(T)| \leq 3p + 2q$. By Observation 3, it holds that for every component C of $G-X$, $N_G(V(C)) \subseteq A_T(x)$ for some $x \in L(T)$. By the pigeonhole principle, we conclude that there is $x \in L(T)$ such that for the components $C_1, \ldots, C_s$ of $G - X$ with $N_G(V(C_i)) \subseteq A_T(x)$ for $i \in \{1, \ldots, s\}$, it holds that $|\bigcup_{i=1}^{s} V(C_i)| \geq p + 1$. Let $S = \alpha(A_T(x))$. Note that $|S| \leq d + 1 \leq q$. Consider $A = S \cup \bigcup_{i=1}^{s} V(C_i)$ and $B = (V(G)\setminus A) \cup S$. We obtain that (A, B) is a separation of G of order at most q and $|A\setminus B| \geq p + 1$. Since G is (p, q)-unbreakable, we have that $|B| \leq p + q$. Notice that $X \subseteq B$. Thus, $|X| \leq p + q$.

To show the second claim, note that $|L(T)| \geq p + q$. In the same way as above, there is $x \in L(T)$ such that for the components $C_1, \ldots, C_s$ of $G - X$ with $N_G(V(C_i)) \subseteq A_T(x)$ for $i \in \{1, \ldots, s\}$, it holds that $|\bigcup_{i=1}^{s} V(C_i)| \geq p + 1$. We show that there is $i \in \{1, \ldots, s\}$ such that $|V(C_i)| \geq p + 1$. For the sake of contradiction, assume that $|V(C_i)| \leq p + 1$ for all $i \in \{1, \ldots, s\}$. Then there is a set of indices $I \subseteq \{1, \ldots, s\}$ such that $|\bigcup_{i\in I} V(C_i)| \geq p+1$ and for every proper $I' \subset I$, $|\bigcup_{i\in I'} V(C_i)| \leq p$. Because each component has at most p vertices, $|\bigcup_{i\in I} V(C_i)| \leq 2p$. Consider $A = S \cup \bigcup_{i=1}^{s} V(C_i)$, where $S = \alpha(A_T(x))$ and $B = (V(G)\setminus A) \cup S$. Note that $|B\setminus S| \geq |V(G)| - 2p - q \geq p + 1$. Then (A, B) is a separation of G of order at most q with $|A\setminus B| \geq p + 1$ and $|B\setminus A| \geq p + 1$; a contradiction with the condition that G is (p, q)-unbreakable. This implies that there is a component C of $G - X$ with $|V(C)| \geq p + 1$.

Because G is a (p, q)-unbreakable graph and $|N_G(V(C))| \leq q$, we have that $|V(G)\setminus N_G[V(C)]| \leq p$. To see it, it is sufficient to consider the separation (A, B) of G with $A = N_G[V(C)]$ and $B = V(G)\setminus V(C)$. Clearly, $|B\setminus A| \leq p$ and, therefore, $|V(G)\setminus N_G[V(C)]| \leq p$. This also implies the uniqueness of a component of $G - S$ with at least $p + 1$, because for every other component C', we have that $V(C') \subseteq B\setminus A$. This concludes the proof. □

Using the notation in Lemma 6, we say that a component C of $G - X$ with at least $p + 1$ vertices is *big* and the other components are *small*.

We can use backtracking to verify, given a set X, whether $\text{depth}(X) \leq d$. For this, we combine Lemmas 1 and 2 with backtracking and obtain the following straightforward lemma:

Lemma 7. *Given a graph G, a set of vertices $X \subseteq V(G)$, and an integer $d \geq -1$, it can be decided in $|X|^{O(d)} \cdot n^{O(1)}$ time whether* $\text{depth}(X) \leq d$.

We have to solve Elimination Distance−(⋆) to φ for $\star \in \{\text{conn}, \text{prop}\}$ on instances where the number of vertices of G is bounded by a polynomial of k. In this case, it is sufficient to have an XP algorithm. It is straightforward to see that such an algorithm can be constructed by backtracking following the definitions of $\text{ed}_\varphi^{\text{conn}}$ and $\text{ed}_\varphi^{\text{prop}}$.

Lemma 8. *Let φ be an FOL formula. Then Elimination Distance−(⋆) to φ can be solved in $n^{O(k+|\varphi|)}$ time for $\star \in \{\text{conn}, \text{prop}\}$.*

We also need the following technical lemma that will allow us to consider inclusion-minimal elimination sets.

Lemma 9. *Let G be a connected graph and let X be a nonempty elimination set with a nice representation (T, α). Let also C be a component of $G - X$ anchored in $x^* \in V(T)$. Suppose that $S \subseteq N_G(V(C))$ and C' is a component of $G - S$ with $V(C) \subseteq V(C')$. Then there is an elimination set $X' \subseteq X$ with a nice representation (T', α') such that $V(T') \subseteq V(T)$ and the following is fulfilled:*

(a) *$S \subseteq X'$ and $(N_G(V(C))\setminus S) \cap X' = \emptyset$,*
(b) *for every component H of $G - X'$, either $V(H) \subseteq V(C')$ or H is a component of $G - X$,*
(c) *for every node $y \in V(T')$, $\text{depth}_{T'}(y) \leq \text{depth}_T(y)$,*
(d) *if a component H of $G - X'$ distinct from C' is anchored in a leaf z of T, then H is anchored in z in T' and z is a leaf of T',*
(e) *if x^* is a leaf of T and $x^* \in S$, then C' is anchored in x^* in T'.*

Proof. Let $R = N_G(V(C))$. The claim is trivial if $S = \emptyset$, because $C' = G$ and we can take $X' = \emptyset$. We assume that this is not the case. The proof is by induction on $|R\setminus S|$. The claim is straightforward if $R = S$ as we can take $X' = X$ and consider the same representation (T, α). The crucial case is the case $|R\setminus S| = 1$. Let u be the unique vertex of $R\setminus S$. We consider two possibilities for u. Let $v = \alpha(r)$, where r is the root of T.

Case 1. $u = v$. Let $W = V(C')$. Notice that a vertex $w \in V(G)$ is in W if and only if either $w \in V(C)$ or $w \notin S \cup V(C)$ and $G - S$ has a (u, w)-path. Define $X' = X\setminus W$. Clearly, (a) holds for this X'. Observe that for a component H of $G - X$, we have that $V(H) \subseteq W$ if $N_G(V(H))$ contains a vertex of W and $V(H) \cap W = \emptyset$ otherwise. In particular, this implies (b).

Next, we construct T' and α'. We set $V(T') = \alpha^{-1}(X')$ and define $\alpha'(x) = \alpha(x)$ for every $x \in V(T')$. Because $S \neq \emptyset$, there is a descendant r' of r of minimum depth such that $\alpha(r') \in S$. For every $w \in X'$ distinct from $\alpha(r')$, we consider $x = \alpha^{-1}(w)$ and find a proper ancestor y of x in T of maximum depth such that $\alpha(y) \in X'$. Then, we define y be the parent of x.

We argue that T' is a tree rooted in r'. We have to show that for every $w \in X'$ distinct from $\alpha(r')$, we have an ancestor y of $x = \alpha^{-1}(w)$ in T such that $\alpha(y) \in X'$. For the sake of contradiction, assume that there is $w \in X'$ such that for every proper ancestors y of $x = \alpha^{-1}(w)$ in T, $\alpha(y) \notin X'$. Clearly, x is not a descendant of r' in T. In particular, r' and x are incomparable. Let z be the lowest proper ancestor of r' and y in T. We have that $\alpha(A_T(z))$ is an $(\alpha(r'), w)$-separator of G and, moreover, S has no vertices in the component $G - \alpha(A_T(z))$ containing $\alpha(x)$. Since (T, α) is nice, this component has an $(\alpha(z), w)$-path. Because $\alpha(z) \notin S$, $G - S$ has a $(u, \alpha(z))$-path. We conclude that $G - S$ has a (u, w)-path and $w \notin X'$; a contradiction. This proves that T' is a tree rooted in r'.

We prove that (T', α') represents X'. Towards a contradiction, assume that this is not the case, that is, there are distinct incomparable $x, y \in V(T')$ whose lowest common ancestor $z \neq x, y$ and $\alpha'(x)$ and $\alpha'(y)$ are in the same component of $G - A_{T'}(z)$. By the definition of T', z has a descendant z' in T such that $z' \neq x, y$ is the lowest common ancestor of x and y in T. Clearly, either $x \notin N_G(V(C))$ or $y \notin N_G(V(C))$. By symmetry, assume that $y \notin N_G(V(C))$. Because (T, α) is a representation of X, $\alpha(A_T(z'))$ is an $(\alpha(x), \alpha(y))$-separator. This means that every $(\alpha'(x), \alpha'(y))$-path in G contains a vertex of $(\alpha(x), \alpha(y))$. In particular, this implies that there is a vertex $z'' \in A_T(z')$ such that G has an $(\alpha(z''), \alpha'(y))$-path P in G such that the internal vertices of the path are in the component $G - \alpha(A_T(z'))$ containing $\alpha'(y)$. Because $y \notin N_G(V(C))$, we have that P avoids the vertices of S. Since $z' \notin X'$, $G - S$ has a $(u, \alpha(z''))$-path P'. Concatenating P' and P, we obtain that $G - S$ has a $(u, \alpha(y'))$-path. However this contradicts that $\alpha'(y) \in X'$. This proves that (T', α') represents X'.

By the construction of T', it is easy to see that T' is nice, because T is nice. Also the construction of T' immediately implies (c)–(e). This concludes the analysis of the first case.

Case 2. $u \neq v$. We show the claim by induction on $d = \text{depth}(T)$. Notice that $\text{depth}(T) \geq 1$, because $X \setminus \{v\} \neq \emptyset$. Let $C_1, \dots, C_s$ be the components of $G - v$ such that $X_i = X \cap V(C_i) \neq \emptyset$ for $i \in \{1, \dots, s\}$. Because T is nice, T has children $r_1, \dots, r_s$ such that for every $i \in \{1, \dots, s\}$, the subtree T_i of T rooted in r_i together with $\alpha_i(x) = \alpha(x)$ for $x \in V(T_i)$ represent X_i in C_i. By Observation 3, we can assume without loss of generality that $V(C) \subseteq V(C_1)$ and $N_G(V(C)) \subseteq V(C_1) \cup \{v\}$. Because $\text{depth}(T_1) < \text{depth}(T)$, we can apply the inductive assumption and construct an elimination set $X'_1 \subset X_1$ with a nice representation (T'_1, α'_1) satisfying (a)–(e). Then, we construct $X' = X'_1 \cup \bigcup_{i=1}^{s} X_i$. Then, we construct T' from T'_1 and $T_2, \dots, T_s$ by making r'_1 and $r_2, \dots, r_s$ the children of r, where r'_1 is the root of T'_1. We set

$$\alpha'(x) = \begin{cases} \alpha_i(x), & \text{if } x \in X_i \text{ for some } i \in \{2, \dots, s\}, \\ \alpha'_1(x), & \text{if } x \in X'_1. \end{cases}$$

It is straightforward to verify that X' and (T', α') satisfy (a)–(e).

This concludes the proof for the base case $|R \setminus S| = 1$. To show the claim for $|R \setminus S| > 1$, consider a vertex $w \in R \setminus S$ and apply the claim for $S' = S \cup \{w\}$ using the inductive assumption. We have that there is an elimination set $X' \subseteq X$ with a nice representation (T', α') such that $V(T') \subseteq V(T)$ and (a)–(e) are fulfilled with respect to S'. Then, we apply the claim for X' and (T', α') with respect to the component C' and S. Clearly, we obtain an elimination set $X'' \subseteq X' \subseteq X$ with a nice representation (T'', α'') such that $V(T'') \subseteq V(T') \subseteq V(T)$ satisfying (a)–(e). This completes the proof. □

In our algorithms, we use the *random separation* technique introduced by Cai, Chan, and Chan in [7]. To avoid dealing with randomized algorithms, we use the following lemma stated by Chitnis et al. in [8]:

Lemma 10 ([8]). *Given a set U of size n and integers $0 \leq a, b \leq n$, one can construct in time $2^{O(\min\{a,b\}\log(a+b))} \cdot n \log n$ a family $\mathcal{F}$ of at most $2^{O(\min\{a,b\}\log(a+b))} \cdot \log n$ subsets of U such that the following holds: For any sets $A, B \subseteq U$, $A \cap B = \emptyset$, $|A| \leq a$, $|B| \leq b$, there exists a set $R \in \mathcal{F}$ with $A \subseteq R$ and $B \cap R = \emptyset$.*

4.2 Algorithm for Unbreakable Graphs

In this subsection, we give FPT-algorithms for Elimination Distance–($\star$) to φ for every $\star \in \{\text{conn}, \text{prop}, \text{depth}\}$ for FOL formulas $\varphi \in \Sigma_3$ on unbreakable graphs. Throughout the subsection, we assume without loss of generality that

$$\varphi = \exists x_1 \cdots \exists x_r \forall y_1 \cdots \forall y_s \exists z_1 \cdots \exists z_t \ \chi,$$

where χ is quantifier-free and r, s, t are positive integers, because we always can write an FOL formula from Σ_3 in this form by adding dummy variables if necessary. We also write $\mathbf{x} = \langle x_1, \ldots, x_r \rangle$, $\mathbf{y} = \langle y_1, \ldots, y_s \rangle$, and $\mathbf{z} = \langle z_1, \ldots, z_t \rangle$.

Notice that $\text{ed}_\varphi^{\text{conn}}(G) = 0$ if and only if for every component C of G, $C \models \varphi$. Also $\text{ed}_\varphi^{\text{prop}}(G) = 0$ ($\text{ed}_\varphi^{\text{depth}}(G) = 0$, respectively) if and only if $G \models \varphi$. This implies that Elimination Distance–($\star$) to φ for $\star \in \{\text{conn}, \text{prop}, \text{depth}\}$ can be solved in time $n^{O(|\varphi|)}$ if $k = 0$ by Observation 1, that is, Theorem 1 trivially holds for $k = 0$. Hence, throughout this subsection, we assume that the parameter k in the considered instances is positive.

By Theorem 3 and Lemma 5, to prove Theorem 1, it is sufficient to demonstrate an FPT algorithm for the considered problems on $(p(k), k)$-unbreakable graphs for a computable function $p\colon \mathbb{N} \to \mathbb{N}$. Slightly abusing notation, we write p instead of $p(k)$.

The algorithms for Elimination Distance–($\star$) to φ for $\star \in \{\text{conn}, \text{prop}, \text{depth}\}$ exploit the same ideas and have similar structures. However, there are differences that make it hard to give a unified description. In particular, for Elimination Distance–(prop) to φ, we have to deal with the properties of elimination sets and their representations given in Lemma 4, and this makes various details of our algorithm for this problem different from the algorithm for Elimination Distance–(conn) to φ. Furthermore, our algorithm for Elimination Distance–(depth) to φ is sufficiently different form the other two, because in this case, we aim for quite different properties of the graph obtained by deleting an elimination set and this rules out a common description. Hence, we first give the details of the algorithm for Elimination Distance–(conn) to φ and then more briefly explain our algorithm for Elimination Distance–(prop) to φ. Further, we present our algorithm for Elimination Distance–(depth) to φ. Then, we derive Theorem 1 from Lemmas 12, 15, and 17 in which we summarize the properties of the algorithms for the considered problems.

*Algorithm for Elimination Distance–(*conn*) to φ.* Let (G, k) be an instance of Elimination Distance–(conn) to φ, where G is a (p, k)-unbreakable graph. We assume without loss of generality that G is connected. Otherwise, because $\text{ed}_\varphi^{\text{conn}}(G) = \max\{\text{ed}_\varphi^{\text{conn}}(C) \mid C \text{ a component of } G\}$, we can solve the problem for each component separately. If $|V(G)| \leq (3p+2k)(p+1)$, then we solve the problem in $(p+k)^{O(k+|\varphi|)}$ time by Lemma 8. From now on, we assume that $|V(G)| > (3p+2k)(p+1)$.

By Lemma 3, (G, k) is a yes-instance of Elimination Distance–(conn) to φ if and only if G contains an elimination set X of depth at most $k - 1$ such that $C \models \varphi$ for every component C of $G - X$. Our algorithm finds such a set X, called a *solution*, if it exists. We verify in $n^{O(|\varphi|)}$ time whether $X = \emptyset$ has the required property and return yes if this holds. Assume that this is not the case, that is, we have to find a nonempty solution.

Suppose that (G, k) is a yes-instance and let X be a solution with a representation (T, α). By Lemma 6, $|X| \leq p+k$ and there is a unique big component C of $G-X$ with at least $p+1$ vertices, the other components are small, and $|V(G) \setminus N_G[V(C)]| \leq p$. By Observation 3, $N_G(V(C)) \subseteq \alpha(A_T(x))$,

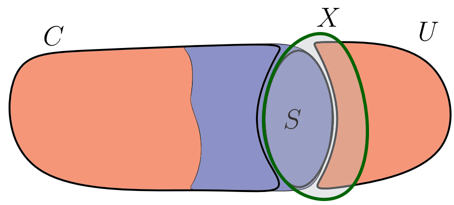

Fig. 2. A visualization of the set X, the component C, the sets S, and U, and the way the red and blue colors are distributed among them.

where x is an anchor of C. In particular, this means that $|N_G(V(C))| \leq k$. We use these properties to identify C. This is done by combining the random separation technique [7] with a recursive branching algorithm.

We use random separation to highlight the hypothetical sets $S = N_G(V(C))$ and $U = V(G) \setminus N_G[V(C)]$ (if they exist). To avoid randomized algorithms, we directly use the derandomization tool from Lemma 10. By this lemma, we can construct in $2^{O(\min\{p,k\}\log(p+k))} \cdot n \log n$ time a family $\mathcal{F}$ of at most $2^{O(\min\{p,k\}\log(p+k))} \cdot \log n$ subsets of $V(G)$ such that there is $R \in \mathcal{F}$ such that $U \subseteq R$ and $S \cap R = \emptyset$. In our algorithm, we go over all sets $R \in \mathcal{F}$ and for each set R, we check whether there is a solution X such that $U \subseteq R$ and $S \cap R = \emptyset$ for the sets S and U corresponding to X (recall that $S = N_G(V(C))$ and $U = V(G) \setminus N_G[V(C)]$, where C is the unique big component of $G-X$ with at least $p+1$ vertices). Clearly, (G, k) is a yes-instance of Elimination Distance–(conn) to φ if and only if there is $R \subseteq \mathcal{F}$ and a solution X with the required property.

From now on, we assume that $R \in \mathcal{F}$ is given. We set $B = V(G) \setminus R$. We say that the vertices of R are *red* and the vertices of B are *blue*. We also call the components of $G[R]$ *red components* of G and we use the same convention for induced subgraphs of G. A solution X is *colorful* if the vertices of U are red and the vertices of S are blue (see Figure 2). The crucial property of colorful solutions is that

> if a red vertex v is in U, then the set of vertices of the red component H containing v is a subset of U.

If $G-X$ has a big component C and $C \models \varphi$, then there is an r-tuple $\mathbf{v} = \langle v_1, \ldots, v_r \rangle$ of vertices of C such that $(C, \mathbf{v}) \models \varphi[\mathbf{x}]$ (recall that $\varphi[\mathbf{x}]$ is the formula with the free variables $x_1, \ldots, x_r$ obtained from φ by the removal of the quantification over $x_1, \ldots, x_r$, and $(C, \mathbf{v}) \models \varphi[\mathbf{x}]$ means that $\varphi[\mathbf{x}]$ evaluates *true* on G when x_i is assigned v_i for all $i \in \{1, \ldots, r\}$). Using brute force, we consider all r-tuples $\mathbf{v} = \langle v_1, \ldots, v_r \rangle$ of vertices of G, and for each $\mathbf{v}$, we explain how to check whether there is a colorful solution X with the big component C such that $v_i \in V(C)$ for all $i \in \{1, \ldots, r\}$. Note that at most n^r r-tuples $\mathbf{v}$ can be listed in $n^{O(|\varphi|)}$ time. The algorithm returns yes if we find a colorful solution for some choice of $\mathbf{v}$, and it concludes that there is no colorful solution for the considered selection of R otherwise.

From now on, we assume that $\mathbf{v} = \langle v_1, \ldots, v_r \rangle$ is fixed. Because these vertices should be in C, we temporarily (i.e., only for the current choice of $\mathbf{v}$) recolor them red to simplify further notation. We apply a recursive branching algorithm to find C and S.

We exploit the property that because $\varphi \in \Sigma_3$, the formula contains exactly one alternation from the existential quantifiers to universal. By definition, we have that $(C, \mathbf{v}) \models \varphi[\mathbf{x}]$ if and only if for every s-tuple $\mathbf{u} = \langle u_1, \ldots, u_s \rangle$ of vertices of C, $(C, \mathbf{vu}) \models \varphi[\mathbf{xy}]$. Suppose that $(C, \mathbf{v}) \not\models \varphi[\mathbf{x}]$. Then there is an s-tuple $\mathbf{u} = \langle u_1, \ldots, u_s \rangle$ of vertices such that $(C, \mathbf{vu}) \not\models \varphi[\mathbf{xy}]$. Notice now that, because $\varphi \in \Sigma_3$, we have that for any induced subgraph C' of C such that $v_i \in V(C')$ for every $i \in \{1, \ldots, r\}$, if $(C', \mathbf{v}) \models \varphi[\mathbf{x}]$, then there is $j \in \{1, \ldots, s\}$ such that $u_j \notin V(C')$. This implies that if $(C, \mathbf{vu}) \not\models \varphi[\mathbf{xy}]$, then there is $j \in \{1, \ldots, s\}$ such that either $u_j \in S$ and should be deleted or u_j is in a component of $G - S$ distinct from C and this component should be deleted together with its neighborhood. Note that u_j is blue in the first case, and u_j is red in the second. Moreover, in the second case, we should delete the red component containing u_j together with its blue neighborhood. We branch on all possible deletions of v_i's, using the following subroutine FindC(C, S, h), where we initially set $C := G$, $S := \emptyset$, and $h := k$:

Subroutine FindC(C, S, h).

- If $(C, \mathbf{v}) \models \varphi[\mathbf{x}]$ and $h \geq 0$, then return C, S, and stop.
- If $(C, \mathbf{v}) \not\models \varphi[\mathbf{x}]$ and $h \leq 0$, then stop.
- If $h \geq 1$ and there is an s-tuple $\mathbf{u} = \langle u_1, \ldots, u_s \rangle$ of vertices of C such that $(C, \mathbf{vu}) \not\models \varphi[\mathbf{xy}]$, then do the following for every $j \in \{1, \ldots, s\}$.
  - If $u_j \in B$ and there is a component C' of $C - u_j$ such that $v_i \in V(C')$ for all $i \in \{1, \ldots, r\}$, then call FindC$(C', S \cup \{u_j\}, h - 1)$.
  - If $u_j \in R$ and there is a red component H of C with the set of vertices W and $S' = N_C(W)$ such that (a) $u_j \in W$, (b) $|S'| \leq h$, and (c) there is a component C' of $C - N_C[W]$ with $v_i \in V(C')$ for all $i \in \{1, \ldots, r\}$, then call FindC$(C', S \cup S', h - |S'|)$.

We show the following lemma:

Lemma 11. *If X is an inclusion-minimal colorful solution to (G, k) with the big component C such that $v_i \in V(C)$ for all $i \in \{1, \ldots, s\}$ and $(C, \mathbf{v}) \models \varphi[\mathbf{x}]$, then there is a leaf of the search tree produced by* FindC$(G, \emptyset, k)$ *for which the subroutine outputs C and $S = N_G(V(C))$.*

Proof. To prove the lemma, we show the following claim: If the subroutine FindC is called for $(\tilde{C}, \tilde{S}, \tilde{h})$ such that (a) $V(C) \subseteq V(\tilde{C})$, (b) $\tilde{S} = N_G(V(\tilde{C}))$, (c) $\tilde{S} \subseteq S$, and (d) $\tilde{h} = k - |\tilde{S}|$, then either the subroutine outputs $\tilde{C}$ and $\tilde{S}$ or it recursively calls FindC$(\tilde{C}', \tilde{S}', \tilde{h}')$, where (a′) $V(C) \subseteq V(\tilde{C}')$, (b′) $\tilde{S} = N_G(V(\tilde{C}'))$, (c′) $\tilde{S}' \subseteq S$, and (d′) $\tilde{h}' = k - |\tilde{S}'|$.

Notice that $\tilde{h} = k - |\tilde{S}| \geq 0$, because $|S| \leq k$. Hence, if $(\tilde{C}, \mathbf{v}) \models \varphi[\mathbf{x}]$, then FindC$(\tilde{C}, \tilde{S}, \tilde{h})$ outputs $\tilde{C}$ and $\tilde{S}$ in the first step, and the claim holds. Assume that $(\tilde{C}, \mathbf{v}) \not\models \varphi[\mathbf{x}]$. Because the subroutine is called only for connected induced subgraphs of G, we have that $\tilde{S} \subset S$ and, therefore, $\tilde{h} > 0$. This implies that the subroutine does not stop in the second step. Then it proceeds to the third step and finds an s-tuple $\mathbf{u} = \langle u_1, \ldots, u_s \rangle$ of vertices of $\tilde{C}$ such that $(\tilde{C}, \mathbf{vu}) \not\models \varphi[\mathbf{xy}]$. Because $(C, \mathbf{v}) \models \varphi[\mathbf{x}]$, there is a $j \in \{1, \ldots, s\}$ such that $u_j \notin V(C)$. We consider the following two cases.

Case 1. $u_j \in S$. Notice that because X is a colorful solution, u_j is blue in this case. Observe also that $\tilde{C} - u_j$ has a component $\tilde{C}'$ such that $V(C) \subseteq V(\tilde{C}')$. Then the subroutine calls FindC$(\tilde{C}', \tilde{S}', \tilde{h}')$, where $\tilde{S}' = \tilde{S} \cup \{u_j\}$ and $\tilde{h}' = \tilde{h} - 1$. It is easy to see that (a′)–(d′) are fulfilled for $\tilde{C}'$, $\tilde{S}'$, and $\tilde{h}'$.

Case 2. $u_j \in U$. As X is colorful, u_j is red in this case. Let H be the red component of $\tilde{C}$ containing u_j and let $W = V(H)$. Because X is a colorful solution, we have that $W \subseteq U$ and $N_{\tilde{C}}(W) \subseteq S$. Then $G - N_G[W]$ has a component $\tilde{C}'$ such that $V(C) \subseteq V(\tilde{C}')$. Then the subroutine calls FindC$(\tilde{C}', \tilde{S}', \tilde{h}')$, where $\tilde{S}' = \tilde{S} \cup N_{\tilde{C}}(W)$ and $\tilde{h}' = \tilde{h} - |N_{\tilde{C}}(W)|$. We obtain that (a′)–(d′) are fulfilled for $\tilde{C}'$, $\tilde{S}'$, and $\tilde{h}'$. This concludes the case analysis and the proof of the claim.

Observe that conditions (a)–(d) of the claim are fulfilled if $C = G$, $S = \emptyset$, and $h = k$. Then the inductive application of the claim proves that there is a leaf of the search tree for which it outputs $\tilde{C}$ and $\tilde{S}$ such that (a) $V(C) \subseteq V(\tilde{C})$, (b) $\tilde{S} = N_G(V(\tilde{C}))$, and (c) $\tilde{S} \subseteq S$. Recall that X is an inclusion-minimal colorful solution. Then Lemma 9 immediately implies that $C = \tilde{C}$ and $S = \tilde{S}$ and this concludes the proof. ◻

Note that the number of branches of every node of the search tree produced by $\textsc{FindC}(G, \emptyset, k)$ is at most s and the depth of the search tree is at most k. This implies that the search tree has at most s^k leaves. By Lemma 11, if (G, k) has an inclusion-minimal colorful solution X, then the subroutine outputs the corresponding big component C containing $v_1, \dots, v_r$ and S. We consider all pairs (C, S) produced by $\textsc{FindC}(G, \emptyset, k)$ and for each of these pairs, we verify whether there is a colorful solution corresponding to it. If we find such a solution, then we return yes (or return the solution), and we return no if we fail to find a colorful solution for each C and S. In the last case, we conclude that we have no colorful solution and discard the current choice of $R \in \mathcal{F}$.

Assume that C and S are given. Recall that $v_i \in V(C)$ for $i \in \{1, \dots, r\}$, $S = N_G(V(C))$, and $(G, \mathbf{v}) \models \varphi[\mathbf{x}]$. First, we check whether C is a big component of $G - S$ by verifying whether $|V(C)| \geq p + 1$. Clearly, if $|V(C)| \leq p$, then C cannot be a big component of $G - X$ for a solution X and we discard the considered choice of C and S. Assume that this is not the case, that is, $|V(C)| \geq p + 1$. Then because G is a (p, k)-unbreakable graph, we have that $|V(G) \setminus N_G[V(C)]| \leq p$. We use brute force and consider every subset $Y \subseteq V(G) \setminus N_G[V(C)]$ and then verify whether (i) $X = S \cup Y$ is an elimination set of depth at most $k - 1$ and (ii) for every component $C' \neq C$ of $G - X$, $C' \models \varphi$. Note that checking (i) can be done by Lemma 7 in $(k + p)^{O(k)} \cdot n^{O(1)}$ time and (ii) can be verified in $n^{O(|\varphi|)}$ time by Observation 1. If we find $X = S \cup Y$ satisfying (i) and (ii), then we conclude that X is a solution and return yes. Otherwise, if we fail to find such a set, then we return no.

This concludes the description of the algorithm for Elimination Distance–(conn) to φ and its correctness proof. We summarize in the following lemma:

Lemma 12. *Elimination Distance–(conn) to φ on (p, k)-unbreakable graphs for $\varphi \in \Sigma_3$ can be solved in $2^{O((p+k)\log(p+k))} \cdot n^{O(|\varphi|)}$ time.*

Proof. Since the correctness of the algorithm was already established, it remains to evaluate the total running time. Recall that if $|V(G)| \leq (3p + 2k)(p + 1)$, then the problem is solved in $(p + k)^{O(k+|\varphi|)}$ time. Otherwise, we construct $\mathcal{F}$ of size at most $2^{O(\min\{p,k\}\log(p+k))} \cdot \log n$ in $2^{O(\min\{p,k\}\log(p+k))} \cdot n \log n$ time. Then for every $R \in \mathcal{F}$, we try to find a colorful solution. For this, we first guess $\mathbf{v}$. Clearly, we have $n^{O(|\varphi|)}$ possibilities for the choice of $\mathbf{v}$. Then, we run the subroutine $\textsc{FindC}(C, S, h)$. Note that the search tree produced by the subroutine has at most $|\varphi|^k$ leaves and each call (without recursive calls) requires $n^{O(|\varphi|)}$ time. Then the running time of the subroutine is $|\varphi|^k \cdot n^{O(|\varphi|)}$. We consider the pairs (C, S) produced by the subroutine, and for each C and S, we verify whether we have a corresponding colorful solution X. The brute force selection of X can be done in $2^{O(p)}$ time. Then checking whether X is a solution requires $(k + p)^{O(k)} \cdot n^{O(1)}$. Then, we conclude that the total running time is $2^{O((p+k)\log(p+k))} \cdot n^{O(|\varphi|)}$. □

Algorithm for Elimination Distance–(prop) to φ. Let (G, k) be an instance of Elimination Distance–(prop) to φ, where G is a (p, k)-unbreakable graph. We check whether $G \models \varphi$ and immediately return yes if this is fulfilled. Assume that this is not the case and that $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) \geq 1$. Then, we can assume without loss of generality that G is connected. Otherwise, because $\mathrm{ed}_\varphi^{\mathrm{prop}}(G) = \max\{1, \max\{\mathrm{ed}_\varphi^{\mathrm{prop}}(C) \mid C \text{ a component of } G\}\}$, we can solve the problem for each component separately. In the same way as with Elimination Distance–(conn) to φ, we solve the problem in $(p + k)^{O(k+|\varphi|)}$ time by Lemma 8 if $|V(G)| \leq (3p + 2k)(p + 1)$. Therefore, from now on, we may assume that $|V(G)| > (3p + 2k)(p + 1)$.

Let (T, α) be a representation of an elimination set X. Recall that $\mathcal{P}_x$ denotes the set of components of $G - X$ anchored in x, where x is a node of T. Also G_x denotes the subgraph of G induced by the vertices of the graphs of $\mathcal{P}_x$, that is, G_x is the union of the components of $G - X$ anchored in x. By Lemma 4, (G, k) is a yes-instance of Elimination Distance–(prop) to φ if and only if G contains an elimination set X of depth at most $k-1$ with a representation (T, α) such that

(i) for every nonleaf node $x \in V(T)$, $C \models \varphi$ for every $C \in \mathcal{P}_x$,
(ii) for every leaf x of T with $\text{depth}_T(x) \leq k-2$, either $G_x \models \varphi$ or $C \models \varphi$ for every $C \in \mathcal{P}_x$, and
(iii) for every leaf x of T with $\text{depth}_T(x) = k-1$, $G_x \models \varphi$.

We call such a set X a *solution*. We observe that, given a set X, we can decide whether X is a solution.

Lemma 13. *Let $X \subseteq V(G)$ be nonempty. It can be decided in $|X|^k \cdot n^{O(|\varphi|)}$ time whether X has a representation (T, α) satisfying (i)–(iii).*

Proof. Because G is connected and $k \geq 1$, it is sufficient to verify the existence of a nice representation. We do it by a recursive algorithm that for a given $x \in X$ finds a nice representation (T, α) such that $\alpha(r) = x$, where r is the root of T. More precisely, given a graph G, a nonempty $X \subseteq V(H)$, a vertex $x \in X$, and a positive integer k, the algorithm finds a nice representation (T, α) of X satisfying (i)–(iii) such that $\alpha(r) = x$ if such a representation exists.

Suppose that $|X| = 1$, that is, $X = \{x\}$. If $G - x \models \varphi$, then the algorithm returns a single-vertex tree rooted in r with $\alpha(r) = x$. If $G - x \not\models \varphi$ and $k \geq 2$, then we check whether $C \models \varphi$ for every component $C - x$. If this holds, then again, the algorithm returns a single-vertex tree rooted in r with $\alpha(r) = x$. In all other cases, the algorithm returns no.

Suppose from now on that $|X| \geq 2$. If $k = 1$, then we immediately return no and stop. Also, if there is a component C of $G - x$ such that $V(C) \cap X = \emptyset$ and $C \not\models \varphi$, then the algorithm returns no and stops. Assume that these are not cases. Let $C_1, \ldots, C_s$ be the components of $G - x$ such that $X_i = X \cap V(C_i) \neq \emptyset$.

For every $i \in \{1, \ldots, s\}$, we call the algorithm recursively for C_i, X_i, every $y \in X_i$, and $k-1$. If there is $i \in \{1, \ldots, s\}$ such that the algorithm failed to produce a representation for every choice of $y \in X_i$, then the algorithm returns no and stops. Otherwise, the algorithm finds for every $i \in \{1, \ldots, s\}$ a vertex $x_i \in X_i$ and a nice representation (T_i, α_i) of X_i in C_i satisfying (i)–(iii) (with respect to the new parameters) such that the root r_i is mapped to x_i by α_i. We construct T from $T_1, \ldots, T_s$ by creating a root r and making it the parent of $r_1, \ldots, r_s$. Then

$$\alpha(z) = \begin{cases} x, & \text{if } z = r, \\ \alpha_i(z), & \text{if } z \in V(T_i) \text{ for some } i \in \{1, \ldots, s\}. \end{cases}$$

This completes the description of the algorithm. It is straightforward to verify its correctness using the definition of a nice representation of an elimination set. To decide whether X has a representation (T, α) satisfying (i)–(iii), we run the algorithm for all $x \in X$. Clearly, a representation exists if and only if the algorithm produces a representation for some choice of x. Since in each call of the algorithm, we make at most $|X|$ recursive calls and the depth of the recursion is at most k, the total running time is $|X|^k \cdot n^{O(|\varphi|)}$. □

Suppose that (G, k) is a yes-instance and let X be a solution with a nice representation (T, α). By Lemma 6, $|X| \leq p+k$ and there is a unique big component C of $G-X$ with at least $p+1$ vertices, the other components are small, and $|V(G) \setminus N_G[V(C)]| \leq p$. By Observation 3, $N_G(V(C)) \subseteq \alpha(A_T(x))$, where x is an anchor of C. In particular, this means that $|N_G(V(C))| \leq k$. As with the algorithm for Elimination Distance–(conn) to φ, our aim is to identify C. We consider two possibilities for C.

First, we try to find C assuming that one of the following holds: either (a) the anchor of C is not a leaf of T or (b) the anchor x is leaf but $\text{depth}_T(x) < k-1$ and $C' \models \varphi$ for every $C' \in \mathcal{P}_x$, or (c) $G_x = C$. In this case, the algorithm is essentially identical to the algorithm for Elimination Distance–(conn) to φ. We use Lemma 10 to highlight S and $U = V(G)\setminus N_G[V(C)]$. Then, we guess $\mathbf{v}$ in C and call the subroutine FindC$(G, \emptyset, k)$ to enumerate all candidate big components C and $S = N_G(V(S))$. The difference occurs only in the last step of the algorithm, where we find a solution X. We use brute force and consider every subset $Y \subseteq V(G)\setminus N_G[V(C)]$ and then verify whether $X = S \cup Y$ is an elimination set of depth at most $k-1$ satisfying (i)–(iii) using Lemma 13. If we find a required X, then we conclude that X is a solution and return yes. Otherwise, if we fail to find such a set for every candidate C, then we return no for the considered set R and discard it. The correctness is proved and the running time is analyzed in exactly the same way as for Elimination Distance–(conn) to φ.

Next, if we failed to find a solution so far, then we consider the remaining possibility that the anchor x of C is a leaf of T and $G_x \models \varphi$, where G_x is a disconnected graph. Our algorithm for this case uses the same approach as the algorithm for Elimination Distance–(conn) to φ but the arguments are more involved, as we aim to identify C together with the other components of G_x. In other words, we find G_x.

Let $S = N_G(V(G_x))$. Note that $S \subseteq \alpha(A_T(x))$ and, therefore, $|S| \leq k$. Observe that $N_G(V(C)) \subseteq S$. Let also $U = V(G)\setminus(V(C) \cup S)$. Because C is a big component and G is (p,k)-unbreakable, $|U| \leq p$.

Similarly to the algorithm for Elimination Distance–(conn) to φ, we use Lemma 10 to highlight hypothetical S and U. By this lemma, we can construct in $2^{O(\min\{p,k\}\log(p+k))} \cdot n \log n$ time a family $\mathcal{F}$ of at most $2^{O(\min\{p,k\}\log(p+k))} \cdot \log n$ subsets of $V(G)$ such that there is $R \in \mathcal{F}$ such that $U \subseteq R$ and $S \cap R = \emptyset$. In our algorithm, we go over all sets $R \subseteq \mathcal{F}$ and for each set R, we check whether there is a solution X such that $U \subseteq R$ and $S \cap R = \emptyset$ for the sets S and U corresponding to X. Clearly, (G,k) is a yes-instance of Elimination Distance–(prop) to φ if and only if there is $R \subseteq \mathcal{F}$ and a solution X with the required property.

From now on, we assume that $R \subseteq \mathcal{F}$ is given. We set $B = V(G)\setminus R$. In the same way as before, we say that the vertices of R are *red* and the vertices of B are *blue*. The components of $G[R]$ are called *red components* of G and the same convention is used for induced subgraphs of G. A solution X is called *colorful* if the vertivces of U are red and the vertices of S are blue. We aim to find a colorful solution.

Assume that a colorful solution X exists. Suppose that $w = \alpha(x)$ for the leaf x of T that is the anchor of G_x. Notice that $w \in B$. Then for every component C' of G_x distinct from C, we have that C' is a red component and $z \in N_G(V(C'))$. We also observe that by the assumption for R, if C' is a red component of G such that $w \in N_G(V(H))$, then either $V(C') \subseteq V(C)$ or C' is a component of G_x distinct from C. Using these observations, we consider all possible choices of w in B, and decide whether there is a colorful solution X such that for the required G_x, the leaf x of T is mapped to w. We say that X is a colorful solution *attached* to w.

From now on, we assume that w is given. Let $W = \bigcup V(H)$, where the union is taken over all red components H of G such that $w \in N_G(V(H))$. Notice that if there is a colorful solution X attached to w for the considered choice of w, then $W \subseteq V(G_x)$ for the corresponding graph G_x.

Since we require that $G_x \models \varphi$, then there is an r-tuple $\mathbf{v} = \langle v_1, \ldots, v_r\rangle$ of vertices of G_x such that $(G_x, \mathbf{v}) \models \varphi[\mathbf{x}]$. Using brute force, we consider all r-tuples $\mathbf{v} = \langle v_1, \ldots, v_r\rangle$ of vertices of G distinct from w, and for each $\mathbf{v}$, we check whether there is a colorful solution X with G_x such that $v_i \in V(G_x)$ for all $i \in \{1, \ldots, r\}$. Note that at most n^r r-tuples $\mathbf{v}$ can be listed in $n^{O(|\varphi|)}$ time. The algorithm returns yes if we find a colorful solution attached to w for some choice of $\mathbf{v}$, and it concludes that there is no colorful solution for the considered selection of R otherwise.

From this point, we assume that $\mathbf{v} = \langle v_1, \ldots, v_r\rangle$ is fixed. Because these vertices should be in G_x, we temporarily (i.e., only for the current choice of $\mathbf{v}$) recolor them red to simplify further notation and recompute W if necessary. We apply a recursive branching algorithm to find $F = G_x$ and

S. Similarly to the subroutine FindC, we construct the subroutine FindF(F, S, h), where initially $F = G - w$, $S = \{w\}$, and $h = k - 1$.

Subroutine FindF(F, S, h).

- If $(F, \mathbf{v}) \models \varphi[\mathbf{x}]$ and $h \geq 0$, then return F, S, and stop.
- If $(F, \mathbf{v}) \not\models \varphi[\mathbf{x}]$ and $h \leq 0$, then stop.
- If $h \geq 1$ and there is an s-tuple $\mathbf{u} = \langle u_1, \dots, u_s \rangle$ of vertices of F such that $(F, \mathbf{vu}) \not\models \varphi[\mathbf{xy}]$, then do the following for every $j \in \{1, \dots, s\}$:
  - If $u_j \in B$ and there is an induced subgraph F' of F that is the disjoint union of the components of $F - u_j$ containing vertices of W and vertices v_i for $i \in \{1, \dots, r\}$, then call FindF$(F', S \cup \{u_j\}, h - 1)$.
  - If $u_j \in R$ and there is a red component H of C with the set of vertices Z and $S' = N_F[Z]$ such that (a) $u_j \in Z$, (b) $Z \cap W = \emptyset$ and $v_i \notin Z$ for all $i \in \{1, \dots, r\}$, (c) $|S'| \leq h$, and (d) there is an induced subgraph F' of F that is a disjoint union of the components of $F - N_F[W]$ containing vertices of W and vertices v_i for some $i \in \{1, \dots, r\}$, then call FindF$(F', S \cup S', h - |S'|)$.

In the same way as with Lemma 11, we show the following:

Lemma 14. *If X is an inclusion-minimal colorful solution attached to w for (G, k) such that (a) X has a representation (T, α) with $\alpha(x) = w$, (b) $W \subseteq V(G_x)$, (c) $v_i \in V(G_x)$ for all $i \in \{1, \dots, s\}$ and $(G_x, \mathbf{v}) \models \varphi[\mathbf{x}]$, then there is a leaf of the search tree produced by FindC$(G, \{w\}, k - 1)$ for which the subroutine outputs F and $S = N_G(V(F))$.*

Since the number of branches of every node of the search tree produced by FindF$(G, \{w\}, k-1)$ is at most s and the depth of the search tree is at most k, the search tree has at most r^k leaves. By Lemma 14, if (G, k) has an inclusion-minimal colorful solution X attached to w with respect to some representation (T, α) of X, then the subroutine outputs the corresponding graph $F = G_x$ containing $v_1, \dots, v_r$ and S. We consider all pairs (F, S) produced by FindF$(G, \{w\}, k - 1)$ and for each of these pairs, we verify whether there is a colorful solution corresponding to it. If we find such a solution, then we return yes (or return the solution), and we return no if we fail to find a colorful solution for each F and S. In the last case, we conclude that we have no colorful solution and discard the current choice of $R \in \mathcal{F}$.

Assume that F and S are given. Recall that $v_i \in V(C)$ for $i \in \{1, \dots, r\}$, $S = N_G(V(C))$, and $(G, \mathbf{v}) \models \varphi[\mathbf{x}]$. First, we check whether F has a big components of $G - S$ by verifying whether F has a component with at least $p + 1$ vertices. If we have no such a component, then we discard the considered choice of F and S. Assume that this is not the case. Then because G is a (p, k)-unbreakable graph, we have that $|V(F) \backslash N_G[V(F)]| \leq p$. We use brute force and consider every subset $Y \subseteq V(G) \backslash N_G[V(F)]$ and then verify whether $X = S \cup Y$ is a solution using Lemma 13. If we find a solution, then we return yes. Otherwise, if we fail to find Y with the required properties, then we return no.

This concludes the description of the algorithm for Elimination Distance–(conn) to φ and its correctness proof. We summarize in the following lemma that is proved in the same way as Lemma 12:

Lemma 15. *Elimination Distance–(prop) to φ on (p, k)-unbreakable graphs for $\varphi \in \Sigma_3$ can be solved in $2^{O((p+k)\log(p+k))} \cdot n^{O(|\varphi|)}$ time.*

Algorithm for Elimination Distance–(depth) to φ. Our final task is to construct an algorithm for Elimination Distance–(depth) to φ. Let (G, k) be an instance of Elimination Distance–(depth) to φ, where G is a (p, k)-unbreakable graph. If $G \models \varphi$, then we return yes. Assume that this is not the case and $\mathrm{ed}_{\varphi}^{\mathrm{depth}}(G) \geq 1$.

Suppose that G is disconnected. Denote by $C_1, \dots, C_s$ the components of G. Because G is a (p,k)-unbreakable graph, at most one component can have more p vertices. Then, we can assume that $|V(C_i)| \le p$ for every $i \in \{2,\dots,s\}$. For each $i \in \{2,\dots,s\}$, we solve Elimination Distance–(depth) to φ for $(C_i, k-1)$ and (C_i,k) in $2^p \cdot p^{O(k+|\varphi|)}$ time using brute force. Let $i \in \{2,\dots,s\}$. For each set $X \subseteq V(C_i)$, we check whether $\mathrm{depth}(X) \le k-2$ ($\mathrm{depth}(X) \le k-1$, respectively) applying Lemma 7, and if this holds, then we verify whether $G - X \models \varphi$. This can be done in $2^p \cdot p^{O(k)} \cdot p^{O(|\varphi|)}$ time. If we find that either there is $i \in \{2,\dots,s\}$ such that $\mathrm{ed}_{\varphi}^{\mathrm{depth}}(C_i) \ge k+1$ or there are two distinct $i,j \in \{2,\dots,s\}$ such that $\mathrm{ed}_{\varphi}^{\mathrm{depth}}(C_i) = \mathrm{ed}_{\varphi}^{\mathrm{depth}}(C_j) = k$, then we return no by Lemma 2. If there is a unique $i \in \{2,\dots,s\}$ with $\mathrm{ed}_{\varphi}^{\mathrm{depth}}(C_i) = k$ and $\mathrm{ed}_{\varphi}^{\mathrm{depth}}(C_j) \le k-1$ for $j \in \{2,\dots,s\}$ distinct from i, (G,k) is a yes-instance if and only if $(C_1, k-1)$ is a yes-instance by Lemma 2. If $\mathrm{ed}_{\varphi}^{\mathrm{depth}}(C_i) \le k-1$ for every $i \in \{2,\dots,s\}$, then by the same lemma, (G,k) is a yes-instance if and only if (C_1,k) is a yes-instance. Thus, we are able to reduce solving the problem on G to solving it on C_1. This implies that we can assume without loss of generality that G is connected.

If $|V(G)| \le (3p+2k)(p+1)$, then we again can solve the problem using brute force in $2^{(3p+2k)(p+1)} \cdot ((3p+2k)(p+1))^{O(k+|\varphi|)}$ time in the same way as above. Then, we assume that $|V(G)| > (3p+2k)(p+1)$.

Given a subset $X \subseteq V(G)$, we can verify in $|X|^{O(k)} \cdot n^{O(1)}$ whether $\mathrm{depth}(X) \le k-1$ by Lemma 7 and then can check whether $G - X \models \varphi$ using Observation 1. Based on this, we aim to find X that we call a *solution* in the same way as for the previously considered problems.

For Elimination Distance–(conn) to φ, we used the random separation technique to highlight a big component of $G - X$ (or rather its complement), and for Elimination Distance–(prop) to φ, besides a big component, we had to highlight some specific small components composing G_x together with the big component. Now, we are highlighting the small components, X and the neighborhood $N_G(V(C)) \subseteq X$ of the big component.

Suppose that (G,k) is a yes-instance with a solution X. By Lemma 6, $|X| \le p+k$ and $|V(G) \backslash N_G[V(C)]| \le p$, where C is a big component of $G - X$. By Lemma 10, we can construct the family $\mathcal{F}$ of subsets of $V(G)$ of size at most $2^{O((p+k)\log(p+k))} \cdot \log n$ in $2^{O((p+k)\log(p+k))} \cdot n \log n$ time such that if (G,k) has a solution X, then $\mathcal{F}$ has a set R such that $V(H) \subseteq R$ for every small component and $R \cap X = \emptyset$. Then for every $R \in \mathcal{F}$, we aim to find a solution X such that the vertices of the small components of $G - X$ are in R and $X \cap R = \emptyset$.

From this point, we assume that R is given. Consider $U = V(G) \backslash R$. If C is a big component of a (hypothetical) solution X satisfying the above conditions, then $N_G(V(C)) \subseteq U$ and $|N_G(V(C))| \le k$. Recall that $|X \backslash N_G(V(C))| \le |V(G) \backslash N_G[V(C)]| \le p$. Since $|U| \le n$, applying Lemma 10 for U, we construct the family $\mathcal{F}'$ of subsets of U of size at most $2^{O(\min\{p,k\}\log(p+k))} \cdot \log n$ in $2^{O(\min\{p,k\}\log(p+k))} \cdot n \log n$ time such that $\mathcal{F}'$ has a set Y with the property that $X \backslash N_G(V(C)) \subseteq Y$ and $N_G(V(C)) \cap Y = \emptyset$. We consider all $Y \in \mathcal{F}'$ and try find a solution X such that

(i) the vertices of the small components of $G - X$ are in R,
(ii) for the big component C of $G - X$, $X \backslash N_G(V(C)) \subseteq Y$,
(iii) for the big component C of $G - X$, $N_G(V(C)) \subseteq B$, where $B = V(G) \backslash (R \cup Y)$.

If a solution X satisfies (i)–(iii), then we say that X is *colorful*.

We say that the vertices of R are *red*, the vertices of Y are yellow, and the vertices of B are *blue*. The components of $G[R]$ are called *red* and the components of $G[R \cup Y]$ are called *red-yellow* components of G, and we use the same term for the induced subgraphs of G.

Assume that X is a colorful solution. Notice that if H is a red component of G, then either H is a small component of $G - X$ with $N_G(V(H)) \subseteq X$ or $V(H) \subseteq V(C)$, where C is the big component. Also, we have that if H is a red-yellow component of G, then either $V(H) \subseteq V(C)$ or every red component of $V(H)$ is a small component of $G - X$. These are the crucial properties of colorful solutions exploited by our algorithm.

Because $G - X \models \varphi$ for a solution X, there should exist an r-tuple $\mathbf{v} = \langle v_1, \ldots, v_r \rangle$ of vertices of $G - X$ such that $(G - X, \mathbf{v}) \models \varphi[\mathbf{x}]$. In the same way as for the previous problem, we use brute force to list all r-tuples $\mathbf{v} = \langle v_1, \ldots, v_r \rangle$ of vertices of G. Then for each $\mathbf{v}$, we check whether there is a colorful solution X such that $v_i \notin X$ for all $i \in \{1, \ldots, r\}$ and $(G - X, \mathbf{v}) \models \varphi[\mathbf{x}]$. There are at most n^r r-tuples $\mathbf{v}$ can be listed $n^{O(|\varphi|)}$ time. Our algorithm returns yes if we find a colorful solution for some choice of $\mathbf{v}$, and it concludes that there is no colorful solution for the considered selection of R otherwise.

From now on, we assume that $\mathbf{v} = \langle v_1, \ldots, v_r \rangle$ is fixed. Again, we observe that these vertices should not belong to X and we recolor them red for the considered choice of R. We use a recursive branching algorithm to find X. The algorithm exploits the subroutine FindX(Z, h), where initially $Z = \emptyset$ and $h = p + k$.

Subroutine FindX(Z, h).

(1) Set $F := G - Z$.
(2) If $(F, \mathbf{v}) \models \varphi[\mathbf{x}]$, $\text{depth}(Z) \leq k-1$, and $h \geq 0$, then return Z, and stop executing the algorithm.
(3) If $(F, \mathbf{v}) \not\models \varphi[\mathbf{x}]$ and $h \leq 0$, then stop executing the subroutine.
(4) If $h \geq 1$ and there is an s-tuple $\mathbf{u} = \langle u_1, \ldots, u_s \rangle$ of vertices of F such that $(F, \mathbf{vu}) \not\models \varphi[\mathbf{xy}]$, then do the following:
   - If $u_j \in R$ for every $j \in \{1, \ldots, s\}$, then stop executing the subroutine.
   - Otherwise, for every $j \in \{1, \ldots, s\}$ such that $u_j \in Y \cup B$, call FindX$(Z \cup \{u_j\}, h - 1)$.
(5) If $\text{depth}(Z) \geq k$, then for every $x \in Z$ such that there is a red-yellow component H of F with the properties (i) $|N_F(V(H))| \leq k$, (ii) $|V(H)| \leq p$, (iii) $x \in V(H)$, and (iv) $N_F[V(H)] \cap (B \cup Y) \neq \emptyset$, set $S := N_F[V(H)] \cap (B \cup Y)$ and call FindX$(Z \cup S, h - |S|)$.

Notice that if the subroutine outputs Z, then we stop the algorithm and report that we found a solution. If we stop in other steps, then we only stop the execution of the subroutine for the current call. The crucial property of the subroutine is proved in the following lemma. Since FindX(Z, h) substantially differs from FindC(C, S, h) and FindF(F, S, h), we provide the proof.

Lemma 16. *If (G, k) has a colorful inclusion-minimal solution X with $v_i \in V(G)\backslash X$ for all $i \in \{1, \ldots, s\}$ such that $(G - X, \mathbf{v}) \models \varphi[\mathbf{x}]$, then FindX$(\emptyset, p + k)$ returns X.*

Proof. The lemma is proved similarly to Lemma 11. Let $t = p + k$. We show that the algorithm maintains the following property: If the subroutine FindX is called for (Z, h) such that (a) $Z \subseteq X$ and (b) $h = t - |Z|$, then either the subroutine outputs $Z = X$ or it recursively calls FindX(Z', h'), where (a′) $Z' \subseteq X$ and (b′) $h' = t - |Z'|$.

In the first step, the algorithms sets $F = G - Z$. If $(F, \mathbf{v}) \models \varphi[\mathbf{x}]$, $\text{depth}(Z) \leq k - 1$, $h \geq 0$, and $(F, \mathbf{v}) \models \varphi[\mathbf{x}]$, then Z is a colorful solution and the algorithm returns return Z. Since X is inclusion-minimal, we have that $X = Z$. Thus, the claim holds. Assume that this is not the case. Since $Z \subseteq X$, we have that $h \geq 1$, that is, the subroutine does not stop in step 3. Clearly, we have that $(F, \mathbf{v}) \not\models \varphi[\mathbf{x}]$ and/or $\text{depth}(Z) \geq k$.

Suppose that $(F, \mathbf{v}) \not\models \varphi[\mathbf{x}]$. Then there is an s-tuple $\mathbf{u} = \langle u_1, \ldots, u_s \rangle$ of vertices of F such that $(F, \mathbf{vu}) \not\models \varphi[\mathbf{xy}]$. This means that the subroutine executes step 4. As $(G - X, \mathbf{v}) \models \varphi[\mathbf{x}]$ and $Z \subseteq X$, there is $j \in \{1, \ldots, s\}$ such that $u_j \in X\backslash X$. Because X is a colorful solution, $u_j \in B$. Therefore, the subroutine calls FindX(Z', h') for $Z' = Y\backslash\{u_j\}$ and $h' = h - 1$. It is easy to see that (a′) and (b′) are fulfilled.

Assume that $(F, \mathbf{v}) \models \varphi[\mathbf{x}]$. Then $\text{depth}(Z) \geq k$. Because $\text{depth}(X) \leq k - 1$, X has a representation (T, α) with $\text{depth}(T) \leq k - 1$. Because $\text{depth}(T) \leq k - 1$ and $\text{depth}(Z) \geq k$, there are vertices

$x, y \in Z$ such that the nodes $x' = \alpha^{-1}(x)$ and $y' = \alpha^{-1}(y)$ have the lowest common ancestor z in T such that $z \neq x, y$ and it holds that $\alpha(A_T(s))\setminus Z \neq \emptyset$ and $\alpha(A_T(z))$ is an (x,y)-separator in G. In particular, x and y cannot be both in $N_G[V(C)]$. By symmetry, we can assume that $x \notin N_G[V(C)]$. This means that there is a red-yellow component H of F such that properties (i)–(iv) of step 5 are fulfilled. Because X is a colorful solution, we have that $S = N_F[V(H)] \cap (B \cup Y) \subseteq X$. Thus, (a′) and (b′) are fulfilled for $Z' = Z \cup S$ and $h' = h - |S|$. As the subroutine calls FindX(Z', h'), we conclude that the claim if fulfilled.

Recall that we call FindX($\emptyset, p + k$) and note that conditions (a) and (b) are trivially fulfilled for $Z = \emptyset$ and $h = t$. Observe also that in each recursive call of the subroutine the parameter h strictly decreases. Thus, we conclude that we output X is some recursive call of FindX(Z, h). □

Lemma 16 concludes the description of the algorithm and its correctness proof. We summarize and evaluate the running time in the following lemma.

Lemma 17. *Elimination Distance–*(depth) *to φ on (p,k)-unbreakable graphs for $\varphi \in \Sigma_3$ can be solved in $2^{O((p+k)(\log(p+k)+p))} \cdot n^{O(|\varphi|)}$ time.*

Proof. If $|V(G)| \leq (3p + 2k)(p + 1)$, then the problem is solved by brute force in $2^{(3p+2k)(p+1)} \cdot ((3p + 2k)(p + 1))^{O(k+|\varphi|)}$ time. Assume that $|V(G)| > (3p + 2k)(p + 1)$. Then, we construct $\mathcal{F}$ in $2^{O((p+k)\log(p+k))} \cdot n \log n$ time. The size of $\mathcal{F}$ is at most $2^{O((p+k)\log(p+k))} \cdot \log n$, and for every $R \in \mathcal{F}$, we construct $\mathcal{F}'$ in $2^{O(\min\{p,k\}\log(p+k))} \cdot n \log n$ time. Recall that the size of $\mathcal{F}'$ is at most $2^{O(\min\{p,k\}\log(p+k))} \cdot \log n$. Then, we consider at most n^r r-tuples of vertices $\mathbf{v}$ that can be listed in $n^{O(|\varphi|)}$ time. Finally, for every $R \in \mathcal{F}$, $Y \in \mathcal{F}'$, and every $\mathbf{v}$, we call FindX($\emptyset, p + k$).

Thus, it remains to evaluate the running time of FindX($\emptyset, p + k$). Notice that in each call, $|Z| \leq p + k$. Then, we can verify in $(p + k)^{O(k)} \cdot n^{O(1)}$ time whether depth$(Z) \leq k - 1$ using Lemma 7. Also, we can check whether $(F, \mathbf{v}) \models \varphi[\mathbf{x}]$ in $n^{O(|\varphi|)}$ by Observation 1. Simultaneously, we find an s-tuple $\mathbf{u}$ of vertices of F such that $(F, \mathbf{vu}) \not\models \varphi[\mathbf{xy}]$ if this is not the case. In step 4, we perform at most s recursive calls. In step 5, finding H can be done in polynomial time. Notice that we have at most $|Z| \leq p + k$ recursive calls in this step. The depth of the recursion is upper bounded by $k + p$. This implies that the running time of FindX($\emptyset, p + k$) is $(p + k)^{O(p+k)} \cdot n^{|\varphi|}$.

Summarizing, we obtain that the total running time is $2^{O((p+k)(\log(p+k)+p))} \cdot n^{O(|\varphi|)}$. □

5 LOWER BOUND FOR Π_3-FORMULAS

In Section 4, we proved that for every FOL formula $\varphi \in \Sigma_3$, Elimination Distance–($\star$) to φ can be solved in $f(k) \cdot n^{O(|\varphi|)}$ time for each $\star \in \{\text{conn}, \text{prop}, \text{depth}\}$. Recall that one of the ingredients of the proof is a recursive algorithm that exploits the following observation: Suppose that $\varphi = \exists x_1 \cdots \exists x_r \forall y_1 \cdots \forall y_s \exists z_1 \cdots \exists z_t\ \chi$ and our task is, given a graph G, to obtain a graph G' such that $G' \models \varphi$ by deleting some vertices of G. We guess an r-tuple $\mathbf{v} = \langle v_1, \dots, v_r\rangle$ of vertices of G' such that $(G', \mathbf{v}) \models \varphi[\mathbf{x}]$. Then if there is an s-tuple $\mathbf{u} = \langle u_1, \dots, u_s\rangle$ of vertices of G such that $(G, \mathbf{vu}) \not\models \varphi[\mathbf{xy}]$, then at least one of the vertices $u_1, \dots, u_s$ should be deleted to satisfy φ. This allows us to branch on the vertices of such s-tuples $\mathbf{u}$. Notice that we cannot apply these arguments for a formula $\varphi = \forall x_1 \cdots \forall x_r \exists y_1 \cdots \exists y_s \forall z_1 \cdots \forall z_t\ \chi \in \Pi_3$. If there is an r-tuple $\mathbf{v} = \langle v_1, \dots, v_r\rangle$ of vertices of G such that $(G, \mathbf{v}) \not\models \varphi[\mathbf{x}]$, then it may happen that neither of the vertices $v_1, \dots, v_r$ should be deleted. In this section, we show that this is crucial and complement Theorem 1 by proving that there are formulas in Π_3 for which Elimination Distance–($\star$) to φ is W[2]-hard. We state now Theorem 2 formally.

Theorem 2. *For every $\star \in \{\text{conn}, \text{prop}, \text{depth}\}$, there is $\varphi \in \Pi_3$ such that Elimination Distance–($\star$) to φ is* W[2]*-hard.*

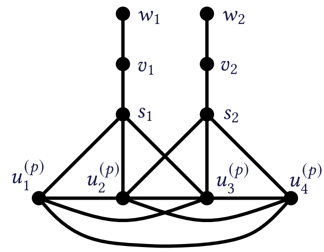

Fig. 3. Construction of G for $n = 2$ and $m = 2$ with $S_1 = \{u_1, u_2, u_3\}$ and $S_2 = \{u_2, u_3, u_4\}$; for simplicity, just one copy of each $u_i^{(p)}$ for $p \in \{0, \dots, k\}$ is shown.

Proof. The proof exploits the same idea as in the W[2]-hardness proof for Deletion to φ for $\varphi \in \Pi_3$ in [16]. However, in [16], the hardness was derived from the result about the edge deletion variant of the problem. Here, we deal only with vertices and consider elimination distances. Thus, the reduction has to be modified. We show that the problems are W[2]-hard for the formula φ expressing the property that for every vertex u of a graph, there is a vertex v of degree at most one that is at distance at most two from u. Notice that the property that a vertex v of a given graph G has degree at most one can be written as follows: For every $z_1, z_2 \in V(G)$, if v is adjacent to z_1 and z_2, then $z_1 = z_2$. Thus, we define the formula

$$\psi(v, z_1, z_2) = [((v \sim z_1) \wedge (v \sim z_2)) \to (z_1 = z_2)]$$

with three free variables and set

$$\varphi = \forall x \exists y_1 \exists y_2 \forall z_1 \forall z_2 \, [\psi(x, z_1, z_2) \vee ((x \sim y_1) \wedge \psi(y_1, z_1, z_2)) \\ \vee ((x \sim y_1) \wedge (y_1 \sim y_2) \wedge \psi(y_2, z_1, z_2))].$$

Clearly, $\varphi \in \Pi_3$.

To show W[2]-hardness, we reduce from the Set Cover problem. The problem asks, given a universe U, a family $\mathcal{S}$ of subsets of U, and a positive integer k, whether there is $\mathcal{S}' \subset \mathcal{S}$ of size at most k that covers U, that is, for every $u \in U$, there is $S \in \mathcal{S}'$ such that $u \in S$. It is well-known that Set Cover is W[2]-complete when parameterized by k [13].

Let $(U, \mathcal{S}, k)$ be an instance of Set Cover with $U = \{u_1, \dots, u_n\}$, $\mathcal{S} = \{S_1, \dots, S_m\}$. We also assume that $n \geq 2$ and $k \leq m$. We construct the following graph G (see Figure 3):

- For every $i \in \{1, \dots, n\}$, construct $k + 2$ vertices $u_i^{(1)}, \dots, u_i^{(k+2)}$, and then for every $i, j \in \{1, \dots, n\}$ and all $p, q \in \{1, \dots, k + 2\}$ such that $(i, p) \neq (j, q)$, make $u_i^{(p)}$ and $u_j^{(q)}$ adjacent.
- For every $j \in \{1, \dots, m\}$, construct three vertices s_j, v_j, w_j and edges $s_j v_j$ and $v_j w_j$.
- For every $i \in \{1, \dots, n\}$ and every $j \in \{1, \dots, m\}$, make s_j adjacent to $u_i^{(1)}, \dots, u_i^{(k+2)}$ if $u_i \in S_j$.

We claim that G has a set cover of size at most k if and only if $\mathrm{ed}_\varphi^\star(G) \leq k$ for $\star \in \{\mathrm{conn}, \mathrm{prop}, \mathrm{depth}\}$. Notice that by the definition of φ, $H \models \varphi$ if and only if $C \models \varphi$ for every component C of H. Therefore, $\mathrm{ed}_\varphi^{\mathrm{conn}}(G) = \mathrm{ed}_\varphi^{\mathrm{prop}}(G) = \mathrm{ed}_\varphi^{\mathrm{depth}}(G)$, and it is sufficient to prove that G has a set cover of size at most k if and only if there is an elimination set X of G with $\mathrm{depth}(G) \leq k - 1$ such that $C \models \varphi$ for every component C of $G - X$.

Suppose that sets $S_{j_1}, \dots, S_{j_k} \in \mathcal{S}$ form a set cover. We define $X = \{w_{j_1}, \dots, w_{j_k}\}$. Since $|X| = k$, $\mathrm{depth}(X) \leq k - 1$. Notice that $H = G - X$ is connected. We claim that $H \models \varphi$. Recall that $H \models \varphi$ if and only if for every vertex $x \in V(H)$ there is a vertex $y \in V(H)$ at distance at most two such that $d_H(y) \leq 1$. This property trivially holds if $x \in \{s_j, v_j, w_j\} \setminus X$ for $j \in \{1, \dots, m\}$. Consider a vertex

$u_i^{(p)}$ for some $i \in \{1, \ldots, n\}$ and $p \in \{1, \ldots, k+2\}$. We have that there is $h \in \{1, \ldots, k\}$ such that $u_i \in S_{j_h}$. Then $u_i s_{j_h} \in E(H)$. Because $w_{j_h} \in X$, we obtain that $d_H(v_{j_h}) = 1$. Since $s_{j_h} v_{j_h} \in E(H)$, v_{j_h} is at distance two from $u_i^{(p)}$ as required. We conclude that $H \models \varphi$.

For the opposite direction, assume that there is an elimination set X of G with $\mathsf{depth}(X) \leq k-1$ such that $C \models \varphi$ for every component C of $G - X$. Consider $Z = \{u_i^{(p)} \mid 1 \leq i \leq n,\ 1 \leq p \leq k+2\}$. Because Z is a clique, we have that $|Z \cap X| \leq k$. To see this, consider a representation (T, α) of X with $\mathsf{depth}(T) \leq k-1$. Then there is a leaf x of T such that $\alpha^{-1}(X \cap Z) \subseteq A_T(x)$. Since $\mathsf{depth}(T) \leq k-1$, we conclude that $|Z \cap X| \leq k$. Note that the vertices of $Z \backslash X$ are in the same component H of $G - X$. Let $W = \{w_1, \ldots, w_m\}$. By Observation 3, $|N_G(V(H)) \cap X| \leq k$. Hence, $|N_G(V(H)) \cap (X \cap W)| \leq k$ as well. Let $\{w_{j_1}, \ldots, w_{j_\ell}\} = N_G(V(H)) \cap (X \cap W)$. We claim that the sets $S_{j_1}, \ldots, S_{j_k}$ cover U.

Consider an arbitrary $i \in \{1, \ldots, n\}$. Because $|Z \cap W| \leq k$, there are two distinct $p, q \in \{1, \ldots, k+2\}$ such that $u_i^{(p)}, u_i^{(q)} \in V(H)$. Since $H \models \varphi$, there is a vertex $z \in V(H)$ at distance at most two from $u_i^{(p)}$ such that $d_H(z) \leq 1$. Since $n \geq 2$, we have that $|Z \backslash X| \geq 3$ and, therefore, $d_H(u_i^{(p)}) \geq 2$. Moreover, for every $h \in \{1, \ldots, n\}$ and $r \in \{1, \ldots, k+2\}$, if $u_h^{(r)} \in V(H)$, then $d_H(u_h^{(r)}) \geq 2$. Then the construction of G implies that there is $h \in \{1, \ldots, m\}$ such that $s_h \in V(H)$ and $u_i^{(p)} s_h \in E(G)$ with the property that either $d_H(s_j) \leq 1$ or s_j has a neighbor in H of degree at most one. As s_h is adjacent to $u_i^{(p)}$, this vertex is adjacent to $u_i^{(q)}$. Hence, $d_H(s_h) \geq 2$. We obtain that $v_h \in V(H)$ and $d_H(v_h) \leq 1$ by the construction of G. This means that $w_h \notin H$, that is, $w_h \in N_G(V(H)) \cap (X \cap W)$. We conclude that there is $t \in \{1, \ldots, \ell\}$ such that $j_t = h$. Finally, because $u_i^{(p)}$ is adjacent to s_{j_t}, $u_i \in S_{j_t}$ and this concludes the proof. □

6 DISCUSSION

We established a parameterized complexity dichotomy for the elimination problems whose aim is to satisfy an FOL formula φ with respect to the quantification structure of the prefix. For this, we considered three variants of the elimination distance to the class of graphs modelling φ and defined the Elimination Distance–($\star$) to φ for $\star \in \{\mathsf{conn}, \mathsf{prop}, \mathsf{depth}\}$ corresponding to the considered type of distance. In Theorem 1, we proved that for every FOL formula $\varphi \in \Sigma_3$, Elimination Distance–($\star$) to φ is FPT for $\star \in \{\mathsf{conn}, \mathsf{prop}, \mathsf{depth}\}$. In Theorem 2, we showed that this result is tight in the sense that there are FOL formulas $\varphi \in \Pi_3$ such that these problems are W[2]-hard.

Notice that the above dichotomy is the same for all the considered variants of the elimination problems. Moreover, it coincides with the structural dichotomy obtained by for Deletion to φ by Fomin, Golovach, and Thilikos in [16]. This leads to the following natural question: Is there an FOL formula φ such that the parameterized complexity of Elimination Distance–($\star$) to φ for $\star \in \{\mathsf{conn}, \mathsf{prop}, \mathsf{depth}\}$ and Deletion to φ differs? In particular, is there a formula φ such that Deletion to φ is FPT but one of the problem Elimination Distance–($\star$) to φ for $\star \in \{\mathsf{conn}, \mathsf{prop}, \mathsf{depth}\}$ turns to be, say, W[1] or W[2]-hard? Note that Lemma 5 holds for every FOL formula φ. Thus, solving Elimination Distance–($\star$) to φ for $\star \in \{\mathsf{conn}, \mathsf{prop}, \mathsf{depth}\}$ can be reduced to solving these problems on unbreakable graphs by Theorem 3. Since Elimination Distance–($\star$) to φ for $\star \in \{\mathsf{conn}, \mathsf{prop}, \mathsf{depth}\}$ is somehow similar to Deletion to φ on unbreakable graphs, it may happen that Elimination Distance–($\star$) to φ for $\star \in \{\mathsf{conn}, \mathsf{prop}, \mathsf{depth}\}$ are FPT whenever Deletion to φ is FPT. However proving this would demand applying different algorithmic tools, as our techniques are tailored for $\varphi \in \Sigma_3$. Also it would be interesting to know whether there are FOL formulas such that Elimination Distance–($\star$) to φ for $\star \in \{\mathsf{conn}, \mathsf{prop}, \mathsf{depth}\}$ differ from the parameterized complexity viewpoint.

In contrast with the same behavior of the elimination and deletion problems with respect to the inclusion in FPT, we would like to point that they behave differently with respect to *kernelization* (we refer to the books cited in [10, 17] for the definition of the notion). It was shown in [16] that Deletion to φ admits a polynomial kernel for $\varphi \in \Sigma_1 \cup \Pi_1$ (in fact, Deletion to φ is polynomial for $\varphi \in \Sigma_1$) and there are formulas $\varphi \in \Pi_2$ and Σ_2 such that Deletion to φ has no polynomial kernel unless NP $\subseteq$ coNP /poly. For the elimination problems, we can show the following lower bound:

Proposition 1. *For every* $\star \in \{\text{conn}, \text{prop}, \text{depth}\}$*, there is* $\varphi \in \Pi_1$ *such that Elimination Distance–($\star$) to* φ *does not admit a polynomial kernel unless* NP $\subseteq$ coNP /poly*.*

Proof. We show the claim for the formula φ expressing the property that a graph has no triangles, that is, cycles of length three:

$$\varphi = \forall x \forall y \forall z\, [(x = y) \vee (y = z) \vee (x = z) \vee \neg(x \sim y) \vee \neg(y \sim z) \vee \neg(x \sim z)].$$

It is straightforward to see that $G \models \varphi$ if and only if G has no triangles.

By the classical results of Lewis and Yannakakis [23], Deletion to φ is NP-complete. Then it is easy to observe that the problem remains NP on instances (G, k), where G is a $(k + 1)$-connected graph. For example, we can reduce from Deletion to φ on general graphs. Let G be an n-vertex graph. We assume that $k < n - 1$, as otherwise the problem is trivial. We construct the graph G' from G by adding $k + 1$ copies of the complete bipartite graph $K_{n,n}$ and making each vertex of one part of the vertex partition to a unique vertex of G. Clearly, G' is $(k + 1)$-connected and it is easy to see that $G - X$ is triangle-free if and only if G' has no triangles for every $X \subseteq V(G')$. This proves the NP-hardness for Deletion to φ on $(k + 1)$-connected graphs. Then Observation 5 implies that Elimination Distance–($\star$) to φ is NP-complete for every $\star \in \{\text{conn}, \text{prop}, \text{depth}\}$.

Let $(G_1, k), \ldots (G_t, k)$ be instances of Elimination Distance–($\star$) to φ for some $\star \in \{\text{conn}, \text{prop}, \text{depth}\}$. Let G be the disjoint union of $G_1, \ldots, G_t$. Then for $\star \in \{\text{conn}, \text{prop}\}$, we have that (G, k) is a yes-instance of Elimination Distance–($\star$) to φ if and only if (G_j, k) is a yes-instance of Elimination Distance–($\star$) to φ for every $j \in \{1, \ldots, t\}$. Then by the result of Bodlaender, Jansen, and Kratsch [3] (see also [17, Part III] for the introduction to the technique), Elimination Distance–($\star$) to φ does not admit a polynomial kernel unless NP $\subseteq$ coNP /poly. For Elimination Distance–(depth) to φ, consider G' that is the disjoint union of G and K_{k+3}. Clearly, $\text{ed}_\varphi^{\text{depth}}(K_{k+2}) = k + 1$. Then by Lemma 2, $(G', k + 1)$ is a yes-instance of Elimination Distance–(depth) to φ if and only if (G_j, k) is a yes-instance of Elimination Distance–(depth) to φ for every $j \in \{1, \ldots, t\}$. This implies that Elimination Distance–(depth) to φ has no polynomial kernel unless NP $\subseteq$ coNP /poly. □

Notice that Proposition 1 does no exclude existence of *Turing kernels* (we again refer to References [10, 17] for the definition of the notion). This makes it natural to ask whether Elimination Distance–($\star$) to φ admit polynomial Turing kernels for $\varphi \in \Sigma_3$ for $\star \in \{\text{conn}, \text{prop}, \text{depth}\}$.

Recall that Elimination Distance–(depth) to φ can be stated as follows: Given a graph G and a nonnegative integer k, is there $X \subseteq V(G)$ whose torso has the tree-depth at most k such that $G - X \models \varphi$? In other words, we ask whether there is a set of vertices whose torso has bounded tree-depth such that the graph obtained by the deletion of this set models our formula. However, we also can consider different "width-measures." In particular, Eiben et al. [14] introduced the notion of $\mathcal{P}$*-tree-width* of a graph G for a graph propoerty $\mathcal{P}$ as the minimum *tree-wdith* (see, e.g, [10] for the definition) of the torso of $X \subseteq V(G)$ such that $G - X \in \mathcal{P}$. Further results in this direction were recently obtained by Jensen, de Kroon, and Wlodarczyk [22]. We believe that Theorem 2 can be extended for the variant of ELIMINATION DISTANCE–(depth) TO φ, where the tree-width of the torso of X should be at most k. Can the same be said about Theorem 1?

Finally, we believe that it could be interesting to consider yet another variant of the elimination distance. Recall that in the definitions of $\mathrm{ed}_\varphi^\star$ for $\star \in \{\mathrm{conn}, \mathrm{prop}, \mathrm{depth}\}$, we considered properties of the components. In particular,

$$\mathrm{ed}_\varphi^{\mathrm{prop}}(G) = \begin{cases} 0, & \text{if } G \models \varphi \text{ or } G = (\emptyset, \emptyset), \\ 1 + \min_{v \in V(G)} \mathrm{ed}_\varphi^{\mathrm{prop}}(G - v), & \text{if} G \not\models \varphi \text{and} G \text{is connected}, \\ \max\{1, \max\{\mathrm{ed}_\varphi^{\mathrm{prop}}(C) \mid C \text{ is a component of } G\}\}, & \text{otherwise.} \end{cases}$$

However, we can consider *unions* of components instead. We say that graphs $G_1, \ldots, G_s$ form a *component-partition* of G if every component of G is a component of G_i for some $i \in \{1, \ldots, s\}$ and G is the disjoint union of $G_1, \ldots, G_s$. Then, we can define

$$\mathrm{ed}_\varphi^{\mathrm{part}}(G) = \begin{cases} 0, & \text{if} G \models \varphi \text{ or } G = (\emptyset, \emptyset), \\ 1 + \min_{v \in V(G)} \mathrm{ed}_\varphi^{\mathrm{part}}(G - v), & \text{if} G \not\models \varphi \text{and} G \text{is connected}, \\ \min\{\max\{1, \mathrm{ed}_\varphi^{\mathrm{part}}(G_1), \ldots, \mathrm{ed}_\varphi^{\mathrm{part}}(G_s)\} \mid G_1, \ldots, G_s & \\ \qquad \text{is a component partition of } G\} & \text{otherwise.} \end{cases}$$

Then, we can define the respective ELIMINATION DISTANCE–(part) TO φ and investigate its parameterized complexity depending of φ. Note that our approach for solving the elimination problems fails in this case. In particular, we cannot express the problem using MSOL.

REFERENCES

[1] Akanksha Agrawal, Lawqueen Kanesh, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. 2021. An FPT algorithm for elimination distance to bounded degree graphs. In *38th International Symposium on Theoretical Aspects of Computer Science, (STACS) (LIPIcs)*, Vol. 187. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 5:1–5:11. DOI: https://doi.org/10.4230/LIPIcs.STACS.2021.5

[2] Akanksha Agrawal and M. S. Ramanujan. 2020. On the parameterized complexity of clique elimination distance. In *15th International Symposium on Parameterized and Exact Computation (IPEC) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 180. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 1:1–1:13. https://doi.org/10.4230/LIPIcs.IPEC.2020.1

[3] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. 2014. Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.* 28, 1 (2014), 277–305. DOI: https://doi.org/10.1137/120880240

[4] Egon Börger, Erich Grädel, and Yuri Gurevich. 2001. *The Classical Decision Problem.* Springer Science & Business Media.

[5] Jannis Bulian and Anuj Dawar. 2016. Graph isomorphism parameterized by elimination distance to bounded degree. *Algorithmica* 75, 2 (2016), 363–382. DOI: https://doi.org/10.1007/s00453-015-0045-3

[6] Jannis Bulian and Anuj Dawar. 2017. Fixed-parameter tractable distances to sparse graph classes. *Algorithmica* 79, 1 (2017), 139–158. DOI: https://doi.org/10.1007/s00453-016-0235-7

[7] Leizhen Cai, Siu Man Chan, and Siu On Chan. 2006. Random separation: A new method for solving fixed-cardinality optimization problems. In *2nd International Workshop Parameterized and Exact Computation (IWPEC) (Lecture Notes in Computer Science)*, Vol. 4169. Springer, 239–250. DOI: https://doi.org/10.1007/11847250_22

[8] Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michal Pilipczuk. 2016. Designing FPT algorithms for cut problems using randomized contractions. *SIAM J. Comput.* 45, 4 (2016), 1171–1229. DOI: https://doi.org/10.1137/15M1032077

[9] Bruno Courcelle and Joost Engelfriet. 2012. *Graph Structure and Monadic Second-order Logic - A Language-Theoretic Approach.* Encyclopedia of Mathematics and Its Applications, Vol. 138. Cambridge University Press.

[10] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms.* Springer. DOI: https://doi.org/10.1007/978-3-319-21275-3

[11] Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2019. Minimum bisection is fixed-parameter tractable. *SIAM J. Comput.* 48, 2 (2019), 417–450. DOI: https://doi.org/10.1137/140988553

[12] Reinhard Diestel. 2012. *Graph Theory, 4th Edition.* Graduate Texts in Mathematics, Vol. 173. Springer.

[13] Rodney G. Downey and Michael R. Fellows. 2013. *Fundamentals of Parameterized Complexity*. Springer. DOI: https://doi.org/10.1007/978-1-4471-5559-1

[14] Eduard Eiben, Robert Ganian, Thekla Hamm, and O-Joung Kwon. 2021. Measuring what matters: A hybrid approach to dynamic programming with treewidth. *J. Comput. Syst. Sci.* 121 (2021), 57–75. DOI: https://doi.org/10.1016/j.jcss.2021.04.005

[15] Jörg Flum and Martin Grohe. 2006. *Parameterized Complexity Theory*. Springer. DOI: https://doi.org/10.1007/3-540-29953-X

[16] Fedor V. Fomin, Petr A. Golovach, and Dimitrios M. Thilikos. 2020. On the parameterized complexity of graph modification to first-order logic properties. *Theory Comput. Syst.* 64, 2 (2020), 251–271. DOI: https://doi.org/10.1007/s00224-019-09938-8

[17] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. 2019. *Kernelization*. Cambridge University Press, Cambridge.

[18] Markus Frick and Martin Grohe. 2004. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic* 130, 1–3 (2004), 3–31. DOI: https://doi.org/10.1016/j.apal.2004.01.007

[19] Georg Gottlob, Phokion G. Kolaitis, and Thomas Schwentick. 2004. Existential second-order logic over graphs: Charting the tractability frontier. *J. ACM* 51, 2 (2004), 312–362. DOI: https://doi.org/10.1145/972639.972646

[20] Jiong Guo, Falk Hüffner, and Rolf Niedermeier. 2004. A structural view on parameterizing problems: Distance from triviality. In *1st International Workshop Parameterized and Exact Computation (IWPEC) (Lecture Notes in Computer Science)*, Vol. 3162. Springer, 162–173. DOI: https://doi.org/10.1007/978-3-540-28639-4_15

[21] Eva-Maria C. Hols, Stefan Kratsch, and Astrid Pieterse. 2020. Elimination distances, blocking sets, and kernels for vertex cover. In *37th International Symposium on Theoretical Aspects of Computer Science (STACS) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 154. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 36:1–36:14. DOI: https://doi.org/10.4230/LIPIcs.STACS.2020.36

[22] Bart M. P. Jansen, Jari J. H. de Kroon, and Michal Wlodarczyk. 2021. Vertex deletion parameterized by elimination distance and even less. In *53rd Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 1757–1769. DOI: https://doi.org/10.1145/3406325.3451068

[23] John M. Lewis and Mihalis Yannakakis. 1980. The node-deletion problem for hereditary properties is NP-Complete. *J. Comput. Syst. Sci.* 20, 2 (1980), 219–230. DOI: https://doi.org/10.1016/0022-0000(80)90060-4

[24] Alexander Lindermayr, Sebastian Siebertz, and Alexandre Vigny. 2020. Elimination distance to bounded degree on planar graphs. In *45th International Symposium on Mathematical Foundations of Computer Science (MFCS) (LIPIcs)*, Vol. 170. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 65:1–65:12. DOI: https://doi.org/10.4230/LIPIcs.MFCS.2020.65

[25] Alexander Lindermayr, Sebastian Siebertz, and Alexandre Vigny. 2020. Elimination distance to bounded degree on planar graphs. *CoRR* abs/2007.02413 (2020).

[26] Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. 2018. Reducing CMSO model checking to highly connected graphs. In *45th International Colloquium on Automata, Languages, and Programming (ICALP) (LIPIcs)*, Vol. 107. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 135:1–135:14. DOI: https://doi.org/10.4230/LIPIcs.ICALP.2018.135

[27] Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. 2018. Reducing CMSO model checking to highly connected graphs. *CoRR* abs/1802.01453 (2018).

[28] Jaroslav Nešetřil and Patrice Ossona de Mendez. 2006. Tree-depth, subgraph coloring and homomorphism bounds. *Eur. J. Combin.* 27, 6 (2006), 1022–1041.

[29] Rolf Niedermeier. 2006. *Invitation to Fixed-parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications, Vol. 31. Oxford University Press.

[30] C. Smorynski. 1977. The incompleteness theorems. In *Handbook of Mathematical Logic*. Stud. Logic Found. Math., Vol. 90. North-Holland, Amsterdam, 821–865.

[31] Moshe Y. Vardi. 1982. The complexity of relational query languages (Extended Abstract). In *14th Annual ACM Symposium on Theory of Computing*. ACM, 137–146. DOI: https://doi.org/10.1145/800070.802186

[32] Ryan Williams. 2014. Faster decision of first-order graph properties. In *Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic (CSL) and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. ACM, 80:1–80:6. DOI: http://doi.acm.org/10.1145/2603088.2603121

Received April 2021; revised October 2021; accepted February 2022