Treewidth with a Quantifier Alternation Revisited

Michael Lampis[1] and Valia Mitsou*[2]

1 **Université Paris-Dauphine, PSL Research University, CNRS, UMR 7243 LAMSADE, 75016, Paris, France,** michail.lampis@dauphine.fr
2 **Université de Lyon, LIRIS, CNRS, UMR 5205, Université Lyon 1, 69622, Villeurbanne, Lyon, France,** vmitsou@liris.cnrs.fr

Abstract

In this paper we take a closer look at the parameterized complexity of ∃∀SAT, the prototypical complete problem of the class Σ_2^p, the second level of the polynomial hierarchy. We provide a number of tight fine-grained bounds on the complexity of this problem and its variants with respect to the most important structural graph parameters. Specifically, we show the following lower bounds (assuming the ETH):

- It is impossible to decide ∃∀SAT in time less than *double-exponential* in the input formula's treewidth. More strongly, we establish the same bound with respect to the formula's primal vertex cover, a much more restrictive measure. This lower bound, which matches the performance of known algorithms, shows that the degeneration of the performance of treewidth-based algorithms to a tower of exponentials already begins in problems with one quantifier alternation.
- For the more general ∃∀CSP problem over a non-boolean domain of size B, there is no algorithm running in time $2^{B^{o(vc)}}$, where vc is the input's primal vertex cover.
- ∃∀SAT is already NP-hard even when the input formula has constant modular treewidth (or clique-width), indicating that dense graph parameters are less useful for problems in Σ_2^p.
- For the two weighted versions of ∃∀SAT recently introduced by de Haan and Szeider, called $\exists_k\forall$SAT and $\exists\forall_k$SAT, we give tight upper and lower bounds parameterized by treewidth (or primal vertex cover) and the weight k. Interestingly, the complexity of these two problems turns out to be quite different: one is double-exponential in treewidth, while the other is double-exponential in k.

We complement the above negative results by showing a double-exponential FPT algorithm for QBF parameterized by vertex cover, showing that for this parameter the complexity never goes beyond double-exponential, for any number of quantifier alternations.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, F.2.2 Nonnumerical Algorithms and Problems, G.2.1 Combinatorics, G.2.2 Graph Theory

Keywords and phrases Treewidth, Exponential Time Hypothesis, Quantified SAT

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

The main goal of this paper is to provide a fine-grained complexity analysis of ∃∀SAT, the prototypical complete problem for the second level of the polynomial hierarchy, with respect to the most important structural graph parameters, and especially treewidth.

* Valia Mitsou was supported by the ANR-14-CE25-0006 project of the French National Research Agency.

Conference title on which this volume is based on.
Editors: Billy Editor, Bill Editors; pp. 1–12

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Treewidth, a graph parameter that roughly measures how tree-like a graph is, is one of the central notions of parameterized complexity theory. Its popularity and success rest largely on the fact that when a graph's treewidth is small a very large variety of problems which are normally intractable can be solved efficiently (often in linear time [3]). This success has strongly motivated research that seeks to apply the ideas of treewidth to other domains, such as satisfiability problems. This can be done by defining a graph that (partially) encodes the structure of the input instance, and then using this graph's structure to design efficient algorithms. Much work has been devoted in this direction, and by now the complexity of satisfiability and related constraint satisfaction problems (counting, maximization, etc.) is well-understood not only with respect to treewidth, but also other related structural graph parameters such as clique-width, modular treewidth, vertex cover, and others [8, 19, 21, 23, 25, 24].

Despite this success, one notable weakness of the treewidth approach is that its effectiveness rapidly deteriorates once one starts considering problems outside NP. Indeed, it has long been known that as one considers problems that need more and more quantifier alternations to be expressed, the "hidden constants" in treewidth-based algorithms become towers of exponentials [10, 16, 20] and for an unbounded number of alternations, even constant treewidth does not help [1]. One response to this weakness has been the search for other graph measures which are more robust in the face of alternating quantifiers [9, 11, 12, 15]. Our goal in this paper on the other hand is to more closely examine whether we can hope to evade this deteriorating performance not necessarily by changing the graph parameter, but by restricting ourselves to problems with only one quantifier alternation, that is, problems in Σ_2^p, the second level of the polynomial hierarchy.

Given the current state of the art, one would probably expect the typical Σ_2^p-complete problem to have double-exponential complexity when parameterized by treewidth, as a result of the extra quantifier alternation, and indeed the best currently known algorithms typically do have this complexity. Nevertheless, almost no *concrete lower bounds* are known showing that natural problems in this class need a double-exponential dependence on treewidth. The lower bounds given in [10, 16, 20] are either asymptotic or only give concrete tight bounds for the *odd* levels of the polynomial hierarchy. Perhaps the only tight double-exponential lower bound for the complexity of a Σ_2^p-complete problem parameterized by treewidth is given in [17], which proves such a result for Choosability. To the best of our knowledge no such bounds are known for other concrete problems, and in particular, no such tight bounds are known for perhaps the most basic problem in this class: $\exists\forall$SAT.

Our contribution: Our main conceptual contribution is a simple direct reduction showing that the complexity of $\exists\forall$SAT when parameterized by the treewidth of the input formula has to be double-exponential, unless the ETH is false (Theorem 1). This essentially matches the running time of the best currently known algorithm [2]. Before this work similar bounds were only known for satisfiability problems complete for odd levels of the polynomial hierarchy [20], and thus this fills a natural hole in the literature. However, beyond advancing our understanding of the complexity of quantified satisfiability, we believe the main value of this result is in providing the first "textbook" example of a double-exponential lower bound for treewidth. Unlike currently known lower bounds of somewhat similar flavor ([15, 17, 20]), our result is a completely self-contained reduction that essentially does not need any gadgets. Furthermore, because of the central role of $\exists\forall$SAT in the class Σ_2^p we expect that our lower bound may serve as a starting point to prove similar bounds for various other problems in Σ_2^p.

Building and extending on the above result we give a number of tight upper and lower

bounds on ∃∀SAT and related problems. Specifically, we observe that we are able to give a similar double-exponential lower bound on the treewidth dependence for ∃∀3-SAT, and that our reduction shows that ∃∀SAT is NP-hard even for formulas of constant modular treewidth or clique-width (meaning that dense graph parameters are likely to be much less useful for ∃∀SAT than they are for SAT). We also observe that the main lower bound for ∃∀SAT applies not only when the problem is parameterized by treewidth, but much more strongly, when it is parameterized by vertex cover.

Finding this latter fact surprising, since most problems are significantly easier when parameterized by vertex cover than by treewidth, we investigate vertex cover as a parameter more closely. We show that, despite matching closely the complexity of treewidth for ∃∀SAT, this parameter is indeed much more algorithmically amenable in other ways: first, it allows a double-exponential algorithm for QBF with any number of quantifier alternations (a problem PSPACE-complete for constant treewidth); and second, it allows a single-exponential algorithm for ∃∀3-SAT. We note that QBF was already known to be FPT parameterized by vertex cover [9] as a corollary of an algorithm for a much more general parameter (prefix pathwidth), but since we concentrate on vertex cover we are able to give an algorithm that is both faster and significantly simpler.

Having analyzed the complexity of ∃∀SAT with respect to some of the most important graph parameters, we move towards two more recently introduced variations with special interest for parameterized complexity: $\exists_k\forall$SAT and $\exists\forall_k$SAT. In these problems, introduced in the works of de Haan and Szeider [6, 5, 7], one of the two quantifiers is weighted, that is, we only consider assignments that set k out of its variables to true. Though both problems are FPT parameterized by treewidth we show that their complexity is quite different: one is double-exponential with respect to treewidth, while the other is double-exponential with respect to k (and single-exponential in the treewidth). Both lower bounds match algorithms that follow from simple adaptations of [2].

Finally, we consider a more general version of our problem: ∃∀CSP where variables are no longer necessarily boolean. The question here is how the complexity of the problem changes not only with respect to treewidth, but also with the size of the domain of a non-boolean CSP. Once again we show a double-exponential bound that essentially matches the performance of the best known algorithm.

2 Definitions and Preliminaries

2.1 Problem Definitions

We recall some standard definitions related to satisfiability problems: a *literal* is a boolean variable, or its negation; a clause is a disjunction of literals; a term is a conjunction of literals; a formula is in conjunctive normal form (CNF) if it is a conjunction of clauses; it is in disjunctive normal form (DNF) if it is a disjunction of terms. We will sometimes overload notation and view clauses and terms as sets of literals.

The main problem we study in this paper is ∃∀SAT, the prototypical complete problem for the second level of the polynomial hierarchy [26]. In this problem we are given a quantified formula of the form $\exists\mathbf{x}\forall\mathbf{y}\phi(\mathbf{x},\mathbf{y})$, where $\mathbf{x},\mathbf{y}$ are disjoint tuples of boolean variables and ϕ is a DNF formula. The question is to decide whether there exists an assignment to the variables of $\mathbf{x}$ so that for all assignments to $\mathbf{y}$, ϕ evaluates to True. We refer to $\mathbf{x}$ as the existential variables and $\mathbf{y}$ as the universal variables. When all terms of ϕ have size at most d we refer to this problem as $\exists\forall d$-SAT. We also consider two *weighted* versions: in $\exists_k\forall$SAT we are asked if there exists an assignment that sets exactly k existential variables

to True, such that for all assignments to $\mathbf{y}$, ϕ evaluates to True; in $\exists\forall_k$SAT we are asked if there exists any assignment to $\mathbf{x}$ such that for all assignments that set exactly k of the variables of $\mathbf{y}$ to True, ϕ evaluates to True. A more general version of $\exists\forall$SAT allows the input formula to have any number of quantifiers, that is, we are given a formula of the form $\exists x_1 \forall x_2 \exists x_3 \ldots Q x_n \phi(x_1, x_2, \ldots, x_n)$, where ϕ is in CNF, and are asked if it evaluates to True. We call this problem QBF.

We will also consider the more general $\exists\forall$CSP problem. Here we are given a CSP instance again involving two tuples of variables $\mathbf{x}, \mathbf{y}$, which can now take values from some finite domain Σ. Furthermore, we are given a set of constraints: a constraint of arity r involves r variables from $\mathbf{x} \cup \mathbf{y}$, and we are given a list of assignments that satisfy the constraint, that is, a subset of Σ^r which determines which combinations of assignments to the involved variable satisfy the constraint. We say that an $\exists\forall$CSP instance is a Yes instance if there exists an assignment to $\mathbf{x}$ such that for all assignments to the variables of $\mathbf{y}$ at least one constraint is satisfied.

2.2 Graph Parameters

Given a propositional formula ϕ we consider two types of graphs associated with it. The primal graph, denoted $G_p(\phi)$ contains a vertex for each variable of ϕ; two vertices of $G_p(\phi)$ are connected if and only if the corresponding variables of ϕ appear in a constraint together. The incidence graph, denoted $G_i(\phi)$ contains a vertex for each variable and for each constraint of ϕ; two vertices of $G_i(\phi)$ are connected if the corresponding variable appears in the corresponding constraint.

In this paper we study various structural restrictions on formulas by considering graph parameters restricting the structure of $G_i(\phi)$ or $G_p(\phi)$. The parameters we consider are the treewidth of $G_i(\phi)$ and $G_p(\phi)$ denoted $tw_i(\phi)$ and $tw_p(\phi)$ respectively; the vertex cover of $G_p(\phi)$, denoted $vc(\phi)$; and the modular treewidth and clique-width of $G_i(\phi)$, denoted $mtw(\phi)$ and $cw(\phi)$ respectively. We refer the reader to standard textbooks for the definitions of treewidth, clique-width and vertex cover [4]. Modular treewidth is the treewidth of a graph after contracting all vertices that share the same neighborhood into single vertices [22]. We recall the following well-known relationships between these parameters. For all formulas ϕ we have $vc(\phi) \geq tw_p(\phi) \geq tw_i(\phi) \geq mtw(\phi)$. Because of this inequality, hardness results which apply for vertex cover automatically carry over to the other (more general) parameters, while algorithmic results which apply to modular treewidth also apply to the other (less general) parameters. We recall that it is also known that $cw(\phi)$ is bounded by some function of $mtw(\phi)$.

2.3 Binary Representations

Let $\mathbf{x}$ be a tuple of variables, and i be an integer such that $i < 2^{|\mathbf{x}|}$. We define the function $B(i, \mathbf{x})$ which produces a set of *literals*, that is, a set which contains for each variable of $\mathbf{x}$ either the variable itself or its negation. Negations will be added according to the binary representation of i. More formally, if $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ we define inductively $B(i, x) = B(\lfloor i/2 \rfloor, \mathbf{x} \setminus \{x_n\}) \cup l_n$, where $l_n = x_n$ if i is even, and $l_n = \neg x_n$ otherwise. For the base case ($n = 0$) we set $B(0, \emptyset) = \emptyset$.

3 Tight Bounds for ∃∀SAT

In this section we consider the complexity of the prototypical Σ_2^p-complete problem ∃∀SAT. Our first result is a fine-grained lower bound which states that any algorithm for ∃∀SAT must either use time exponential in the number of variables of the formula, or run in time double-exponential in the input formula's primal vertex cover (assuming the ETH). This result is obtained through a direct reduction from an n-variable instance of 3-SAT that produces an instance of ∃∀SAT with roughly the same number of variables, but vertex cover $O(\log n)$. In other words, our reduction trades the additional complexity of the problem (caused by the extra level of quantification), to encode the formula in such a way that its structure is significantly simplified. We note that this lower bound is tight, not only for vertex cover, but even for much more general parameters, such as treewidth, for which double-exponential algorithms for ∃∀SAT are known [2].

We then go on to give a second version of the same reduction that gives a similar running-time bound for ∃∀3-SAT parameterized by the input formula's primal treewidth. Here we use a standard trick to reduce the size of all terms to 3; the non-trivial part is to prove that this does not significantly increase the primal treewidth of the instance. Interestingly, the lower bound for ∃∀3-SAT does *not* apply for primal vertex cover. As we show in Section 4 there is a good reason for this, as ∃∀3-SAT is solvable in single-exponential time for this parameter.

We also observe that our main reduction produces instances with constant modular treewidth (and hence clique-width), indicating that these two dense parameters (for which SAT is in XP) are unlikely to be of help with satisfiability problems in Σ_2^p.

▶ **Theorem 1.** *There is no algorithm which, given an* ∃∀SAT *instance ϕ with n variables, can decide if ϕ is True in time $2^{2^{o(vc(\phi))}}2^{o(n)}$, unless the ETH is false.*

Proof. We consider an instance of 3-SAT ψ with n variables and m clauses. Recall that the ETH states that there is no $2^{o(n+m)}$ algorithm that decides 3-SAT[13, 14]. We will construct a quantified DNF formula $\exists \mathbf{x} \forall \mathbf{y} \phi(\mathbf{x}, \mathbf{y})$, whose primal vertex cover will be $O(\log n)$.

Let $\mathbf{x} = \{x_1, \ldots, x_n\}$ be the set of variables of ψ. We retain the same variables as the existential variables of ϕ, and we introduce $\log m$ universally quantified variables $\mathbf{y} = (y_1, \ldots, y_{\log m})$. We assume here without loss of generality that m is a power of 2, otherwise some clauses of ψ can be repeated.

Suppose that the clauses of ψ are numbered $C_0, C_1, \ldots, C_{m-1}$. Let C_i be a clause of ψ of length $l \leq 3$. We construct l terms in ϕ as follows: each term contains one distinct literal of C_i, and all the literals of $B(i, \mathbf{y})$ (the binary encoding defined in Section 2.3). This completes the construction.

To see the correctness of the reduction, suppose that ψ is satisfiable. We take a satisfying assignment and use the same values for the $\mathbf{x}$ variables in ϕ. Now, for any assignment of the $\mathbf{y}$ variables there will be an i such that all the literals in $B(i, \mathbf{y})$ are true. Consider the terms we added representing the clause C_i. One of them contains a true literal from the original clause, so it is a satisfied term. For the other direction, if $\exists \mathbf{x} \forall \mathbf{y} \phi(\mathbf{x}, \mathbf{y})$ is true, we use the same assignment for $\mathbf{x}$ in ψ. If some clause C_i of ψ is not satisfied, this would imply that setting $\mathbf{y}$ to the assignment that agrees with $B(i, \mathbf{y})$ would also make ϕ false, a contradiction.

Observe that the primal graph of the formula ϕ becomes an independent set if we remove the $\log m$ vertices corresponding to the variables of $\mathbf{y}$. Thus, $vc(\phi) = O(\log m) = O(\log n)$. Furthermore, the size of ϕ is linear in the size of ψ. Thus, if there exists an algorithm which

can solve ∃∀SAT in time $2^{2^{o(vc(\phi))}}2^{o(|\phi|)}$ then, with the above reduction, this implies a $2^{o(n)}$ algorithm for 3-SAT. ◀

▶ **Corollary 2.** ∃∀SAT *is NP-hard when restricted to instances with incidence graphs of constant modular treewidth or clique-width.*

Proof. We observe that in the incidence graph of the instances of Theorem 1 the universal variables form a class of false twins. Contracting them to a single vertex results in a graph with constant treewidth. ◀

▶ **Theorem 3.** *There is no algorithm which, given an* ∃∀3-SAT *instance ϕ with n variables can decide if ϕ is True in time $2^{2^{o(tw_p(\phi))}}2^{o(n)}$, unless the ETH is false.*

Proof. We build on the proof of Theorem 1. Recall that in that reduction we started from a 3-SAT instance ψ with n variables and m clauses and constructed an ∃∀SAT instance ϕ with n existential variables, $\log m$ universal variables, such that all terms contain all universal variables and exactly one existential variable. We will edit this instance to make the size of each term at most 3, without affecting the value of ϕ and without increasing its primal treewidth.

We perform the following modification: as long as there exists a term t of ϕ with size greater than 3, we introduce to ϕ a new universally quantified variable z, remove t from ϕ and replace it with two new terms. The first contains z and two of the literals of t, while the second contains $\neg z$ and the remaining literals of t. We repeat this until all terms have size at most 3. It is not hard to see that this transformation does not affect the answer (we performed the standard reduction from SAT to 3-SAT). Furthermore, it is not hard to perform this transformation in such a way that every term of the resulting instance contains at most 2 of the new z variables, or at most one existential variable and one z variable.

Let us now examine the primal treewidth of the modified instance. As previously, we remove all the $\log m$ universal variables of ϕ. What remains is made up of the existential variables and the universal z variables added while breaking up large terms. We observe however, that all vertices corresponding to z variables have degree at most 2, because each appears in only two terms, and each term may only contain either at most one other z variable or one existential variable. Furthermore, among the z variables introduced while breaking up a term t, there must now exist one with degree one, since one of them appears in the same term as an existential variable. We now use the fact that deleting a leaf does not change the value of a graph's treewidth, and applying this rule repeatedly we can eventually delete all the z variables. This only leaves the existential variables in the primal graph, and these form an independent set. ◀

4 Algorithms

One surprising aspect of the lower bound given in Theorem 1 is that, if one takes into account known double-exponential algorithms for ∃∀SAT parameterized by treewidth [2], it indicates that ∃∀SAT has the same complexity parameterized by incidence treewidth, primal treewidth, or primal vertex cover. This is unexpected, because primal vertex cover is a significantly more restrictive measure than treewidth, and hence we would expect things to be significantly easier for this parameter. One indication in this direction is given by the fact that the lower bound for ∃∀3-SAT given in Theorem 3 does not apply to vertex cover.

In this section we give two algorithmic results that confirm that vertex cover is indeed a much more algorithmically amenable parameter. We first show in Theorem 4 that the

complexity of QBF is double-exponential in the formula's vertex cover *for any* number of quantifier alternations. This is in sharp contrast with the case of treewidth, where it is known that (under the ETH) the complexity of solving QBF rapidly degenerates into a tower of exponentials as the number of quantifier alternations increases [20]. Second, we show that there is a good reason why Theorem 3 cannot be extended to vertex cover, as the complexity of $\exists\forall d$-SAT for any fixed d is "only" single-exponential in the input formula's vertex cover raised to the d-th power.

Both of the algorithms we give are quite simple, and rely on the standard branching procedure which can decide QBF in exponential time (hence both algorithms only need polynomial space), together with the elementary observation that when working with a CNF formula we can always discard a clause that is a proper superset of another clause (or a term that is a superset of another term for DNFs). The key idea in both cases is to measure progress as a function of the number of clauses the formula contains that use only variables from the vertex cover. Let us also recall that the fixed-parameter tractability of QBF parameterized by vertex cover was already shown in [9] as a corollary of a more general algorithm using the notion of prefix pathwidth. However, the algorithm following from these results is triple-exponential in the input formula's vertex cover [18], while here we give an algorithm which is much simpler and whose running time is optimal (under Theorem 1).

▶ **Theorem 4.** *There is an algorithm that, given a* QBF *instance ϕ, decides ϕ in time* $O^*(2^{2^{O(vc(\phi))}})$.

Proof. We run the standard branching algorithm for quantified boolean formulas, where we branch on the variables of ϕ in the order that they appear quantified. We prove that the branching depth can be bounded by $3^k + k$, where k is the size of the primal vertex cover. We assume that we are given an optimal vertex cover of the primal graph (otherwise, we can find one in time 2^k).

Let V be the set of variables and $S \subseteq V$ be the variables corresponding to a vertex cover of the primal graph ($|S| = k$). Let further C' be a set containing all possible clauses that can be constructed using only variables from S. We observe that $|C'| \leq 3^k$, as each variable might appear positive, negative, or not appear in such a clause.

The essential reason that the standard branching algorithm works is that branching on a variable x either decreases the size of the vertex cover (if $x \in S$) or adds to the formula clauses of C' which are not already there. Indeed, since each clause of ϕ is represented by a clique in the primal graph, for every clause c of ϕ there can be at most one variable that doesn't belong in S, so branching on a variable that doesn't belong in S creates clauses in C'. The key observation now is that we should only branch on a variable $x \notin S$ if it is going to create clauses of C' that do not already belong in ϕ:

▶ Observation 1. If c, c' are clauses that both belong in ϕ and $c \supseteq c'$, then the formula ϕ' created by deleting c from ϕ is equivalent to ϕ.

Let us now describe the algorithm more formally. Let our quantified formula be $\phi = Q_1 x_1 Q_2 x_2 \dots Q_n x_n \psi$, where Q_i is the i^{th} quantifier. We define $\phi^T = \phi[x_1 = T]$ and $\phi^F = \phi[x_1 = F]$. In order to evaluate ϕ we shall first evaluate at least one of ϕ^T, ϕ^F (recursively) and take their disjunction if $Q_1 = \exists$ or conjunction if $Q_1 = \forall$. We consider the following cases.

If x_1 appears only positive (resp. negative) in ψ then there is no need to branch on x_1 as setting it to the suitable truth value depending on whether Q_1 is existential or universal simplifies the formula: if $Q_1 = \exists$ then $\phi \equiv \phi^T$ (resp. $\phi \equiv \phi^F$), whereas if $Q_1 = \forall$ then $\phi \equiv \phi^F$ (resp. $\phi \equiv \phi^T$).

On the other hand, if x_1 appears both positive and negative, we argue that we only branch if $x_1 \in S$ or if doing so creates at least one clause c' of C' that doesn't already belong in ψ. It is clear that the branching depth should be bounded by $3^k + k$.

Let us describe the branching for $Q_1 = \exists$ (the case $Q_1 = \forall$ is similar). Step 1 is to remove all the clauses $c \in \psi$ which are supersets of some other clause $c' \in \psi$ (we can do this because of Observation 1). Now, if x_1 appears positive (resp. negative) in some clause c of ψ, setting $x_1 = T$ (resp. $x_1 = F$) makes c true. In this case c doesn't appear in ϕ^T (resp. ϕ^F) at all. If x_1 appears negative (resp. positive) in c and we set $x_1 = T$ (resp. $x_1 = F$), then c is replaced in ϕ^T (resp. ϕ^F) by an equivalent clause c', where $c = c' \vee (\neg)x$.

Observe that none of ϕ^T, ϕ^F contains x_1, so if $x_1 \in S$ then the primal vertex cover of ψ is decreased by one. If $x_1 \notin S$, both branches ϕ^T, ϕ^F contain at least one positive and one negative appearance of x_1, thus they should each contain at least one clause $c' \in C' \setminus \psi$ (if $c = (\neg)x_1 \vee c'$ for some $c' \in C' \cap \psi$ then Step 1 removes c from ψ).

Since the branching depth is $3^k + k$, the algorithm will run in time $O^*(2^{3^k+k})$. ◀

▶ **Theorem 5.** *For all fixed $d \geq 3$ there exists an algorithm that decides an input $\exists\forall d$-SAT formula ϕ in time $O^*(2^{O(vc(\phi)^d)})$.*

Proof. The proof uses similar arguments as the proof of Theorem 4. There are two important observations that need to be added: first, during the course of the execution of the standard branching algorithm we never increase the size of any clause. Hence, if we started with an instance of $\exists\forall d$-SAT, we always maintain an instance of $\exists\forall d$-SAT. Second, the set C' that contains all terms that only use variables from the vertex cover now has size at most $O(k^d)$, where $k = vc(\phi)$. To see this, observe that there are at most $2^d\binom{k}{d}$ clauses of size exactly d that use only variables from the vertex cover. Hence, whenever the algorithm is forced to branch, it either decreases the vertex cover or increases the number of terms from C' in the formula, giving the promised running time. ◀

5 Tight Bounds for Weighted Problems

In this section we consider two variations of $\exists\forall$SAT that have recently attracted attention in the parameterized complexity community: $\exists_k\forall$SAT and $\exists\forall_k$SAT. Our main results are ETH-based lower bounds which provide evidence that the complexity of these two problems is quite different.

We first consider $\exists_k\forall$SAT. This is a problem that easily admits an algorithm running in time $n^k 2^{O(tw_i(\phi))}$: one could simply guess a weight k assignment for the existential variables and then solve the remaining (single-quantifier) instance with standard algorithms. The problem also admits an algorithm running in time (roughly) $2^{2^{tw_i(\phi)}}$, simply by adapting the algorithms given in [2] to only consider existential assignments of weight k. Our first result is that neither of these algorithms can be significantly improved: any algorithm that runs in time $n^{o(k)}$ must have complexity double-exponential in treewidth (in fact, more strongly, double-exponential in primal vertex cover). We also extend this result to $\exists_k\forall$3-SAT, as in Section 3.

We then move on to $\exists\forall_k$SAT. For this problem it is possible to adapt the algorithm of [2] to run in time (roughly) $2^{tw_i(\phi)^k}$. Informally, the reason for this is that the double-exponential running time in the algorithm of [2] comes from the need to consider all subsets of all possible assignments to universal variables in a bag. Here we only need to worry about assignments of weight (at most) k, hence there are at most $\binom{tw_i(\phi)}{k}$ of them to consider. We prove that obtaining a running time better than this would contradict the ETH, again even in

the case of vertex cover. Interestingly, we note that it is not immediate here to obtain similar bounds for $\exists\forall_k$3-SAT, because the standard method to break down terms by introducing universal variables does notnecessarily preserve the weight of universal assignments.

▶ **Theorem 6.** *There is no algorithm which, given an* $\exists_k\forall$SAT *instance* ϕ*, can decide if* ϕ *is True in time* $2^{2^{o(vc(\phi))}}|\phi|^{o(k)}$*, unless the ETH is false.*

Proof. We begin with a 3-SAT instance ψ with n variables and m clauses, and we first explain how to produce from this an equivalent $\exists_k$SAT instance ψ' with $k2^{n/k}$ variables and $m+k$ clauses, for any k. We partition the variables $\mathbf{x}$ into k sets $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}$ of n/k variables each (suppose without loss of generality that k divides n). Now, for each $i \in \{1, \ldots, k\}$, for each assignment of truth values to the variables of $\mathbf{x_i}$ we construct a variable that will appear in ψ'. Call the set of such variables $\mathbf{x'}$, and we have $|\mathbf{x'}| = k2^{n/k}$. For every clause C_i of ψ we construct a clause C'_i of ψ' which contains all the variables of $\mathbf{x'}$ that agree with at least one of the literals of C_i. Finally, for each $i \in \{1, \ldots, k\}$ we add a clause that contains all the variables that represent an assignment of $\mathbf{x_i}$. This completes the construction, and we observe that ψ' has $m+k$ clauses in total. It is not hard to see how a satisfying assignment of ψ can be transformed into a satisfying assignment of ψ' that sets exactly k variables to true: for each $\mathbf{x_i}$ we set to true the unique variable of $\mathbf{x'}$ that represents its partial assignment, and set all other variables of $\mathbf{x'}$ to false. The m clauses that were constructed from clauses of ψ are satisfied, because we started with a satisfying assignment of ψ, while the k remaining clauses are satisfied by definition. For the other direction, suppose that we have a satisfying assignment to ψ' which sets exactly k of the $\mathbf{x'}$ variables to True. The k additional clauses ensure that any satisfying assignment to ψ' that sets at most k variables to true must select exactly one partial assignment for each $\mathbf{x_i}$. We can therefore obtain an assignment to the variables of ψ from a weight-k satisfying assignment to ψ', and it is not hard to see that this assignment will satisfy ψ.

We now apply the construction of Theorem 1 to the weighted formula ψ'. Namely, we introduce a set $\mathbf{y}$ of $\log(m+k)$ universal variables (assume without loss of generality that $m+k$ is a power of two), and construct for each clause C'_i of ψ', for each of its literals l_j a term that contains l_j and the literals of $B(i, \mathbf{y})$. This completes the construction of the $\exists_k\forall$SAT instance ϕ. As in Theorem 1, there exists a satisfying assignment of weight k for ψ' if and only if the same assignment to the existential variables of ϕ makes the formular true for all assignments to the universal variables.

We observe that, if the original 3-SAT formula had n variables, then $|\phi| = 2^{O(n/k)}$ and $vc(\phi) = O(\log n)$. Thus, if there was an algorithm running in $2^{2^{o(vc(|\phi|)}}|\phi|^{o(k)}$ for $\exists_k\forall$SAT, then this would give a $2^{o(n)}$ algorithm for 3-SAT. ◀

▶ **Corollary 7.** *There is no algorithm which, given an* $\exists_k\forall$3-SAT *instance* ϕ*, can decide if* ϕ *is True in time* $2^{2^{o(tw_p(\phi))}}|\phi|^{o(k)}$*, unless the ETH is false.*

Proof. The proof is identical to that of Theorem 3, except we start with an instance produced by the reduction of Theorem 6. In particular, we again introduce a new universal variable for each term of size larger than 3 and use it to break it down into two terms. The arguments that we used to bound the primal treewidth in Theorem 3 also apply here. ◀

▶ **Theorem 8.** *There is no algorithm which, given an* $\exists\forall_k$SAT *instance* ϕ *with* n *variables, can decide if* ϕ *is True in time* $2^{vc(\phi)^{o(k)}}2^{o(n)}$*, unless the ETH is false.*

Proof. We begin with a 3-SAT formula ψ with n variables and m clauses. Our aim is to construct an $\exists\forall_k$SAT formula ϕ with $vc(\phi) = m^{O(1/k)}$ and $|\phi| = O(n+m)$. Let $M = m^{1/k}$, and we assume without loss of generality that M is an integer.

We construct ϕ as follows: let $\mathbf{x}$ be the set of variables of ψ; we retain these as the existential variables of ϕ. We also introduce k disjoint sets of universal variables, each of which contains exactly M distinct variables. We label these variables y_{j_1,j_2}, with $j_1 \in \{0,\ldots,k-1\}$ and $j_2 \in \{0,\ldots,M-1\}$.

Suppose that the clauses of ψ are numbered $C_0,\ldots,C_{m-1}$. For each clause C_i we do the following: for each of its literals l we construct a term in ϕ that contains l. Consider now the integer i written in base M; this representation of i contains k digits, each between 0 and $M-1$. We add to the term that contains l all variables y_{j_1,j_2} such that the j_1-th digit of i written in base M is j_2. We repeat this process for all literals of C_i, and clauses of ψ.

To complete the construction, we add to ψ k additional terms: for each $j_1 \in \{0,\ldots,k-1\}$ we add the term $(\wedge_{j_2=0}^{M-1} \neg y_{j_1,j_2})$.

Let us now argue for the correctness of the reduction. If ψ has a satisfying assignment, we use the same assignment to the existential variables of ϕ. We claim that any assignment of weight k to the universal variables must make a term true. To see this, observe that because of the k additional terms, we can assume that the assignment to the universal variables sets for each $j_1 \in \{0,\ldots,k-1\}$ exactly one $j_2 \in \{0,\ldots,M-1\}$ with $y_{j_1,j_2} = 1$. Consider now any assignment to the universal variables with this property. We can now find an integer i with the following property: when i is written in base M, for all $j_1 \in \{0,\ldots,k-1\}$, if the j_1-th digit of i has value j_2, then the assignment has set $y_{j_1,j_2} = 1$. The terms we constructed for clause C_i have all their universal variables set to true, and each contains one literal from C_i, hence at least one of these terms is true. It is not hard to see that the converse direction follows with a similar reasoning.

For the running time bound, we note that the primal graph of ϕ becomes an independent set if we remove the $km^{1/k}$ universal variables, hence $vc(\phi) = O(m^{1/k})$ and the lower bound follows. ◀

6 Non-binary ∃∀CSP

In this section we consider the more general ∃∀CSP problem. The difference between this problem and ∃∀SAT is that in ∃∀CSP the variables don't necessarily have a boolean domain, but may take values from some finite set Σ. As a result, that size of Σ must be factored into the complexity. The algorithm of [2] runs in time (roughly) $2^{|\Sigma|^{tw_i(I)}}$, for an ∃∀CSP instance I, because it considers all possible subsets of all possible $|\Sigma|^{tw_i(I)}$ assignments to the variables of a bag. In other words, the second exponent is linear in $tw(I)\log|\Sigma|$. Our main result is that this dependence is optimal, even for the more restricted case of vertex cover, unless the ETH is false.

▶ **Theorem 9.** *There is no algorithm which, given an* ∃∀CSP *instance I with n variables and domain Σ, can decide if I is a Yes instance in time $2^{|\Sigma|^{o(vc(I))}}|\Sigma|^{o(n)}$, unless the ETH is false.*

Proof. We give a reduction in two steps, beginning from a 3-SAT formula ψ. First, we construct from ψ a CSP instance I_1 with alphabet Σ, and show that this instance cannot be solved in time $|\Sigma|^{o(n)}$, where n is the number of variables of I_1. We then construct from I_1 an instance I_2 of ∃∀CSP for which we obtain the promised lower bound.

Suppose that ψ has n_1 variables and m_1 clauses. We partition its clauses into $n = \lceil m_1/\log_7|\Sigma|\rceil$ groups, each containing at most $\log_7|\Sigma|$ clauses. For each one of these groups we construct a variable in I_1. Consider now a group of clauses and list all assignments of the variables that appear in this group and satisfy all clauses of the group. There are at

most $7^{\log_7 |\Sigma|}$ such assignments, since each clause has size at most three, and therefore at most 7 satisfying assignments. As a result, we can make an injective mapping to values in the domain Σ for the variable that represents this group of clauses in I_1, from the set of partial assignments that satisfy all the clauses. We now add some constraints to our CSP instance to ensure that the assignment to ψ must be consistent. In particular, let x_1, x_2 be two variables of I that represents groups of clauses of ψ that share some variables. We add a constraint to I which only allows the variables x_1, x_2 to receive a pair of values from Σ such that the corresponding partial assignments to the variables of ψ is consistent and satisfies all the clauses of the two groups. This completes the construction. Since the new instance has $n = O(m_1/\log|\Sigma|)$ variables, if there was an algorithm solving CSP in time $|\Sigma|^{o(n)}$ the ETH would be false.

Let us now use I_1 to produce an instance I_2 of $\exists\forall$CSP. We note that I_1 has n variables and therefore, since all constraints have arity two, at most $m = O(n^2)$ constraints. We retain all variables of I_1 as existential variables, and introduce a set of $\lceil \log(m|\Sigma|^2)/\log|\Sigma| \rceil$ universal variables $\mathbf{y}$. The main observation now is that the total number of distinct assignments to $\mathbf{y}$ is at least $m|\Sigma|^2$. We can therefore define an injective mapping which, given a constraint of I_1 and an assignment to the variables of this constraint that *falsifies* the constraint, produces an assignment to the variables of $\mathbf{y}$. We now define the constraints of I_2 as follows: let x_1, x_2 be two variables involved in a constraint C of I_1 and let (v_1, v_2) be an assignment to x_1, x_2 that is not allowed by the constraint. Let $\mathbf{v}$ be the assignment to v to which we have mapped the constraint C and its falsifying assignment (v_1, v_2). We add to I_2 the two following constraints: $(x_1 \neq v_1 \wedge \mathbf{y} = \mathbf{v})$ and $(x_2 \neq v_2 \wedge \mathbf{y} = \mathbf{v})$. We perform this step for every falsifying assignment of every constraint. To complete the construction, for every assignment $\mathbf{v}$ to $\mathbf{y}$ that is not mapped to, we add the constraint $(\mathbf{y} = \mathbf{v})$. We finally note that, because every constraint involves at most $\lceil \log(m|\Sigma|^2)/\log|\Sigma| \rceil + 1$ variables, all constraints can be explicitly described in polynomial time by listing all of their at most $O(m|\Sigma|^2)$ satisfying assignments.

Let us now argue for correctness. If there is a satisfying assignment to I_1, we select the same assignment for the existential variables of I_2. If the universal variables receive some assignment that is not mapped to by a constraint of I_1, then the new instance has a satisfied constraint, because of the constraints of the form $(\mathbf{y} = \mathbf{v})$. Otherwise, the assignment to $\mathbf{y}$ corresponds to a falsifying assignment (v_1, v_2) to a constraint C of I_1. However, since we started with a satisfying assignment to I_1, either $x_1 \neq v_1$ or $x_2 \neq v_2$, so again at least one constraint is satisfied. It is not hard to see that the converse direction follows with similar arguments.

For the running time lower bound, we note that the primal graph of I_2 has vertex cover at most $|\mathbf{y}| = O(\log m/\log|\Sigma|)$, from which the bound follows. ◀

References

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

2 H. Chen. Quantified constraint satisfaction and bounded treewidth. In *ECAI*, pages 161–165. IOS Press, 2004.

3 B. Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

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

5 R. de Haan and S. Szeider. Compendium of parameterized problems at higher levels of the polynomial hierarchy. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:143, 2014.

6 R. de Haan and S. Szeider. Fixed-parameter tractable reductions to SAT. In *SAT*, volume 8561 of *Lecture Notes in Computer Science*, pages 85–102. Springer, 2014.

7 R. de Haan and S. Szeider. Parameterized complexity classes beyond para-np. *J. Comput. Syst. Sci.*, 87:16–57, 2017.

8 H. Dell, E. J. Kim, M. Lampis, V. Mitsou, and T. Mömke. Complexity and approximability of parameterized max-csps. In *IPEC*, volume 43 of *LIPIcs*, pages 294–306. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

9 E. Eiben, R. Ganian, and S. Ordyniak. Using decomposition-parameters for QBF: mind the prefix! In *AAAI*, pages 964–970. AAAI Press, 2016.

10 M. Frick and M. Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic*, 130(1-3):3–31, 2004.

11 R. Ganian. Twin-cover: Beyond vertex cover in parameterized algorithmics. In *IPEC*, volume 7112 of *Lecture Notes in Computer Science*, pages 259–271. Springer, 2011.

12 R. Ganian, P. Hlinený, J. Nesetril, J. Obdrzálek, P. O. de Mendez, and R. Ramadurai. When trees grow low: Shrubs and fast MSO1. In *MFCS*, volume 7464 of *Lecture Notes in Computer Science*, pages 419–430. Springer, 2012.

13 R. Impagliazzo and R. Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

14 R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

15 M. Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012.

16 M. Lampis. Model checking lower bounds for simple graphs. *Logical Methods in Computer Science*, 10(1), 2014.

17 D. Marx and V. Mitsou. Double-exponential and triple-exponential bounds for choosability problems parameterized by treewidth. In *ICALP*, volume 55 of *LIPIcs*, pages 28:1–28:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

18 S. Ordyniak. Private communication, 2017.

19 S. Ordyniak, D. Paulusma, and S. Szeider. Satisfiability of acyclic and almost acyclic CNF formulas. *Theor. Comput. Sci.*, 481:85–99, 2013.

20 G. Pan and M. Y. Vardi. Fixed-parameter hierarchies inside PSPACE. In *LICS*, pages 27–36. IEEE Computer Society, 2006.

21 D. Paulusma, F. Slivovsky, and S. Szeider. Model counting for CNF formulas of bounded modular treewidth. *Algorithmica*, 76(1):168–194, 2016.

22 D. Paulusma, F. Slivovsky, and S. Szeider. Model counting for CNF formulas of bounded modular treewidth. *Algorithmica*, 76(1):168–194, 2016.

23 S. H. Sæther, J. A. Telle, and M. Vatshelle. Solving #sat and MAXSAT by dynamic programming. *J. Artif. Intell. Res.*, 54:59–82, 2015.

24 M. Samer and S. Szeider. Algorithms for propositional model counting. *J. Discrete Algorithms*, 8(1):50–64, 2010.

25 M. Samer and S. Szeider. Constraint satisfaction with bounded treewidth revisited. *J. Comput. Syst. Sci.*, 76(2):103–114, 2010.

26 L. J. Stockmeyer. The polynomial-time hierarchy. *Theor. Comput. Sci.*, 3(1):1–22, 1976.