Strong Backdoors to Bounded Treewidth SAT

Serge Gaspers[1] and Stefan Szeider[2]

[1]The University of New South Wales and NICTA, Sydney, Australia.
sergeg@cse.unsw.edu.au
[2]Institute of Information Systems, Vienna University of Technology, Vienna, Austria.
stefan@szeider.net

Abstract

There are various approaches to exploiting "hidden structure" in instances of hard combinatorial problems to allow faster algorithms than for general unstructured or random instances. For SAT and its counting version #SAT, hidden structure has been exploited in terms of decomposability and strong backdoor sets. Decomposability can be considered in terms of the treewidth of a graph that is associated with the given CNF formula, for instance by considering clauses and variables as vertices of the graph, and making a variable adjacent with all the clauses it appears in. On the other hand, a strong backdoor set of a CNF formula is a set of variables such that each assignment to this set moves the formula into a fixed class for which (#)SAT can be solved in polynomial time.

In this paper we combine the two above approaches. In particular, we study the algorithmic question of finding a small strong backdoor set into the class $\mathcal{W}_{\leq t}$ of CNF formulas whose associated graphs have treewidth at most t. The main results are positive:

(1) There is a cubic-time algorithm that, given a CNF formula F and two constants $k, t \geq 0$, either finds a strong $\mathcal{W}_{\leq t}$-backdoor set of size at most 2^k, or concludes that F has no strong $\mathcal{W}_{\leq t}$-backdoor set of size at most k.

(2) There is a cubic-time algorithm that, given a CNF formula F, computes the number of satisfying assignments of F or concludes that $\mathsf{sb}_t(F) > k$, for any pair of constants $k, t \geq 0$. Here, $\mathsf{sb}_t(F)$ denotes the size of a smallest strong $\mathcal{W}_{\leq t}$-backdoor set of F.

We establish both results by distinguishing between two cases, depending on whether the treewidth of the given formula is small or large. For both results the case of small treewidth can be dealt with relatively standard methods. The case of large treewidth is challenging and requires novel and sophisticated combinatorial arguments. The main tool is an auxiliary graph whose vertices represent subgraphs in F's associated graph. It captures various ways to assemble large-treewidth subgraphs in F's associated graph. This is used to show that every backdoor set of size k intersects a certain set of variables whose size is bounded by a function of k and t. For any other set of k variables, one can use the auxiliary graph to find an assignment τ to these variables such that the graph associated with $F[\tau]$ has treewidth at least $t + 1$.

The significance of our results lies in the fact that they allow us to exploit algorithmically a hidden structure in formulas that is not accessible by any one of the two approaches (decomposability, backdoors) alone. Already a backdoor size 1 on top of treewidth 1 (i.e., $\mathsf{sb}_1(F) = 1$) entails formulas of arbitrarily large treewidth and arbitrarily large cycle cutsets (variables whose deletion makes the instance acyclic).

Keywords: algorithms, #SAT, parameterized complexity, graph minors

1 Introduction

Background. Satisfiability (SAT) is probably one of the most important NP-complete problems [8, 26]. Despite the theoretical intractability of SAT, heuristic algorithms work surprisingly fast on real-world SAT instances. A common explanation for this discrepancy between theoretical hardness and practical feasibility is the presence of a certain "hidden structure" in industrial SAT instances [20, 22]. There are various approaches to capturing the vague notion of a "hidden structure" with a mathematical concept.

One widely studied approach is to consider the hidden structure in terms of *decomposability*. The basic idea is to decompose a SAT instance into small parts that can be solved individually, and to put solutions for the parts together to a global solution. The overall complexity depends only on the maximum overlap of the parts, the *width* of the decomposition. Treewidth and branchwidth are two decomposition width measures (related by a constant factor) that have been applied to satisfiability. The width measures are either applied to the *primal graph* of the formula (variables are vertices, two variables are adjacent if they appear together in a clause) or to the *incidence graph* (a bipartite graph on the variables and clauses, a clause is incident to all the variables it contains). If the treewidth or branchwidth of any of the two graphs is bounded, then SAT can be decided in polynomial time; in fact, one can even count the number of satisfying assignments in polynomial time. This result has been obtained in various contexts, e.g., resolution complexity [2] and Bayesian Inference [5] (branchwidth of primal graphs), and Model Checking for Monadic Second-Order Logic [14] (treewidth of incidence graphs).

A complementary approach is to consider the hidden structure of a SAT instance in terms of a small set of key variables, called *backdoor set*, that when instantiated moves the instance into a polynomial class. More precisely, a *strong* backdoor set of a CNF formula F into a polynomially solvable class $\mathcal{C}$ (or strong $\mathcal{C}$-backdoor set, for short) is a set B of variables such that for all partial assignments τ to B, the reduced formula $F[\tau]$ belongs to $\mathcal{C}$ (*weak* backdoor sets apply only to satisfiable formulas and will not be considered in this paper). Backdoor sets where introduced by Williams *et al.* [43] to explain favorable running times and the heavy-tailed behavior of SAT and CSP solvers on practical instances. In fact, real-world instances tend to have small backdoor sets (see [27] and references). Of special interest are base classes for which we can find a small backdoor set efficiently, if one exists. This is the case, for instance, for the base classes based on the tractable cases in Schaefer's dichotomy theorem [39]. In fact, for any constant b one can decide in linear time whether a given CNF formula admits a backdoor set of size b into any Schaefer class [18].

Contribution. In this paper we combine the two above approaches. Namely, we study the algorithmic question of finding a small strong backdoor set into a class of formulas of bounded treewidth. Let $\mathcal{W}_{\leq t}$ denote the class of CNF formulas whose incidence graph has treewidth at most t. Since SAT and #SAT can be solved in linear time for formulas in $\mathcal{W}_{\leq t}$ [14, 38], we can also solve these problems efficiently for a formula F if we know a strong $\mathcal{W}_{\leq t}$-backdoor set of F of small size k.

However, finding a small strong backdoor set into the class $\mathcal{W}_{\leq t}$ is a challenging problem. What makes the problem difficult is that applying partial assignments to variables is a much more powerful operation than just deleting the variables from the formula, as setting a variable to true may remove a large set of clauses, setting it to false removes a different set of clauses, and for a strong backdoor set B we must ensure that for all the $2^{|B|}$ possible assignments the resulting formula is in $\mathcal{W}_{\leq t}$. The brute force algorithm tries out all possible sets B of at most k variables, and checks for each set whether all the $2^{|B|}$ reduced formulas belong to $\mathcal{W}_{\leq t}$. The number of membership checks is of order $2^k n^k$ for an input formula with n variables. This number is polynomial for constant k, but the order of the polynomial depends on the backdoor size k. Is it possible to get k out of the exponent and to have the same polynomial for every fixed k and t? Our main result provides an affirmative answer to this question. We show the following.

Theorem 1. *There is a cubic-time algorithm that, given a CNF formula F and two constants $k, t \geq 0$, either finds a strong $\mathcal{W}_{\leq t}$-backdoor set of size at most 2^k, or concludes that F has no strong $\mathcal{W}_{\leq t}$-backdoor set of size at most k.*

Our algorithm distinguishes for a given CNF formula between two cases: (A) the formula has small treewidth, or (B) the formula has large treewidth. Case A we use model checking for monadic second order logic [3] to find a smallest backdoor set. Roberson and Seymour's theory of graph minors [34] guarantees a finite set of forbidden minors for every minor-closed class of graphs. Although their proof is non-constructive, for the special case of bounded treewidth graphs the forbidden minors can be computed in constant time [1, 25]. These forbidden minors are used in our monadic second order sentence to describe a strong backdoor set to the base class $\mathcal{W}_{\leq t}$. A model checking algorithm [3] then computes a strong $\mathcal{W}_{\leq t}$-backdoor set of size k if one exists.

Whereas Case A is relatively standard, completely new combinatorial methods are needed in Case B. First, we compute many vertex-disjoint obstructions in the incidence graph, so-called wall-obstructions, whose existence is guaranteed by a theorem by Robertson and Seymour [36]. A backdoor set needs to "kill" all these obstructions, where an obstruction is killed either internally because it contains a backdoor variable, or externally because it contains two clauses containing the same backdoor variable with opposite signs. We exhibit an extremely helpful combinatorial beast, combining one so-called obstruction-template for each wall-obstruction. We prove that, once we have found the beast, we can easily derive a small set of variables intersecting each strong $\mathcal{W}_{\leq t}$-backdoor set of size at most k. Fortunately, we can actually compute the beast in quadratic time. The obstruction-template is a bipartite graph with external killers on one side and vertices representing vertex-disjoint connected subgraphs of a wall-obstruction on the other side of the bipartition. It is used to guarantee that for sets of k variables excluding a bounded set S^*, every assignment to these k variables produces a formula whose incidence graph has treewidth at least $t+1$. Thus, the algorithm can use S^* to guide its branching strategy.

Combining both cases leads to an algorithm producing a strong $\mathcal{W}_{\leq t}$-backdoor set of a given formula F of size at most 2^k if F has a strong $\mathcal{W}_{\leq t}$-backdoor set of size k.

For our main applications of Theorem 1, the problems SAT and #SAT, we can solve Case A actually without recurring to the list of forbidden minors of bounded treewidth graphs and to model checking for monadic second order logic. Namely, when the formula has small treewidth, we can directly apply one of the known linear-time algorithms to count the number of satisfying truth assignments [14, 38], thus avoiding the issue of finding a backdoor set.

We arrive at the following statement where $\mathsf{sb}_t(F)$ denotes the size of a smallest strong $\mathcal{W}_{\leq t}$-backdoor set of a formula F.

Theorem 2. *There is a cubic-time algorithm that, given a CNF formula F, computes the number of satisfying assignments of F or concludes that $\mathsf{sb}_t(F) > k$, for any pair of constants $k, t \geq 0$.*

This is a *robust* algorithm in the sense of [40] since for every instance, it either solves the problem (SAT, #SAT) or concludes that the instance is not in the class of instances that need to be solved (the CNF formulas F with $\mathsf{sb}_t(F) \leq k$). In general, a robust algorithm solves the problem on a superclass of those instances that it guarantees to solve, and it does not necessarily check whether the given instance is in this class.

Theorem 2 applies to formulas of *arbitrarily large treewidth*. We would like to illustrate this with the following example. Take a CNF formula F_n whose incidence graph is obtained from an

$n \times n$ square grid containing all the variables of F_n by subdividing each edge by a clause of F_n. It is well-known that the $n \times n$ grid, $n \geq 2$, has treewidth n and that a subdivision of an edge does not decrease the treewidth. Hence $F_n \notin \mathcal{W}_{\leq n-1}$. Now take a new variable x and add it positively to all horizontal clauses and negatively to all vertical clauses. Here, a *horizontal* (respectively, a *vertical*) clause is one that subdivides a horizontal (respectively, a vertical) edge in the natural layout of the grid. Let F_n^x denote the new formula. Since the incidence graph of F_n is a subgraph of the incidence graph of F_n^x, we have $F_n^x \notin \mathcal{W}_{\leq n-1}$. However, setting x to true removes all horizontal clauses and thus yields a formula whose incidence graph is acyclic, hence $F_n^x[x=1] \in \mathcal{W}_{\leq 1}$. Similarly, setting x to false yields a formula $F_n^x[x=0] \in \mathcal{W}_{\leq 1}$. Hence $\{x\}$ is a strong $\mathcal{W}_{\leq 1}$-backdoor set of F_n^x. Conversely, it is easy to construct, for every $t \geq 0$, formulas that belong to $\mathcal{W}_{\leq t+1}$ but require arbitrarily large strong $\mathcal{W}_{\leq t}$-backdoor sets.

One can also define a *deletion $\mathcal{C}$-backdoor set B* of a CNF formula F by requiring that deleting all literals $x, \neg x$ with $x \in B$ from F produces a formula that belongs to the base class [31]. For many base classes it holds that every deletion backdoor set is a strong backdoor set, but in most cases, including the base class $\mathcal{W}_{\leq t}$, the reverse is not true. In fact, it is easy to see that if a CNF formula F has a deletion $\mathcal{W}_{\leq t}$-backdoor set of size k, then $F \in \mathcal{W}_{t+k}$. In other words, the parameter "size of a smallest deletion $\mathcal{W}_{\leq t}$-backdoor set" is dominated by the parameter "treewidth of the incidence graph" and therefore of limited theoretical interest, except for reducing the space requirements of dynamic programming procedures [6] and analyzing the effectiveness of polynomial time preprocessing [10, 23].

A common approach to solve #SAT is to find a small *cycle cutset* (or feedback vertex set) of variables of the given CNF formula, and by summing up the number of satisfying assignments of all the acyclic instances one gets by setting the cutset variables in all possible ways [11]. We would like to note that such a cycle cutset is nothing but a deletion $\mathcal{W}_{\leq 1}$-backdoor set. By considering strong $\mathcal{W}_{\leq 1}$-backdoor sets instead, one can get super-exponentially smaller backdoor sets, and hence a more powerful method. A strong $\mathcal{W}_{\leq 1}$-backdoor set can be considered as a an *implied cycle cutset* as it cuts cycles by removing clauses that are satisfied by certain truth assignments to the backdoor variables. By increasing the treewidth bound from 1 to some fixed $t > 1$ one can further dramatically decrease the size of a smallest backdoor set.

Our results can also be phrased in terms of *Parameterized Complexity* [13, 15]. Theorem 2 states that #SAT is uniformly fixed-parameter tractable (FPT) for parameter (t, sb_t). Theorem 1 states that there is a uniform FPT-approximation algorithm for the detection of strong $\mathcal{W}_{\leq t}$-backdoor sets of size k, for parameter (t, k), as it is a fixed-parameter algorithm that computes a solution that approximates the optimum with an error bounded by a function of the parameter [29].

Related work. Williams *et al.* [43] introduced the notion of backdoor sets and the parameterized complexity of finding small backdoor sets was initiated by Nishimura *et al.* [30]. They showed that with respect to the classes of Horn formulas and of 2CNF formulas, the detection of strong backdoor sets is fixed-parameter tractable. Their algorithms exploit the fact that for these two base classes strong and deletion backdoor sets coincide. For other base classes, deleting literals is a less powerful operation than applying partial truth assignments. This is the case, for instance, for RHorn, the class of renamable Horn formulas. In fact, finding a deletion RHorn-backdoor set is fixed-parameter tractable [32], but it is open whether this is the case for the detection of strong RHorn-backdoor sets. For the base class q-Horn, which generalizes RHornand 2CNF, an FPT-approximation algorithm is known for the detection of deletion backdoor sets [16]. For clustering formulas, detection of deletion backdoor sets is fixed-parameter tractable, detection of strong backdoor sets is most probably not [31]. Very recently, the authors of the present paper showed [17, 19] that there are FPT-approximation algorithms for the detection of strong backdoor sets with respect to (i) the base class of formulas with acyclic incidence graphs, i.e., $\mathcal{W}_{\leq 1}$, and (ii) the base class of nested formulas (a proper subclass of $\mathcal{W}_{\leq 3}$ introduced by Knuth [24]). The present paper generalizes this approach to base classes of bounded treewidth which requires new ideas and significantly more involved combinatorial arguments.

We conclude this section by referring to a recent survey on the parameterized complexity of backdoor sets [18].

2 Preliminaries

Graphs. Let G be a simple, undirected, finite graph with vertex set $V = V(G)$ and edge set $E = E(G)$. Let $S \subseteq V$ be a subset of its vertices and $v \in V$ be a vertex. We denote by $G - S$ the graph obtained from G by removing all vertices in S and all edges incident to vertices in S. We denote by $G[S]$ the graph $G - (V \setminus S)$. The *(open) neighborhood* of v in G is $N_G(v) = \{u \in V : uv \in E\}$, the *(open) neighborhood* of S in G is $N_G(S) = \bigcup_{u \in S} N_G(u) \setminus S$, and their *closed neighborhoods* are $N_G[v] = N_G(v) \cup \{v\}$ and $N_G[S] = N_G(S) \cup S$, respectively. The *degree* of v in G is $d_G(v) = |N_G(v)|$. Subscripts may be omitted if the graph is clear from the context.

A *tree decomposition* of G is a pair $(\{X_i : i \in I\}, T)$ where $X_i \subseteq V$, $i \in I$, and T is a tree with elements of I as nodes such that:

1. for each edge $uv \in E$, there is an $i \in I$ such that $\{u, v\} \subseteq X_i$, and
2. for each vertex $v \in V$, $T[\{i \in I : v \in X_i\}]$ is a (connected) tree with at least one node.

The *width* of a tree decomposition is $\max_{i \in I} |X_i| - 1$. The *treewidth* [35] of G is the minimum width taken over all tree decompositions of G and it is denoted by $\mathsf{tw}(G)$.

For other standard graph-theoretic notions not defined here, we refer to [12].

CNF formulas and satisfiability. We consider propositional formulas in conjunctive normal form (CNF) where no clause contains a complementary pair of literals. For a clause c, we write $\mathsf{lit}(c)$ and $\mathsf{var}(c)$ for the sets of literals and variables occurring in c, respectively. For a CNF formula F we write $\mathsf{cla}(F)$ for its set of clauses, $\mathsf{lit}(F) = \bigcup_{c \in \mathsf{cla}(F)} \mathsf{lit}(c)$ for its set of literals, and $\mathsf{var}(F) = \bigcup_{c \in \mathsf{cla}(F)} \mathsf{var}(c)$ for its set of variables. The *size* of F is $|F| = |\mathsf{var}(F)| + \sum_{c \in \mathsf{cla}(F)} (1 + |\mathsf{lit}(c)|)$.

For a set $X \subseteq \mathsf{var}(F)$ we denote by 2^X the set of all mappings $\tau : X \to \{0, 1\}$, the *truth assignments* on X. A truth assignment $\tau \in 2^X$ can be extended to the literals over X by setting $\tau(\neg x) = 1 - \tau(x)$ for all $x \in X$. The formula $F[\tau]$ is obtained from F by removing all clauses c such that τ sets a literal of c to 1, and removing the literals set to 0 from all remaining clauses.

A CNF formula F is *satisfiable* if there is some $\tau \in 2^{\mathsf{var}(F)}$ with $\mathsf{cla}(F[\tau]) = \emptyset$. SAT is the NP-complete problem of deciding whether a given CNF formula is satisfiable [8, 26]. #SAT is the #P-complete problem of determining $\#(F)$, the number of distinct $\tau \in 2^{\mathsf{var}(F)}$ with $\mathsf{cla}(F[\tau]) = \emptyset$ [42].

Formulas with bounded incidence treewidth. The *incidence graph* of a CNF formula F is the bipartite graph $\mathsf{inc}(F) = (V, E)$ with $V = \mathsf{var}(F) \cup \mathsf{cla}(F)$ and for a variable $x \in \mathsf{var}(F)$ and a clause $c \in \mathsf{cla}(F)$ we have $xc \in E$ if $x \in \mathsf{var}(c)$. The *sign* of the edge xc is *positive* if $x \in \mathsf{lit}(c)$ and *negative* if $\neg x \in \mathsf{lit}(c)$. Note that $|V| + |E| = |F|$.

The class $\mathcal{W}_{\leq t}$ contains all CNF formulas F with $\mathsf{tw}(\mathsf{inc}(F)) \leq t$. For any fixed $t \geq 0$ and any CNF formula $F \in \mathcal{W}_{\leq t}$, a tree decomposition of $\mathsf{inc}(F)$ of width at most t can be found by

Bodlaender's algorithm [7] in time $O(|F|)$. Given a tree decomposition of width at most t of $\mathsf{inc}(F)$, the number of satisfying assignments of F can be determined in time $O(|F|)$ [14, 38].

Finally, note that, if $\tau \in 2^X$ is a partial truth assignment for a CNF formula F, then $\mathsf{inc}(F[\tau])$ is an induced subgraph of $\mathsf{inc}(F)$, namely $\mathsf{inc}(F[\tau])$ is obtained from $\mathsf{inc}(F) - X$ by removing each vertex corresponding to a clause that contains a literal ℓ with $\tau(\ell) = 1$.

Backdoors. Backdoor sets are defined with respect to a fixed class $\mathcal{C}$ of CNF formulas, the *base class*. Let F be a CNF formula and $B \subseteq \mathsf{var}(F)$. The set B is a *strong $\mathcal{C}$-backdoor set* of F if $F[\tau] \in \mathcal{C}$ for each $\tau \in 2^B$. The set B is a *deletion $\mathcal{C}$-backdoor set* of F if $F - B \in \mathcal{C}$, where $F - B$ is obtained from F by removing all literals in $\{x, \neg x : x \in B\}$ from its clauses.

If we are given a strong $\mathcal{C}$-backdoor set of F of size k, we can reduce the satisfiability of F to the satisfiability of 2^k formulas in $\mathcal{C}$. If $\mathcal{C}$ is clause-induced (i.e., $F \in \mathcal{C}$ implies $F' \in \mathcal{C}$ for every CNF formula F' with $\mathsf{cla}(F') \subseteq \mathsf{cla}(F)$), any deletion $\mathcal{C}$-backdoor set of F is a strong $\mathcal{C}$-backdoor set of F. The interest in deletion backdoor sets is motivated for base classes where they are easier to detect than strong backdoor sets. The challenging problem is to find a strong or deletion $\mathcal{C}$-backdoor set of size at most k if it exists. Denote by $\mathsf{sb}_t(F)$ the size of a smallest strong $\mathcal{W}_{\leq t}$-backdoor set.

Graph minors. The operation of *merging* a subgraph H or a vertex subset $V(H)$ of a graph G into a vertex v produces the graph G' such that $G' - \{v\} = G - V(H)$ and $N_{G'}(v) = N_G(H)$. The *contraction* operation merges a connected subgraph. The *dissolution* operation contracts an edge incident to a vertex of degree 2.

A graph H is a *minor* of a graph G if H can be obtained from a subgraph of G by contractions. If H is a minor of G, then one can find a model of H in G. A *model* of H in G is a set of vertex-disjoint connected subgraphs of G, one subgraph C_u for each vertex u of H, such that if uv is an edge in H, then there is an edge in G with one endpoint in C_u and the other in C_v.

A graph H is a *topological minor* of a graph G if H can be obtained from a subgraph of G by dissolutions. If H is a topological minor of G, then G has a topological model of H. A *topological model* of H in G is a subgraph of G that can be obtained from H by replacing its edges by independent paths. A set of paths is *independent* if none of them contains an interior vertex of another. We also say that G contains a *subdivision* of H as a subgraph.

Obstructions to small treewidth. It is well-known that $\mathsf{tw}(G) \geq \mathsf{tw}(H)$ if H is a minor of G. We will use the following three (classes of) graphs to lower bound the treewidth of a graph containing any of them as a minor. See Figure 1. The complete graph K_r has treewidth $r-1$. The complete bipartite graph $K_{r,r}$ has treewidth r. The *r-wall* is the graph $W_r = (V, E)$ with vertex set $V = \{(i,j) : 1 \leq i \leq r, 1 \leq j \leq r\}$ in which two vertices (i,j) and (i',j') are adjacent iff either $j' = j$ and $i' \in \{i-1, i+1\}$, or $i' = i$ and $j' = j + (-1)^{i+j}$. We say that a vertex $(i,j) \in V$ has horizontal index i and vertical index j. The r-wall has treewidth at least $\lfloor \frac{r}{2} \rfloor$ (it is a minor of the $\lfloor \frac{r}{2} \rfloor \times r$-grid, which has treewidth $\lfloor \frac{r}{2} \rfloor$ [37]).

We will also need to find a large wall as a topological minor if the formula has large incidence treewidth. Its existence is guaranteed by a theorem of Robertson and Seymour.

Theorem 3 ([36])**.** *For every positive integer r, there exists a constant $f(r)$ such that if a graph G has treewidth at least $f(r)$, then G contains an r-wall as a topological minor.*

By [33], $f(r) \leq 20^{64r^5}$. For any fixed r, we can use the cubic algorithm by Grohe *et al.* [21] to find a topological model of an r-wall in a graph G if G contains an r-wall as a topological minor.

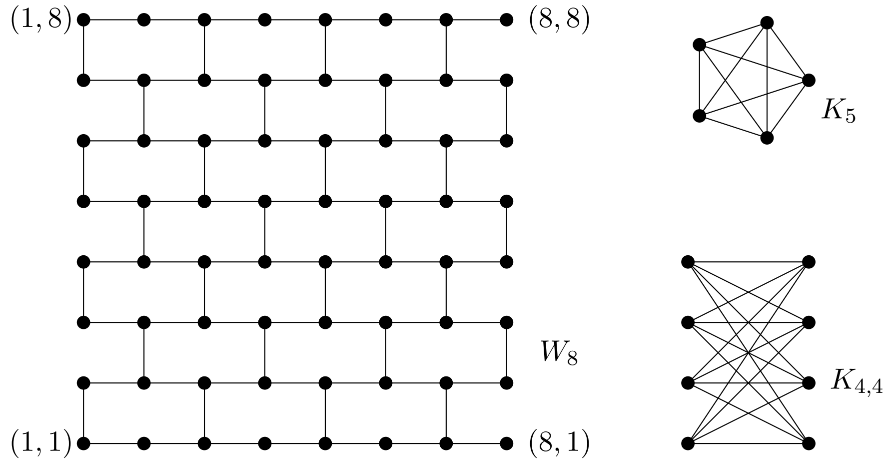

Figure 1: Some graphs with treewidth 4.

3 The algorithms

We start with the overall outline of our algorithms. We rely on the following two lemmas whose proofs we defer to the next two subsections.

Lemma 1. *There is a quadratic-time algorithm that, given a CNF formula F, two constants $t \geq 0$, $k \geq 1$, and a topological model of a $\mathsf{wall}(k,t)$-wall in $\mathsf{inc}(F)$, computes a set $S^* \subseteq \mathsf{var}(F)$ of constant size such that every strong $\mathcal{W}_{\leq t}$-backdoor set of size at most k contains a variable from S^*.*

Lemma 2. *There is a linear-time algorithm that, given a CNF formula F, a constant $t \geq 0$, and a tree decomposition of $\mathsf{inc}(F)$ of constant width, computes a smallest strong $\mathcal{W}_{\leq t}$-backdoor set of F.*

Lemma 2 will be invoked with a tree decomposition of $\mathsf{inc}(F)$ of width at most $\mathsf{tw}(k,t)$. The functions $\mathsf{wall}(k,t)$ and $\mathsf{tw}(k,t)$ are related by the bound from [33], implying that every graph either has treewidth at most $\mathsf{tw}(k,t)$, or it has a $\mathsf{wall}(k,t)$-wall as a topological minor. Here,

$$\begin{aligned}
\mathsf{tw}(k,t) &:= 20^{64\cdot(\mathsf{wall}(k,t))^5},\\
\mathsf{wall}(k,t) &:= (2t+2)\cdot(1+\sqrt{\mathsf{obs}(k,t)}),\\
\mathsf{obs}(k,t) &:= 2^k\cdot\mathsf{same}(k,t)+k,\\
\mathsf{same}(k,t) &:= 3(\mathsf{nb}(t))^2 t 2^{2k}, \text{ and}\\
\mathsf{nb}(t) &:= \lceil 16(t+2)\log(t+2)\rceil.
\end{aligned}$$

The other functions of k and t will be used in Subsection 3.1.

Theorem 1 can now be proved as follows.

Proof of Theorem 1. Let $t, k \geq 0$ be constants, let F be the given CNF formula, with $|F| = n$ and let $G := \mathsf{inc}(F)$. Using Bodlaender's algorithm [7] we can decide in linear time whether $\mathsf{tw}(G) \leq \mathsf{tw}(k,t)$, and if so, compute a tree decomposition of G of smallest width in linear time. If indeed $\mathsf{tw}(G) \leq \mathsf{tw}(k,t)$, we use Lemma 2 to find a smallest strong $\mathcal{W}_{\leq t}$-backdoor set B of F. If $|B| \leq k$ we output B, otherwise we output NO.

If $\mathsf{tw}(G) > \mathsf{tw}(k,t)$ then we proceed as follows. If $k = 0$, we output NO. Otherwise, by [33], G has a $\mathsf{wall}(k,t)$-wall as a topological minor, and by means of Grohe *et al.*'s algorithm [21], we can compute a topological model of a $\mathsf{wall}(k,t)$-wall in G in time $O(n^3)$. By Lemma 1, we can find in time $O(n^2)$ a set $S^* \subseteq \mathsf{var}(F)$ of constant size such that every strong $\mathcal{W}_{\le t}$-backdoor set of F of size at most k contains a variable from S^*. For each $x \in S^*$, the algorithm recurses on both formulas $F[x=0]$ and $F[x=1]$ with parameter $k-1$. If both recursive calls return strong $\mathcal{W}_{\le t}$-backdoor sets $B_{\neg x}$ and B_x, then $\{x\} \cup B_x \cup B_{\neg x}$ is a strong $\mathcal{W}_{\le t}$-backdoor set of F. We can upper bound its size $s(k)$ by the recurrence $s(k) \le 1 + 2 \cdot s(k-1)$, with $s(0) = 0$ and $s(1) = 1$. The recurrence is satisfied by setting $s(k) = 2^k - 1$. In case a recursive call returns NO, no strong $\mathcal{W}_{\le t}$-backdoor set of F of size at most k contains x. Thus, if for some $x \in S^*$, both recursive calls return backdoor sets, we obtain a backdoor set of F of size at most $2^k - 1$, and if for every $x \in S^*$, some recursive call returns NO, F has no strong $\mathcal{W}_{\le t}$-backdoor set of size at most k.

The number of nodes of the search tree modeling the recursive calls of this algorithm is a function of k and t only (and therefore constant), and at each node, the algorithm performs $O(n^2)$ steps. The overall running time is thus dominated by the cubic running time of Grohe *et al.*'s algorithm, hence we arrive at a total running time of $O(n^3)$. □

Theorem 2 follows easily from Theorem 1, by computing first a backdoor set and evaluating the number of satisfying assignments for all reduced formulas. We present an alternative proof that does not rely on Lemma 2. Instead of computing a backdoor set, one can immediately compute the number of satisfying assignments of F by dynamic programming if $\mathsf{tw}(\mathsf{inc}(F)) \le \mathsf{tw}(k,t)$.

Proof of Theorem 2. Let $k, t \ge 0$ be two integers and assume we are given a CNF formula F with $|F| = n$ and $\mathsf{sb}_t(F) \le k$. We will compute the number of satisfying truth assignments of F, denoted $\#(F)$. As before we use Bodlaender's linear-time algorithm [7] to decide whether $\mathsf{tw}(G) \le \mathsf{tw}(k,t)$, and if so, to compute a tree decomposition of smallest width. If $\mathsf{tw}(G) \le \mathsf{tw}(k,t)$ then we use the tree decomposition and, for instance, the algorithm of [38] to compute $\#(F)$ in time $O(n)$.

If $\mathsf{tw}(G) > \mathsf{tw}(k,t)$ then we compute, as in the proof of Theorem 1, a strong $\mathcal{W}_{\le t}$-backdoor set B of F of size at most 2^k in time $O(n^3)$. For each $\tau \in 2^B$ we have $F[\tau] \in \mathcal{W}_{\le t}$. Hence we can compute $\#(F[\tau])$ in time $O(n)$ by first computing a tree decomposition of width at most t, and then applying the counting algorithm of [38]. We obtain $\#(F)$ by taking $\sum_{\tau \in 2^B} 2^{d(F,\tau)} \#(F[\tau])$ where $d(F,\tau) = |\mathsf{var}(F) \setminus (B \cup \mathsf{var}(F[\tau]))|$ denotes the number of variables that disappear from $F[\tau]$ without being instantiated. □

3.1 The incidence graph has a large wall as a topological minor

This subsection is devoted to the proof of Lemma 1 and contains the main combinatorial arguments of this paper. Let $G = (V, E) = \mathsf{inc}(F)$ and suppose we are given a topological model of a $\mathsf{wall}(k,t)$-wall in G. We start with the description of the algorithm.

A *wall-obstruction* is a subgraph of G that is a subdivision of a $(2t+2)$-wall. Since a wall-obstruction, and any graph having a wall-obstruction as a subgraph, has treewidth at least $t+1$, we have that for each assignment to the variables of a strong $\mathcal{W}_{\le t}$-backdoor set, at least one vertex from each wall-obstruction vanishes in the incidence graph of the reduced formula. Using the $\mathsf{wall}(k,t)$-wall, we now find a set $\mathcal{O}$ of $\mathsf{obs}(k,t)$ vertex-disjoint wall-obstructions in G.

Lemma 3. *Given a topological model of a* $\mathsf{wall}(k,t)$*-wall in* G*, a set of* $\mathsf{obs}(k,t)$ *vertex-disjoint wall-obstructions can be found in linear time.*

Proof. For any two integers i and j with $1 \le i, j \le \mathsf{wall}(k,t)/(2t+2)$, the subgraph of a $\mathsf{wall}(k,t)$-wall induced on all vertices (x,y) with $(i-1)\cdot(2t+2)+1 \le x \le i\cdot(2t+2)$ and $(j-1)\cdot(2t+2)+1 \le y \le j\cdot(2t+2)$ is a $(2t+2)$-wall. A corresponding wall-obstruction can be found in G by replacing edges by the independent paths they model in the given topological model. The number of wall-obstructions defined this way is $\left\lfloor \frac{\mathsf{wall}(k,t)}{2t+2} \right\rfloor^2 \geq \left(\frac{\mathsf{wall}(k,t)}{2t+2} - 1\right)^2 \geq \mathsf{obs}(k,t)$. $\square$

Denote by $\mathcal{O}$ a set of $\mathsf{obs}(k,t)$ vertex-disjoint wall-obstructions obtained via Lemma 3. A backdoor variable can destroy a wall-obstruction either because it participates in the wall-obstruction, or because every setting of the variable satisfies a clause that participates in the wall-obstruction.

Definition 1. *Let x be a variable and W a wall-obstruction in G. We say that x* kills *W if neither $\mathsf{inc}(F[x=1])$ nor $\mathsf{inc}(F[x=0])$ contains W as a subgraph. We say that x kills W* internally *if $x \in V(W)$, and that x kills W* externally *if x kills W but does not kill it internally. In the latter case, W contains a clause c containing x and a clause c' containing $\neg x$ and we say that x kills W (externally)* in *c and c'.*

Our algorithm will perform a series of 3 nondeterministic steps to guess some properties about the strong $\mathcal{W}_{\leq t}$-backdoor set it searches. Each such guess is made out of a number of choices that is upper bounded by a function of k and t. At any stage of the algorithm, a *valid* strong $\mathcal{W}_{\leq t}$-backdoor set is one that satisfies all the properties that have been guessed. For a fixed series of guesses, the algorithm will compute a set $S \subseteq \mathsf{var}(F)$ such that every valid strong $\mathcal{W}_{\leq t}$-backdoor set of size at most k contains a variable from S. To make the algorithm deterministic, execute each possible combination of nondeterministic steps. The union of all S, taken over all combinations of nondeterministic steps, forms a set S^* and each strong $\mathcal{W}_{\leq t}$-backdoor set of size at most k contains a variable from S^*. Bounding the size of each S by a function of k and t enables us to bound $|S^*|$ by a function of k and t, and this will prove Lemma 1.

For each strong $\mathcal{W}_{\leq t}$-backdoor set of size at most k, at most k wall-obstructions from $\mathcal{O}$ are killed internally since they are vertex-disjoint. The algorithm guesses a set $\mathcal{O}''$ of k wall-obstructions from $\mathcal{O}$ that may be killed internally. Let $\mathcal{O}' := \mathcal{O} \setminus \mathcal{O}''$. All wall-obstructions in $\mathcal{O}'$ need to be killed externally by any valid strong $\mathcal{W}_{\leq t}$-backdoor set.

Suppose F has a valid strong $\mathcal{W}_{\leq t}$-backdoor set B of size k. Then, B defines a partition of $\mathcal{O}'$ into 2^k parts where for each part, the wall-obstructions contained in this part are killed externally by the same set of variables from B. Since $|\mathcal{O}'| = \mathsf{obs}(k,t) - k = 2^k \cdot \mathsf{same}(k,t)$, at least one of these parts contains at least $\mathsf{same}(k,t)$ wall-obstructions from $\mathcal{O}'$. The algorithm guesses a subset $\mathcal{O}_s \subseteq \mathcal{O}'$ of $\mathsf{same}(k,t)$ wall-obstruction from this part and it guesses how many variables from the strong $\mathcal{W}_{\leq t}$-backdoor set kill the wall-obstructions in this part externally.

Suppose each wall-obstruction in $\mathcal{O}_s$ is killed externally by the same set of ℓ backdoor variables, and no other backdoor variable kills any wall-obstruction from $\mathcal{O}_s$. Clearly, $1 \leq \ell \leq k$. Compute the set of external killers for each wall-obstruction in $\mathcal{O}_s$. Denote by Z the common external killers of the wall-obstructions in $\mathcal{O}_s$. The presumed backdoor set contains exactly ℓ variables from Z and no other variable from the backdoor set kills any wall-obstruction from $\mathcal{O}_s$.

We will use the following lemma to compute a small set S such that every valid strong $\mathcal{W}_{\leq t}$-backdoor set of size at most k contains a variable from S. The lemma will follow from Lemma 9, which is proven in Subsubsection 3.1.1 and contains the main arguments of this paper.

Lemma 4. *There is a quadratic time algorithm, that, given a propositional formula F, a set $\mathcal{O}_s$ of $\mathsf{same}(k,t)$ vertex-disjoint wall-obstructions in $\mathsf{inc}(F)$, and a set Z of vertices such that for each $z \in Z$ and each $W \in \mathcal{O}_s$, the vertex z is an external killer of W but not an internal killer of W, finds a set $S \subseteq \mathsf{var}(F)$ of size at most $6k\mathsf{nb}(t)$ such that every strong $\mathcal{W}_{\leq t}$-backdoor set B that contains ℓ variables from Z and no other variable that externally kills a wall-obstruction from $\mathcal{O}_s$, also contains a variable from S.*

Lemma 4 will enable us to compute S, and thereby S^*. The number of choices the algorithm has in the nondeterministic steps is upper bounded by $\binom{\mathsf{obs}(k,t)}{k} \cdot \binom{2^k \cdot \mathsf{same}(k,t)}{\mathsf{same}(k,t)} \cdot k$, and each series of guesses leads to a set S of at most $6k\mathsf{nb}(t)$ variables. Thus, the set S^*, the union of all such S, contains $2^{O(t^3 \cdot k \cdot 4^k \cdot \mathsf{polylog}(t))}$ variables, where $\mathsf{polylog}$ is a polylogarithmic function. Concerning the running time, each set S is computed in $O(n^2)$ time, by Lemma 4. Thus, the running time of the algorithm is $O(n^2)$. This proves Lemma 1.

3.1.1 Finding the beast, and taming it

We will now introduce graph-theoretic tools based on the theory of graph minors that we need for the algorithm of Lemma 4. We will give a polynomial-time algorithm for a more general problem (Lemma 9) defined in terms of colorings of graphs. It can handle general graphs instead of bipartite incidence graphs, instead of instantiations of boolean variables it allows for coloring vertices by an arbitrary constant number of colors, and apart from wall-obstruction it supports any degree-bounded vertex-disjoint subgraphs with treewidth at least $t+1$.

Let $t \geq 0$, $k \geq 1$, $\kappa \geq 2$, and $\Delta \geq 3$ be integers. A κ-edge coloring $\chi' : E \to \{0, \dots, \kappa - 1\}$ of a graph $G = (V, E)$ is a function assigning one of κ possible colors to the edges of G.

Let us define the concepts of obstructions and external killers in this more general setting.

Definition 2. *A* (t, Δ)-obstruction set $\mathcal{O}_s$ *of a graph* G *is a set of vertex-disjoint connected subgraphs of* G*, each with treewidth at least* $t+1$ *and maximum degree at most* Δ[1].

The elements of $\mathcal{O}_s$ will simply be called obstructions.

Definition 3. *A* common external $\mathcal{O}_s$-killer *in a* κ*-edge colored graph* $G = (V, E)$ *is a vertex* $z \in V$ *such that for each obstruction* $W \in \mathcal{O}_s$*, we have that* $z \notin V(W)$ *and* G *has at least two edges with distinct colors from* z *to a vertex in* W*.*

Recall the definition of the function $\mathsf{nb}(t)$,

$$\mathsf{nb}(t) := \lceil 16(t+2)\log(t+2) \rceil.$$

The following function is equal to $\mathsf{same}(k,t)$ when $\kappa = 2$ and $\Delta = 3$,

$$\mathsf{same}(k, t, \kappa, \Delta) := \Delta(\mathsf{nb}(t))^2 t \kappa^{2k}.$$

Suppose we know a (t, Δ)-obstruction set $\mathcal{O}_s$ of size $\mathsf{same}(k, t, \kappa, \Delta)$ and a set Z of common external $\mathcal{O}_s$-killers of a κ-edge colored graph G. We will be interested in partial colorings of Z and the treewidth of the subgraphs of G resulting from applying these partial colorings to G as follows. A coloring of a set $B \subseteq Z$ is a function $\chi : B \to \{0, \dots, \kappa - 1\}$ and the graph reduced by this coloring, $G[\chi]$, is the graph G induced on the vertex set

$$(V \setminus B) \setminus \{v \in V : \exists u \in B \cap N_G(v) \text{ s.t. } \chi'(uv) \neq \chi(v)\},$$

i.e., for every vertex $u \in B$, we remove u and those neighbors that are connected to u by an edge whose color differs from u's color. The aim is to design a polynomial time algorithm finding a small subset $S \subseteq Z$ such that at least one coloring of every set $B \subseteq Z \setminus S$ of size at most k reduces G to a graph with treewidth at least $t+1$. We call such a set a *relevant vertex set.*

[1] Alternatively, it would suffice to require that each $W \in \mathcal{O}_s$ has a spanning tree with maximum degree at most Δ that can be computed in polynomial time.

Relevant Vertex Set
Input: A graph $G = (V, E)$, a κ-edge coloring $\chi' : E \to \{0, \dots, \kappa - 1\}$ of G, a (t, Δ)-obstruction set $\mathcal{O}_s$ of G of size $\mathsf{same}(k, t, \kappa, \Delta)$, and a set Z of common external $\mathcal{O}_s$-killers.
Output: A relevant vertex set of size at most $2\Delta k\mathsf{nb}(t)$, i.e., a set $S \subseteq Z$ of at most $2\Delta k\mathsf{nb}(t)$ vertices such that for every $B \subseteq Z \setminus S$ of size at most k, there exists a coloring χ_B for B such that the treewidth of $G[\chi_B]$ is at least $t + 1$.

Note that the problem is trivial if $|Z| \leq 2\Delta k\mathsf{nb}(t)$ because then Z is a relevant vertex set. Therefore, we assume from now on that $|Z| > 2\Delta k\mathsf{nb}(t)$.

In terms of Lemma 4, G will correspond to $\mathsf{inc}(F)$, $\kappa = 2$, $\Delta = 3$, and an edge is colored 0 if its sign is positive and colored 1 if its sign is negative. Then, a coloring χ of a set $B \subseteq Z$ corresponds to a partial truth assignment in F, and we have that $G[\chi] = \mathsf{inc}(F[\chi])$.

We come now to the definition of a central combinatorial object that we use to find a relevant vertex set.

Definition 4. *An* obstruction-template $\mathsf{OT}(W)$ *of an obstruction* $W \in \mathcal{O}_s$ *is a triple* $(\mathcal{B}(W), P, R)$*, where*

- $\mathcal{B}(W)$ *is a bipartite graph whose vertex set is bipartitioned into the two independent sets* Z *and* Q_W*, where* Q_W *is a set of new vertices,*
- P *is a partition of* $V(W)$ *into* regions *such that for each region* $A \in P$*, we have that* $W[A]$ *is connected, and*
- $R : Q_W \to P$ *is a function associating a region of* P *with each vertex in* Q_W*.*

Definition 5. *An obstruction-template* $\mathsf{OT}(W) = (\mathcal{B}(W), P, R)$ *of an obstruction* $W \in \mathcal{O}_s$ *is* valid *if it satisfies the following properties:*

(1) only existing edges: *for each* $q \in Q_W$ *we have that* $N_{\mathcal{B}(W)}(q) \subseteq N_G(R(q))$*,*

(2) private neighbor: *for each* $q \in Q_W$*, there is a vertex* $z \in N_{\mathcal{B}(W)}(q)$*, called* q*'s* private neighbor*, such that there is no other* $q' \in N_{\mathcal{B}(W)}(z)$ *with* $R(q') = R(q)$*,*

(3) degree-Z: *for each* $z \in Z$ *we have that* $d_{\mathcal{B}(W)}(z) \geq 1$*,*

(4) degree-Q_W: *for each* $q \in Q_W$ *we have that* $\mathsf{nb}(t) \leq d_{\mathcal{B}(W)}(q) \leq \Delta\mathsf{nb}(t)$*, and*

(5) vulnerable vertex: *for each* $q \in Q_W$*, there is at most one vertex* $v \in R(q)$*, called* q*'s* vulnerable vertex*, such that* $N_G(v) \cap Z \not\subseteq N_{\mathcal{B}(W)}(q)$*.*

We will use the obstruction-templates to identify a relevant vertex set S. Intuitively, an obstruction-template chops up the vertex set of an obstruction into regions. We will suppose the existence of a set $B \subseteq Z \setminus S$ of size at most k such that $G[\chi]$ has treewidth at most t for every coloring χ of B and then derive a contradiction using the obstruction-templates. This is done by showing that for at least one coloring of B, many regions remain in $G[\chi]$, so that we can contract each of them and construct a treewidth obstruction using the contracted vertices. Each vertex from Q_W models a contraction of a region, and its neighborhood models vertices that could potentially hinder the contraction of the region. This explains Property (1). Property (2) becomes handy when a region has many vertices from Q_W that are associated with it. Namely, when we contract regions, we would like to be able to guarantee a lower bound on the number of edges of the resulting graph in terms of $|Q_W|$. To ensure that this lower bound translates into a lower bound in terms of $|Z|$, we need Property (3). The degree lower bound of Property (4) will be needed so we can patch together a treewidth obstruction out of the pieces modeled by the vertices in Q_W. The upper bound on the degree is required to guarantee that sufficiently many vertices from Q_W are not neighboring B. Finally, the last property will be used to guarantee that for every $q \in Q_W$, if $B \cap N_{\mathcal{B}(W)}(q) = \emptyset$, then there is a coloring χ of B such that no vertex from q's region is removed from G by reducing the graph according to χ (see Lemma 6).

In the following lemma, we give a procedure to compute valid obstruction-templates.

Lemma 5. *For each obstruction $W \in \mathcal{O}_s$, a valid obstruction-template can be computed in time $O(|V(W)|^2 + |V(W)| \cdot |Z|)$.*

Proof. We describe a procedure to compute a valid obstruction-template $(\mathcal{B}(W), P, R)$. It starts with Q_W initially empty. Compute an arbitrary rooted spanning tree T of W. For a node v from T, denote by T_v the subtree of T rooted at v. The set of children in T of a node v is denoted $C_T(v)$ and its parent $p_T(v)$. For a subforest $T' \subseteq T$, denote by $Z(T') = Z \cap N_G(T')$ the subset of vertices from Z that have a neighbor from T' in G. The *weight* $w(T')$ of T' is $|Z(T')|$. We denote by $B_v = Z(T_v) \setminus Z(T_v - \{v\})$ the vertices from Z that are incident to v in G but to no other node from T_v. If $uv \in E(T)$, then denote by $T_u(uv)$ the subtree obtained from T by removing all nodes that are closer to v than to u in T (removing the edge uv decomposes T into $T_u(uv)$ and $T_v(uv)$).

(A) If $w(T) > \Delta\mathsf{nb}(t)$, then set the root $r(T)$ of T such that for every child $c \in C_T(r(T))$ of $r(T)$ we have that $w(T - V(T_c)) \geq \mathsf{nb}(t)$.

(B) Select a node v in T as follows. If $w(T) \leq \Delta\mathsf{nb}(t)$, then set $v := r(T)$. Otherwise, select the node v at maximum depth in T such that $w(T_v) \geq \mathsf{nb}(t)$. The vertices from T_v will constitute one region A_v of P. Denoting $s = \Delta\mathsf{nb}(t) - w(T_v - \{v\})$, we will now add a set of $\left\lceil \frac{|B_v|}{s} \right\rceil$ vertices to Q_W. All of them are associated with the region A_v. Denote these new vertices $q_1, \ldots, q_{\lceil |B_v|/s \rceil}$, and denote the vertices in B_v by $b_1, \ldots, b_{|B_v|}$. For each $i, 1 \leq i \leq \lceil |B_v|/s \rceil$, we set $N(q_i) := Z(T_v - \{v\}) \cup \{b_{(i-1) \cdot s+1}, \ldots, b_{i \cdot s}\}$; indices are taken modulo $|B_v|$. If $v \neq r(T)$, then set $T := T_{p_T(v)}(vp_T(v))$ (i.e., remove $V(T_v)$ from T) and go to Step (A).

Now, we prove that this procedure computes a valid obstruction-template.

First we show that in case $w(T) > \Delta\mathsf{nb}(t)$, Step (A) is able to find a root $r(T)$ such that there is no $c \in C_T(r(T))$ with $w(T - V(T_c)) < \mathsf{nb}(t)$. Suppose that T has no node u such that $w(T_u(uv)) \geq \mathsf{nb}(t)$ for every $v \in N_T(u)$. Then, starting with an arbitrary root u_1 and replacing a root u_i by a child u_{i+1} with $w(T - V(T_{u_{i+1}})) < \mathsf{nb}(t)$ defines an infinite sequence of nodes $u_1, u_2, \ldots$ such that u_i neighbors u_{i+1} and $w(T_{u_i}(u_i u_{i+1})) < \mathsf{nb}(t)$. Let j be the smallest integer such that $u_i = u_j$ for some integer i with $1 \leq i < j$. Since T is acyclic, we have that $i + 2 = j$. But then, $w(T) \leq w(T_{u_i}(u_i u_{i+1})) + w(T_{u_{i+1}}(u_i u_{i+1})) \leq 2\mathsf{nb}(t) - 2$, contradicting the assumption that $w(T) > \Delta\mathsf{nb}(t) \geq 3\mathsf{nb}(t)$.

We observe that all edges of $\mathcal{B}(W)$ have one endpoint in Z and the other in Q_W. Thus, $Z \uplus Q_W$ is a bipartition of $\mathcal{B}(W)$ into independent sets. The set $V(W)$ is partitioned into disjoint connected regions since each execution of Step (B) defines a new region equal to the vertices of a subtree T_v of T, the tree T is initially a spanning tree of W, and T_v is removed from T at the end of Step (B).

Consider one execution of Step (B) of the procedure above. We will show that Properties (1)–(5) of a valid obstruction-template hold for the vertices considered in this execution of the step, and this will guarantee these properties for all vertices. Property (1) is ensured for all new vertices introduced in Q_W since $B_v \cup Z(T_v - \{v\}) = Z(T_v) \subseteq N_G(A_v)$. The private neighbor of a vertex q_i

is $b_{i\cdot s}$ if $i < \lceil |B_v|/s \rceil$ and $b_{|B_v|}$ if $i = \lceil |B_v|/s \rceil$. Property (3) is ensured for all vertices in $Z \cap N_G(T_v)$ since all of them receive at least one new neighbor in $\mathcal{B}(W)$. For the lower bound of Property (4), we first show that at any time during the execution of this procedure, either T is empty or $w(T) \geq \mathsf{nb}(t)$. Initially, this is true since $|Z| \geq \mathsf{nb}(t)$ and every vertex from Z has a neighbor in W. This remains true since Step (A) makes sure that whenever the vertex v chosen by Step (B) is not the root of T, $w(T - V(T_v)) \geq \mathsf{nb}(t)$. Thus, Step (B) always finds a node v such that $w(T_v) \geq \mathsf{nb}(t)$. Therefore, every vertex that is added to Q_W has at least $\mathsf{nb}(t)$ neighbors. For the upper bound of Property (4), observe that since W has maximum degree at most Δ, T also has maximum degree at most Δ. Thus, $w(T_v - \{v\}) \leq \Delta(\mathsf{nb}(t) - 1)$ since each tree in $T_v - \{v\}$ has weight at most $\mathsf{nb}(t) - 1$ by the selection of v. Therefore, $d_{\mathcal{B}(W)}(q_i) \leq \Delta \mathsf{nb}(t)$. Property (5) holds since $Z(T_v - \{v\}) \subseteq N_{\mathcal{B}(W)}(q_i)$ and thus v is the only vertex that can be vulnerable for q_i.

We upper bound the running time of the procedure as follows. A spanning tree of W can be computed in time $O(|V(W)|^2)$. In a bottom-up fashion starting at the leaves of T, one can precompute $Z(T_u(uv))$ for all $uv \in E(T)$ in time $O(|V(W)| \cdot |Z|)$. Then, Step (A) can be implemented such that each execution runs in time $O(|V(W)|)$. One execution of Step (B) takes time $O(|V(W)| + |Z|)$. Since Steps (A) and (B) are executed at most $|V(W)|$ times, the running time is $O(|V(W)| \cdot (|V(W)| + |Z|))$. □

The following lemma shows that Property (5) ensures that a region $R(q)$ is unaffected by at least one coloring of any subset of Z containing no neighbor of q.

Lemma 6. *Let $W \in \mathcal{O}_s$ be an obstruction, $\mathsf{OT}(W)$ be a valid obstruction-template of W and $q \in Q_W$. Let $B \subseteq Z \setminus N_{\mathcal{B}(W)}(q)$. There is a coloring χ of B such that $G[\chi]$ contains all vertices from $R(q)$.*

Proof. By definition B contains no vertex from $R(q)$. A coloring χ removes a vertex $c \in R(q)$ from G iff c has an edge to a vertex $b \in B$ with $\chi(b) \neq \chi'(bc)$. We will show that no vertex from B has two edges with distinct colors to vertices in $R(q)$. Therefore there is a coloring χ of B such that $G[\chi]$ contains all vertices from $R(q)$.

For the sake of contradiction, assume there is a vertex $b \in B$ with two neighbors $c, c' \in R(q)$ such that $\chi'(bc) \neq \chi'(bc')$. Since $b \in (N_G(R(q)) \cap Z) \setminus N_{\mathcal{B}(W)}(q)$, we conclude that q has a vulnerable vertex v. But, since v is the only vulnerable vertex of q, by Property (5), $v = c = c'$. We arrive at a contradiction, since G has no multiedges. □

The bipartite graph $\mathcal{B}_m(\mathcal{O}_s)$ is obtained by taking the union of all $\mathcal{B}(W)$, $W \in \mathcal{O}_s$. Its subgraphs $\mathcal{B}(W)$, $W \in \mathcal{O}_s$, share the same vertex subset Z but the vertex subsets Q_W, $W \in \mathcal{O}_s$, are pairwise disjoint. The vertex set of $\mathcal{B}_m(\mathcal{O}_s)$ is $Z \uplus Q_m$, where $Q_m = \bigcup_{W \in \mathcal{O}_s} Q_W$.

The bipartite graph $\mathcal{B}(\mathcal{O}_s)$ is obtained from $\mathcal{B}_m(\mathcal{O}_s)$ by repeatedly and exhaustively deleting vertices from Q_m whose neighborhood equals the neighborhood of some other vertex from Q_m. Denote the vertex set of the resulting graph $\mathcal{B}(\mathcal{O}_s)$ by $Z \uplus Q$. The multiplicity $\mu(q)$ of a vertex $q \in Q$ is the number of distinct vertices $q' \in Q_m$ such that $N_{\mathcal{B}_m(\mathcal{O}_s)}(q') = N_{\mathcal{B}(\mathcal{O}_s)}(q)$.

Lemma 7. *If there is a vertex $q \in Q$ with $\mu(q) \geq t \cdot \kappa^k + 1$, then $N_{\mathcal{B}(\mathcal{O}_s)}(q)$ is a relevant vertex set.*

Proof. Let $Q_L \subseteq Q_m$ be a set of $t \cdot \kappa^k + 1$ vertices such that for each $q \in Q_L$, $N_{\mathcal{B}_m(\mathcal{O}_s)}(q) = L$. For the sake of contradiction, suppose L is not a relevant vertex set. Then, there exists a set $B \subseteq Z \setminus L$ of size at most k such that for each coloring χ of B, the treewidth of $G[\chi]$ is at most t. By Lemma 6, for each $q \in Q_L$, there is a coloring χ of B such that $G[\chi]$ contains all vertices from $R(q)$. But there are at most κ^k colorings of B. Therefore, for at least one coloring χ of B, there is a set $Q'_L \subseteq Q_L$

of at least $\lceil |Q_L|/\kappa^k \rceil = t+1$ vertices such that $G[\chi]$ contains all vertices from $R(q), q \in Q'_L$. By Property (2), no two distinct $q, q' \in Q_L$ are assigned to the same region. Consider the subgraph of $G[\chi]$ induced on all vertices in L and $R_q, q \in Q'_L$. Contracting each region $R(q), q \in Q'_L$, one obtains a supergraph of a $K_{t+1,t+1}$. Thus, $G[\chi]$ has a $K_{t+1,t+1}$ as a minor, implying that its treewidth is at least $t+1$, a contradiction. □

Lemma 8. *If every vertex from Q has multiplicity at most $t \cdot \kappa^k$, then the $2\Delta k \mathsf{nb}(t)$ vertices from Z of highest degree in $\mathcal{B}(\mathcal{O}_s)$ (ties are broken arbitrarily) form a relevant vertex set.*

We will prove the lemma with the use of a theorem by Mader.

Theorem 4 ([28])**.** *Every graph $G = (V, E)$ with $|E| \geq c(x) \cdot |V|$ has a K_x-minor, where $c(x) = 8x \log x$.*

For large x, the function $c(x)$ can actually be improved to $c(x) = (\alpha + o(1))x\sqrt{\log x}$ where $\alpha = 0.319\ldots$ is an explicit constant, and random graphs are extremal [41].

Proof of Lemma 8. Denote by S the set of $2\Delta k \mathsf{nb}(t)$ vertices from Z of highest degree in $\mathcal{B}(\mathcal{O}_s)$. Suppose $B \subseteq Z \setminus S$ is a set of size at most k such that each coloring χ of B reduces G to a graph with treewidth at most t. To arrive at a contradiction, we exhibit a coloring χ to B such that $G[\chi]$ has treewidth at least $t+1$.

Claim 1. *There is a coloring χ of B and a set $Q' \subseteq Q$ with $|Q'| \geq \frac{|Z| \cdot |\mathcal{O}_s|}{2\Delta \mathsf{nb}(t) t \kappa^{2k}}$ such that $G[\chi]$ contains all vertices from $\bigcup_{q \in Q'} R(q)$.*

To prove the claim, we first show a lower bound on $|Q \setminus N_{\mathcal{B}(\mathcal{O}_s)}(B)|$ in terms of $|Z|$ and $|\mathcal{O}_s|$.

Since, by Property (3), each vertex $z \in Z$ has degree at least one in $\mathcal{B}(W)$, $W \in \mathcal{O}_s$, there are at least $|Z| \cdot |\mathcal{O}_s|$ edges in $\mathcal{B}_m(\mathcal{O}_s)$. Since, by Property (4), each vertex from Q_m has degree at most $\Delta \mathsf{nb}(t)$, we have that $|Q_m| \geq \frac{|Z| \cdot |\mathcal{O}_s|}{\Delta \mathsf{nb}(t)}$. By assumption, no set of $t \cdot \kappa^k + 1$ vertices from Q_m has the same neighborhood. Therefore, $|Q| \geq \frac{|Z| \cdot |\mathcal{O}_s|}{\Delta \mathsf{nb}(t) t \kappa^k}$. Let d denote the number of edges in $\mathcal{B}(\mathcal{O}_s)$ with one endpoint in B. Thus, $|N_{\mathcal{B}(\mathcal{O}_s)}(B)| \leq d$. Since $|S| \geq 2\Delta|B|\mathsf{nb}(t)$ and the degree of any vertex in S is at least the degree of any vertex in B, we have that the number of edges incident to S is at least $2\Delta \mathsf{nb}(t) d$ in $\mathcal{B}(\mathcal{O}_s)$. Thus, $|Q| \geq \frac{2\Delta \mathsf{nb}(t) d}{\Delta \mathsf{nb}(t)} = 2d$. Therefore, $N_{\mathcal{B}(\mathcal{O}_s)}(B)$ contains at most half the vertices of Q, and we have that $|Q \setminus N_{\mathcal{B}(\mathcal{O}_s)}(B)| \geq \frac{|Z| \cdot |\mathcal{O}_s|}{2\Delta \mathsf{nb}(t) t \kappa^k}$.

By Lemma 6, for every $q \in Q \setminus N_{\mathcal{B}(\mathcal{O}_s)}(B)$ there is a coloring χ of B such that $G[\chi]$ contains all vertices from $R(q)$. Since B has at most κ^k colorings, there is a coloring χ of B and a subset $Q' \subseteq Q \setminus N_{\mathcal{B}(\mathcal{O}_s)}(B)$ of at least $|Q \setminus N_{\mathcal{B}(\mathcal{O}_s)}(B)|/\kappa^k \geq \frac{|Z| \cdot |\mathcal{O}_s|}{2\Delta \mathsf{nb}(t) t \kappa^{2k}}$ vertices such that $G[\chi]$ contains all vertices from $R(q)$ for every $q \in Q'$. This proves Claim 1. ⌟

Let $H' := \mathcal{B}(\mathcal{O}_s)[Z' \cup Q']$ where $Z' := Z \setminus B$ and Q' is as in Claim 1. Thus, no vertex from $Z' \cup \bigcup_{q \in Q'} R(q)$ is removed from G by applying the coloring χ. We will now merge vertices from H' in such a way that we obtain a minor of $G[\chi]$. To achieve this, we repeatedly merge a part $A \in P$ into a vertex $z \in Z$ such that z has a neighbor q in H' such that $R(q) = A$. In G, this corresponds to contracting $R(q) \cup \{z\}$ into the vertex z. After having contracted all vertices from Q' into vertices from Z', we obtain therefore a minor of $G[\chi]$. Our objective will be to show that the treewidth of this minor is at least $t+1$, which implies that $G[\chi]$ has treewidth at least $t+1$ as well.

Claim 2. *$G[\chi]$ has a K_{t+2}-minor.*

To prove the claim, we start with H'' and Q'' as copies of H' and Q', respectively. We use the invariant that every connected component of $H''[Z']$ is a minor of $G[\chi]$.

For any part A of the partition P, let R_A denote the set of vertices $\{q \in Q'' : R(q) = A\}$. As long as $Q'' \neq \emptyset$, select a part A of P such that $|R_A| \geq 1$. Let $U := \bigcup_{q \in R_A} N_{H''}(q)$. By the construction of H' and $\mathcal{B}(\mathcal{O}_s)$ (Property (2)), we have that $|U| \geq \mathsf{nb}(t) + |R_A| - 1$.

If for every vertex $u \in U$, $|N_{H''}(u) \cap U| \geq \mathsf{nb}(t)$, then $H''[U]$ has at least $\mathsf{nb}(t) \cdot |U|/2 = \lceil 8(t+2)\log(t+2) \rceil \cdot |U|$ edges. Then, by Theorem 4, $H''[U]$ has a K_{t+2}-minor. By our invariant, $G[\chi]$ has a K_{t+2}-minor.

Otherwise, there exist a vertex $z \in U$ such that z has less than $\mathsf{nb}(t)$ neighbors in U. But then, merging A into z adds at least $|U| - \mathsf{nb}(t) + 1 \geq |R_A|$ edges to $H''[Z']$.

In the end, if no K_{t+2}-minor was found before $Q'' = \emptyset$, each merge of a part A of P into a vertex from Z' added at least $|R_A|$ edges to $H''[Z']$. Therefore, the final graph $H''[Z']$ contains at least $|Q'|$ edges. By Claim 1, $|Q'| \geq \frac{|Z| \cdot |\mathcal{O}_s|}{2\Delta \mathsf{nb}(t) t \kappa^{2k}}$ and $|\mathcal{O}_s| = \mathsf{same}(k, t, \kappa, \Delta) = \Delta(\mathsf{nb}(t))^2 t \kappa^{2k}$. Thus, $H''[Z']$ has at least $(8(t+2)\log(t+2)) \cdot |Z'|$ edges. Consequently, $H''[Z']$ has a K_{t+2}-minor by Theorem 4, which is a minor of $G[\chi]$ by our invariant. This proves Claim 2. ⌟

Claim 2 entails that $G[\chi]$ has treewidth at least $t+1$. This proves the lemma. □

The following lemma summarizes these results.

Lemma 9. *The* Relevant Vertex Set *problem can be solved in $O(n^2)$ time, where n is the number of vertices of G.*

Proof. If $|Z| \leq 2\Delta k \mathsf{nb}(t)$, then Z is a relevant vertex set of size at most $2\Delta k \mathsf{nb}(t)$. Suppose from now on that $|Z| > 2\Delta k \mathsf{nb}(t)$.

Compute an obstruction-template for each obstruction in $\mathcal{O}_s$ by Lemma 5. Since $\mathcal{O}_s = O(1)$, this takes time $O(n^2)$. Compute $\mathcal{B}_m(\mathcal{O}_s)$ and $\mathcal{B}(\mathcal{O}_s)$. The construction of $\mathcal{B}(\mathcal{O}_s)$ needs to compare the neighborhoods of a quadratic number of vertices from Q_m. Since each vertex from Q_m has a constant sized neighborhood, this takes $O(n^2)$ time.

If there is a vertex $q \in Q$ with $\mu(q) \geq t \cdot \kappa^k + 1$, then, by Lemma 7, $N_{\mathcal{B}(\mathcal{O}_s)}(q)$ is a relevant vertex set and $|N_{\mathcal{B}(\mathcal{O}_s)}(q)| \leq \Delta \mathsf{nb}(t)$. Otherwise, by Lemma 8 the $2\Delta k \mathsf{nb}(t)$ vertices from Z of highest degree in $\mathcal{B}(\mathcal{O}_s)$ form a relevant vertex set. □

Lemma 4 will now follow as a special case of Lemma 9.

Proof of Lemma 4. We reduce the problem to the Relevant Vertex Set problem whose input is defined as follows. Let $G = \mathsf{inc}(F)[Z \cup V(\mathcal{O}_s)]$, where $V(\mathcal{O}_s) = \bigcup_{W \in \mathcal{O}_s} V(W)$. Note that G is a simple graph since no clause contains a variable and its negation. Let $\kappa = 2$ and $\Delta = 3$. If $xc \in E(G)$ with $x \in \mathsf{var}(F)$ and $c \in \mathsf{cla}(F)$, then set

$$\gamma'(xc) := \begin{cases} 0 & \text{if } x \in \mathsf{lit}(c), \text{ and} \\ 1 & \text{if } \neg x \in \mathsf{lit}(c). \end{cases} \tag{1}$$

The set $\mathcal{O}_s$ is a (t, Δ)-obstruction set of G of size $\mathsf{same}(k, t, \kappa, \Delta) = \mathsf{same}(k, t)$, and the set Z is a set of common external $\mathcal{O}_s$-killers.

Now, since $G[\chi]$ is a subgraph of $\mathsf{inc}(F[\chi])$ for every 2-coloring χ of every $B \subseteq Z$, a solution to the Relevant Vertex Set problem is a set $S \subseteq Z$ of size at most $6k\mathsf{nb}(t)$ such that for every $B \subseteq Z \setminus S$ of size at most k, there exists a partial truth assignment $\tau \in 2^B$ such that the treewidth of $\mathsf{inc}(F[\tau])$ is at least $t+1$. Since $\ell \leq k$, every strong $\mathcal{W}_{\leq t}$-backdoor set B that contains ℓ variables from Z and no other variable that externally kills a wall-obstruction from $\mathcal{O}_s$, also contains a variable from S. □

3.2 The incidence graph has small treewidth

This subsection is devoted to the proof of Lemma 2.

We are going to use Arnborg *et al.*'s extension [3] of Courcelle's Theorem [9]. It gives, amongst others, a linear-time algorithm that takes as input a graph $\mathcal{A}$ with labeled vertices and edges, a tree decomposition of $\mathcal{A}$ of constant width, and a fixed Monadic Second Order (MSO) sentence $\varphi(X)$, and computes a minimum-sized set of vertices X such that $\varphi(X)$ is true in $\mathcal{A}$.

First, we define the labeled graph $\mathcal{A}_F$ for F. The set of vertices of $\mathcal{A}_F$ is $\mathsf{lit}(F) \cup \mathsf{cla}(F)$. The vertices are labeled by LIT and CLA, respectively. The vertices from $\mathsf{var}(F)$ are additionally labeled by VAR. The set of edges is the union of the sets $\{\, x\neg x : x \in \mathsf{var}(F) \,\}$ and $\{\, c\ell : c \in \mathsf{cla}(F), \ell \in \mathsf{lit}(c) \,\}$, edges in the first set are labeled NEG, and edges in the second set are labeled IN.

Since a tree decomposition of $\mathcal{A}_F$ may be obtained from a tree decomposition of $\mathsf{inc}(F)$ by replacing each variable by both its literals, we have that $\mathsf{tw}(\mathcal{A}_F) \le 2 \cdot \mathsf{tw}(\mathsf{inc}(F)) + 1$ and we obtain a constant-width tree decomposition of $\mathcal{A}_F$ in this way.

The goal is to find a minimum size subset X of variables such that for each truth assignment τ to X the incidence graph of $F[\tau]$ belongs to $\mathcal{G}_{\le t}$, where $\mathcal{G}_{\le t}$ denotes the class of all graphs of treewidth at most t. For testing membership in $\mathcal{G}_{\le t}$ we use a forbidden-minor characterization. As proved in a series of papers by Robertson and Seymour [34], every minor-closed class $\mathcal{G}$ of graphs is characterized by a finite set $\mathsf{obs}(\mathcal{G})$ of forbidden minors. That is, $\mathsf{obs}(\mathcal{G})$ is a finite set of graphs such that a graph G belongs to $\mathcal{G}$ if and only if G does not contain any graph from $\mathsf{obs}(G)$ as a minor. Clearly $\mathcal{G}_{\le t}$ is minor-closed. We denote its finite set of obstructions by $\mathsf{obs}(t) = \mathsf{obs}(\mathcal{G}_{\le t})$. The set $\mathsf{obs}(t)$ is explicitly given in [4] for $t \le 3$ and it can be computed in constant time [1, 25] for all other values of k.

Next we formulate an MSO sentence that checks whether for each truth assignment τ to X, the incidence graph of $F[\tau]$ does not contain any of the graphs in $\mathsf{obs}(t)$ as a minor. We break up our MSO sentence into several simpler sentences and we use the notation of [15]. The following sentence checks whether X is a set of variables.

$$\mathrm{var}(X) = \forall x(Xx \to \mathrm{VAR}x)$$

We associate a partial truth assignment to X with a subset Y of $\mathsf{lit}(F)$, the literals set to 1 by the partial truth assignment. This subset Y contains no complementary literals, every literal in Y is either a variable from X or its negation, and for every variable x from X, x or $\neg x$ is in Y. The following sentence checks whether Y is an assignment to X.

$$\begin{aligned}\mathrm{ass}(X,Y) = \forall y[Yy \to ((Xy \vee (\exists z(Xz \wedge \mathrm{NEG}yz))) \\ \wedge (\forall z(Yz \to \neg\mathrm{NEG}yz)))] \\ \wedge \forall x[Xx \to (Yx \vee \exists y(Yy \wedge \mathrm{NEG}xy))]\end{aligned}$$

To test whether $\mathsf{inc}(F[\tau])$ has a graph G with $V(G) = \{v_1, \ldots, v_n\}$ as a minor, we will check whether it contains n disjoint sets $A_1, \ldots, A_n$ of vertices, where each set A_i corresponds to a vertex v_i of G, such that the following holds: each set A_i induces a connected subgraph in $\mathsf{inc}(F[\tau])$, and for every two vertices v_i, v_j that are adjacent in G, the corresponding sets A_i, A_j are connected by an edge in $\mathsf{inc}(F[\tau])$. Deleting all vertices that are in none of the n sets, and contracting each of the sets into one vertex, one obtains G as a minor of $F[\tau]$. To test whether $\mathcal{A}_F$ has G as a minor can be done similarly, except that we need to ensure that each set A_i is closed under the complementation of literals (i.e., $x \in A_i$ iff $\neg x \in A_i$). The following sentence checks whether A is disjoint from B.

$$\mathrm{disjoint}(A,B) = \neg\exists x(Ax \wedge Bx)$$

To check whether A is connected, we check that there is no set B that is a proper nonempty subset of A such that B is closed under taking neighbors in A.

$$\begin{aligned}\text{connected}(A) = \neg\exists B\ [&\exists x(Ax \wedge \neg Bx) \wedge \exists x(Bx)\\ &\wedge \forall x(Bx \rightarrow Ax) \wedge \forall x, y((Bx \wedge Ay\\ &\wedge (\text{IN}xy \vee \text{NEG}xy)) \rightarrow By)]\end{aligned}$$

The next sentence checks whether A is closed under complementation of literals.

$$\text{closed}(A) = \forall x, y(\text{NEG}xy \rightarrow (Ax \leftrightarrow Ay))$$

The following sentence checks whether some vertex from A and some vertex from B have a common edge labeled IN.

$$\text{edge}(A, B) = \exists x, y(Ax \wedge Bx \wedge \text{IN}xy)$$

An assignment removes from the incidence graph all variables that are assigned and all clauses that are assigned correctly. Therefore, the minors we seek must not contain any variable that is assigned nor any clause that is assigned correctly. The following sentence checks whether all vertices from a set A survive when assigning Y to X.

$$\begin{aligned}\text{survives}(A, X, Y) = \neg\exists x(Ax \wedge (&Xx \vee \exists y(Xy \wedge \text{NEG}xy)\\ &\vee \exists y(Yy \wedge \text{IN}yx)))\end{aligned}$$

We can now test whether a G-minor survives in the incidence graph as follows:

$$\begin{aligned}G\text{-minor}(X, Y) = \exists A_1, \ldots, A_n[&\bigwedge_{i=1}^{n}(\text{survives}(A_i)\\ &\wedge \text{connected}(A_i) \wedge \text{closed}(A_i))\\ &\wedge \bigwedge_{1\le i<j\le n} \text{disjoint}(A_i, A_j) \wedge \bigwedge_{\substack{1\le i<j\le n\\ v_iv_j\in E(G)}} \text{edge}(A_i, A_j)]\end{aligned}$$

Our final sentence checks whether X is a strong $\mathcal{W}_{\le t}$-backdoor set of F.

$$\begin{aligned}\text{Strong}_t(X) = \text{var}(X) \wedge \forall Y[&\text{ass}(X, Y)\\ &\rightarrow \bigwedge_{G\in\mathsf{obs}(t)} \neg G\text{-minor}(X, Y)))]\end{aligned}$$

Recall that we assume t to be a constant. Hence $|\text{Strong}_t| = O(1)$. Moreover, the tree decomposition of $\mathcal{A}_F$ that we described has width $O(1)$. We can now use the result of Arnborg *et al.* [3] that provides a linear time algorithm for finding a smallest set X of vertices of $\mathcal{A}_F$ for which $\text{Strong}_t(X)$ holds. This completes the proof of Lemma 2.

4 Conclusion

We have combined two very prominent (#)SAT solving techniques in a new way, using the advantages of both methods to efficiently solve a much larger class of instances. Our cubic-time algorithm solves SAT and #SAT for those CNF formulas F that have a strong backdoor set of size at most

k into the class of formulas with incidence treewidth at most t, where k and t are constants. Our algorithm can be seen as an efficient way to compute a "super" feedback vertex set, where "super" stands for a generalization in several ways: (i) instead of producing a forest, it produces a graph of bounded treewidth, (ii) instead of deleting vertices to kill the cycles, it applies partial assignments, which is a much stronger operation, as demonstrated in the introduction (cf. the discussion of implied cycle cutsets).

We also designed an approximation algorithm for finding an actual strong backdoor set. Can our backdoor detection algorithm be improved to an exact algorithm? In other words, is there an $O(n^c)$-time algorithm finding a k-sized strong $\mathcal{W}_{\leq t}$-backdoor set of any formula F with $\mathsf{sb}_t(F) \leq k$ where k, t are two constants and c is an absolute constant independent of k and t? This question is even open for $t = 1$. An orthogonal question is how far one can generalize the class of tractable (#)SAT instances. For example, improving the dependence on t of the running time of our algorithm could significantly enlargen the class of known subexponential-time solvable (#)SAT instances.

5 Acknowledgments

Both authors acknowledge support by the European Research Council (ERC), project COMPLEX REASON 239962. Serge Gaspers acknowledges support by the Australian Research Council, grant DE120101761.

References

[1] I. Adler, M. Grohe, and S. Kreutzer. Computing excluded minors. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008)*, pages 641–650. SIAM, 2008.

[2] M. Alekhnovich and A. A. Razborov. Satisfiability, branch-width and Tseitin tautologies. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2002)*, pages 593–603, 2002.

[3] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12(2):308–340, 1991.

[4] S. Arnborg, A. Proskurowski, and D. G. Corneil. Forbidden minors characterization of partial 3-trees. *Discrete Math.*, 80(1):1–19, 1990.

[5] F. Bacchus, S. Dalmao, and T. Pitassi. Algorithms and complexity results for #SAT and Bayesian inference. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2003)*, pages 340–351, 2003.

[6] B. Bidyuk and R. Dechter. Cutset sampling for Bayesian networks. *J. Artif. Intell. Res.*, 28:1–48, 2007.

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

[8] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC 1971)*, pages 151–158, 1971.

[9] B. Courcelle. Graph rewriting: an algebraic and logic approach. In *Handbook of theoretical computer science, Vol. B*, pages 193–242. Elsevier Science Publishers, North-Holland, Amsterdam, 1990.

[10] M. Cygan, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. On the hardness of losing width. In D. Marx and P. Rossmanith, editors, *Proceedings of the 6th International Symposium on Parameterized and Exact Computation (IPEC 2011)*, volume 7112 of *Lecture Notes in Computer Science*, pages 159–168. Springer, 2012.

[11] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[12] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer Verlag, New York, 4th edition, 2010.

[13] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer Verlag, New York, 1999.

[14] E. Fischer, J. A. Makowsky, and E. R. Ravve. Counting truth assignments of formulas of bounded tree-width or clique-width. *Discr. Appl. Math.*, 156(4):511–529, 2008.

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

[16] S. Gaspers, S. Ordyniak, M. S. Ramanujan, S. Saurabh, and S. Szeider. Backdoors to q-Horn. In *Proceedings of the 30th Symposium on Theoretical Aspects of Computer Science (STACS 2013)*, volume 20 of *LIPIcs*, pages 67–79. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013.

[17] S. Gaspers and S. Szeider. Backdoors to acyclic SAT. In *Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP 2012)*, volume 7391 of *Lecture Notes in Computer Science*, pages 363–374. Springer, 2012.

[18] S. Gaspers and S. Szeider. Backdoors to satisfaction. In H. L. Bodlaender, R. G. Downey, F. V. Fomin, and D. Marx, editors, *The Multivariate Algorithmic Revolution and Beyond: Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*, pages 287–317. Springer Verlag, 2012.

[19] S. Gaspers and S. Szeider. Strong backdoors to nested satisfiabiliy. In *Proceedings of the 15th International Conference on Theory and Applications of Satisfiability Testing (SAT 2012)*, volume 7317 of *Lecture Notes in Computer Science*, pages 72–85. Springer, 2012.

[20] C. P. Gomes, H. Kautz, A. Sabharwal, and B. Selman. Satisfiability solvers. In *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 89–134. Elsevier, 2008.

[21] M. Grohe, K. ichi Kawarabayashi, D. Marx, and P. Wollan. Finding topological subgraphs is fixed-parameter tractable. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, (STOC 2011)*, pages 479–488. ACM, 2011.

[22] L. A. Hemaspaandra and R. Williams. SIGACT News Complexity Theory Column 76: an atypical survey of typical-case heuristic algorithms. *SIGACT News*, 43(4):70–89, 2012.

[23] E. J. Kim, A. Langer, C. Paul, F. Reidl, P. Rossmanith, I. Sau, and S. Sikdar. Linear kernels and single-exponential algorithms via protrusion decompositions. Technical Report 1207.0835, arXiv CoRR, 2012.

[24] D. E. Knuth. Nested satisfiability. *Acta Informatica*, 28(1):1–6, 1990.

[25] J. Lagergren. Upperbounds on the size of obstructions and intertwines. *Journal of Combinatorial Theory, Series B*, 73(1):7–40, 1998.

[26] L. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.

[27] Z. Li and P. van Beek. Finding small backdoors in SAT instances. In *Proceedings of the 24th Canadian Conference on Artificial Intelligence (Canadian AI 2011)*, volume 6657 of *Lecture Notes in Computer Science*, pages 269–280. Springer, 2011.

[28] W. Mader. Homomorphiesätze für Graphen. *Mathematische Annalen*, 178:154–168, 1968.

[29] D. Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1):60–78, 2008.

[30] N. Nishimura, P. Ragde, and S. Szeider. Detecting backdoor sets with respect to Horn and binary clauses. In *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing (SAT 2004)*, pages 96–103, 2004.

[31] N. Nishimura, P. Ragde, and S. Szeider. Solving #SAT using vertex covers. *Acta Informatica*, 44(7-8):509–523, 2007.

[32] I. Razgon and B. O'Sullivan. Almost 2-SAT is fixed parameter tractable. *Journal of Computer and System Sciences*, 75(8):435–450, 2009.

[33] N. Robertson, P. Seymour, and R. Thomas. Quickly excluding a planar graph. *J. Combin. Theory Ser. B*, 62(2):323–348, 1994.

[34] N. Robertson and P. D. Seymour. Disjoint paths—a survey. *SIAM J. Algebraic Discrete Methods*, 6(2):300–305, 1985.

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

[36] N. Robertson and P. D. Seymour. Graph minors. V. Excluding a planar graph. *J. Combin. Theory Ser. B*, 41(1):92–114, 1986.

[37] N. Robertson and P. D. Seymour. Graph minors X. Obstructions to tree-decomposition. *J. Combin. Theory Ser. B*, 52(2):153–190, 1991.

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

[39] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC 1978)*, pages 216–226. ACM, 1978.

[40] J. P. Spinrad. *Efficient Graph Representations.* Fields Institute Monographs. AMS, 2003.

[41] A. Thomason. The extremal function for complete minors. *Journal of Combinatorial Theory, Series B*, 81(2):318–338, 2001.

[42] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[43] R. Williams, C. Gomes, and B. Selman. Backdoors to typical case complexity. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 1173–1178, 2003.