Using Decomposition-Parameters for QBF: Mind the Prefix!

Eduard Eiben[c], Robert Ganian[a], Sebastian Ordyniak[b]

[a]*Algorithms and Complexity Group, Vienna University of Technology, Austria*
[b]*Department of Computer Science, University of Sheffield, United Kingdom*
[c]*Royal Holloway, University of London, United Kingdom*

Abstract

Similar to the satisfiability (SAT) problem, which can be seen to be the archetypical problem for NP, the quantified Boolean formula problem (QBF) is the archetypical problem for PSPACE. Recently, Atserias and Oliva (2014) showed that, unlike for SAT, many of the well-known decompositional parameters (such as treewidth and pathwidth) do not allow efficient algorithms for QBF. The main reason for this seems to be the lack of awareness of these parameters towards the dependencies between variables of a QBF formula. In this paper we extend the ordinary pathwidth to the QBF-setting by introducing prefix pathwidth, which takes into account the dependencies between variables in a QBF, and show that it leads to an efficient algorithm for QBF. We hope that our approach will help to initiate the study of novel tailor-made decompositional parameters for QBF and thereby help to lift the success of these decompositional parameters from SAT to QBF.

1. Introduction

Many important computational tasks such as verification, planning, and several questions in knowledge representation and automated reasoning can be naturally encoded as the problem of evaluating quantified Boolean formulas [15, 24, 28, 31], a generalization of the propositional satisfiability problem (SAT). In recent years quantified Boolean formulas have become a very active research area. The problem of evaluating quantified Boolean formulas, called QBF, is the archetypical PSPACE-complete problem and is therefore believed to be computationally harder than the NP-complete propositional satisfiability problem [21, 27, 37].

In spite of the close connection between QBF and SAT, many of the tools and techniques that work for SAT are not known to help for QBF, and this is especially true for so-called decomposition-based techniques [2]. Such techniques use various kinds of decompositions to capture the structure of the input, leading to efficient algorithms for computing solutions with run-time guarantees. Decomposition-based techniques

Email addresses: eduard.eiben@gmail.com (Eduard Eiben), rganian@gmail.com (Robert Ganian), sordyniak@gmail.com (Sebastian Ordyniak)

are tied to a numerical *parameter* k, which represents the fitness of the decomposition. The goal is then to obtain algorithms whose running time is polynomial in the input size n and exponential only in k, i.e., with a running time of $f(k) \cdot n^{\mathcal{O}(1)}$ where f is some computable function; such algorithms are called *FPT algorithms*, and problems that admit an FPT algorithm w.r.t. some parameter belong to the class *FPT*. Prominent examples of decompositions used in such techniques include decompositions for the structural parameters treewidth [30], pathwidth [29], clique-width [10] and rank-width [25]; all of these are known to support FPT algorithms for SAT [38, 19], but the same is not true for QBF. Indeed QBF remains PSPACE-complete even on instances with constant pathwidth [3]. As a consequence, many classes of QBFs that have a natural and seemingly "simple" structure remained beyond the reach of current algorithmic techniques; this is also witnessed by previous work of Pan and Vardi [26] that established strong lower bounds for the problem.

In this work we introduce and develop *prefix pathwidth*, which is a novel decomposition-based parameter that allows an FPT algorithm for QBF. Prefix pathwidth is an extension of pathwidth, which takes into account not only the structure of clauses in the formula, but also the structure contained in the quantification of variables. To achieve the latter, we make use of the *dependency schemes* introduced by Samer and Szeider ([32, 34]), see also the work of Biere and Lonsing ([6]). Dependency schemes capture how the assignment of individual variables in a QBF depends on other variables, and research in this direction has uncovered a large number of distinct dependency schemes. The most basic dependency scheme is called the *trivial dependency scheme* [32], which stipulates that each variable depends on all variables with distinct quantification that come before it in the prefix. When using this dependency scheme, we obtain (by combining Theorem 2 with Theorem 34):

Theorem 1. *QBF is FPT parameterized by the prefix pathwidth with respect to the trivial dependency scheme.*

All of our results can also be applied to prefix pathwidth with respect to so-called permutation dependency schemes [35]. Informally, a dependency scheme is a *permutation dependency scheme* if the satisfiability of a QBF formula is independent of the ordering of the variables in the quantifier prefix as long as the ordering is consistent with the ordering implied by the dependency scheme. Since almost all known dependency schemes are permutation dependency schemes all our results apply to a wide range of dependency schemes. In practice, using different dependency schemes may lead to better prefix path-decompositions, in turn resulting in significantly faster algorithms.

In their full generality, our main results on solving QBF using prefix pathwidth can be separated into two components:

1. using a prefix path-decomposition of small prefix pathwidth to solve the given QBF I, and
2. finding a suitable prefix path-decomposition to be used for step 1.

We resolve the first task by applying advanced dynamic programming techniques on partial existential strategies for the Hintikka game (see e.g., the work of Grädel et

al. [14]) played on the QBF. Essentially, the game approach allows us to translate the question of whether a QBF is true to the question of whether there exists a winning strategy for one player in the Hintikka game. We show that although the number of such strategies is unbounded, at each point in the prefix path-decomposition there is only a small number of partial strategies on the processed vertices that need to be considered. Thus we obtain:

Theorem 2. QBF *is FPT parameterized by the width of a prefix path-decomposition w.r.t. any permutation dependency scheme, when such a decomposition is provided as part of the input.*

Resolving step 2 boils down to an algorithmic problem on graphs, which is related to the problem of computing various established parameters of directed graphs, such as directed pathwidth or directed treewidth. It is an important open problem whether computing these parameters is FPT or not [39] and the same obstacles seem to also be present for computing our parameter in the general sense. To bypass this barrier, we develop new algorithmic techniques to obtain three distinct algorithms for computing prefix path-decompositions. The first of these algorithms, presented in Theorem 34, works for the trivial dependency poset as well as other posets that have a similar "layered" structure. The latter two of our algorithms then focus on general posets, but their performance depends on the *poset-width* (i.e., the size of a maximum anti-chain) of the dependency relation; on a high level, the poset-width captures the density of dependencies between variables. In particular, we obtain one polynomial-time approximation algorithm (Theorem 30) and one FPT algorithm (Theorem 29). In combination with the previous Theorem 2, Theorem 30 yields one of our main contributions, formalized in Theorem 3 below. Observe that here we do not require a decomposition to be part of the input.

Theorem 3. *Let τ be a fixed permutation dependency scheme. There exists an FPT algorithm that takes as input a* QBF *I and decides whether I is true in time $f(k, w) \cdot |I|^{\mathcal{O}(1)}$, where f is a computable function, k is the prefix pathwidth and w is the poset-width of I w.r.t. τ.*

We remark that our results have implications for the tractability of QBF with respect to already established structural parameters. We provide an example of this in the Concluding Notes, where we show that QBF is FPT when parameterized by the *vertex cover number* of the matrix (irrespective of the prefix), i.e., the QBF formula without the prefix.

1.1. Related Work

After showing that QBF remains PSPACE-complete on graphs of bounded pathwidth [3], the authors introduced a width parameter based on treewidth, which is called *respectful treewidth*, that allows to take into account dependencies between the variables in a QBF formula. They showed that QBF is fixed-parameter tractable parameterized by respectful treewidth provided that a corresponding tree decomposition is given as part of the input. Similar results have been shown for first-order model checking [1] and quantified constraint satisfaction [9]. In the former reference, it is

also shown that computing an optimal respectful tree decomposition is fixed-parameter tractable, showing that QBF is fixed-parameter tractable parameterized by respectful treewidth. As we will show in Section 3, respectful treewidth is incomparable to our new parameter prefix pathwidth. Informally, there are two main differences between respectful treewidth and prefix pathwidth: (1) whereas respectful treewidth requires the ordering in which the variables are introduced (along the tree decomposition) to be compatible with the dependencies, prefix pathwidth needs the ordering in which the variables are forgotten (along the path decomposition) to be compatible with the dependencies and (2) respectful treewidth is solely defined for the trivial dependency scheme, while prefix pathwidth allows the use of arbitrary permutation dependency schemes. Other structural parameters such as backdoors have also been studied in the context of QBF [32].

Recent follow-up work that builds upon the results established in this paper include a paper by Lampis and Mitsou [23], which (1) investigates upper and lower bounds when using treewidth to solve QBF with a single quantifier alternation, and (2) improves the running time of the fixed-parameter algorithm parameterized by the vertex cover number presented in our Theorem 40. In another recent follow-up [16], the authors of this paper investigate a different and incomparable parameter related to treewidth that can be used for solving QBF; the incomparability of that parameter with the one presented here is established in Lemma 4 in [16].

2. Preliminaries

For $i \in \mathbb{N}$, we let $[i]$ denote the set $\{1, \ldots, i\}$. We refer to the book by Diestel ([12]) for standard graph-theoretic terminology. Given an undirected graph G, we denote by $V(G)$ and $E(G)$ its vertex and edge set, respectively. We use ab as a shorthand for the edge $\{a, b\}$. For a set of vertices $V' \subseteq V(G)$ the *guards* of V', denoted by $\partial(V')$, are the vertices in V' with at least one neighbor in $V(G) \setminus V'$. For a vertex $v \in V(G)$, we denote by $N(v)$ the set of its neighbors (excluding v) and for a vertex set $V' \subseteq V(G)$, we denote by $N(V')$ the set $\bigcup_{v \in V'} N(v) \setminus V'$.

We refer to the standard textbooks [13, 11] for an in-depth overview of parameterized complexity theory. Here, we only recall that a *parameterized problem* (Q, κ) is a *problem* $Q \subseteq \Sigma^*$ together with a (computable) *parameterization* $\kappa \colon \Sigma^* \to \mathbb{N}$, where Σ is a finite alphabet. A parameterized problem (Q, κ) is *fixed-parameter tractable (w.r.t. κ)*, in short *FPT*, if there exists a decision algorithm for Q, a computable function $f \colon \mathbb{N} \to \mathbb{N}$, and a polynomial function $p \colon \mathbb{N} \to \mathbb{N}$, such that for all $x \in \Sigma^*$, the running time of the algorithm on x is at most $f(\kappa(x)) \cdot p(|x|)$. Algorithms with this running time are then referred to as *FPT algorithms*.

2.1. Quantified Boolean Formulas

For a set of propositional variables K, a *literal* is either a variable $x \in K$ or its negation $\neg x$, where $\mathrm{var}(x) = \mathrm{var}(\neg x) = x$ denote the variable of a literal. A *clause* is a disjunction over literals. A *propositional formula in conjunctive normal form* (i.e., a *CNF formula*) is a conjunction of clauses. We say that a CNF formula ϕ is *over* a variable set K if each literal x in ϕ satisfies $\mathrm{var}(x) \in K$, and denote the set of variables

that occur in ϕ by var(ϕ). For notational purposes, we will view a clause as a set of literals and a CNF formula as a set of clauses. We refer interested readers to the Handbook of Satisfiability for extended definitions and an in-depth discussion of the notions presented in this subsection [5].

A *quantified Boolean formula* is a tuple (ϕ, τ) where ϕ is a CNF formula and τ is a sequence of quantified variables (containing any variable at most once), denoted var(τ), that satisfies var(τ) $\supseteq$ var(ϕ); then ϕ is called the *matrix* and τ is called the *prefix*. A QBF (ϕ, τ) is true if the formula $\tau\phi$ is true. An *(partial) assignment* is a mapping from (a subset of) the variables of var(ϕ) to $\{0, 1\}$.

Given a QBF $I = (\phi, \tau)$ and a partial assignment $\omega\colon Q \to \{0, 1\}$ where $Q \subseteq$ var(ϕ), we denote by I_ω the instance obtained by applying the partial assignment ω; similarly, for a clause $c \in \phi$ we let c_ω denote the clause obtained from c by applying ω.

The *primal graph* of a QBF $I = (\phi, \tau)$ is the graph G_I defined as follows. The vertex set of G_I consists of the set of variables that occur in ϕ, and st is an edge in G_I iff there exists a clause in ϕ containing both s and t.

2.2. Dependency Schemes and Posets for QBF

We use *dependency posets* to provide a general and formal way of speaking about the various *dependency schemes* introduced for QBF [32].

We begin by formally defining dependency schemes. For a binary relation $\mathcal{R}$ over some set V we write $\overline{\mathcal{R}}$ to denote its inverse , i.e., $\overline{\mathcal{R}} = \{(y, x) : (x, y) \in \mathcal{R}\}$, and we write $\mathcal{R}^*$ to denote the reflexive and transitive closure of $\mathcal{R}$ i.e., the smallest set $\mathcal{R}^*$ such that $\mathcal{R}^* = \mathcal{R} \cup \{(x, x) : x \in V\} \cup \{(x, y) : \exists z$ such that $(x, z) \in \mathcal{R}^*$ and $(z, y) \in \mathcal{R}\}$. Given a relation $\mathcal{R}$ over some set V such that $\mathcal{R}^*$ is antisymmetric, a *linear extension* of $\mathcal{R}$ is any total order $T_\mathcal{R}$ over V such that if $(a, b) \in \mathcal{R}^*$, then $(a, b) \in T_\mathcal{R}$. Moreover, we let $\mathcal{R}(x) = \{y : (x, y) \in \mathcal{R}\}$ for $x \in V$ and $\mathcal{R}(X) = \cup_{x \in X}\mathcal{R}(x)$ for $X \subseteq V$. Given a QBF $I = (\phi, \tau)$ we will also need the following binary relation over var(τ): $\mathcal{R}_I = \{\, (x, y) \mid x, y \in \text{var}(\tau), x$ is before (to the left of) y in $\tau\,\}$.

To define dependency schemes we need also the notion of *shifting*, which takes some subset of the variables of I in the prefix and puts them together with their quantifiers, in the same relative order, to the end (*down-shifting*) or to the beginning (*up-shifting*) of the prefix.

Definition 4 (Shifting, [32])**.** *Let $I = (\phi, \tau)$ be a QBF and $A \subseteq$ var(τ). We say that $I' = (\phi, \tau')$ is obtained from I by* down-shifting *(*up-shifting*) A, in symbols $I' = S^{\downarrow}(I, A)$ ($I' = S^{\uparrow}(I, A)$), if I' is obtained from I by reordering quantifiers together with their variables in the prefix such that the following holds:*

1. *$A = \mathcal{R}_{I'}(x)$ ($A = \overline{\mathcal{R}_{I'}}(x)$) for some $x \in$ var(τ) and*
2. *$(x, y) \in \mathcal{R}_{I'}$ iff $(x, y) \in \mathcal{R}_I$ for all $x, y \in A$ and*
3. *$(x, y) \in \mathcal{R}_{I'}$ iff $(x, y) \in \mathcal{R}_I$ for all $x, y \in$ var(τ) $\setminus A$*

Definition 5 (Dependency Scheme, [32])**.** *A* dependency scheme *D assigns to each QBF I a binary relation $D_I \subseteq \mathcal{R}_I$ such that I and $S^{\downarrow}(I, D^*_I(x))$ are satisfiability-equivalent (i.e., logically equivalent) for every variable x of I.*

Observe that since $D_I \subseteq \mathcal{R}_I$, D_I^* is always antisymmetric. It is important to note that dependency schemes in general are too general a notion for our purposes. Namely, for our algorithm for QBF using prefix pathwidth, we require dependency schemes that lead to satisfiability-equivalent QBF instances even after several (up-)shifting operations; without updating the dependency scheme D_I after up-shifting. This is why we will focus our attention on so-called *permutation dependency schemes* [35], which are known to satisfy this condition.

Definition 6 (Permutation dependency scheme, [35])**.** *A dependency scheme D is a* permutation dependency scheme *if for every* QBF $I = (\phi, \tau)$ *and every linear extension D'_I of D_I^*, it holds that the QBF obtained by permuting the prefix τ according to D'_I is equivalent with I.*

Note that the definition above implies that permutation dependency schemes preserve satisfiability (i.e., result in satisfiability-equivalent instances) after several (up- or down-) shifting operations because each shifting operation leads to an ordering of the prefix that is a linear extension of the dependency scheme applied to the original formula.

A partially ordered set (*poset*) $\mathcal{V}$ is a pair $(V, \leq^V)$ where V is a set and $\leq^V$ is a reflexive, antisymmetric, and transitive binary relation over V. A *chain* W of $\mathcal{V}$ is a subset of V such that $x \leq^V y$ or $y \leq^V x$ for every $x, y \in W$. An *anti-chain* A of $\mathcal{V}$ is a subset of V such that for all $x, y \in V$ neither $x \leq^V y$ nor $y \leq^V x$. A *chain partition* of $\mathcal{V}$ is a tuple $(W_1, \ldots, W_k)$ such that $\{W_1, \ldots, W_k\}$ is a partition of V and for every i with $1 \leq i \leq k$ the poset induced by W_i is a chain of $\mathcal{V}$. The *width* (or *poset-width*) of a poset $\mathcal{V}$, denoted by $\text{width}(\mathcal{V})$ is the maximum cardinality of any anti-chain of $\mathcal{V}$, which by Dilworth's theorem is equal to the minimum size of any chain partition of $\mathcal{V}$. A subset A of V is *downward-closed* if for every $a \in A$ it holds that $b \leq^V a \implies b \in A$. For brevity we will often write $\leq^V$ for the poset $\mathcal{V} := (V, \leq^V)$.

Proposition 7 ([18])**.** *Let $\mathcal{V}$ be a poset. Then in time $O(\text{width}(\mathcal{V}) \cdot \|\mathcal{V}\|^2)$, it is possible to compute both* $\text{width}(\mathcal{V}) = w$ *and a corresponding chain partition $(W_1, \ldots, W_w)$ of $\mathcal{V}$.*

Definition 8 (Dependency poset)**.** *Given a QBF $I = (\phi, \tau)$ a* dependency poset $\mathcal{V} = (var(\phi), \leq^I)$ *of I is a poset over $var(\phi)$ such that for every linear extension $<$ of $\mathcal{V}$, it holds that I and the QBF instance obtained from I after reordering its prefix according to $<$ are equivalent.*

Note that every permutation dependency scheme gives rise to an dependency poset. Namely, given a QBF $I = (\phi, \tau)$ and a permutation dependency scheme D, the poset over $\text{var}(\phi)$ such that $x \leq^I y$ iff $(x, y) \in D_I^*$ for all $x, y \in \text{var}(\phi)$ is a dependency poset.

The *trivial dependency scheme* assigns to each variable x the closest variables on the right of x in the prefix with different quantification; this corresponds to the fact that variables within the same quantifier block can be permuted without changing the semantics of the formula. This gives rise to the *trivial dependency poset*, which has a certain "layered" structure; more details about these posets are presented in Subsection 5.2. However, more refined dependency posets are known to exist and can be computed efficiently [32].

To illustrate these definitions, consider the following QBF I:

$$\forall x \exists y \forall u \exists v (x \vee \neg y \vee v) \wedge (\neg u \vee \neg v \vee y) \wedge (\neg x \vee u \vee \neg v).$$

As an example, consider the following dependency poset on variables of I: $x \leq^I u \leq^I v$, and y is incomparable to all other variables. Up-shifting of the downward-closed set $\{x, u\}$ yields the QBF I':

$$\forall x \forall u \exists y \exists v (x \vee \neg y \vee v) \wedge (\neg u \vee \neg v \vee y) \wedge (\neg x \vee u \vee \neg v).$$

One can readily see that I and I' are both true. The trivial dependency poset over I is the poset given by the chain $x \leq^I y \leq^I u \leq^I v$, where every downward-closed set cannot be further up-shifted.

2.3. Pathwidth and Treewidth

Definition 9 (Tree decomposition)**.** *A tree-decomposition of a graph G is a pair $(T, \{X_t\}_{t \in V(T)})$, where T is a rooted tree whose every vertex t is assigned a vertex subset $X_t \subseteq V(G)$, called a* bag*, such that the following properties hold:*

(P1) $\cup_{t \in V(T)} X_t = V(G)$*,*

(P2) for every $u \in V(G)$, the set $T_u = \{t \in V(T) : u \in X_t\}$ induces a connected subtree of T, and

(P3) for each $uv \in E(G)$ there exists $t \in V(T)$ such that $u, v \in X_t$.

To distinguish between the vertices of the tree T and the vertices of the graph G, we will refer to the vertices of T as *nodes*. The *width* of the tree-decomposition $\mathcal{T}$ is $\max_{t \in T} |X_t| - 1$. The *treewidth* of G, $tw(G)$, is the minimum width over all tree-decompositions of G.

For our definition of respectful treewidth the following alternative characterization of the treewidth of a graph becomes handy.

Definition 10 (Elimination ordering)**.** *An* elimination ordering *of a graph is a linear order $\pi = (v_1, \ldots, v_n)$ of its vertices. Given an elimination ordering π of the graph G, the* fill-in graph *H of G w.r.t. π is the unique minimal graph such that:*

- $V(G) = V(H)$.
- $E(H) \supseteq E(G)$.
- *If $0 \leq k < i < j \leq n$ and $v_i, v_j \in N_H(v_k)$, then $v_i v_j \in E(H)$.*

The width *of elimination ordering π is the maximum number of neighbors of any vertex v that are larger than v (w.r.t. π) in H.*

The following proposition shows that tree decompositions and elimination orderings are two equivalent characterizations of treewidth.

Proposition 1 ([22])**.** *Let G be a graph. The following two statements are equivalent:*

- *G has treewidth k,*
- *G has an elimination ordering of width k,*

A path-decomposition $\mathcal{P}$ is a tree decomposition where the tree T is a path (rooted at one of the endpoints). Observe that any path-decomposition can be fully characterized by the order of appearance of its bags along T, and hence we will consider succinct representations of path-decompositions in the form $\mathcal{P} = (P_1, \ldots, P_d)$, where P_i is the i-th bag in $\mathcal{P}$. The width of a path-decomposition and the *pathwidth* of G, $pw(G)$, are defined analogously.

We say that a path-decomposition $\mathcal{P} = (P_1, \ldots, P_d)$ is *nice* if $P_1 = P_d = \emptyset$, and furthermore for all $i = 1, \ldots, d-1$ either $|P_{i+1}| = |P_i| + 1$ and $P_i \subseteq P_{i+1}$ (in which case we call the node P_{i+1} an *introduce node*) or $|P_{i+1}| = |P_i| - 1$ and $P_i \supseteq P_{i+1}$ (in which case we call the node P_{i+1} a *forget node*). We note that there exists a polynomial-time algorithm that converts a given arbitrary path-decomposition into a nice path-decomposition of the same width [22, 7].

2.4. Respectful Treewidth

For our comparison to prefix pathwidth, we need to define *respectful treewidth* as it has for instance been introduced in [9, 1, 3].

Let $I = (\phi, \tau)$ be an QBF instance. An elimination ordering of G_I is *respectful* if it is a linear extension of the reverse of the trivial dependency scheme for I; intuitively, this corresponds to being forced to eliminate variables that have the most dependencies first. The respectful treewidth is then defined as the minimum width of any respectful elimination ordering of G_I.

3. Prefix Pathwidth for QBF

Let $G = (V, E)$ be a graph and $\leq^V$ be a partial order of V. For a vertex $v \in V$, we denote by $D_{\leq^V}(v)$ the *downward closure* of v w.r.t. $\leq^V$, i.e., the set $\{\, u \in V(G) \mid u \leq^V v \,\}$. Similarly, for $W \subseteq V$ we let $D_{\leq^V}(W) = \bigcup_{v \in W} D_{\leq^V}(v)$.

Let $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ be a tree decomposition of G. For a node t of T we denote by T_t the subtree of T rooted at t, by $T_{\leq t}$ the set $\bigcup_{s \in T_t} X_s$, and by $T_{<t}$ the set $T_{\leq t} \setminus X_t$. For a vertex $v \in V(G)$ we denote by $f_{\mathcal{T}}(v)$ the unique node t satisfying $v \in X_t$ and $v \notin X_s$, where s is the parent of t in T. For a path decomposition $\mathcal{P} = (P_1, \ldots, P_n)$ of G we define $P_{\leq i}$, $P_{<i}$, and $f_{\mathcal{P}}(v)$ analogously; here and in the following, we will assume that a path decomposition is rooted in the right-most node, which in this example is P_n.

A *prefix tree-decomposition* of $G = (V, E)$ w.r.t. $\leq^V$ is a tree-decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ that has the *downward closure property*, i.e., for every vertex $v \in V$ it holds that $D_{\leq^V}(v) \subseteq T_{\leq f_{\mathcal{T}}(v)}$. Analogously, a *prefix path-decomposition* of $G = (V, E)$ w.r.t. $\leq^V$ is a path-decomposition $\mathcal{P}$ that has the *downward closure property*. The *prefix treewidth* of G w.r.t. $\leq^V$, denoted by $ptw_{\leq^V}(G)$, is the minimum width over all prefix tree decompositions of G. The *prefix pathwidth*, denoted by $ppw(G)$, is defined analogously.

The following theorem shows us that if the width of the dependency poset is small, then prefix pathwidth is actually a good approximation of the prefix treewidth w.r.t. the same dependency poset and hence by using the simpler path-decompositions we can get the same result.

Theorem 11. *Let $G = (V, E)$ be a graph and w the width of the poset $(V, \leq^V)$. Then $ppw_{\leq^V}(G) \leq w \cdot ptw_{\leq^V}(G)$.*

Proof. Let $\mathcal{T} = (T, \{X_t\}_{t \in T})$ be an optimal prefix tree-decomposition of G w.r.t. $\leq^V$. We begin our proof by showing the following claim:

Claim 1. *Let W be a chain of the poset $(V, \leq^V)$. Then there exists a leaf-to-root path in T that contains $f_{\mathcal{T}}(v)$ for every $v \in W$.*

Proof of Claim. Suppose for a contradiction that the set $\{ f_{\mathcal{T}}(v) \mid v \in W \}$ does not lie on a leaf-to-root path. This means that there exist two vertices $u, v \in W$ such that $f_{\mathcal{T}}(v) \notin T_{f_{\mathcal{T}}(u)}$ and $f_{\mathcal{T}}(u) \notin T_{f_{\mathcal{T}}(v)}$. W.l.o.g. we can assume that $u \leq^V v$. The downward closure property of $\mathcal{T}$ then implies that $u \in T_{\leq f_{\mathcal{T}}(v)}$, but then either $u \in X_{f_{\mathcal{T}}(v)}$ and $f_{\mathcal{T}}(v) \in T_{f_{\mathcal{T}}(u)}$, or $u \notin X_{f_{\mathcal{T}}(v)}$ and by then the downward closure property also $f_{\mathcal{T}}(u) \in T_{f_{\mathcal{T}}(v)}$, in either case leading to a contradiction. ■

From the above claim it follows that if $\mathcal{T}$ does not contain unnecessary nodes, i.e., a node t of $\mathcal{T}$ is unnecessary if $X_t \subseteq X_p$ for the parent p of t in $\mathcal{T}$, then $\mathcal{T}$ has at most w leaves. Hence, the prefix path-decomposition $\mathcal{P} := (P_1, \ldots, P_h)$, where h is the height of T, defined by $P_i = \{ v \mid v \in B$, where B is a bag of distance i from the root of $T \}$ is a prefix path-decomposition of G w.r.t. $\leq^V$ of width at most w times the width of $\mathcal{T}$. □

We build on the above definitions to define the notions we need on QBFs. A prefix path-decomposition of a QBF $I = (\phi, \tau)$ w.r.t. a dependency poset $\mathcal{V} = (\text{var}(\phi), \leq^I)$ is a prefix path-decomposition of the primal graph G_I w.r.t. $\leq^I$. The prefix pathwidth of I is then the minimum width over all prefix path-decompositions of G_I w.r.t. $\mathcal{V}$. We note that using the same technique as for path-decomposition, one can show that every prefix path-decomposition of G can be turned into a nice prefix path-decomposition of the same width in polynomial time. We also remark that if the dependency poset imposes no restrictions, as it is the case, e.g., if all variables are quantified in the same way, then prefix pathwidth is equal to the pathwidth (of the primal graph).

In order to compute prefix path-decompositions, we will later (in Section 5) introduce other equivalent characterizations of prefix pathwidth, and these may provide additional insight into the notion. For instance, readers familiar with the notion of *directed pathwidth* [39] may be interested in the characterization provided by Observation 27.

3.1. Comparison to Respectful Treewidth

Respectful treewidth is based on Q-resolution [8] and thus decomposes the dependency structure *beginning from variables that have the most dependencies* (i.e., could be down-shifted to the end of the prefix). In contrast, our parameter prefix pathwidth is based on bounding the number of viable strategies in the classical two-player game

characterization of the QBF problem. As such, it decomposes the dependency structure of a QBF instance *beginning from variables that have the least dependencies* (i.e., may be up-shifted to the beginning of the prefix). Lemma 12 shows that both approaches are, in principle, incomparable: there exist classes of QBF instances where one approach leads to polynomial-time algorithms and the other does not, and vice-versa.

Lemma 12. *Let $\mathcal{P}$ be the trivial dependency poset. There exist infinite classes $\mathcal{A}, \mathcal{B}$ of QBF instances such that:*

a. $\mathcal{A}$ has unbounded respectful treewidth but prefix pathwidth w.r.t. $\mathcal{P}$ at most 1;

b. $\mathcal{B}$ has unbounded prefix pathwidth (and prefix treewidth) w.r.t. $\mathcal{P}$ but respectful treewidth at most 1.

Proof. **a.** Let

$$A_i = \exists x_1, \ldots, \exists x_i \forall y \exists x (y \vee x) \wedge \bigwedge_{j=1}^{i} (x_j \vee x).$$

The trivial dependency poset $\mathcal{P}_i$ for A_i is $\{x_1, \ldots, x_i\} \leq y \leq x$. Hence every respectful elimination ordering must start with x, and then the width of such an ordering would be $i + 1$. On the other hand, it is straightforward to verify that the path decomposition $\mathcal{Q} = (Q_1, \ldots, Q_{i+1})$, where $Q_j = \{x_j, x\}$ for $1 \leq j \leq i$ and $Q_{i+1} = \{y, x\}$, is a prefix path-decomposition w.r.t. $\mathcal{P}_i$ of width 1.

b. Consider the following formula with alternating prefix:

$$B_i = \exists x_1 \forall x_2 \exists x_3 \ldots \forall x_{2^i} \exists x_{2^i+1} \bigwedge_{j=1}^{2^{i-1}} ((x_j \vee x_{2j}) \wedge (x_j \vee x_{2j+1})).$$

Since the quantifiers in the prefix of B_i alternate, the trivial dependency poset $\mathcal{P}_i$ for B_i would be the linear order $x_1 \leq x_2 \leq \cdots \leq x_{2^i}$. It is readily observed that the primal graph of B_i is a balanced binary tree of depth i, and it is known that the pathwidth of such trees is $\lceil (i-1)/2 \rceil$ [33]. From the fact that pathwidth is a trivial lower bound for prefix pathwidth together with Theorem 11, it follows that $i - 1$ is a lower bound on the prefix treewidth of B_i.

On the other hand, since $\mathcal{P}_i$ is a linear order, the only respectful elimination ordering is the reverse of $\mathcal{P}_i$. Moreover, from the definition of B_i, it is easily seen that x_j has at most 1 neighbor that is smaller w.r.t. $\mathcal{P}_i$, namely $x_{\lfloor j/2 \rfloor}$. Therefore, the respectful treewidth of B_i is one. □

4. Using Prefix Pathwidth

In this section we will show that deciding the satisfiability of a QBF is fixed-parameter tractable parameterized by the width of a prefix path-decomposition, which is assumed to be provided as part of the input. The next section will then show how such a prefix path-decomposition can be computed efficiently.

4.1. Section Overview

The route to the main goal of this section, i.e., an FPT algorithm for QBF, can be conceptually separated into three parts, each corresponding to one subsection. First, our techniques essentially rely on the well-known Hintikka Games [14], which we introduce in the next subsection. In particular, the notion of a "winning existential strategy" will be crucial for the algorithm; a QBF instance is true if and only if the existential player has a winning strategy. Second, we show that even though the number of existential strategies can be potentially unbounded, they can be grouped into a small (i.e., bounded by k) number of equivalence classes. This equivalence is formalized in Definition 17 via the use of so-called "signatures". The final subsection then presents the dynamic programming algorithm itself; the algorithm maintains and dynamically computes records of relevant signatures, which contain all the needed information about existential strategies on the dynamically processed variables.

4.2. Hintikka Games

Alternating prenex form. For the definition of Hintikka Games (and in particular their strategies), it will be convenient to use an equivalent but more structured representation of QBFs. A QBF is in *alternating prenex form* if the prefix has the form $\forall y_1 \exists x_1, \ldots, \forall y_\ell \exists x_\ell$. Any QBF in alternating prenex form can then be represented as a tuple (ϕ, Y, X) where ϕ is the matrix and $Y = (y_1, \ldots, y_\ell)$ and $X = (x_1, \ldots, x_\ell)$ are disjoint ordered sets of the variables in the prefix.

We remark that any QBF can be transformed into alternating prenex form in linear time by the addition of *dummy variables*, i.e., variables which do not occur in the matrix. It is readily observed that if two dummy variables occur consecutively in the prefix, then they can both be deleted without changing the truth value of the QBF. As a consequence, we may freely assume that the number of dummy variables will never be greater than $\frac{|\mathrm{var}(\phi)|}{2}$. Moreover, adding the dummy variables does not change the prefix path-decomposition since the dummy variables do not occur in the matrix but solely in the prefix of the QBF formula. In the remainder of this section, we will assume that every QBF is in alternating prenex form.

Hintikka Games. Given a QBF (ϕ, Y, X) such that $|X| = |Y| = \ell$, a *strategy for Eloise* (an *existential strategy*) is a sequence of mappings $\Gamma = (\tau_i : \{0,1\}^i \to \{0,1\})_{i=1,\ldots,\ell}$. An existential strategy Γ is *winning* if, for any mapping $\delta : \{y_1, \ldots, y_\ell\} \to \{0,1\}$, the formula ϕ is true under the assignment $y_i \mapsto \delta(y_i)$ and $x_i \mapsto \tau_i(\delta(y_1), \ldots, \delta(y_i))$ for $1 \leq i \leq \ell$. A *partial existential strategy* is a sequence of mappings $\Gamma = (\tau_i : \{0,1\}^i \to \{0,1\})_{i=1,\ldots,\ell'}$, for some $\ell' \leq \ell$.

A *strategy for Abelard* (a *universal strategy*) is defined analogously, whereas the mappings δ and τ are swapped, and we call a universal strategy winning if ϕ is not true when variables are valuated according to the strategy. Formally, it is a sequence of mappings $\Lambda = (\lambda_i : \{0,1\}^{i-1} \to \{0,1\})_{i=1,\ldots,\ell}$. A universal strategy Λ is *winning* if, for any mapping $\delta : \{x_1, \ldots, x_\ell\} \to \{0,1\}$, the formula ϕ is false under the assignment $x_i \mapsto \delta(x_i)$ for $1 \leq i \leq \ell$, $y_1 \mapsto \lambda_1$, and $y_i \mapsto \lambda_i(\delta(x_1), \ldots, \delta(x_{i-1}))$ for $2 \leq i \leq \ell$.

A mapping δ from a subset of Y to $\{0,1\}$ is called a *universal play*, and similarly a mapping δ from a subset of X to $\{0,1\}$ is called an *existential play*. It will sometimes

be useful to view plays as binary strings, and in this context we will use the symbol $\circ$ to denote the concatenation of two strings; for instance, if $\delta(x_1) = 1$ and $\delta(x_2) = 0$, then one can represent δ as $(1, 0)$, and $(1, 0) \circ (0) = (1, 0, 0)$. It is easily observed that plays on dummy variables do not need to be taken into account by a winning existential strategy.

Proposition 13 (folklore)**.** *A* QBF I *is true iff there exists a winning existential strategy on* I *iff there exists no winning universal strategy on* I*.*

Let α be a partial existential strategy restricted to $X' = (x_1, \ldots, x_a)$ and let β be a universal play over $Y' = \{y_1, \ldots, y_b\}$. Then the pair (β, α) results in a partial assignment δ of $X' \cup Y'$, formally given as follows (for i up to $\min(a, b)$): $\delta(y_i) = \beta(y_i)$ and $\delta(x_i) = \alpha(\beta(y_1), \beta(y_2), \ldots, \beta(y_i))$. We denote this as $(\beta, \alpha) \rightsquigarrow \delta$. For brevity, we also sometimes just write (β, α) for the assignment δ given by $(\beta, \alpha) \rightsquigarrow \delta$.

For the remainder of this section, we fix the following notions. Let $I = (\phi, Y, X)$ be a QBF, let $\leq^I$ be a partial order forming a dependency poset of I (w.r.t. some permutation dependency scheme), and let $\mathcal{P} := (P_1, \ldots, P_n)$ be a prefix path-decomposition of I w.r.t. $\leq^I$ of width k. Moreover, for every i with $1 \leq i \leq n$, let $D_i = D_{\leq^I}(P_{<i})$, $C_i = P_{<i}$ (see Figure 1), and let I be up-shifted on D_i.

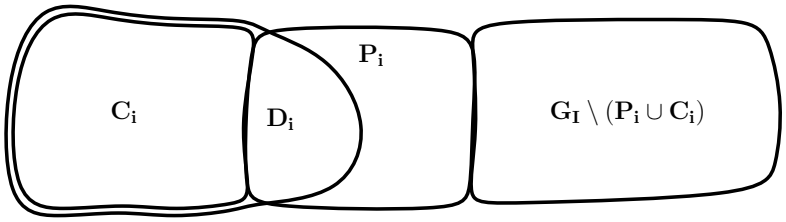

Figure 1: P_i is a bag in $\mathcal{P}$ that separates C_i, i.e., vertices forgotten in some bag before P_i, from the rest of the graph. D_i is the downward closure of C_i w.r.t. $\leq^I$.

Observation 14. *For any* i *with* $1 \leq i \leq n$*,* P_i *forms a separator in* G_I *and hence each clause in* ϕ *either contains only variables in* $P_{\leq i}$ *or only variables in* $(Y \cup X) \setminus C_i$*. Furthermore,* $D_i \subseteq P_{\leq i}$ *and* $C_i \subseteq D_i$*.*

Note that C_i can be a proper subset of D_i since the downward closure $D_{\leq^I}(P_{<i})$ can contain a vertex in $P_i \cap P_{i-1}$.

Hintikka games allow us to decide the truthfulness of a QBF by computing all strategies for the existential player. We will show next that even though the number of possible strategies that can be used for the variables in each $P_{\leq i}$ is huge, it is sufficient to only remember a small number of "representative strategies" that can be used on $P_{\leq i}$ to allow dynamic programming along the prefix path-decomposition. The proof of this claim is based on considering two layers of equivalences and showing that they both only have a small number of equivalence classes.

4.3. Equivalence of Assignments

Recall that for a partial assignment δ the notion I_δ refers to I after δ has been applied. Also recall that we view clauses as sets of literals and CNF formulas as sets

of clauses. Therefore, it is possible that $I_{\delta_1} = I_{\delta_2}$ even if $\delta_1 \neq \delta_2$.

The first equivalence, which serves as the building block for the latter one, considers partial assignments that are defined exactly on the variable set D_i.

Definition 15. *Let δ_1 and δ_2 be two partial assignments defined exactly on the variables in D_i. Then we say $\delta_1 \approx \delta_2$ if $I_{\delta_1} = I_{\delta_2}$.*

It is readily observed that $\approx$ is an equivalence. We prove that its index is bounded by a function of k.

Lemma 16. *$\approx$ has at most $2^{2^{\mathcal{O}(k)}}$ equivalence classes.*

Proof. Consider two partial assignments δ_1, δ_2 defined exactly on the variables in D_i. Let U be the set of all possible clauses over $P_i \setminus D_i$ (including the empty clause); clearly $|U| \leq 3^k$. Let U_1 contain the clauses from U that occur in I_{δ_1}, and similarly for U_2 and I_{δ_2}. Let δ_i' be the restriction of δ_i to variables from $D_i \cap P_i$.

Claim 2. *If $U_1 = U_2$ and $\delta_1' = \delta_2'$, then $\delta_1 \approx \delta_2$.*

Proof of Claim. Clearly, I_{δ_1} and I_{δ_2} are defined on the same variables. It remains to show that both contain the same clauses. Let c be a clause of I. Because of Observation 14 either $c \subseteq (X \cup Y) \setminus C_i$ or $c \subseteq P_{\leq i}$. In the first case, it follows from $\delta_1' = \delta_2'$ that $c_{\delta_1} = c_{\delta_2}$ and hence I_{δ_1} contains c_{δ_1} if and only if so does I_{δ_2}. In the later case, we obtain that $c_{\delta_1} \subseteq P_i \setminus D_i$. Hence, because $U_1 = U_2$, we obtain that I_{δ_1} contains c_{δ_1} if and only if so does I_{δ_2}. ■

The Lemma then follows from the above claim because there are at most $2^{|U|} = 2^{3^k}$ possible choices of U_i and at most 2^k possible choices of δ_i'. □

4.4. Equivalence of Strategies

For a partial existential strategy α on D_i, we denote by S_α (referred to as the *signature*) the set containing each instance I' such that there exists a universal play β that together with α results in I'; formally, $S_\alpha = \{\, I_\delta \mid \exists \text{ universal play } \beta \text{ such that } (\beta, \alpha) \rightsquigarrow \delta \,\}$.

Definition 17. *Let α_1 and α_2 be two partial existential strategies on D_i. Then $\alpha_1 \equiv \alpha_2$ iff $S_{\alpha_1} = S_{\alpha_2}$.*

Once again, it is easy to verify that $\equiv$ is transitive, reflexive and symmetric, and hence is an equivalence relation. We show that its index is also upper-bounded by a function of k.

Lemma 18. *For any partial existential strategy α on D_i it holds that $|S_\alpha| \leq 2^{2^{\mathcal{O}(k)}}$. Furthermore, $\equiv$ has at most $2^{2^{2^{\mathcal{O}(k)}}}$ equivalence classes.*

Proof. By Lemma 16 each partial assignment δ that is defined exactly on the variables in D_i results in one of at most $2^{2^{\mathcal{O}(k)}}$ many distinct QBF instances I_δ. Because S_α is a subset of the set of all these instances for any partial existential strategy α, it follows that $|S_\alpha| \leq 2^{2^{\mathcal{O}(k)}}$ and moreover that there are at most $2^{2^{2^{\mathcal{O}(k)}}}$ distinct choices for S_α. □

Below we prove that the signatures of partial strategies obtained from winning strategies contain only true instances.

Lemma 19. *Let I be a QBF and let α be a winning existential strategy for I. Then, for any partial existential strategy α' that is a subset of α, it holds that I' is true for any $I' \in S_{\alpha'}$.*

Proof. Assume for a contradiction that this is not case and let α' be defined on the variables in $X' \cup Y'$, where $X' \subseteq X$ and $Y' \subseteq Y$ and let $I' \in S_{\alpha'}$ be a no-instance. Because $I' \in S_{\alpha'}$ there is a universal play β_0 on the variables in Y' such that $I' = I_{\delta_0}$, where $(\beta_0, \alpha') \rightsquigarrow \delta_0$. Because I' is a no-instance there is an universal play β_1 on the variables in $Y \setminus Y'$ such that the instance I_δ, where $(\beta_0 \circ \beta_1, \alpha) \rightsquigarrow \delta$, contains the empty clause. Hence, the universal play $\beta_0 \circ \beta_1$ wins against α on I, contradicting our assumption that α is a winning strategy. □

We will also need the converse of the above claim, formulated below.

Observation 20. *If there exists a partial existential strategy α on D_i such that each $I' \in S_\alpha$ is true, then I is true.*

Proof. A winning existential strategy for I can simply apply α until it reaches a true (sub)-instance $I' \in S_\alpha$. From there on, it can continue with a winning existential strategy for I'. □

4.5. The Algorithm

In this subsection, we develop a dynamic programming algorithm on a nice prefix path-decomposition $\mathcal{P} = (P_1, \ldots, P_n)$ of I to decide whether I is true. Recall that by the above, for each partial existential strategy α on D_i there is a signature S_α. For each D_i, we will compute the set K_i of all signatures corresponding to any partial existential strategy on D_i; formally, $K_i = \{\, S_\alpha \mid \alpha \text{ is an existential strategy on } D_i \,\}$. We call K_i the *signature set* of D_i, and the algorithm proceeds by computing the sets $K_1, \ldots, K_n$ for the bags $P_1, \ldots P_n$. One key observation is that for the construction of the sets K_i one only needs to consider a special type of partial existential strategies on D_i, which we will call oblivious.

A (partial) existential strategy α on $X_0 = (x_1, \ldots, x_j)$ is *oblivious* if it does not distinguish between universal plays that lead to the same reduced instance. More precisely, if two universal plays on the first l universal variables result in the same reduced instance, then an oblivious (partial) existential strategy α shall not distinguish between these two universal plays in the moves following after l. Formally, α is oblivious if it satisfies the following condition for every partial existential strategy α' obtained as a restriction of α to $(x_1, \ldots, x_l)$, $l < j$, and for every two universal plays β_1, β_2 on $(y_1, \ldots, y_l)$ such that $I_{\delta_1} = I_{\delta_2}$ where $(\beta_1, \alpha') \rightsquigarrow \delta_1$ and $(\beta_2, \alpha') \rightsquigarrow \delta_2$. Let p satisfy $l < p \leq j$, and for each $\beta_p = \{0,1\}^{p-l}$ let $(\beta_1 \circ \beta_p, \alpha) \rightsquigarrow \delta_1''$ and similarly $(\beta_2 \circ \beta_p, \alpha) \rightsquigarrow \delta_2''$. Then, for every x_i where $l < i \leq p$, it holds that $\delta_1''(x_i) = \delta_2''(x_i)$. The following shows we can compute K_i, by merely considering signatures of oblivious partial existential strategies.

Lemma 21. *Let I be a* QBF. *For any partial existential strategy there is an oblivious partial existential strategy that has the same signature.*

Proof. Let α be a partial existential strategy of I. If α is oblivious, then the claim of the lemma holds. So assume that α is not oblivious. We will show how to transform α into an oblivious partial existential strategy without changing the signature. Let l be the smallest number such that the restriction α' of α to $(x_1, \ldots, x_l)$ violates the definition of obliviousness for some universal plays β_1, β_2 on $(y_1, \ldots, y_l)$. Let $S = \{\, I_\delta \mid \exists \beta' \in \{0,1\}^l : (\beta', \alpha') \rightsquigarrow \delta \,\}$, and for each $I_\delta \in S$ let us choose an arbitrary representative β_δ such that $(\beta_\delta, \alpha') \rightsquigarrow \delta$.

Now consider the strategy α'' which copies α in all mappings except for the following. For each universal play β_1 on $(y_1, \ldots, y_l)$ such that β_1 is distinct from each of the representatives β_δ, for each p where $l < p$, and for each $\beta_p \in \{0,1\}^{p-l}$, we set $\alpha''_p(\beta_1 \circ \beta_p) := \alpha_p(\beta_\delta \circ \beta_p)$, where δ satisfies $I_\delta = I_{(\beta_1, \alpha)}$. Observe that α'' no longer violates the condition of obliviousness for any β_1 on $(y_1, \ldots, y_l)$. Additionally, if the condition of obliviousness was satisfied by α for a pair of universal plays β'_1, β'_2 then it remains satisfied also by α''. For now, assume α'' has the same signature as α; then by repeating the above procedure until we obtain an oblivious strategy would prove the claim of the lemma.

To complete the proof, we argue that α'' has the same signature as α. Let β be an assignment of Y partitioned into β_1 (on $(y_1, \ldots, y_l)$) and β_r (on the remaining universal variables) and let δ_1 be the assignment obtained as $(\beta_1, \alpha'') \rightsquigarrow \delta_1$. Moreover, let β_δ be the representative of I_{δ_1} and let δ_2 be the assignment obtained as $(\beta_\delta, \alpha) \rightsquigarrow \delta_2$. Then, $I_{\delta_1} = I_{\delta_2}$ and $(\beta_1 \circ \beta_r, \alpha'')$ is equal to $(\beta_\delta \circ \beta_r, \alpha)$ on all variables x_i, y_i with $i > l$ (by definition of α''). Hence, $I_{\delta'_1} = I_{\delta'_2}$, where $(\beta_1 \circ \beta_r, \alpha'') \rightsquigarrow \delta'_1$ and $(\beta_\delta \circ \beta_r, \alpha) \rightsquigarrow \delta'_2$, as required. □

The algorithm then consists of the following four procedures, each tied to a specific claim:

1. *Initialization*(Observation 22): this is the procedure that is called at the beginning of the algorithm, i.e., for the empty bag P_1.

2. *Introduce* (Observation 23): this is the procedure that is called whenever we have computed K_{i-1} and P_i is an introduce node.

3. *Forget* (Lemma 24): this is the procedure that is called whenever we have computed K_{i-1} and P_i is a forget node.

4. *Termination* (Observation 25): this is the procedure that is called when we have computed K_n.

The claims are provided below. We remark that each procedure not only computes the next signature set, but also implicitly ensures that I is up-shifted on D_i.

Observation 22. *There exists a constant-time algorithm which takes as input a QBF instance I and a prefix path-decomposition $\mathcal{P}$ and outputs K_1.*

Proof. Since D_1 is empty, K_1 contains only a single signature (the signature of the empty strategy), which contains I. In other words, $K_1 = \{\{I\}\}$. Observe that I is up-shifted on D_1. □

Observation 23. *There exists a constant-time algorithm which takes as input a QBF instance I, a prefix path-decomposition $\mathcal{P}$ and the signature set K_{i-1} and outputs K_i when P_i is an introduce node.*

Proof. Assume $P_i = P_{i-1} \cup \{z\}$, where $z \in X \cup Y$. Then $P_{<i-1} = P_{<i}$ and in particular $z \notin D_i$. Hence $K_i = K_{i-1}$. Observe that if I was up-shifted on D_{i-1}, then I will also be up-shifted on D_i. □

Lemma 24. *There exists an FPT algorithm (with run-time $2^{2^{2^{\mathcal{O}(k)}}} \cdot |\phi|$) which takes as input a QBF I, a prefix path-decomposition $\mathcal{P}$ such that I is up-shifted on D_{i-1} and the signature set K_{i-1} and outputs K_i when P_i is a forget node.*

Proof. Assume $P_i = P_{i-1} \setminus \{z\}$, where $z \in X \cup Y$. Then $z \in P_{<i}$ but $z \notin P_{<i-1}$, which implies that $D_i = D_{i-1} \cup \mathcal{D}_{\leq^I}(z)$. The algorithm checks whether $z \in D_{i-1}$ or not. If $z \in D_{i-1}$, then $\mathcal{D}_{\leq^I}(z) \subseteq D_{i-1}$ and hence $D_i = D_{i-1}$. This means that $K_i = K_{i-1}$.

If $z \notin D_{i-1}$, then let $Z = \mathcal{D}_{\leq^I}(z) \setminus D_{i-1} = D_i \setminus D_{i-1}$. Observe that $Z \subseteq P_i$ and hence $|Z| \leq k$. We apply up-shifting on D_i; since I was already up-shifted on D_{i-1}, this means that after up-shifting the prefix of I will contain first the variables in D_{i-1}, followed by the variables in Z, and then all remaining variables. Our goal is now to expand the signature set K_{i-1} by considering all possible results of an existential strategy and universal play on Z. We first formalize the notion of extended signature below, and then describe how the algorithm proceeds.

Let S be a signature in K_{i-1} and let $I_\delta \in S$. Let $\mathcal{A}$ be the set of all partial existential strategies in I_δ on the variable set $Z \cap X$. Since Z forms a prefix of I_δ and $|Z| \leq k$, it follows that $|\mathcal{A}| \leq 2^{2^{\mathcal{O}(k)}}$. Similarly, let $\mathcal{B}$ be the set of all universal plays in I_δ on the variable set $Z \cap Y$, and observe $|\mathcal{B}| \leq 2^k$. The *extended signature* w.r.t. I_δ and $\alpha_0 \in \mathcal{A}$ is the set $S^{I_\delta}_{\alpha_0} = \{\, I_\omega \mid \omega = \delta \cup \delta',$ whereas $\exists \beta_0 \in \mathcal{B} :$ $(\beta_0, \alpha_0) \rightsquigarrow \delta'$ within $I_\delta \,\}$. Observe that, by the bound on $|\mathcal{B}|$, it follows that each extended signature can be computed from a given I_δ and α_0 in $2^k \cdot \mathcal{O}(|\phi|)$ time.

The algorithm begins by setting $K'_i := \emptyset$ and iteratively processes each $S \in K_{i-1}$ as follows. Let $S = \{I_1, \ldots, I_m\}$. The algorithm branches over all m-tuples of (possibly non-distinct) partial existential strategies from $\mathcal{A}$, and for each such $\tau = (\alpha_1, \alpha_2, \ldots, \alpha_m) \in \mathcal{A}^m$ it produces m pair-wise extended signatures $S^{I_1}_{\alpha_1}, S^{I_2}_{\alpha_2}, \ldots, S^{I_m}_{\alpha_m}$. It then computes their union $S_\tau = \bigcup_{j \in [m]} S^{I_j}_{\alpha_j}$, and updates $K'_i := K'_i \cup \{S_\tau\}$. Therefore, for every $S \in K_{i-1}$, the run-time for updating K'_i is equal to $(|\mathcal{A}|)^{|S|} 2^k |\phi|$. Since $|\mathcal{A}| \leq 2^{2^{\mathcal{O}(k)}}$, $|S| \leq 2^{2^{\mathcal{O}(k)}}$ (by Lemma 18), and $|K_{i-1}| \leq 2^{2^{2^{\mathcal{O}(k)}}}$ (by Lemma 18), we obtain that the total run-time of this algorithm is at most $2^{2^{2^{\mathcal{O}(k)}}} \cdot |\phi|$.

It remains to show that $K'_i = K_i$. We first show that if $S \in K'_i$, then $S \in K_i$. Because $S \in K'_i$, there exists $S_0 \in K_{i-1}$, where $S_0 = \{I_1, \ldots, I_m\}$, along with an m-tuple $\tau := (\alpha_1, \ldots, \alpha_m)$ of partial existential strategies such that $S = \bigcup_{j \in [m]} S^{I_j}_{\alpha_j}$.

Let us fix one arbitrary partial existential strategy α_0 on D_{i-1} with the signature S_0. Then, α_0 partitions all possible assignments $\mathcal{B}$ of $Y \cap D_{i-1}$ into sets $\mathcal{B}_1, \ldots, \mathcal{B}_m$ where $\beta \in \mathcal{B}_j$ iff $(\beta, \alpha_0) \rightsquigarrow I_j$.

Consider the partial existential strategy α on D_i which proceeds as follows. On D_{i-1}, it copies α_0. On $D_i \setminus D_{i-1}$, it takes each universal play β and decomposes it into $\beta_0 \circ \beta_z$, where β_0 is the universal play on D_{i-1} and β_z is the universal play on the remaining variables, i.e., a subset of $D_i \setminus D_{i-1}$ and copies α_j on β_z. By construction, it follows that α has the signature S, which implies that $S \in K_i$, as required.

Now assume that $S \in K_i$. Because of Lemma 21, it holds that S is the signature of an oblivious partial existential strategy α on D_i. Let α_0 be the restriction of α to D_{i-1}, and let $S_0 = \{I_1, \ldots, I_m\}$ be the signature of α_0 in K_{i-1}. Then α_0 once again partitions all possible assignments $\mathcal{B}$ of $Y \cap D_{i-1}$ into sets $\mathcal{B}_1, \ldots, \mathcal{B}_m$ where $\beta \in \mathcal{B}_j$ iff $I_{(\beta,\alpha_0)} = I_j$.

For any $j \in [m]$, let β_j be an arbitrary universal play in $\mathcal{B}_j$. Moreover, let α_j be the partial existential strategy (operating on I_j) defined as follows. For each universal play β_z on (a subset of) $Y \cap Z$, we let α_j copy the move of α against the universal play $\beta_j \circ \beta_z$. Observe that because α is oblivious, α_j is independent of the actual choice of β_j in $\mathcal{B}_j$.

We claim that $S = S_\tau$ for $\tau = (\alpha_1, \ldots, \alpha_m)$, which implies that $S \in K'_i$. We first show that if $I \in S$ then $I \in S_\tau$. Hence, let $I \in S$. Then, there is a universal play β on D_i such that $I = I_{(\beta,\alpha)}$. Let β_0 be the restriction of β to $D_{i-1} \cap Y$, let β_1 be the restriction of β to $Z \cap Y$, and let j be such that $\beta_0 \in \mathcal{B}_j$. Then, because α is oblivious, it holds that $I = I_{(\beta_j \circ \beta_1, \alpha)} \in S^{I_j}_{\alpha_j}$. This shows that $I \in S_\tau$, as required.

For the reverse direction let $I \in S_\tau$. Then, there is a $j \in [m]$ such that $I \in S^{I_j}_{\alpha_j}$. Hence, there is a universal play β_z on $Z \cap Y$ such that $I = I_{(\beta_j \circ \beta_z, \alpha)}$, which implies $I \in S$, as required. □

Observation 25. *There exists a constant-time algorithm which takes as input a QBF instance I and a prefix path-decomposition $\mathcal{P} = \{P_1, \ldots, P_n\}$ and the signature set K_n and decides whether I is true.*

Proof. After processing the last bag P_n of $\mathcal{P}$, it holds that $D = X \cup Y$ and hence every signature in K_n can only contain two possible instances: the trivial true instance I_T which contains no variables and no clauses, and the trivial false instance I_F, which contains no variables and the empty clause. If $\{I_T\} \in K_n$, then the algorithm outputs true, and otherwise it outputs false. Correctness follows from Lemma 19 and Observation 20. □

Having established the procedures for the individual nodes of the path-decomposition, we can now prove the correctness of the whole dynamic programming algorithm. We note that the following theorem applies to arbitrary QBF instances (not only those in alternating prenex form).

Theorem 26. *There exists an FPT algorithm which takes as input a* QBF *I, an integer parameter k, and a prefix path-decomposition $\mathcal{P}$ of I of width at most k (w.r.t. to some dependency poset) and decides whether I is true.*

Proof. If I is not already in alternating prenex form, we convert it by adding dummy variables as described in Subsection 4.2. As mentioned in the preliminaries, it is possible to transform a prefix path-decomposition $\mathcal{P}$ into a nice prefix path-decomposition $\mathcal{P}'$ containing at most $\mathcal{O}(k|I|)$ bags; let $n = |\mathcal{P}'|$. It then computes the signature set K_1 by Observation 22 and proceeds by iteratively computing $K_2, K_3, \ldots, K_n$ by Observation 23 and Lemma 24. The run-time for each step is dominated by the run-time of Lemma 24 and therefore the total run-time is equal to $2^{2^{2^{\mathcal{O}(k)}}} \cdot |\phi| \cdot n$, which shows that the algorithm runs in fpt-time w.r.t k. Once the algorithm computes the signature set K_n, it outputs based on Observation 25. □

5. Computing Prefix Pathwidth

This section is devoted to parameterized and approximation algorithms for computing the prefix pathwidth. Observe that the prefix pathwidth of the graph G w.r.t. the empty partial ordering is the same as the pathwidth of G. Therefore, since computing pathwidth is NP-complete so is computing prefix pathwidth.

Before we present our algorithms, we will state some interesting observations about prefix path-decompositions. For the remainder of this section let G be a graph and $(V(G), \leq^V)$ a poset on $V(G)$ of width w. The first observation relates prefix pathwidth with a well-known decompositional parameter for directed graphs, i.e., directed pathwidth [4].

Definition 26.1 (Directed path-decomposition ([4]))**.** *Let D be a directed graph. A* directed path-decomposition *is a sequence $\mathcal{P} := (P_1, \ldots, P_n)$ of subsets of vertices of D such that:*

(D1) $\bigcup_{1 \leq i \leq n} P_i = V(D)$*,*

(D2) for every $u \in V(D)$, the set $D_u = \{\, i \in \{1, \ldots, n\} \mid u \in W_i \,\}$ induces an interval, and

(D3) for each $uv \in E(D)$ there are i and j with $1 \leq i \leq j \leq n$ such that $u \in W_i$ and $v \in W_j$.

The width *of a directed path-decomposition and the* directed pathwidth *of D, denoted by $dpw(D)$ are defined analogously to the corresponding notions for path-decompositions.*

Observation 27. *Let D be the directed graph obtained from G by replacing every edge by two anti-parallel arcs and adding an arc uv for every $u, v \in V(G)$ such that $u \leq^V v$. Then, $ppw_{\leq^V}(G) = dpw(D)$.*

Proof. We will show an even stronger statement, namely that any prefix path-decomposition of G w.r.t. $\leq^V$ is also a directed path-decomposition of the graph D and vice versa.

Suppose that $\mathcal{P} := (P_1, \ldots, P_n)$ is a prefix path-decomposition of G w.r.t. $\leq^V$. Then, $\mathcal{P}$ satisfies Properties D1 and D2, because $\mathcal{P}$ satisfies Properties P1 and P2. Towards showing Property D3, let uv be any arc in D. If uv is also an edge of G, then

Property D3 follows because $\mathcal{P}$ satisfies Property P3. Otherwise, $u \leq^V v$ and because of the downward closure property of $\mathcal{P}$, we obtain that $u \in P_{\leq f_{\mathcal{P}}(v)}$. Hence, there is an i with $i \leq f_{\mathcal{P}}(v)$ and $u \in P_i$, which because $v \in P_{f_{\mathcal{P}}(v)}$ implies Property D3.

On the other hand, let $\mathcal{P}$ be a directed path-decomposition of D. Then, properties P1 and P2 follow immediately from properties D1 and D2 of $\mathcal{P}$. Towards showing Property P3, let $uv \in E(G)$ be an edge of G. Then, both uv and vu are arcs in D and it follows from Property D3 that there are i, j, i', and j' with $1 \leq i < j \leq n$ and $1 \leq i' \leq j' \leq n$ such that $u \in P_i$, $v \in P_j$, $v \in P_{i'}$, and $u \in P_{j'}$. Because of Property D2, we obtain that $u \in P_l$ for every l between i and j' and $v \in P_l$ for every l between i' and j. Hence, because $i < j$ and $i' < j'$ there exists an h with $1 \leq h \leq n$ such that $u, v \in P_h$, showing Property P3.

Towards showing the downward closure property for $\mathcal{P}$, recall that for every $u \in D_{\leq^V}(v)$ there exists a directed edge uv in D. It hence follows from Property D3 that either there exists i with $1 \leq i \leq n$ such that $u, v \in W_i$, or there are i and j with $1 \leq i < j \leq n$ such that $u \in W_i$ and $v \in W_j$. Therefore, $u \in P_{\leq f_{\mathcal{P}}(v)}$ for every $u \in D_{\leq^V}(v)$ and $\mathcal{P}$ has the downward closure property. □

Since it has been shown [39] that deciding whether the directed pathwidth of a digraph is at most k is solvable in polynomial-time for every fixed k, the above observation implies that the same holds for the prefix pathwidth (w.r.t. an arbitrary dependency poset). It is an important open question, however, whether computing directed pathwidth is fixed-parameter tractable.

Let G be a graph and $\leq$ a total ordering on $V(G)$. We say that G is a *minor* of a graph G' if G can be obtained from G' by a sequence of operations of any of the following kind: (1) vertex deletion, (2) edge deletion, or (3) edge-contraction, i.e. replacing the two endpoints of an edge e with a new vertex adjacent to the union of the neighbors of the endpoints of e. We call the pair $(G, \leq)$ an *ordered graph*. We say that an ordered graph $(G, \leq)$ is an *ordered minor* of an ordered graph $(G', \leq')$, if $(G, \leq)$ can be obtained from $(G', \leq')$ by a sequence of operations of any of the following kind: (1) vertex deletion, (2) edge deletion, or (3) ordered edge-contraction, i.e. an edge-contraction for which the resulting vertex can take the place of any of the endpoints of the contracted edge in the ordering $\leq'$. Let X be a set and $\leq$ a reflexive and transitive binary relation on X. Then, X is *well-quasi ordered* w.r.t. $\leq$ if every infinite sequence $x_1, x_2, \ldots$ of elements of X contains an increasing pair $x_i \leq x_j$ where $i < j$. If the set of ordered graphs of prefix-pathwidth at most k was well-quasi ordered w.r.t. the ordered minor relation, this would be an important step towards a FPT algorithm for determining whether a graph has prefix-pathwidth at most k [20]. However, we show that this is actually not the case.

Observation 28. *The set of ordered graphs is not well-quasi ordered w.r.t. the ordered minor relation.*

Proof. It has been shown that the set of all permutations of the natural numbers is not well-quasi ordered w.r.t. the removal of entries [36]. In particular, Bona and Spielman [36] constructed an infinite antichain of permutations for this setting. We will make use of this antichain to construct an infinite antichain of ordered graphs w.r.t. the ordered minor relation.

Let $\mathcal{P} = (p_1, p_2, \dots)$ be the infinite sequence of permutations (also called an antichain) witnessing that the set of all permutations is not well-quasi ordered w.r.t. to the removal operation as defined by Bona and Spielman [36]. We define a corresponding sequence $\mathcal{G} = (g_1, g_2, \dots)$ of ordered graphs as follows: for every permutation $p \in \mathcal{P}$, let $g(p)$ be the ordered graph $(G, \leq)$, where G is a path given by the sequence $(v_0, v_1, \dots, v_{|p|})$ of vertices (with endpoints v_0 and $v_{|p|}$) and the partial ordering $\leq$ is defined by $v_i \leq v_j$ if and only if $p^{-1}[i] \leq p^{-1}[j]$ for every $0 < i, j \leq n$. Note that because G is a path and the endpoint v_0 is the only vertex of G, which is not comparable (w.r.t. $\leq$) to any other vertex, it holds that $g(p) = g(p')$ if and only if $p = p'$ (hence g is a bijection). We set $\mathcal{G} := (g(p_1), g(p_2), \dots)$. It remains to show that $\mathcal{G}$ is an infinite antichain w.r.t. to the ordered minor relation, i.e. there is no pair i, j with $i < j$ such that g_i is an ordered minor of g_j.

Suppose for a contradiction that there is such a pair say i, j such that $i < j$ and $g_i := (G_i, \leq_i)$ is an ordered minor of $g_j := (G_j, \leq_j)$. Then, in particular, G_i is a minor of G_j and because both G_i and G_j are paths we can assume w.l.o.g. that g_i is obtained from g_j by a sequence of ordered edge-contractions. Moreover, because any ordered edge-contraction of an edge $e = \{u, v\} \in G_j$ for which the resulting vertex takes the place of say u in the ordering leads to an ordered graph isomorphic to $g(p'_j)$, where p'_j is the permutation obtained from p_j after removing the element corresponding to v in p_j, we obtain that any permutation p such that $g_i = g(p)$ is can be obtained from any permutation p' such that $g_j = g(p')$. Finally, because $g(p)$ is a bijection, we obtain that p_i can be obtained from p_j via the removal of elements, contradicting our assumption that the set of all permutations is well-quasi ordered under removal of elements. □

The above observations suggest that deciding whether computing prefix pathwidth (w.r.t. an arbitrary poset) is fixed-parameter tractable is a difficult question.

5.1. Algorithms for Posets of Bounded Width

In the following we will give two algorithms that compute the prefix path-decomposition of a graph that are efficient in the case that the given poset has small width. Our first algorithm shows that if the width of the poset $\leq^V$ is bounded by a constant, then deciding whether G has a prefix path-decomposition w.r.t. $\leq^V$ of width at most k is fixed-parameter tractable (in k). Recall that as before we assume that we are given a graph G together with a poset $(V(G), \leq^V)$ on the vertices of G of width w.

Theorem 29. *Finding a prefix path-decomposition of a graph G w.r.t. a partial order $\leq^V$ of width at most k or deciding that no such prefix path-decomposition exists can be done in time $O((|V(G)|^w k^{2k})^2 |V(G)|)$, where $w = \text{width}(V(G), \leq^V)$. Hence, for any constant w computing a prefix path-decomposition is fixed-parameter tractable in k.*

Whereas the above algorithm allows us to compute the prefix pathwidth exactly it has one serious drawback since it does not run in fpt-time w.r.t. the combined parameter prefix pathwidth k and the width w of the poset and hence is not sufficient to show an fpt-result for QBF parameterized these two parameters. Our second main result of this section circumvents this issue at the expense of obtaining only an approximate solution. Namely, our second algorithm allows us to compute a prefix path-decomposition

in polynomial-time whose relative difference from the optimal width can be bounded in terms of k and w. Using this approximate prefix path-decomposition as an input to the algorithm in Theorem 26, then shows that QBF is fixed-parameter tractable parameterized by k and w.

Theorem 30. *There is a polynomial-time algorithm that, given a graph G, a poset $(V(G), \leq^V)$ of width w, and an integer k, outputs a prefix path-decomposition of G w.r.t. $\leq^V$ of width at most $2w(2k^2 + k) + 1$ or outputs correctly that no prefix path-decomposition of G w.r.t. $\leq^V$ of width at most k exists.*

The remainder of this section is devoted to a proof of the above theorems. For a subset $V' \subseteq V$, let $P_{\leq^V}(V')$ be the *prefix* of V' w.r.t. $\leq^V$, i.e., the set of all vertices v in V' with $D_{\leq^V}(v) \subseteq V'$, and let $S_{\leq^V}(V')$ be the suffix of V' w.r.t. $\leq^V$, i.e., the set $V' \setminus P_{\leq^V}(V')$.

Let $B(V')$ be the bipartite graph with bipartition $(\partial(V'), N(\partial(V')) \setminus V')$ containing all edges of G with one endpoint in $\partial(V')$ and the other endpoint in $N(\partial(V')) \setminus V'$. Moreover, for a prefix path-decomposition $\mathcal{P} := (P_1, \ldots, P_n)$ of G w.r.t. $\leq^V$ and for every i with $1 \leq i \leq n$, let $B_{\mathcal{P}}(i)$ be the bipartite graph $B(P_{\leq^V}(P_{\leq i}))$.

Before we proceed to prove the required statements for the algorithm, let us first provide an informal overview of the algorithm. The main observation behind the algorithm (which is shown in Lemma 31 below) is that in any prefix path-decomposition of G w.r.t. $\leq^V$, the intersection between any two bags, say P_i and P_{i+1}, can be characterized by a pair (D, C), where D is a downward closed set of vertices of G equal to $P_{\leq^V}(P_{\leq i})$ and C is a minimal vertex cover of the bipartite graph between the guards of D and the neighbors of these guards in the remainder of G, i.e. the bipartite graph $B_{\mathcal{P}}(i)$. Given this crucial observation, it is then straightforward to define simple conditions for deciding whether a pair (D, C) can be the intersection of two bags in a prefix path-decomposition of width at most k as well as conditions for deciding whether the intersection of two bags corresponding to a pair (D, C) can be followed by (in some prefix path-decomposition of width at most k) the intersection of two bags corresponding to the pair (D', C') (these conditions are defined in Lemma 32). Computing a prefix path-decomposition then boils down to deciding whether there is a directed path from the pair $(\emptyset, \emptyset)$ to the pair $(V(G), \emptyset)$ in the digraph whose vertex set consists of all pairs (D, C) such that (D, C) can be the intersection between two bags in some prefix path decomposition of width at most k and whose arcs are defined using the above mentioned conditions (see the proof of Theorem 29). Because the number of downward closed sets is bounded by $|V(G)|^w$ and one can show (see Lemma 33) that the number of possible minimal vertex covers (for each downward closed set) is bounded by k^{2k}, this then leads to the required result.

We are now ready to prove the formal statements required by the algorithm as outlined above.

Lemma 31. *Let $\mathcal{P} := (P_1, \ldots, P_n)$ be a prefix path decomposition of G w.r.t. $\leq^V$ of width at most k, which is minimal in the sense that no bag of $\mathcal{P}$ contains unnecessary vertices. Then, for every i with $1 \leq i < n$, it holds that $P_i \cap P_{i+1}$ is a minimal vertex cover of $B_{\mathcal{P}}(i)$.*

Proof. We first show that $P_i \cap P_{i+1}$ contains a vertex cover of $B_{\mathcal{P}}(i)$ for every i with $1 \leq i < n$. Suppose not, then there is an i with $1 \leq i < n$ such that $P_i \cap P_{i+1}$ is not a vertex cover of $B_{\mathcal{P}}(i)$. Hence, there is an edge $\{v, u\} \in E(B_{\mathcal{P}}(i)) \subseteq E(G)$ with $v \in P_{\leq^V}(P_{\leq i})$ and $u \in N(\partial(P_{\leq^V}(P_{\leq i}))) \setminus (P_{\leq^V}(P_{\leq i}))$ such that $v, u \notin P_i \cap P_{i+1}$. Hence, either $u \notin P_{\leq i}$ or $u \in S_{\leq^V}(P_{\leq i})$. In the former case, because $v \in P_{\leq i}$ but $v \notin P_{i+1}$, we obtain that property P3 of $\mathcal{P}$ is violated. In the latter case, because $S_{\leq^V}(P_{\leq i}) \subseteq P_i$, we obtain that $u \notin P_{i+1}$ and hence $f_{\mathcal{P}}(u) = i$. Consequently, $D_{\leq^V}(u) \not\subseteq P_{\leq i} = P_{\leq f(u)}$, contradicting the downward closure property of $\mathcal{P}$.

Towards showing the minimality of the vertex cover, assume for a contradiction that this is not the case and there is an i with $1 \leq i < n$ such that $P_i \cap P_{i+1}$ is not a minimal vertex cover of $B_{\mathcal{P}}(i)$. Then, there is a vertex $v \in P_i \cap P_{i+1}$ such that $(P_i \cap P_{i+1}) \setminus \{v\}$ is still a vertex cover of $B_{\mathcal{P}}(i)$. Then, either $v \in P_{\leq^V}(P_{\leq i})$ or $v \in S_{\leq^V}(P_{\leq i})$.

In the first case, let $\mathcal{P}' := (P'_1, \ldots, P'_n)$ be obtained from $\mathcal{P}$ after removing v from every bag P_j with $j \geq i + 1$. We show that $\mathcal{P}'$ is still a prefix path-decomposition, contradicting our assumption that no bag of $\mathcal{P}$ contained unnecessary vertices. It is easy to verify that conditions (P1) and (P2) as well as the downward closure property still hold for $\mathcal{P}'$. Towards showing (P3), let $\{v, u\} \in E(G)$. Because $(P_i \cap P_{i+1}) \setminus \{v\}$ is a vertex cover of $B_{\mathcal{P}}(i)$, we obtain that if $\{v, u\} \in B_{\mathcal{P}}(i)$, then $u \in P'_i$, as required. If $\{v, u\} \notin B_{\mathcal{P}}(i)$, then $u \in P_{\leq^V}(P_{\leq i})$ and because $\mathcal{P}$ satisfies condition (P3), it follows that $\{u, v\}$ is covered by some bag P_j with $j \leq i$, as required.

In the second case, let $\mathcal{P}' := (P'_1, \ldots, P'_n)$ be obtained from $\mathcal{P}$ after removing v from every bag P_j with $j \leq i$. We show that $\mathcal{P}'$ is still a prefix path-decomposition, contradicting our assumption that no bag of $\mathcal{P}$ contained unnecessary vertices. It is easy to verify that conditions (P1) and (P2) as well as the downward closure property still hold for $\mathcal{P}'$. Towards showing (P3), let $\{v, u\} \in E(G)$. Because $(P_i \cap P_{i+1}) \setminus \{v\}$ is a vertex cover of $B_{\mathcal{P}}(i)$, we obtain that if $\{v, u\} \in B_{\mathcal{P}}(i)$, then $u \in P'_{i+1}$, as required. If $\{v, u\} \notin B_{\mathcal{P}}(i)$, then either $u \in S_{\leq^V}(P_{\leq i})$ in which case because of the downward closure property of $\mathcal{P}$ also $u \in P_{i+1}$ and hence $u \in P'_{i+1}$, as required, or $u \in V(G) \setminus P_{\leq i}$ and because $\mathcal{P}$ satisfies condition (P3), we obtain that $\{v, u\}$ is covered by some bag P_j with $j \geq i + 1$, as required. □

Lemma 32. *There is a prefix path-decomposition of G w.r.t. $\leq^V$ of width at most k if and only if there is a sequence $\mathcal{S} := ((D_0, C_0), \ldots, (D_n, C_n))$ of pairs (D_i, C_i) such that:*

C1 $D_0 = C_0 = \emptyset$, $D_n = V(G)$, $C_n = \emptyset$,

C2 for every i with $0 \leq i \leq n$:

- *C2A* D_i *is downward closed,*
- *C2B* C_i *is a minimal vertex cover of* $B(D_i)$,

C3 for every i with $0 \leq i < n$:

- *C3A* $|(D_{i+1} \setminus D_i) \cup C_i \cup C_{i+1}| \leq k$,
- *C3B* $C_i \setminus C_{i+1} \subseteq D_{i+1}$,

C3C $D_i \subseteq D_{i+1}$.

Proof. For the forward direction, suppose there is a prefix path-decomposition $\mathcal{P} := (P_1, \ldots, P_n)$ of G w.r.t. $\leq^V$ of width at most k. W.l.o.g. we can assume that $\mathcal{P}$ is minimal (in the sense that no bag contains unnecessary vertices). It hence follows from Lemma 31 that $P_i \cap P_{i+1}$ is a minimal vertex cover of $B_{\mathcal{P}}(i)$ for every i with $1 \leq i < n$.

We claim that $\mathcal{S} := ((D_0, C_0), (D_1, C_1), \ldots, (D_n, C_n))$ with $D_0 := \emptyset$, $C_0 := \emptyset$, $D_i := P_{\leq^V}(P_{\leq i})$, $C_i := P_i \cap P_{i+1}$ for every i with $1 \leq i < n$, $D_n = V(G)$, and $C_n = \emptyset$ is the required sequence. Conditions C1 and C2A are trivially satisfied. Condition C2B follows from Lemma 31 applied to $\mathcal{P}$. Towards condition C3A observe that

$$\begin{aligned}
& |(D_{i+1} \setminus D_i) \cup C_i \cup C_{i+1}| \\
= \; & |((P_{\leq^V}(P_{\leq i+1})) \setminus (P_{\leq^V}(P_{\leq i}))) \cup \\
& (P_i \cap P_{i+1}) \cup (P_{i+1} \cap P_{i+2})| \\
\leq \; & |((P_{\leq^V}(P_{\leq i+1})) \setminus (P_{\leq^V}(P_{\leq i}))) \cup P_{i+1}| \\
\leq \; & |((P_{\leq i+1}) \setminus (P_{\leq^V}(P_{\leq i}))) \cup P_{i+1}| \\
\leq \; & |P_{i+1} \cup P_{i+1}| \\
\leq \; & k
\end{aligned}$$

The second to last inequality follows because $P_{\leq i+1} \setminus P_{\leq^V}(P_{\leq i}) \subseteq P_{i+1}$, which in turn can be seen as follows. Let $v \in P_{\leq i+1} \setminus P_{\leq^V}(P_{\leq i})$, then either $v \in P_{\leq i+1} \setminus P_{\leq i}$ or $v \in S_{\leq^V}(P_{\leq i})$. In the former case $v \in P_{i+1}$ (due to the definition of $P_{\leq i+1}$) and in the later case $v \in P_{i+1}$ because of the downward closure property of $\mathcal{P}$.

Towards showing C3B, it suffices to show that $v \in C_i \setminus C_{i+1}$ implies $v \in D_{i+1}$ for every $v \in V(G)$. Because $C_i \setminus C_{i+1} = (P_i \cap P_{i+1}) \setminus (P_{i+1} \cap P_{i+2}) = (P_i \cap P_{i+1}) \setminus P_{i+2}$, we obtain that $v \in C_i \setminus C_{i+1}$ implies that $f_{\mathcal{P}}(v) = i + 1$. Hence, using the fact that $\mathcal{P}$ satisfies the downward closure property, we obtain that $D_{\leq^V}(v) \subseteq D_{i+1}$. In particular, $v \in D_{i+1}$, as required.

Condition C3C, i.e., $D_i \subseteq D_{i+1}$ or equivalently $P_{\leq^V}(P_{\leq i}) \subseteq P_{\leq^V}(P_{\leq i+1})$ is satisfied because $P_{\leq i} \subseteq P_{\leq i+1}$ and the downward closed subset of a set contains the downward closed subset of any subset of the set.

For the reverse direction suppose that we are given a sequence $\mathcal{S} := ((D_0, C_0), \ldots, (D_n, C_n))$ satisfying the conditions given in the statement of the lemma. We claim that $\mathcal{P} := (P_1, \ldots, P_n)$ with $P_i := D_i \setminus D_{i-1} \cup C_{i-1} \cup C_i$ for every i with $1 \leq i \leq n$ is a prefix path-decomposition of G w.r.t. $\leq^V$ of width at most k. Because of condition C1, it holds that $P_{\leq i} = \bigcup_{1 \leq j \leq i} D_j \setminus D_{j-1} \cup C_{j-1} \cup C_j = \bigcup_{1 \leq j \leq i} D_j \cup C_j$ and again using condition C1, we obtain $P_{\leq n} = V(G)$. It follows that $\mathcal{P}$ satisfies property P1.

To show the remaining properties for $\mathcal{P}$, we need the following claim.

Claim 3. *Let $v \in V(G)$ be any vertex such that $v \in P_i \setminus P_{i+1}$, then $v \in D_i$, $v \notin C_i$, and all neighbors of v outside of D_i are in C_i.*

Proof of Claim. Let $v \in V(G)$ be any vertex such that $v \in P_i \setminus P_{i+1} = (D_i \setminus D_{i-1} \cup C_{i-1} \cup C_i) \setminus (D_{i+1} \setminus D_i \cup C_i \cup C_{i+1})$ for some i with $1 \leq i < n$, i.e., v is forgotten at position i. There are two cases for this to happen, i.e., either $v \in D_i \setminus D_{i-1}$ or $v \in C_{i-1}$. Note that in both cases $v \notin C_i$. In the latter case, we obtain from C3B that $v \in D_i$. Hence, we have already shown that $v \in D_i$ and $v \notin C_i$. Moreover, because $v \notin C_i$ we obtain from C2B that C_i contains all the neighbors of v outside of D_i. This completes the proof of the above claim. ■

To show property P2 for $\mathcal{P}$, it suffices to show that for any $v \in V(G)$ with $v \in P_i \setminus P_{i+1}$ for some position i with $1 \leq i < n$, it holds that $v \notin P_j = D_j \setminus D_{j-1} \cup C_{j-1} \cup C_j$ for any $j > i+1$. From Claim 3, we obtain that $v \in D_i$, $v \notin C_i$, and all the neighbors of v outside of D_i are in C_i. Because $v \in D_i$, we obtain from C3C that $v \notin D_j \setminus D_{j-1}$. Consequently, it is sufficient to show that $v \notin C_j$ for any $j > i+1$. Because C_i contains all the neighbors of v outside of D_i, we obtain from C3B that for every $j > i+1$ it holds that C_j contains all the neighbors of v outside of D_j. Using C2B (in particular the minimality of C_j), we obtain that C_j does not contain v, as required.

We will show property P3 for $\mathcal{P}$ by showing (by induction on the index of the forgotten vertices) that whenever a vertex is forgotten at position i, then every edge incident to it is contained in some bag $j \leq i$. This clearly holds if v is the first vertex that is forgotten, because all neighbors of v in D_i must still be in P_i and all neighbors of v outside of D_i are in $C_i \subseteq P_i$ (because of Claim 3). So assume that v is the l-th vertex that is forgotten and v is forgotten at some position i. Again, all neighbors of v in D_i are either still in P_i or have been forgotten before v and hence (by the induction hypotheses) every edge incident to the forgotten neighbors is contained in some bag P_j with $j \leq i$. Moreover, because of Claim 3 all the neighbors of v outside of D_i are in C_i and hence also in P_i, as required.

We show next that $\mathcal{P}$ has the downward closure property. Again, because of Claim 3, if a vertex v is forgotten at some position i, then $v \in D_i$. The downward closure property for $\mathcal{P}$, then follows from C2A.

Because of C3A, we have that the width of $\mathcal{P}$ is at most k. Hence, $\mathcal{P}$ is a prefix path decomposition of G w.r.t. $\leq^V$ of width at most k, as required. □

Lemma 33. *Let $D \subseteq V(G)$ be a downward closed set w.r.t. $\leq^V$, then there are at most $O(k^{2k})$ minimal vertex covers of $B(D)$ of size at most k and these can be enumerated in time $O(k^{2k}|B(D)|)$*

Proof. Let C_0 be all vertices in $B(D)$ whose degree is greater then k. Note that C_0 has to be contained in any vertex cover of size at most k. It follows that $B(D) \setminus C_0$ contains at most $k(k-|C_0|)$ edges because every vertex in $B(D) \setminus C_0$ can cover at most k edges (since it has degree at most k) and one needs to cover all edges in $B(D) \setminus C_0$ with at most $k - |C_0|$ vertices. It follows that $B(D) \setminus C_0$ contains at most $2k(k-|C_0|)$ non-isolated vertices. Furthermore, the isolated vertices of $B(D)$ do not occur in any minimal vertex cover. Hence, there are at most $\binom{2k(k-|C_0|)}{k-|C_0|} \in O(k^{2k})$ vertex covers of $B(D)$ and for each of those one can check in $O(k(2k(k-|C_0|)))$ time, whether it is a minimal vertex cover. □

In the following let $w = \text{width}((V(G), \leq^V)$ and let $(W_1, \ldots, W_w)$ be a chain partition of $(V(G), \leq^V)$ (which due to Proposition 7 can be computed in polynomial-time). For a vector $v \in [|W_1|] \times \ldots \times [|W_w|]$, let $D_{\leq^V}(v)$ be the set of vertices of $V(G)$ that contains the first $v[i]$ vertices from every chain W_i. Note that for every downward closed set V' there is a vector $v \in [|W_1|] \times \ldots \times [|W_w|]$ such that $V' = D_{\leq^V}(v)$, which we denote by $v_{\leq^V}(V')$.

Let $\mathcal{P} := (P_1, \ldots, P_n)$ be a prefix path decomposition of G w.r.t. $\leq^V$ of width at most k. Note that because $P_{\leq^V}(P_{\leq i})$ is downward closed the vector $v(P_{\leq^V}(P_{\leq i}))$ is well-defined. We are now ready to show our two main results of this section.

Proof of Theorem 29. To decide whether G has a prefix path-decomposition w.r.t. $\leq^V$ of width at most k, we first build an auxiliary directed graph H as follows.

The vertex set of H consists of all pairs (D, C) such that $D \subseteq V(G)$ is a downward closed set and C is a minimal vertex cover of $B(D)$ of size at most k. Furthermore, there is an arc from (D, C) to (D', C') of H if and only if (D, C) and (D', C') satisfy the conditions C3A-C3C given in Lemma 32 (with $D_i = D$, $C_i = C$, $D_{i+1} = D'$, and $C_{i+1} = C'$). This completes the construction of H. Because of Lemma 32, we obtain that G has a prefix path decomposition w.r.t. $\leq^V$ of width at most k if and only if there is a directed path in H from $(\emptyset, \emptyset)$ to $(V(G), \emptyset)$. Hence, given H we can decide whether G has a prefix path-decomposition w.r.t. $\leq^V$ of width at most k (and output such a path-decomposition if it exists) in time $O(|V(H)| \log(|V(H)|) + E(H))$ (e.g., by using Dijkstra's algorithm). It remains to analyze the time required to construct H (as well as its size).

Because every downward closed set D can be characterized by a vector $v \in [|W_1|] \times \ldots \times [|W_w|]$ (namely by the vector $v(D)$), there are at most $|V(G)|^w$ such sets D. Moreover, due to Lemma 33 for each such D there are at most $O(k^{2k})$ minimal vertex covers of $B(D)$ of size at most k. Hence, H has at most $O(|V(G)|^w k^{2k})$ vertices and again using Lemma 33, the vertex set of H can be constructed in time $O(|V(G)|^{w+1} k^{2k})$. To compute the arc set of H, we go over all of the at most $O((|V(G)|^w k^{2k})^2)$ pairs of vertices of H and check the conditions C3A–C3C, which can be done in time $O(|V(G)|)$ (for each of these pairs). Hence, the total time required to construct H is $O((|V(G)|^w k^{2k})^2 |V(G)|)$ and since this dominates the time required to find a shortest path in H, this is also the total running time of the algorithm. □

Proof of Theorem 30. For a vector $v \in [|W_1|] \times \ldots \times [|W_w|]$ and i with $1 \leq i \leq w$, let $A(v, i)$ be the set of vertices $\partial(D_{\leq^V}(v)) \cap W_i$, let $B(v, i)$ be the set of vertices $N(\partial(D_{\leq^V}(v)) \cap W_i) \setminus D_{\leq^V}(v))$, and let $G(v, i)$ be the bipartite graph with bipartition $(A(v, i), B(v, i))$ containing all edges of G between $A(v, i)$ and $B(v, i)$. Let $H(G(v, i))$ be the set of vertices of $G(v, i)$ with degree larger than k and let $C(G(v, i))$ consist of $H(G(v, i))$ and all vertices in $A(v, i) \setminus H(G(v, i))$ with at least one neighbor in $G(v, i) \setminus H(G(v, i))$. We also set $P(v)$ to be the set of vertices $\bigcup_{1 \leq i \leq w} C(G(v, i))$.

We are now ready to describe the algorithm that outputs a prefix path-decomposition of G w.r.t. to $\leq^V$ of width at most $2w(2k^2 + k) + 1$ or outputs "No" if no prefix path-decomposition of G w.r.t. $\leq^V$ of at width at most k exists. The algorithms tries to iteratively construct a sequence $(v_1, \ldots, v_n)$ of vectors $v_i \in [|W_1|] \times \ldots \times [|W_w|]$ such that v_1 is the all zero vector, v_n is the vector

$(|W_1|, \ldots, |W_w|)$, v_{i+1} is obtained from v_i by increasing exactly one component of v_i by one, and $|C(G(v_i, j))| \leq 2k^2 + k$ for every i and j with $1 \leq i \leq n$ and j with $1 \leq j \leq w$. The algorithm tries to find the sequence in a greedy manner, i.e., after having constructed the sequence $(v_1, \ldots, v_i)$ it first checks whether v_i is the vector $(|W_1|, \ldots, |W_w|)$ (in which case the algorithm has succeeded). Otherwise it goes over all j with $1 \leq j \leq w$ and checks whether the vector v' obtained from v_i by increasing the j-th entry by one satisfies $v'[j] \leq |W_j|$ and $|C(G(v', j))| \leq 2k^2 + k$. If so it continues on the sequence $(v_1, \ldots, v_i, v')$. Otherwise, it outputs that G does not have a prefix path-decomposition w.r.t. $\leq^V$ of width at most k.

If the algorithm succeeds, it outputs $\mathcal{P} := (P_1, \ldots, P_n)$ with $P_1 = \emptyset$ and $P_i := P(v_i) \cup P(v_{i-1}) \cup (D_{\leq^V}(v_i) \setminus D_{\leq^V}(v_{i-1}))$ for every i with $1 < i \leq n$ as the prefix path-decomposition of G w.r.t. $\leq^V$. Otherwise, it returns that G does not have a prefix path-decomposition w.r.t. of width at most k. This completes the description of the algorithm.

It is straightforward to check that the algorithm runs in polynomial-time. It remains to show correctness of the algorithm.

We start by showing that if the algorithm succeeds and outputs the sequence $\mathcal{P} := (P_1, \ldots, P_n)$, then $\mathcal{P}$ is a prefix path-decomposition of G w.r.t. $\leq^V$ of width at most $2w(2k^2 + k) + 1$. Let $(v_1, \ldots, v_n)$ be the sequence of vectors computed by the algorithm. Recall that v_1 is the all-zero vector, v_n is the vector $(|W_1|, \ldots, |W_n|)$, v_{i+1} is obtained from v_i by increasing exactly one entry by one, and $|C(G(v_i, j)| \leq 2k^2 + k$ for every i and j with $1 \leq i \leq n$ and j with $1 \leq j \leq w$. Because for every i with $1 < i \leq n$, P_i contains (at least) the vertex $D_{\leq^V}(v_i) \setminus D_{\leq^V}(v_{i-1})$, it holds that $\bigcup_{1 \leq i \leq n} P_i = V(G)$, which shows property P1 for $\mathcal{P}$.

Claim 4. *Let $v \in V(G)$ be any vertex such that $v \in P_i \setminus P_{i+1}$, then $v \in D_{\leq^V}(v_i)$, and all neighbors of v outside of $D_{\leq^V}(v_i)$ are in P_i and have degree more than k in $G(v_i, j)$ for every $1 \leq j \leq w$.*

Proof of Claim. We first show that $v \in D_{\leq^V}(v_i)$. Assume for a contradiction that $v \notin D_{\leq^V}(v_i)$. Then, because $v \in P_i$, it holds that $v \in P(v_i) \cup P(v_{i-1})$. If $v \in P(v_i)$, then $v \in P_{i+1}$ contradicting our assumption. Hence, $v \in P(v_{i-1})$. In particular, there is some j with $1 \leq j \leq w$ such that $v \in B(v_{i-1}, j)$ and either v has degree larger than k in $G(v_{i-1}, j)$. Note that because $v \notin P_{i+1}$, we obtain that $v \notin D_{\leq^V}(v_{i+1}) \setminus D_{\leq^V}(v_i)$ and hence also $v \notin D_{\leq^V}(v_{i+1})$. It follows that $v \in B(v_{i+1}, j)$. Furthermore, if v had degree more than k in $G(v_{i-1}, j)$, then because $v_{i-1} \leq v_{i+1}$, v still has degree more than k in $G(v_{i+1}, j)$ and hence $v \in C(v_{i+1}, j) \subseteq P_{i+1}$ contradicting our assumption. This shows that $v \in D_{\leq^V}(v_i)$.

We show next that all neighbors of v outside of $D_{\leq^V}(v_i)$ are in P_i. Because $v \notin P_{i+1}$, we obtain that $v \notin P(v_i)$. Consequently, for every j with $1 \leq j \leq w$, all neighbors of v in $G(v_i, j)$ have degree more than k. Hence, all neighbors of v outside of $D_{\leq^V}(v_i)$ are in $C(v_i, j) \subseteq P_i$. This completes the proof of the claim. ■

To show property P2 for $\mathcal{P}$, it suffices to show that for any $v \in V(G)$ with $v \in P_i \setminus P_{i+1}$ for some position i with $1 \leq i < n$, it holds that $v \notin P_j = D_{\leq^V}(v_j) \setminus D_{\leq^V}(v_{j-1}) \cup P(v_{j-1}) \cup P(v_j)$ for any $j > i + 1$. From Claim 4, we obtain that $v \in D_{\leq^V}(v_i)$, and all the neighbors of v outside of $D_{\leq^V}(v_i)$ have degree more than k

in $G(v_i, l)$ for every l with $1 \leq l \leq w$. Because $v \in D_{\leq^V}(v_i)$ and $v_i \leq v_j$, we obtain that $v \notin D_{\leq^V}(v_j) \setminus D_{\leq^V}(v_{j-1})$. Consequently, it is sufficient to show that $v \notin P(v_j)$ for any $j > i+1$. Because all the neighbors of v outside of $D_{\leq^V}(v_i)$ have degree more than k in $G(v_i, l)$ for every l with $1 \leq l \leq w$ and $v_i \leq v_j$, we obtain that for every $j > i+1$ and every l with $1 \leq l \leq w$, it holds that all the neighbors of v outside of $D_{\leq^V}(v_j)$ have degree more than k in $G(v_j, l)$. It follows that $v \notin P(v_j)$, as required.

We will show property P3 for $\mathcal{P}$ by showing (by induction on the index of the forgotten vertices) that whenever a vertex is forgotten at position i, then every edge incident to it is contained in some bag $j \leq i$. This clearly holds if v is the first vertex that is forgotten, because all neighbors of v in $D_{\leq^V}(v_i)$ must still be in P_i and all neighbors of v outside of $D_{\leq^V}(v_i)$ are in P_i by Claim 4. So assume that v is the l-th vertex that is forgotten and v is forgotten at some position i. Again, all neighbors of v in $D_{\leq^V}(v_i)$ are either still in P_i or there have been forgotten before v and hence (by the induction hypotheses) the edges incident to the forgotten neighbors are contained in some bag P_j with $j \leq i$. Moreover, because of Claim 4 all the neighbors of v outside of $D_{\leq^V}(v_i)$ are in P_i, as required.

The downward closure property of $\mathcal{P}$ follows immediately from Claim 3 and the fact that $D_{\leq^V}(v_i)$ is downward closed for every i with $1 \leq i \leq n$.

By construction the width of $\mathcal{P}$ is at most $2w(2k^2+k)+1$. This shows that $\mathcal{P}$ is a prefix path-decomposition of G w.r.t. $\leq^V$ of width at most $2w(2k^2+k)+1$.

We show next that if the algorithm is not successful, then G does not have a prefix path-decomposition w.r.t. $\leq^V$ of width at most k. Because the algorithm is not successful, there is a vector $v \in [|W_1|] \times \ldots \times [|W_w|]$ with $v \neq (|W_1|, \ldots, |W_w|)$ such that for every j with $1 \leq j \leq w$ either $v'[j] > |W_j|$ or $|C(G(v', j))| > 2k^2 + k$, where v' is the vector obtained from v by increasing the j-th entry of v by one. Assume for a contradiction that there is a prefix path-decomposition $\mathcal{P} := (P_1, \ldots, P_n)$ of G w.r.t. $\leq^V$ of width at most k. W.l.o.g. we will assume that $|P_i \triangle P_{i+1}| \leq 1$. Let i be the smallest integer with $1 \leq i \leq n$ such that there is a l with $1 \leq l \leq w$ with $v_P[l] > v[l]$, where $v_P := v(P_{\leq^V}(P_{\leq i}))$. Observe that because $|P_i \triangle P_{i+1}| \leq 1$, the integer l is unique and moreover $v_P[l] \leq v[l] + k$. Hence, $v_P[l'] \leq v[l']$ for any $l' \neq l$ and $1 \leq l' \leq w$ and $v[l] < v_P[l] \leq v[l] + k$. It follows that there is a set D of at most k vertices of $G(v, l)$ in $B(v, l)$ such that $G(v, l) \setminus D$ is an induced subgraph of $G(v_P, l)$. Because every vertex in D can reduce the size of $C(v, l)$ by at most k, i.e., if its degree in $G(v, l)$ is exactly k, we obtain that $|C(v_P, l)| \geq |C(v, l)| - k^2 > 2k^2 + k - k^2 = k^2 + k$.

Because of Lemma 31, P_i has to contain a vertex cover of $B_{\mathcal{P}}(i)$. In particular, because $G(v_P, l)$ is a subgraph of $B_{\mathcal{P}}(i)$, P_i has to contain a vertex cover of $G(v_P, l)$. It remains to show that because $|C(v_P, l)| > k^2 + k$, $G(v_P, l)$ cannot have a vertex cover of size at most k. Assume for a contradiction that $G(v_P, l)$ has a vertex cover C of size at most k. Clearly, $H(v_P, l) \subseteq C$. Furthermore, because every vertex in $G(v_P, l) \setminus H(v_P, l)$ has degree at most k, any such vertex can cover the edges of at most k non-isolated vertices in $A(v_P, l) \cap C(v_P, l)$. It follows that $|C(v_P, l)| \leq k + k^2$ a contradiction. □

5.2. FPT for Trivial Dependency Scheme

We say that a dependency poset $\leq^V$ is *layered* if there is a partition of the elements into *layers* $V_1, \ldots, V_\ell$ such that for every $x_i \in V_i, x_j \in V_j$ we have $x_i < x_j$ if and

only if $i < j$. Informally, such posets are divided into a chain of layers of incomparable elements. We note that every trivial dependency poset is a layered poset; however, other dependency posets may in some cases also be layered.

The remainder of this section is devoted to a proof of the following theorem.

Theorem 34. *Let $\leq^V$ be a layered dependency poset and k a natural number. There exists an FPT algorithm (parameterized by k) that either finds a prefix path-decomposition of G w.r.t. $\leq^V$ of width at most k or determines that no such path-decomposition exists.*

Our general proof strategy will be to first show that any prefix path-decomposition w.r.t. a layered dependency poset has a simple structure, which then can be exploited for an fpt-algorithm that finds such a decomposition of small width. In particular, we will show that any prefix path-decomposition can be naturally divided into path-decompositions of the parts $(V_1, \ldots, V_l)$ given by a layered dependency poset that are held together via (minimal) separators between these parts.

In the following let G be a graph, $\leq^V$ be a layered dependency poset, let $(V_1, \ldots, V_l)$ be the ordered partition associated with $\leq^V$, and let $\mathcal{P} = (P_1, \ldots, P_m)$ be a prefix path-decomposition of G w.r.t. $\leq^V$. Moreover, for every layer i with $1 \leq i \leq l$, we denote by $V_{\leq i}$ the set $\bigcup_{1 \leq j \leq i} V_j$ and similarly by $V_{>i}$ the set $V_{>i} = \bigcup_{i < j \leq l} V_j$. We will be interested in (minimal) separators between $V_{\leq i}$ and $V_{>i}$ in G. In particular, we will show that any such separator corresponds to a vertex cover in the bipartite graph $B(i)$ defined as follows.

- $V(B(i))$ contains the vertices of G that either lie in $V_{\leq i}$ and have at least one neighbor in $V_{>i}$, or are in $V_{>i}$ and have at least one neighbor in $V_{\leq i}$.
- $\{a, b\} \in E(B(i))$ if and only if $\{a, b\} \in E(G)$ and $a \in V_{\leq i}$ and $b \in V_{>i}$.

Proposition 35. *For every i with $1 \leq i \leq l$, it holds that a set C of vertices of G is a separator between $V_{\leq i}$ and $V_{>i}$ in G if and only if C is a vertex cover of $B(i)$.*

Proof. Towards showing the forward direction, let C be a separator between $V_{\leq i}$ and $V_{>i}$ in G. Then C is also a vertex cover of $B(i)$ since otherwise there would be an edge e between $V_{\leq i}$ and $V_{>i}$ in G with $C \cap e = \emptyset$, contradicting our assumption that C is a separator .

Towards showing the reverse direction, let C be a vertex cover of $B(i)$. Then C is also a separator between $V_{\leq i}$ and $V_{>i}$ in G since otherwise there would be an edge e between $V_{\leq i}$ and $V_{>i}$ in $B(i)$ with $C \cap e = \emptyset$, contradicting our assumption that C is a vertex cover of $B(i)$. □

In the following we will show that for every i with $1 \leq i \leq l$, $\mathcal{P}$ contains at least one bag that contains all vertices of a minimal separator between $V_{\leq i}$ and $V_{>i}$ in G. Namely, we will show that this holds for the first bag of $\mathcal{P}$ which precedes a bag that forgets a vertex from $V_{>i}$. We will denote this bag by $p(i)$; formally, we set $p(i) = \min\{ f_{\mathcal{P}}(v) \mid v \in V_{>i} \}$ where "min" returns the left-most bag out of the set.

Lemma 36. *For every i with $1 \leq i \leq l$, it holds that the bag $p(i)$ of $\mathcal{P}$ contains a minimal separator between $V_{\leq i} = \bigcup_{1 \leq j \leq i} V_j$ and $V_{>i} = \bigcup_{i < j \leq l} V_j$ in G.*

Proof. Because of Proposition 35 it is sufficient to show that the bag $p(i)$ contains a vertex cover of the bipartite graph $B(i)$. Towards showing this consider an edge $\{a, b\} \in E(B(i))$ with $a \in V_{\leq i}$ and $b \in V_{>i}$. It follows from the choice of $p(i)$ that:

O1 Every vertex in $V(B(i)) \cap V_{>i}$ that has been introduced in some bag of $\mathcal{P}$ to the left of $p(i)$ or in $p(i)$ is also in the bag $p(i)$,

O2 Every vertex in $V(B(i)) \cap V_{\leq i}$ has been introduced in some bag of $\mathcal{P}$ to the left of $p(i)$ or in $p(i)$.

If b is contained in the bag $p(i)$ then there is nothing to show. So suppose that b is not in the bag $p(i)$. It follows from Observation O1 that b has not yet been introduced. Hence, because of Property P3 of a path-decomposition, we obtain that the vertex a needs to occur in some bag to the right of $p(i)$ and it now follows from Observation O2 together with Property P2 of a path-decomposition that a is contained in the bag $p(i)$. □

Using Lemma 36, we are ready to show that there always exists an optimal prefix path-decomposition of a graph G w.r.t. a dependency poset that is layered, which contains a minimal separator between $V_{\leq i}$ and $V_{>i}$ (for each $i \leq l$) in G as a bag.

Lemma 37. *There exists an optimal prefix path-decomposition $\mathcal{Q}$ of G w.r.t. $\leq^V$, such that $\mathcal{Q} = (P_1, \ldots, P_{n_1}, \ldots, P_{n_2}, \ldots, P_{n_i}, \ldots, P_{n_l})$, where for every i:*

Q1 P_{n_i} is a minimal separator between $V_{\leq i}$ and $V_{>i}$;

Q2 $(P_{n_{i-1}}, \ldots, P_{n_i})$ is a path-decomposition of $V_i \cup P_{n_{i-1}} \cup P_{n_i}$;

Q3 every vertex in $V_{>i}$ is forgotten after the bag P_{n_i}, and $V_{\leq i} \subseteq P_{\leq n_i}$.

Proof. We start with a given optimal prefix path-decomposition $\mathcal{P}_0 = \mathcal{P}$ and we transform it inductively to prefix path-decompositions $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_l$ such that $\mathcal{P}_j$ satisfies properties Q1, Q2 and Q3 up to index j. Clearly, $\mathcal{P}_0$ satisfies the conditions Q1, Q2 and Q3 up to index 0. As our inductive hypothesis, we assume we are given a prefix path-decomposition $\mathcal{P}_{i-1} = (P_1, \ldots, P_{n_1}, \ldots, P_{n_2}, \ldots, P_{n_{i-1}}, \ldots P_r)$ which satisfies the conditions Q1, Q2 and Q3 up to $i - 1$.

We proceed by showing how to construct $\mathcal{P}_i$ from $\mathcal{P}_{i-1}$. Let $P_{i'}$ be the bag $p(i)$ (i.e., the first bag of $\mathcal{P}_{i-1}$ that precedes the bag forgetting a vertex from $V_{>i}$). Since Q3 holds for $i - 1$, it is easy to see that $i' \geq n_{i-1}$.

By Lemma 36 $P_{i'}$ contains a minimal separator X_i between $V_{\leq i}$ and $V_{>i}$. We set $X_{\leq i} = P_{i'} \cap (V_{\leq i} \setminus X_i)$, and $X_{>i} = P_{i'} \cap (V_{>i} \setminus X_i)$. Next, we construct $\mathcal{P}_i = (P'_1, \ldots, P'_{r+2})$ such that

- $P'_j = P_j$ for $j \leq n_{i-1}$,
- $P'_j = P_j \setminus X_{>i}$ for $n_{i-1} < j \leq i'$,
- $P'_{i'+1} = X_i$, and
- $P'_{j+2} = P_j \setminus X_{\leq i}$ for $j \geq i'$.

Moreover, we set $n_i = i' + 1$, hence Q1 holds for i. Since we do not change the path decomposition up to $P_{n_{i-1}}$, it is easy to see that Q1, Q2, Q3 still hold for all $j < i$. From the choice of $P_{i'}$ it follows that all vertices of $V_{>i}$ are forgotten after the bag $P'_{i'+1}$ and all vertices of $V_{\leq i}$ are introduced before the bag $P'_{i'+1}$ (because of the downward closure property). Hence, Q3 holds for i as well. Now we show that Q2 holds. Since Q3 holds for both $i-1$ and i, it follows that $P'_{n_{i-1}} \cup P'_{n_{i-1}+1} \cup \cdots \cup P'_{n_i} = V_i \cup P'_{n_{i-1}} \cup P'_{n_i}$. From the construction of $\mathcal{P}_i$, it follows that for every vertex x in $V_i \cup P'_{n_{i-1}} \cup P'_{n_i}$ it holds that if $x \in P_j$ for $n_{i-1} \leq j \leq i'$, then $x \in P'_j$. Hence, the bags containing x form an interval. We are left to show that every edge in $V_i \cup P'_{n_{i-1}} \cup P'_{n_i}$ is contained in some bag P_j for $n_{i-1} \leq j \leq i'$, hence it is also in the bag P'_j. Let e be an arbitrary edge with both endpoints in $V_i \cup P'_{n_{i-1}} \cup P'_{n_i}$. Since $\mathcal{P}_{i-1}$ is a path-decomposition, there is a bag, say P_{j_e}, that contain both endpoints of e. However note that all vertices of $V_i \cup P'_{n_{i-1}} \cup P'_{n_i}$ are forgotten in $\mathcal{P}_{i-1}$ after the bag $P_{n_{i-1}}$ and introduced before the bag $P_{i'}$. Hence if $j_e < n_{i-1}$, then e is contained also in $P_{n_{i-1}}$ and similarly if $j_e > i'$ then e is also in the bag $P_{i'}$. Therefore, $\mathcal{P}_i$ satisfies all the conditions Q1, Q2, Q3 for i.

We are left to verify that $\mathcal{P}_i$ is also a prefix path-decomposition w.r.t. $\leq^V$. It is easy to verify that since each vertex of G is in a bag of $\mathcal{P}_{i-1}$, the same holds for $\mathcal{P}_i$ as well.

Next, we show that property P2 holds. The only way our operations could have violated monotonicity is by deletion of $X_{>i}$ from a certain interval of the prefix path-decomposition, and in particular only for a vertex $x \in X_{>i}$ which occurs in $P_{n_{i-1}}$ and also in $P_{i'}$. By minimality of the separator in $P_{n_{i-1}}$ and Q3, there must exist a neighbor y of x in $V_{\leq i-1}$ which has already been forgotten in $P_{n_{i-1}}$. But then every minimal separator between $V_{\leq i}$ and $V_{>i}$ that is contained in $P_{i'}$ has to also contain x, since it has to cover the edge xy. As a consequence, $x \in X_i$ which contradicts $x \in X_{>i}$.

We proceed by showing that property P3 holds. Since X_i is a minimal separator between $V_{>i}$ and $V_{\leq i}$, all edges between $X_{\leq i}$ and $V_{>i}$ are contained in the edges between $X_{\leq i}$ and X_i, and hence these are in the bag $P'_{i'}$. Similarly, all edges between $X_{>i}$ and $V_{\leq i}$ are in $P'_{i'+2}$. Furthermore, all vertices of $V_{>i}$ are forgotten after $P'_{i'+2}$ and all vertices of $V_{\leq i}$ are introduced before $P'_{i'}$. Therefore, the fact that every edge of G is in a bag of $\mathcal{P}_i$ follows from the construction of $\mathcal{P}_i$ and the fact that $\mathcal{P}_{i-1}$ is a path-decomposition.

Finally, we show that $\mathcal{P}_i$ has the downward closure property. We consider three possibilities. If $v \in V_{\leq i} \setminus P_{i'}$, then clearly $f_{\mathcal{P}_i}(v) = P'_j$ for some $j < i'$ and $P'_{\leq j} = P_{\leq j}$ or $P'_{\leq j} = P_{\leq j} \setminus X_{>i}$. Since $v \in V_{\leq i}$ and all vertices in $X_{>i}$ are from $V_{>i}$, $D_{\leq}(v) \subseteq P_{\leq j}$ implies $D_{\leq}(v) \subseteq P'_{\leq j}$. If $v \in X_{\leq i}$, then $f_{\mathcal{P}_i}(v) = P'_{i'}$ and $D_{\leq}(v) \subseteq V_{\leq i} \subseteq P'_{\leq i'}$ follows from Q3. Otherwise, if $f_{\mathcal{P}_i}(v) = P'_j$, then $f_{\mathcal{P}_{i-1}}(v) = P_{j-2}$ and $D_{\leq}(v) \subseteq P'_{\leq j} = P_{\leq j-2}$. This concludes the proof of the Lemma. □

To later be able to compute a prefix path-decomposition, we need to be able to enumerate all possible minimal separators between $V_{\leq i}$ and $V_{>i}$ for any i with $1 \leq i \leq l$. The next lemma shows that this is indeed possible.

Lemma 38. *For every i with $1 \leq i \leq l$, there are at most $O(k^{2k})$ minimal separators between $V_{\leq i}$ and $V_{>i}$ in G and these can be enumerated in time $O(k^{2k}|B(i)|)$.*

Proof. Because of Proposition 35 it is sufficient to show the claim for the minimal vertex covers of $B(i)$. Let C_0 be all vertices in $B(i)$ whose degree is greater then k.

Note that C_0 has to be contained in any vertex cover of size at most k. It follows that $B(i) \setminus C_0$ contains at most $k(k - |C_0|)$ edges because every vertex in $B(i) \setminus C_0$ can cover at most k edges (since it has degree at most k) and one needs to cover all edges in $B(i) \setminus C_0$ with at most $k - |C_0|$ vertices. It follows that $B(i) \setminus C_0$ contains at most $2k(k - |C_0|)$ non-isolated vertices. Furthermore, the isolated vertices of $B(i)$ do not occur in any minimal vertex cover. Hence, there are at most $\binom{2k(k-|C_0|)}{k-|C_0|} \in O(k^{2k})$ vertex covers of $B(i)$ and for each of those one can check in $O(k(2k(k - |C_0|)))$ time, whether it is a minimal vertex cover. □

Our proof requires one last technical observation, which states that it is possible to find path-decompositions starting and ending with a specific bag.

Observation 39. *Given a graph G, integer k, and vertex subsets P, P' of size at most $k + 1$, there exists an FPT algorithm (parameterized by k) which either constructs a path-decompositions which starts with a bag P and ends with a bag P' of width at most k or determines that no such path-decompositions exists.*

Proof. Let us represent an instance (G, P, P') as a labeled graph G' obtained from G by the addition of two labels: the label P denotes vertices in P and the label P' denotes vertices in P'. We will show that the existence of a path-decomposition $\mathcal{P}$ with the required properties is closed under the minor operation on the labeled graph G'. Indeed, any path-decomposition of G' can be modified to a path-decomposition for any minor of G' in exactly the same way as in the case of standard path-decompositions. The claim then follows from the results of Kawarabayashi et al. [20]. □

We remark that a more efficient algorithm for Observation 39 could be obtained by adapting Bodlaender's algorithm [7]; however, a formal proof of this claim is outside the scope of this paper.

Proof of Theorem 34. By Lemma 37 we can just concentrate on finding a prefix path-decomposition such that $\mathcal{P} = (\emptyset = P_0, \ldots, P_{n_1}, \ldots, P_{n_2}, \ldots, P_{n_i}, \ldots, P_{n_l} = \emptyset)$, where P_{n_i} is a minimal separator between $V_{\leq i}$ and $V_{>i}$ and $(P_{n_{i-1}}, \ldots, P_{n_i})$ is a prefix path decomposition of $V_i \cup P_{n_{i-1}} \cup P_{n_i}$. We say that minimal separators C between $V_{\leq i-1}$ and $V_{>i-1}$ and D between $V_{\leq i}$ and $V_{>i}$ are compatible if

- if C contains a vertex $v \in V_{>i}$, then D contains v.
- if D contains a vertex $v \in V_{<i}$, then C contains v.

Since by Lemma 37 (property Q3) all vertices in $V_{>i}$ are forgotten in $\mathcal{P}$ after the bag P_{n_i} and $V_{\leq i} \subseteq P_{\leq n_i}$ it follows that $P_{n_{i-1}}$ and P_{n_i} are compatible for all i. Since all vertices of V_i are incomparable, every proper path decomposition of $V_i \cup P_{n_{i-1}} \cup P_{n_i}$ starting with the bag $P_{n_{i-1}}$ and ending with the bag $P_{n_{i-1}}$ is also a prefix path decomposition. To decide whether G has such prefix path-decomposition w.r.t. V of width k we first construct an auxiliary directed graph H as follows. H contains a vertex $(i, P, P', \mathcal{P})$ for every natural number i with $1 \leq i \leq l$ and for every two compatible minimal separators P, P' of size at most k between $V_{\leq i-1}$ and $V_{>i-1}$ and between $V_{\leq i}$ and $V_{>i}$, respectively, and $\mathcal{P}$ is a path-decomposition of $V_i \cup P \cup P'$ of width at most k that starts with P and ends with P'.

Moreover, there are two special vertices $(0, \emptyset, \emptyset, \emptyset)$ and $(l+1, \emptyset, \emptyset, \emptyset)$. There is an edge in H from every vertex $(i, P, P', \mathcal{P})$ to any vertex $(i+1, P', P'', \mathcal{P}')$, where $0 \leq i \leq l+1$. By Lemma 38 there are at most $\mathcal{O}(k^{2k})$ minimal separators between $V_{\leq i}$ and $V_{>i}$ in G and these can be enumerated in time $\mathcal{O}(k^{2k}|B(i)|)$. Therefore, H has $\mathcal{O}(l \cdot k^{4k})$ vertices. Moreover, for each i, we can enumerate all pairs P and P' and check their compatibility in time $\mathcal{O}(k^{4k}|V(G)|)$. Furthermore, for each such pair P, P', we can compute an optimal path-decomposition of of $V_i \cup P \cup P'$ starting with P and ending with P' by Observation 39. Once H is constructed, it is easily observed that each path from $(0, \emptyset, \emptyset, \emptyset)$ to $(l+1, \emptyset, \emptyset, \emptyset)$ gives us a prefix path-decomposition of G w.r.t. $\leq^V$ of width at most k. On the other hand, if there exists a prefix path-decomposition of G w.r.t. $\leq^V$, then by Lemma 37 there exists one of the form $\mathcal{P} = (\emptyset = P_0, \ldots, P_{n_1}, \ldots, P_{n_2}, \ldots, P_{n_i}, \ldots, P_{n_l} = \emptyset)$, where $P_{n_{i-1}}$ and P_{n_i} are compatible minimal separators for all i. But then H contains a path $(0, \emptyset, \emptyset, \emptyset)$, $(0, \emptyset, P_{n_1}, \mathcal{P}_1)$, $(0, P_{n_1}, P_{n_2}, \mathcal{P}_2)$,..., $(l+1, P_{n_l}, \emptyset, \emptyset)$. Since finding such a path in H takes time time at most $|V(H)|^2$, this concludes the proof of the theorem. □

6. Concluding Notes

Our results push the frontiers of tractability for QBF to new natural classes of instances. We provide one specific example of this below. A *vertex cover* of a graph G is a vertex set of G which is incident to each edge in G, and the *vertex cover number* of G is the minimum size of a vertex cover in G. The vertex cover number has often been used as a structural parameter for graph problems which do not have FPT algorithms parameterized by treewidth (see for instance [17]).

Theorem 40. QBF *is fixed parameter tractable parameterized by the vertex cover number of the primal graph.*

Proof. Let $I = (\phi, \tau)$ be an instance of QBF and let k be the vertex cover number of G_I. Let $n = |\text{var}(\tau)|$ and let Z be a vertex cover of G_I of cardinality k; recall that a vertex cover of cardinality k can be computed in time at most $\mathcal{O}(2^k \cdot n)$ [13]. For each $i \in [n]$, let v_i be the i-th variable in the prefix of I, and let $P_i = Z \cup \{v_i\}$. Now consider $\mathcal{P} = (P_1, \ldots, P_n)$. We claim that $\mathcal{P}$ is a prefix path-decomposition of width at most k.

Indeed, clearly each P_i in $\mathcal{P}$ contains at most $k+1$ vertices. It is straightforward to verify that conditions (P1) and (P2) hold, and condition (P3) must also hold since every edge has at most one endpoint outside of Z. Finally, $\mathcal{P}$ also has the downward closure property, since for each vertex v_i the set $D_{\leq^V}(v_i)$ is a subset of $Z \cup \{ v_j \mid j < i \}$, and hence $D_{\leq^V}(v) \subseteq P_{\leq f_{\mathcal{P}}(v)}$.

It follows that I has prefix pathwidth (w.r.t. the trivial dependency poset) at most k. To conclude the proof, we merely need to use the construction above to obtain a prefix path-decomposition of small width and then apply Theorem 26. □

A number of interesting research questions still remain open in the area, and perhaps the most prominent of these is whether one can lift our results towards prefix treewidth. This would be especially interesting for posets of unbounded width, since on

bounded-width posets these parameters differ only by a constant factor. The exact complexity of computing prefix treewidth and prefix pathwidth on general posets remains a challenging open problem. Finally, obtaining tight runtime bounds via improving the running times of the presented algorithms and/or by establishing lower bounds linked to the Exponential Time Hypothesis would also certainly be a worthwhile endeavour.

Acknowledgments. Robert Ganian was supported by the Austrian Science Fund (FWF, Project P31336).

[1] I. Adler and M. Weyer. Tree-width for first order formulae. *Logical Methods in Computer Science*, 8(1), 2012.

[2] M. Aschinger, C. Drescher, G. Gottlob, P. Jeavons, and E. Thorstensen. Structural decomposition methods and what they are good for. In T. Schwentick and C. Dürr, editors, *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*, volume 9 of *LIPIcs*, pages 12–28, 2011.

[3] A. Atserias and S. Oliva. Bounded-width QBF is pspace-complete. *J. Comput. Syst. Sci.*, 80(7):1415–1429, 2014.

[4] J. Barát. Directed path-width and monotonicity in digraph searching. *Graphs and Combinatorics*, 22(2):161–172, 2006.

[5] A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh. *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 2009.

[6] A. Biere and F. Lonsing. Integrating dependency schemes in search-based QBF solvers. In *Theory and Applications of Satisfiability Testing - SAT 2010*, volume 6175 of *LNCS*, pages 158–171. Springer, 2010.

[7] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[8] H. K. Büning, M. Karpinski, and A. Flögel. Resolution for quantified Boolean formulas. *Information and Computation*, 117(1):12–18, 1995.

[9] H. Chen and V. Dalmau. Decomposing quantified conjunctive (or disjunctive) formulas. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 205–214. IEEE Computer Society, 2012.

[10] B. Courcelle and S. Olariu. Upper bounds to the clique-width of graphs. *Discr. Appl. Math.*, 101(1-3):77–114, 2000.

[11] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

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

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

[14] E. Grädel et al. *Finite Model Theory and Its Applications (Texts in Theoretical Computer Science. An EATCS Series)*. Springer, 2005.

[15] U. Egly, T. Eiter, H. Tompits, and S. Woltran. Solving advanced reasoning tasks using quantified boolean formulas. In H. A. Kautz and B. W. Porter, editors, *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA.*, pages 417–422. AAAI Press / The MIT Press, 2000.

[16] E. Eiben, R. Ganian, and S. Ordyniak. Small resolution proofs for QBF using dependency treewidth. In R. Niedermeier and B. Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPIcs*, pages 28:1–28:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[17] M. R. Fellows, D. Lokshtanov, N. Misra, F. A. Rosamond, and S. Saurabh. Graph layout problems parameterized by vertex cover. In S. Hong, H. Nagamochi, and T. Fukunaga, editors, *Algorithms and Computation, 19th International Symposium, ISAAC 2008, Gold Coast, Australia, December 15-17, 2008. Proceedings*, volume 5369 of *Lecture Notes in Computer Science*, pages 294–305. Springer, 2008.

[18] S. Felsner, V. Raghavan, and J. Spinrad. Recognition algorithms for orders of small width and graphs of small dilworth number. *Order*, 20(4):351–364, 2003.

[19] R. Ganian, P. Hliněný, and J. Obdržálek. Better algorithms for satisfiability problems for formulas of bounded rank-width. *Fundam. Inform.*, 123(1):59–76, 2013.

[20] K. Kawarabayashi, Y. Kobayashi, and B. A. Reed. The disjoint paths problem in quadratic time. *J. Comb. Theory, Ser. B*, 102(2):424–435, 2012.

[21] H. Kleine Büning and T. Lettman. *Propositional logic: deduction and algorithms*. Cambridge University Press, Cambridge, 1999.

[22] T. Kloks. *Treewidth: Computations and Approximations*. Springer Verlag, Berlin, 1994.

[23] M. Lampis and V. Mitsou. Treewidth with a quantifier alternation revisited. In D. Lokshtanov and N. Nishimura, editors, *12th International Symposium on Parameterized and Exact Computation, IPEC 2017, September 6-8, 2017, Vienna, Austria*, volume 89 of *LIPIcs*, pages 26:1–26:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[24] C. Otwell, A. Remshagen, and K. Truemper. An effective QBF solver for planning problems. In *Proceedings of the International Conference on Modeling, Simulation & Visualization Methods, MSV '04 & Proceedings of the International*

Conference on Algorithmic Mathematics & Computer Science, AMCS '04, June 21-24, 2004, Las Vegas, Nevada, USA, pages 311–316. CSREA Press, 2004.

[25] S. Oum and P. Seymour. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006.

[26] G. Pan and M. Y. Vardi. Fixed-parameter hierarchies inside PSPACE. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, pages 27–36, 2006.

[27] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[28] J. Rintanen. Constructing conditional plans by a theorem-prover. *J. Artif. Intell. Res.*, 10:323–352, 1999.

[29] N. Robertson and P. D. Seymour. Graph minors. i. excluding a forest. *J. Comb. Theory, Ser. B*, 35(1):39–61, 1983.

[30] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.

[31] A. Sabharwal, C. Ansótegui, C. P. Gomes, J. W. Hart, and B. Selman. QBF modeling: Exploiting player symmetry for simplicity and efficiency. In A. Biere and C. P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, volume 4121 of *Lecture Notes in Computer Science*, pages 382–395. Springer, 2006.

[32] M. Samer and S. Szeider. Backdoor sets of quantified Boolean formulas. *Journal of Autom. Reasoning*, 42(1):77–97, 2009.

[33] P. Scheffler. *Die Baumweite von Graphen als ein Maß für die Kompliziertheit algorithmischer Probleme*, volume 4. Akademie der Wissenschaften der DDR, Karl-Weierstrass-Institut für Mathematik, 1989.

[34] F. Slivovsky and S. Szeider. Variable dependencies and q-resolution. In *Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8561 of *Lecture Notes in Computer Science*, pages 269–284. Springer, 2014.

[35] F. Slivovsky and S. Szeider. Quantifier reordering for QBF. *J. Autom. Reasoning*, 56(4):459–477, 2016.

[36] D. A. Spielman and M. Bóna. An infinite antichain of permutations. *Electron. J. Combin*, 7:N2, 2000.

[37] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1973, Austin, Texas, USA*, pages 1–9. ACM, 1973.

[38] S. Szeider. On fixed-parameter tractable parameterizations of SAT. In E. Giunchiglia and A. Tacchella, editors, *Theory and Applications of Satisfiability, 6th International Conference, SAT 2003, Selected and Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, pages 188–202. Springer Verlag, 2004.

[39] H. Tamaki. A polynomial time algorithm for bounded directed pathwidth. In P. Kolman and J. Kratochvíl, editors, *Graph-Theoretic Concepts in Computer Science - 37th International Workshop, WG 2011, Teplá Monastery, Czech Republic, June 21-24, 2011. Revised Papers*, volume 6986 of *Lecture Notes in Computer Science*, pages 331–342. Springer, 2011.