Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Using Decomposition-Parameters for QBF: Mind the Prefix!

Eduard Eiben, Robert Ganian, Sebastian Ordyniak
TU Wien, Vienna, Austria

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.

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 (Egly et al. 2000; Otwell, Remshagen, and Truemper 2004; Rintanen 1999; Sabharwal et al. 2006), 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 (Kleine Büning and Lettman 1999; Papadimitriou 1994; Stockmeyer and Meyer 1973).

In spite of the close connection between QBF and SAT, many of the tools and techniques which work for SAT are not known to help for QBF, and this is especially true for so-called decomposition-based techniques (Aschinger et al. 2011). Such techniques use various kinds of decompositions to capture the structure of the input, leading to efficient algorithms for computing solutions with runtime guarantees. Decomposition-based techniques 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 which 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 (Robertson and Seymour 1986), pathwidth (Robertson and Seymour 1983), clique-width (Courcelle and Olariu 2000) and rank-width (Oum and Seymour 2006); all of these are known to support FPT algorithms for SAT (Szeider 2004; Ganian, Hliněný, and Obdržálek 2013), but the same is not true for QBF (Atserias and Oliva 2014) under established complexity assumptions. As a consequence, many classes of QBFs which have a natural and seemingly "simple" structure remained beyond the reach of current algorithmic techniques. For completeness, we remark that formally our results consider QBFs which have been transformed into so-called *alternating prenex form* (see the Preliminaries).

In this work we introduce and develop *prefix pathwidth*, which is the first 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 (2009; 2014), see also the work of Biere and Lonsing (2010). 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* (Samer and Szeider 2009), which stipulates that each variable depends on all variables with distinct quantification, which come before it in the prefix. When using this dependency scheme, our main result (Theorem 3) yields:

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

However, prefix pathwidth can be used in conjunction with any dependency scheme that is cumulative (Samer and

Szeider 2009), which holds for almost all known dependency schemes; this is reflected in all of our technical results, where we do not fix any particular dependency scheme. In practice, using different dependency schemes may lead to better prefix path-decompositions, in turn resulting in significantly faster algorithms.

Our use of dependency schemes is also related to the reason we use prefix pathwidth instead of a prefix extension of treewidth, even though treewidth has generally seen more use than pathwidth in other fields of computer science. For the trivial dependency scheme, as well as for any QBF and dependency scheme where we can efficiently compute prefix path-decompositions, our Theorem 6 establishes that using an analogously defined prefix version of treewidth would not lead to any substantial advantages over the conceptually simpler prefix pathwidth. Furthermore, we show that prefix pathwidth is closely related to the notion of directed pathwidth studied on directed graphs.

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

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. (2005)) 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 cumulative dependency scheme, when such a decomposition is provided as part of the input.*

Resolving step 2 boils down to a graph-algorithmic problem 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 (Tamaki 2011) 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 two distinct algorithms for computing prefix path-decompositions—one polynomial-time approximation algorithm (Theorem 15) and one FPT algorithm (Theorem 14). The efficiency of these algorithms 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; for example, the poset-width is always at most one when the trivial dependency scheme is used, and hence our algorithms are highly efficient for this dependency scheme. In combination with the previous Theorem 2, Theorem 15 yields our main general contribution, 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 cumulative dependency scheme. There exists an FPT algorithm which 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). The full version of proofs marked as proof sketches can be found in the full version of this paper.

Related Work

We are not aware of many successful attempts making use of decompositional parameters for QBF. Chen and Dalmau (2012) showed that QCSP becomes fixed-parameter tractable parameterized by the length of the formula given that a specific width notion is bounded by a constant in the input graph. Other structural parameters such as backdoors have also been studied in the context of QBF (Samer and Szeider 2009).

Preliminaries

For $i \in \mathbb{N}$, we let $[i]$ denote the set $\{1, \ldots, i\}$. We refer to the book by Diestel (2012) for standard graph-theoretic terminology. Given a 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' are the vertices in V' with at least one neighbor in $V(G) \setminus V'$.

We refer to Flum and Grohe; Downey and Fellows (2006; 2013) 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 *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*.

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 $v(x) = v(\neg x) = x$ denotes 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 over clauses. We say that a CNF formula ϕ is *over* a variable set K if each literal x in ϕ satisfies $v(x) \in K$, and denote the set of variables which occur in ϕ by $\mathrm{var}(\phi)$. For notational purposes, we will view a clause as a set of literals and a CNF formula as a set of clauses.

A *quantified Boolean formula* is a tuple (ϕ, τ) where ϕ is a CNF formula and τ is a sequence of quantified variables, denoted $\text{var}(\tau)$, which satisfies $\text{var}(\tau) \supseteq \text{var}(\phi)$; 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 *assignment* is a mapping from (a subset of) the variables to $\{0, 1\}$.

To provide succinct proofs for our statements, it will later 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 $2 \cdot |\text{var}(\phi)| + 1$.

Given a QBF $I = (\phi, Y, X)$ and a partial assignment $\omega : Q \to \{0, 1\}$ where $Q \subseteq X \cup Y$, we denote by I_ω the subinstance 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 $I = (\phi, Y, X)$ is the graph G_I defined as follows. The vertex set of G_I consists of every variable which occurs in ϕ, and st is an edge in G_I if there exists a clause in ϕ containing both s and t.

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 (Samer and Szeider 2009). Intuitively, a dependency scheme assigns to each variable x the set of variables that depend on x. Here, we omit the formal definition of the dependency scheme and focus on dependency posets, which can be obtained from any so-called *cumulative* dependency scheme (Samer and Szeider 2009).

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 $\mathcal{V}$. 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 4 (Felsner, Raghavan, and Spinrad (2003))**.** *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 $\mathcal{V}$.*

From now on, it is implicitly assumed that every QBF is in alternating prenex form. To define dependency posets we need also the notion of *shifting*, which takes some subset of variables of QBF I in the prefix and moves them together with their quantification, in the same relative order, to the end (*down-shifting*) or to the beginning (*up-shifting*) of the prefix. If a QBF is obtained by shifting, it is assumed to have subsequently been transformed into alternating prenex form.

Given a QBF $I = (\phi, Y, X)$, a *dependency poset* $\mathcal{V} = (\text{var}(\phi), \leq^I)$ of I is a poset over $\text{var}(\phi)$ with the following properties:

1. if $x \leq^I y$, then x is before y in the prefix for all $x, y \in \text{var}(\phi)$ and
2. for every $A \subseteq \text{var}(\phi)$, if A is downward-closed w.r.t. $\leq^I$, then the QBF I' obtained by up-shifting A is true iff I is true.

The *trivial dependency scheme* assigns to each variable x the closest variables on the right of x with different quantification. This gives rise to the *trivial dependency poset*, which forms a complete total ordering on the variables. However, more refined dependency schemes which give rise to other dependency posets are known to exist and can be computed efficiently (Samer and Szeider 2009).

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. Up-shifting of the downward-closed set $\{x, u\}$ yields the QBF I':

$$\forall x \exists t_1 \forall u \exists y \forall t_2 \exists v (x \vee \neg y \vee v) \wedge (\neg u \vee \neg v \vee y) \wedge (\neg x \vee u \vee \neg v),$$

where t_1, t_2 are new dummy variables, not occurring in the matrix of I. 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.

Pathwidth and Treewidth

Definition 5 (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.

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 (Kloks 1994).

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 with t as a root, 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 with $v \in T_{\leq t} \setminus X_s$, 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_i, $P_{\leq i}$, $P_{<i}$, and $f_{\mathcal{P}}(v)$ analogously.

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 then the minimum width over all prefix tree decompositions of G. The *prefix pathwidth*, denoted by $ppw(G)$, is then defined analogously.

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.

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

We build on the above definitions to define the notions we need on QBFs. A prefix path-decomposition of a QBF $I = (\phi, Y, X)$ w.r.t. a dependency poset $\mathcal{V} = (\mathrm{var}(\phi), \leq^I)$ 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. $\mathcal{V}$.

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.

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 (E. Grädel et al. 2005), 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 10 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.

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 $\mathcal{T} = (\tau_i : \{0,1\}^i \to \{0,1\})_{i=1,\ldots,\ell}$. An existential strategy $\mathcal{T}$ is *winning* if, for any mapping $\delta : \{y_1, \ldots, y_n\} \to \{0,1\}$, 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 $\mathcal{T} = (\tau_i : \{0,1\}^i \to \{0,1\})_{i=1,\ldots,\ell'}$, for some $\ell' \leq \ell$.

A mapping δ from a subset of Y to $\{0,1\}$ is called a *universal 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 7 (folklore)**.** *A QBF I is true iff there exists a winning existential 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 cumulative 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 $B_i = P_i$ be a bag in $\mathcal{P}$, $D_i = D_{\leq^I}(P_{<i})$, $C_i = P_{<i}$ (see Figure 1), and let I be up-shifted on D.

Observation 8. For any i with $1 \leq i \leq n$, B_i forms a separator in G_I and hence each clause in ϕ either contains only

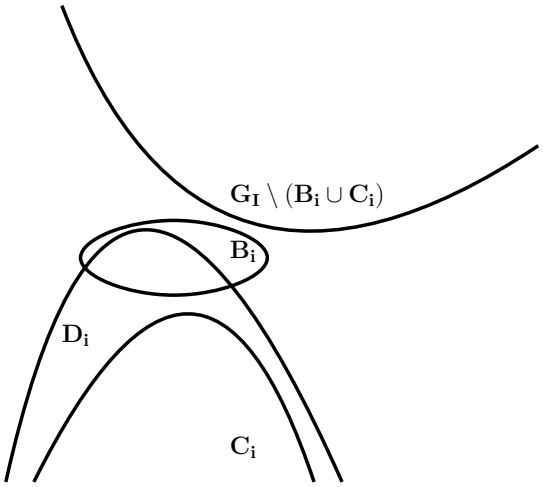

Figure 1: B_i is a bag in $\mathcal{P}$ that separates C_i, i.e., vertices forgotten in some bag before B_i, from the rest of the graph. D_i is the downward closure of C_i w.r.t. $\leq^I$.

variables in $P_{\leq i}$ or only variables in $(Y \cup X) \setminus C_i$. Furthermore, $D_i \subseteq P_{\leq i}$.

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.

Equivalence of Assignments and Strategies

The first equivalence, which serves as the building block for the latter one, considers assignments of the variable set D_i.

Definition 9. Let δ_1 and δ_2 be two partial assignments of D_i. Then $\delta_1 \approx \delta_2$ iff $I_{\delta_1} = I_{\delta_2}$.

It can be proved that the number of equivalence classes of $\approx$ can be upper-bounded by a function of k. Next, 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 which 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 10. Let α_1 and α_2 be two partial existential strategies on D_i. Then $\alpha_1 \equiv \alpha_2$ iff $S_{\alpha_1} = S_{\alpha_2}$.

It can be shown that $\equiv$ is an equivalence and that its number of equivalence classes is upper-bounded by a function of k. Furthermore, it is possible to prove that the signature of any partial existential strategy which is obtained from a winning strategy must contain only true instances.

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

The algorithm consists of the following procedures:

1. *Initialization*: this is the procedure that is called at the beginning of the algorithm, i.e., for the empty bag P_1.
2. *Introduce*: this is the procedure that is called whenever we have computed K_{i-1} and P_i is an introduce node.
3. *Forget*: this is the procedure that is called whenever we have computed K_{i-1} and P_i is a forget node.
4. *Termination*: this is the procedure that is called when we have computed K_n.

All procedures except for *Forget* are straightforward at this point, since the signature set can only change if new variables are added into D_i. We develop an FPT algorithm for *Forget* which proceeds as follows. We let Z be the variables which were added into D_i in this step, i.e., $Z = D_i \setminus D_{i-1}$. We show that $|Z| \leq k$ and we alter the prefix of the instance so that Z comes immediately after D_{i-1}. Then for each signature S in K_{i-1}, we take each instance $I_\delta \in S$ and loop over all possible partial existential strategies on Z which play on I_δ; in other words, for each signature in K_{i-1} we consider all ways an oblivious strategy could proceed on Z depending on the play in D_{i-1} and the play of universal variables in Z. This allows us to compute the signature set K_i. Observe that considering only oblivious strategies is crucial for the above to work—the number of reduced instances in each signature $S \in K_{i-1}$ is bounded by k, but the number of universal plays in D_{i-1} is not.

Having established the procedures for the individual nodes of the path-decomposition, we can now prove the correctness of the whole dynamic programming algorithm.

Theorem 12. *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 and decides whether I is true.*

Computing Prefix Pathwidth

This section is devoted to parameterized and approximation algorithms for computing the prefix pathwidth. Observe that if the given partial ordering is empty, then the prefix pathwidth of the graph G is the same as the pathwidth of G. Thus, computing the prefix pathwidth is NP-complete.

Before we present our algorithms, we will state some interesting observations about prefix path-decompositions (Please refer to the full version of this paper for a more detailed exposition of the following observations). 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 (Barát 2006).

Observation 13. 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 distinct $u, v \in V(G)$ such that $u \leq^V v$. Then, $ppw_{\leq^V}(G) = dpw(D)$, where $dpw(D)$ denotes the directed pathwidth of D.

Since it has been shown (Tamaki 2011) 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. It is an important open question, however, whether computing directed pathwidth is fixed-parameter tractable.

In the following we will give two algorithms that compute a 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). Since the width of the trivial dependency scheme is at most one, this in particular implies an fpt-algorithm in the case of the trivial dependency scheme.

Theorem 14. *Finding a prefix path-decomposition of G w.r.t. $\leq^V$ of width at most k or deciding that no such prefix path-decomposition exists can be done in time $O((|V(G)|^w k^{2k})^2 |V(G)|)$. Hence, for any constant w computing a prefix path-decomposition is fixed-parameter tractable in k.*

Proof Sketch. The main observation behind the algorithm is that in any prefix path-decomposition of G w.r.t. $\leq^V$, the intersection between any two bags can be characterized by a pair (D, C), where D is a downward closed set of vertices of G 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. 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'). Computing an 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. Because the number of downward closed sets is bounded by $|V(G)|^w$ and one can show 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. □

Our second result shows that the prefix pathwidth of G w.r.t. $\leq^V$ can be approximated in polynomial-time up to $2w(2k^2 + k) + 1$, where k is the optimum prefix pathwidth. Note that this algorithm together with Theorem 12 implies an FPT algorithm to decide a QBF parameterized by the width of its poset and its prefix pathwidth.

Theorem 15. *There is a polynomial-time algorithm that 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.*

Proof Sketch. The algorithm starts from the empty prefix path-decomposition and at each step it tries to extend the current prefix path-decomposition by at least the currently smallest (unprocessed) vertex (w.r.t. $\leq^V$) from some chain of $(V(G), \leq^V)$. It does so as long as there is a chain such that after adding its currently smallest (unprocessed) vertex the number of vertices required to guard (separate) the already processed vertices on that chain from the set of all unprocessed vertices (on every chain) does not exceed $2k^2 + k$. A delicate argument then shows that in this manner the algorithm is either able to compute a prefix path-decomposition of width at most $2w(2k^2 + k) + 1$ or one can show that no prefix path-decomposition of G w.r.t. $\leq^V$ of width at most k can exist. □

Concluding Notes

The notion of prefix pathwidth is, to the best of our knowledge, the first decomposition-based parameter which supports an FPT algorithm for QBF. Our results, specifically Theorem 12 and Theorem 15, together 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 Fellows et al. (2008)).

Theorem 16. QBF *is fixed parameter tractable parameterized by the vertex cover number of the primal graph.*

A number of interesting research questions still remain open in the area. In particular, is it possible to compute (or approximate) an optimal prefix path-decomposition in FPT time parameterized *only* by the prefix pathwidth, i.e., regardless of the poset-width? And can our results be generalized towards prefix treewidth on instances of unbounded poset-width? Do there exist other natural structural parameters for QBF? These and other questions form important challenges for future research.

Acknowledgments The authors wish to thank the anonymous reviewers for their helpful comments. Eduard Eiben acknowledges support by the Austrian Science Fund (FWF, projects P26696 and W1255-N23). Robert Ganian is also affiliated with FI MU, Brno, Czech Republic.

References

Aschinger, M.; Drescher, C.; Gottlob, G.; Jeavons, P.; and Thorstensen, E. 2011. Structural decomposition methods and what they are good for. In *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*, 12–28.

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

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

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

Chen, H., and Dalmau, V. 2012. Decomposing quantified conjunctive (or disjunctive) formulas. In *Proceedings of the 2012 27th Annual IEEE/ACM Symposium on Logic in Computer Science*, 205–214. IEEE Computer Society.

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

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

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

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

Egly, U.; Eiter, T.; Tompits, H.; and Woltran, S. 2000. Solving advanced reasoning tasks using quantified boolean formulas. In *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.*, 417–422.

Fellows, M. R.; Lokshtanov, D.; Misra, N.; Rosamond, F. A.; and Saurabh, S. 2008. Graph layout problems parameterized by vertex cover. In *Algorithms and Computation, 19th International Symposium, ISAAC 2008, Gold Coast, Australia, December 15-17, 2008. Proceedings*, 294–305.

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

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Berlin: Springer Verlag.

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

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

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

Otwell, C.; Remshagen, A.; and Truemper, K. 2004. 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*, 311–316. CSREA Press.

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

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

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

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

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

Sabharwal, A.; Ansótegui, C.; Gomes, C. P.; Hart, J. W.; and Selman, B. 2006. QBF modeling: Exploiting player symmetry for simplicity and efficiency. In *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, 382–395.

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

Slivovsky, F., and Szeider, S. 2014. 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*, 269–284. Springer.

Stockmeyer, L. J., and Meyer, A. R. 1973. 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*, 1–9. ACM.

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

Tamaki, H. 2011. A polynomial time algorithm for bounded directed pathwidth. In *Graph-Theoretic Concepts in Computer Science - 37th International Workshop, WG 2011, Teplá Monastery, Czech Republic, June 21-24, 2011. Revised Papers*, 331–342.