Solving #SAT Using Vertex Covers

Naomi Nishimura[1,⋆], Prabhakar Ragde[1,⋆], and Stefan Szeider[2,⋆⋆]

[1] School of Computer Science, University of Waterloo,
Waterloo, Ontario, N2L 3G1, Canada
{nishi, plragde}@uwaterloo.ca

[2] Department of Computer Science, Durham University,
Durham DH1 3LE, England, United Kingdom
stefan.szeider@durham.ac.uk

Abstract. We propose an exact algorithm for counting the models of propositional formulas in conjunctive normal form (CNF). Our algorithm is based on the detection of strong backdoor sets of bounded size; each instantiation of the variables of a strong backdoor set puts the given formula into a class of formulas for which models can be counted in polynomial time. For the backdoor set detection we utilize an efficient vertex cover algorithm applied to a certain "obstruction graph" that we associate with the given formula. This approach gives rise to a new hardness index for formulas, the clustering-width. Our algorithm runs in uniform polynomial time on formulas with bounded clustering-width.

It is known that the number of models of formulas with bounded clique-width, bounded treewidth, or bounded branchwidth can be computed in polynomial time; these graph parameters are applied to formulas via certain (hyper)graphs associated with formulas. We show that clustering-width and the other parameters mentioned are incomparable: there are formulas with bounded clustering-width and arbitrarily large clique-width, treewidth, and branchwidth. Conversely, there are formulas with arbitrarily large clustering-width and bounded clique-width, treewidth, and branchwidth.

1 Introduction

#SAT is the problem of determining the number of satisfying truth assignments or models of a given propositional formula in conjunctive normal form (CNF). This problem is computationally equivalent to several problems that arise in automatic reasoning and artificial intelligence. However, since the problem is #P-complete (Valiant [27]), it is very unlikely that it can be solved in polynomial time. #SAT remains #P-hard even for monotone 2CNF formulas and Horn 2CNF formulas, and it is NP-hard to approximate the number of models of a formula with n variables within $2^{n^{1-\epsilon}}$ for $\epsilon > 0$. This approximation hardness holds also for monotone 2CNF formulas and Horn 2CNF formulas [23].

⋆ Supported by the Natural Science and Engineering Research Council of Canada.
⋆⋆ Research partially supported by the Nuffield Foundation (NAL/01012/G).

A. Biere and C.P. Gomes (Eds.): SAT 2006, LNCS 4121, pp. 396–409, 2006.

An alternative to restricting the language of formulas is to impose *structural restrictions* in terms of certain (hyper)graphs associated with formulas. In particular, graph parameters that restrict the structure of associated primal graphs, incidence graphs, and formula hypergraphs have been considered; see Sect. 8 for definitions of the various graphs and graph parameters. Bacchus, Dalmao, and Pitassi [1] propose an algorithm that solves #SAT in time $n^{O(1)}2^{O(k)}$ for formulas with n variables whose formula hypergraphs have *branchwidth* k. The algorithm is based on the DPLL procedure and uses caching techniques for an efficient reuse of solutions for subproblems. A similar time complexity can be achieved by restricting the *treewidth* of primal graphs and by dynamic programming on tree-decompositions; this approach is described by Gottlob, Scarcello, and Sideri [12] for SAT and can be extended to #SAT in a straight-forward way. Bounding the *clique-width* of directed incidence graphs yields larger classes of formulas for which #SAT is tractable: Fisher, Makowsky, and Ravve [8] obtain an algorithm for #SAT by combining Oum and Seymour's approximation algorithm for clique-width [21] with a general result of Courcelle, Makowsky, and Rotics [4] on counting problems expressible in a certain fragment of Monadic Second Order Logic. The algorithm solves #SAT in time $n^{O(1)}O(f(k))$ for formulas with n variables whose directed incidence graphs have clique-width k; here f denotes a simply exponential function. The latter result is more general than the results for bounded treewidth and branchwidth in the sense that every class of formulas with bounded treewidth or bounded branchwidth also has bounded clique-width; however, there are classes of formulas with bounded clique-width but unbounded treewidth and unbounded branchwidth, see Sect. 8. Practical application of the clique-width based algorithm is, however, very limited due to a huge hidden constant in the estimation of its running time.

Note that the algorithms considered above are so-called *fixed-parameter algorithms*, since the bound on the running time is, although exponential in the parameter k, uniformly polynomial in n. The main advantage of fixed-parameter algorithms is that the running time increases moderately when n becomes large, in contrast to algorithms with running time $n^{O(k)}$. We will review the basic concepts of parameterized complexity in Sect. 2.2.

1.1 Our Approach: Backdoor Sets

The concept of strong backdoor sets with respect to a base class $\mathcal{C}$ of formulas was introduced by Williams, Gomes, and Selman [28] as a tool for analyzing the performance of local search SAT algorithms. Backdoor sets have recently received a lot of attention in satisfiability research [14,16,18,20,24,26].

A set B of variables of a formula F is a *strong* $\mathcal{C}$*-backdoor set* if for all truth assignments $\tau : B \to \{0,1\}$, the restriction $F[\tau]$ of F to τ belongs to the base class $\mathcal{C}$. Note that if a strong $\mathcal{C}$-backdoor set of size k is found, then we can decide the satisfiability of the given formula by deciding the satisfiability of 2^k formulas that belong to the base class $\mathcal{C}$. Based on this concept, Nishimura, Ragde, and Szeider [20] propose algorithms for SAT that search for strong backdoor sets of bounded size with respect to the base classes HORN and 2CNF. The detection of strong backdoor sets is based on the fact that a set B of variables is a strong HORN-backdoor set (strong 2CNF-backdoor set) of a formula F if and only if $F - B$ is a Horn formula (2CNF formula, respectively); here $F - B$ denotes the formula obtained from F by removing all the literals $x, \overline{x}$ for $x \in B$ from the clauses of F. We also say that B is a *deletion $\mathcal{C}$-backdoor set* if $F - B \in \mathcal{C}$. In general, deletion $\mathcal{C}$-backdoor sets are not necessarily strong $\mathcal{C}$-backdoor sets. However, if all subsets of a formula in $\mathcal{C}$ also belong to $\mathcal{C}$ ($\mathcal{C}$ is *clause-induced*), then indeed deletion $\mathcal{C}$-backdoor sets are strong $\mathcal{C}$-backdoor sets.

In this paper we extend the algorithmic use of backdoor sets for SAT to the counting problem #SAT. It is easy to see that the number of models of a formula F equals the sum over the number of models of the restrictions $F[\tau]$ for all truth assignments $\tau : B \to \{0,1\}$ for a set B of variables of F. Hence, if we can solve #SAT for the elements of a base class $\mathcal{C}$ in polynomial time, then we can solve #SAT for a formula F in time $O(2^k n^{O(1)})$ provided that we know a strong $\mathcal{C}$-backdoor set of F of size at most k. Hence, to convert the above considerations into an algorithm for #SAT, we need to identify a base class $\mathcal{C}$ for which the following holds:

1. #SAT can be solved in polynomial time for formulas in $\mathcal{C}$, and
2. for a given formula F we can find strong $\mathcal{C}$-backdoor sets of bounded size efficiently.

The second condition can be relaxed to deletion $\mathcal{C}$-backdoor sets if $\mathcal{C}$ is clause-induced.

To this end, we introduce the clause-induced class CLU of *cluster formulas*. A cluster formula is a variable-disjoint union of so-called *hitting formulas*; any two clauses of a hitting formula clash in at least one literal. The known polynomial-time algorithm for computing the number of models of a hitting formula can be extended in a straight-forward way to compute the number of models of a cluster formula.

A strong CLU-backdoor set of size k of a formula F with n variables can obviously be found by exhaustive search, considering all $O(n^k)$ sets of k variables. This approach does not yield a fixed-parameter algorithm and becomes inefficient for large n even if k is small. We show in Sect. 5 that under a certain complexity theoretic assumption, there is no algorithm that is significantly faster than exhaustive search. We overcome this limitation by restricting by k the size of a smallest deletion CLU-backdoor set. We propose a fixed-parameter algorithm that either finds for a given formula a *strong* CLU-backdoor set of size at most k or decides that the given formula has no *deletion* CLU-backdoor set of size at most k.

To develop such an algorithm, we proceed as follows. We associate with every formula F a certain graph $G(F)$, the *obstruction graph* of F, which can be obtained in polynomial time. The vertex set of $G(F)$ is the set of variables of F. We show that every *vertex cover* of $G(F)$ is a strong CLU-backdoor set of F; recall that a vertex cover is a set S of vertices such that every edge is incident with a vertex in S. Now we can apply known vertex cover algorithms, e.g., the algorithm of Chen, Kanj, and Xia [3] for the detection of strong CLU-backdoor sets. Of related interest is Gramm et al.'s work [11] on a graph editing problem involving *cluster graphs* (i.e., disjoint unions of cliques).

1.2 Clustering-Width

We define the *clustering-width* of a formula F as the size of a smallest vertex cover of the obstruction graph of F. It follows from our results that the clustering-width of a formula F is a lower bound on the size of a smallest deletion CLU-backdoor set of F and an upper bound on the size of a smallest strong CLU-backdoor set of F.

Finally, we exhibit a class of formulas of bounded clustering-width for which all the parameters clique-width, branchwidth, and treewidth are unbounded. We also exhibit a class of formulas with unbounded clustering-width for which all the parameters clique-width, branchwidth, and treewidth are bounded. In other words, there are formulas that are easy for our algorithm and arbitrarily hard for the known algorithms, and formulas where the converse prevails.

It would be interesting to complement our theoretical results with empirical evidence on the significance of our new parameter. In particular, it would be interesting to know the clustering-width of CNF formulas that encode real-world instances from different domains. However, one must choose the encoding carefully in order to avoid a large clustering-width caused by the gadgets of the encoding itself. On the other hand, as indicated above, it can be checked very efficiently whether a CNF formula has small clustering-width. Hence, any other #SAT algorithm can be extended by a subroutine that checks the clustering-width and performs our algorithm if the clustering-width is small.

2 Preliminaries

2.1 SAT and #SAT

We consider propositional formulas in conjunctive normal form (CNF), represented as sets of clauses. That is, a *literal* is a (propositional) variable x or a negated variable $\overline{x}$; a *clause* is a finite set of literals not containing a complementary pair x and $\overline{x}$; a *formula* is a finite set clauses. For a literal $\ell = \overline{x}$ we write $\overline{\ell} = x$; for a clause C we put $\overline{C} = \{\, \overline{\ell} : \ell \in C \,\}$. For a clause C, $\mathrm{var}(C)$ denotes the set of variables x with $x \in C$ or $\overline{x} \in C$. Similarly, for a formula F we write $\mathrm{var}(F) = \bigcup_{C \in F} \mathrm{var}(C)$.

We say that two clauses C, D *overlap* if $C \cap D \neq \emptyset$; we say that C and D *clash* if C and $\overline{D}$ overlap. Note that two clauses can clash and overlap at the same time.

A *truth assignment* (or *assignment*, for short) is a mapping $\tau : X \to \{0,1\}$ defined on some set X of variables. We extend τ to literals by setting $\tau(\overline{x}) = 1 - \tau(x)$ for $x \in X$. $F[\tau]$ denotes the formula obtained from F by removing all clauses that contain a literal x with $\tau(x) = 1$ and by removing from the remaining clauses all literals y with $\tau(y) = 0$; $F[\tau]$ is the *restriction* of F to τ. Note that $\mathrm{var}(F[\tau]) \cap X = \emptyset$ holds for every assignment $\tau : X \to \{0,1\}$ and every formula F. A truth assignment $\tau : X \to \{0,1\}$ *satisfies* a formula F if $F[\tau] = \emptyset$. A truth assignment $\tau : \mathrm{var}(F) \to \{0,1\}$ that satisfies F is a *model* of F. We denote by $\#(F)$ the number of models of F. A formula F is *satisfiable* if $\#(F) > 0$. The satisfiability problem SAT is the problem of deciding whether a given formula is satisfiable. #SAT, the counting version of SAT, is the problem of determining $\#(F)$ for a given formula F. SAT and #SAT are complete problems for the complexity classes NP and #P, respectively.

The following concept of connectedness of formulas will be useful below. We call a formula F *connected* if for any two clauses $C, D \in F$ there exists a sequence of clauses $C_1, \ldots, C_r \in F$ such that $C_1 = C$, $C_r = D$, and $\mathrm{var}(C_i) \cap \mathrm{var}(C_{i+1}) \neq \emptyset$ holds for all $i \in \{1, \ldots, r-1\}$. A maximal connected subset of a formula is a *connected component.*

2.2 Parameterized Complexity

Next we give a brief and rather informal review of the most important concepts of parameterized complexity. For an in-depth treatment of the subject we refer the reader to other sources [7,19].

The instances of a parameterized problem can be considered as pairs (I, k) where I is the *main part* of the instance and k is the *parameter* of the instance; the latter is usually a non-negative integer. A parameterized problem is *fixed-parameter tractable* if instances (I, k) of size n can be solved in time $O(f(k)n^c)$ where f is a computable function and c is a constant independent of k.

The framework of parameterized complexity offers a *completeness theory*, similar to the theory of NP-completeness, that allows the accumulation of strong theoretical evidence that a parameterized problem is *not* fixed-parameter tractable. This completeness theory is based on the *weft hierarchy* of equivalence classes $\mathrm{W}[1], \mathrm{W}[2], \ldots, \mathrm{W}[\mathrm{P}]$ of certain parameterized decision problems under *parameterized reductions.* A parameterized reduction is a straightforward extension of a polynomial-time many-one reduction that ensures a parameter for one problem maps into a parameter for another (see [7] for details).

Below we will refer to the following parameterized decision problem, which is known to be W[2]-complete [7].

HITTING SET
Instance: A family $\mathcal{S}$ of finite sets $S_1, \ldots, S_m$.
Parameter: An integer $k \geq 0$.
Question: Is there a subset $R \subseteq \bigcup_{i=1}^{m} S_i$ of size at most k such that $R \cap S_i \neq \emptyset$ for all $i = 1, \ldots, m$? (R is a *hitting set* of $\mathcal{S}$)

3 Backdoor Sets

Consider a *base class* $\mathcal{C}$ of formulas for which the problems #SAT and recognition can be solved in polynomial time. Furthermore, consider a formula F and a set B of variables of F. A set $B \subseteq \mathrm{var}(F)$ is a *strong backdoor set* of F with respect to $\mathcal{C}$ (or *strong $\mathcal{C}$-backdoor set*, for short) if $B \subseteq \mathrm{var}(F)$ and for every truth assignment $\tau : B \to \{0,1\}$ we have $F[\tau] \in \mathcal{C}$. For every formula F and every set $B \subseteq \text{var}(F)$ we have $\#(F) = \sum_{\tau:B\to\{0,1\}} \#(F[\tau])$. Thus, if B is a strong $\mathcal{C}$-backdoor set of a formula F, then determining $\#(F)$ reduces to determining the number of satisfying assignments for $2^{|B|}$ formulas of the base class $\mathcal{C}$. Thus, when we have found a small strong $\mathcal{C}$-backdoor set of F, we can compute $\#(F)$ efficiently. A key question is whether we can find a small backdoor set if it exists. To study this question, we define for every base class $\mathcal{C}$ the following parameterized problem.

STRONG $\mathcal{C}$-BACKDOOR
Input: A formula F.
Parameter: A positive integer k.
Question: Does F have a strong $\mathcal{C}$-backdoor set of size at most k?

For base classes that have a certain property, we can relax the problem STRONG $\mathcal{C}$-BACKDOOR as follows. For a formula F and a set X of variables let $F - X$ denote the formula obtained from F by removing all literals x and $\overline{x}$ from the clauses of F. We call a set $B \subseteq \text{var}(F)$ a *deletion backdoor set* with respect to a base class $\mathcal{C}$ (or *deletion $\mathcal{C}$-backdoor set*, for short) if $F - B \in \mathcal{C}$. Furthermore, we define a base class $\mathcal{C}$ to be *clause-induced* if for every $F \in \mathcal{C}$ and every $F' \subseteq F$, also $F' \in \mathcal{C}$.

Lemma 1. *Let F be a formula and $\mathcal{C}$ a clause-induced base class. Every deletion $\mathcal{C}$-backdoor set of F is also a strong $\mathcal{C}$-backdoor set.*

Proof. The result follows directly from the fact that $F[\tau] \subseteq F - X$ holds for every truth assignment $\tau : X \to \{0,1\}$. $\square$

For a base class $\mathcal{C}$, deletion backdoor sets can be larger than strong backdoor sets. However, if the detection of strong $\mathcal{C}$-backdoor sets is fixed-parameter intractable, we can still hope that the detection of deletion $\mathcal{C}$-backdoor sets is fixed-parameter tractable. We state the corresponding parameterized problem:

DELETION $\mathcal{C}$-BACKDOOR
Input: A formula F.
Parameter: A positive integer k.
Question: Does F have a deletion $\mathcal{C}$-backdoor set of size at most k?

4 Hitting Formulas and Cluster Formulas

A formula is a *hitting formula* if any two of its clauses clash (see [17]). A *cluster formula* is the variable-disjoint union of hitting formulas. In other words, a formula is a cluster formula if and only if all its connected components are hitting formulas. We denote the class of all hitting formulas by HIT and the class of all cluster formulas by CLU.

The next lemma is due to an observation of Iwama [15].

Lemma 2. *A hitting formula F with n variables has exactly $2^n - \sum_{C \in F} 2^{n-|C|}$ models.*

Proof. Let F be a hitting formula with n variables. For a clause $C \in F$ let T_C denote the set of all truth assignments $\tau : \mathrm{var}(F) \to \{0,1\}$ that *do not* satisfy C. Obviously $|T_C| = 2^{n-|C|}$ since T_C contains exactly those assignments that set all literals in C to 0. Since F is a hitting formula, the sets T_C and $T_{C'}$ are disjoint for any two distinct clauses $C, C' \in F$. Hence the lemma follows. □

Lemma 3. #SAT *can be solved in polynomial time for cluster formulas.*

Proof. If a formula F is the variable-disjoint union of formulas $F_1, \dots, F_q$, then $\#(F) = \prod_{i=1}^{q} \#(F_i)$. Thus the result follows directly from Lemma 2. □

By means of the previous lemma we can consider CLU as the base class for a backdoor set approach to #SAT. Observe that CLU is clause-induced.

5 Finding Smallest Strong CLU-Backdoor Sets

In this section we show that the detection of strong CLU-backdoor sets is fixed-parameter intractable.

We shall use the following construction. Let D be a directed graph. We associate with D a formula F_D where every arc a of D corresponds to a variable x_a of F, and every vertex v of D corresponds to a clause C_v of F. The clause C_v contains the literals x_a for outgoing arcs a of v, and the literals $\overline{x_b}$ for incoming arcs b of v. Note that if D is the orientation of a complete graph, then F_D is a hitting formula.

Theorem 1. *The problem* STRONG CLU-BACKDOOR is W[2]-hard.

Proof. (Sketch.) We give a parameterized reduction from the W[2]-complete problem HITTING SET as defined in Sect. 2.2. Let $\mathcal{S} = S_1, \dots, S_m$ be an instance of HITTING SET; $\bigcup_{i=1}^{m} S_i = \{x_1, \dots, x_n\}$. Let D be an orientation of a complete graph with $r = (m+1)(k+1)$ vertices. Consider the hitting formula F_D. We partition F_D into formulas $F_1, \dots, F_m, H$ such that each of the partite sets contains exactly $k+1$ clauses. For $i = 1, \dots, m$ we put

$$F'_i = \{\, C \cup S_i : C \in F_i \,\}.$$

Finally, we put $C^* = \{\overline{x_1}, \dots, \overline{x_n}\}$ and

$$F = \{C^*\} \cup \bigcup_{i=1}^{m} F'_i \cup H.$$

We can show that $\mathcal{S}$ has a hitting set of size at most k if and only if F has a strong CLU-backdoor set of size at most k. □

The NP-hardness of the non-parameterized version of STRONG CLU-BACKDOOR (where the parameter is taken as part of the input) follows from the proof of Theorem 1.

We will show in sections below that the concept of deletion backdoor sets can be used to find small strong backdoor sets with respect to CLU. Next we give an example that shows that for the base class CLU, smallest deletion backdoor sets can be larger that smallest strong backdoor sets.

Consider the formula

$$F = \{\{x_1, \ldots, x_n\}, \{\overline{x_1}, \ldots, \overline{x_n}, y_1, \ldots, y_n\}, \{\overline{y_1}, \ldots, \overline{y_n}\}\}.$$

Note that each of the variables of F forms a strong CLU-backdoor set of F; e.g., $B = \{x_1\}$ is a strong CLU-backdoor set. However, we need to delete at least n variables in order to obtain a cluster formula. Thus a smallest strong CLU-backdoor set of F has size 1, but every deletion CLU-backdoor set of F has size at least n.

6 Obstructions

In the following results, it is helpful to characterize cluster formulas in terms of obstructions. An *overlap obstruction* is a formula $\{C_1, C_2\}$ consisting of two clauses that overlap but do not clash. With an overlap obstruction we associate the following pair of sets of variables:

$$\{\mathrm{var}(C_1 \cap C_2),\ \mathrm{var}(C_1 \bigtriangleup C_2)\}.$$

Here $C_1 \bigtriangleup C_2$ denotes the symmetric difference $(C_1 \setminus C_2) \cup (C_2 \setminus C_1)$ of C_1 and C_2. A *clash obstruction* is a formula $\{C_1, C_2, C_3\}$ where C_1 and C_2 clash such that $(C_1 \setminus C_3) \cap \overline{C_2} \neq \emptyset$, C_2 and C_3 clash such that $(C_3 \setminus C_1) \cap \overline{C_2} \neq \emptyset$, and C_1 and C_3 do not clash. (Any two of the three clauses may overlap.) With a clash obstruction we associate the following pair of sets of variables:

$$\{\mathrm{var}((C_1 \setminus C_3) \cap \overline{C_2}),\ \mathrm{var}((C_3 \setminus C_1) \cap \overline{C_2})\}.$$

We say that an overlap or clash obstruction F' is an *obstruction of* a formula F if F' is a subset of F. A pair $\{X, Y\}$ of sets of variables is a *deletion pair* of F if the pair is associated with an overlap or clash obstruction of F. It follows from the definitions of overlap and clash obstructions that the two sets in a deletion pair are nonempty and disjoint.

Lemma 4. *A formula is a cluster formula if and only if it has no overlap or clash obstruction.*

Proof. If a formula F contains an overlap or clash obstruction, then there are two clauses $C, D \in F$ that belong to the same connected component of F but do not clash. Hence F is not a cluster formula.

Conversely, consider a formula F that does not contain any overlap or clash obstructions. We show that F is a cluster formula. Consider a connected component F' of F. If $|F| = 1$ then F' is a hitting formula; hence assume $|F| > 1$. We show that any two clauses of F' clash. Choose two arbitrary clauses $C, D \in F'$.

Since F' is connected, there is a sequence of clauses $C_1, \dots, C_r \in F$ such that $C_1 = C$, $C_r = D$, and $\mathrm{var}(C_i) \cap \mathrm{var}(C_{i+1}) \neq \emptyset$ holds for all $i \in \{1, \dots, r-1\}$. We observe that C_i and C_{i+1} clash for all and $i \in \{1, \dots, r-1\}$ since otherwise C_i and C_{i+1} would form an overlap obstruction. It now follows inductively that the clauses C_1 and C_i clash for all $i \in \{3, \dots, r\}$ since otherwise C_1, C_{i-1}, and C_i would form a clash obstruction. Thus, indeed, C and D clash. Whence F' is a hitting formula. □

The next result is a consequence of Lemma 4. We omit the proof due to space limitations.

Lemma 5. *Let F be a formula and $B \subseteq \mathrm{var}(F)$. If $F - B$ is a cluster formula, then $X \subseteq B$ or $Y \subseteq B$ holds for every deletion pair $\{X, Y\}$ of F.*

7 Finding Backdoor Sets Using Vertex Covers

For a formula F let G_F denote the graph with vertex set $\mathrm{var}(F)$; two variables x and y are joined in G_F by an edge if and only if there is a deletion pair $\{X, Y\}$ of F with $x \in X$ and $y \in Y$. We call G_F the *obstruction graph* of F. Note that the obstruction graph of a formula can be constructed in polynomial time.

We consider vertex covers of obstruction graphs. Recall that a *vertex cover* of a graph is a set of vertices that contains at least one end of every edge of the graph. It is NP-hard to determine, given a graph and an integer k, whether the graph has a vertex cover of size at most k. Parameterized by the size of the vertex cover, however, the problem is fixed-parameter tractable. In fact, vertex cover is the best studied problem in parameterized complexity with a long history of improvements. The current best worst-case time complexity for the parameterized vertex cover problem is due to Chen, Kanj, and Xia [3]:

Theorem 2. *Given a graph G on n vertices, one can find in time $O(1.273^k + nk)$ (and in polynomial space) a vertex cover of G of size at most k, or determine that no such vertex cover exists.*

The next two lemmas relate backdoor sets and vertex covers of obstruction graphs. The first is a direct consequence of Lemma 5.

Lemma 6. *Every deletion* CLU*-backdoor set of a formula F is a vertex cover of the obstruction graph of F.*

Lemma 7. *Every vertex cover of the obstruction graph of a formula F is a strong* CLU*-backdoor set of F.*

Proof. (Sketch.) Let B be a vertex cover of the obstruction graph of a formula F. Assume to the contrary that B is not a strong CLU-backdoor set of F. Thus, there is an assignment $\tau : B \to \{0, 1\}$ such that $F[\tau] \notin \mathrm{CLU}$. Let $B_0 = \{\, y \in B \cup \overline{B} : \tau(y) = 0 \,\}$; i.e., B_0 is the set of all literals over variables of B that are mapped to 0 under τ. By Lemma 4, $F[\tau]$ contains overlap or clash obstructions.

We assume that $F[\tau]$ contains an overlap obstruction; for clash obstructions the argument is similar. Let C_1, C_2 be two clauses of $F[\tau]$ that overlap but do not clash. For the associated obstruction pair $\{X, Y\}$ with $X = \mathrm{var}(C_1 \cap C_2)$ and $Y = \mathrm{var}(C_1 \bigtriangleup C_2)$ choose $x \in X$ and $y \in Y$. By definition of $F[\tau]$ it follows that F contains clauses C_1', C_2' with $C_1 = C_1' \setminus B_0$ and $C_2 = C_2' \setminus B_0$. It follows that C_1' and C_2' overlap but do not clash, thus $\{C_1', C_2'\}$ is an overlap obstruction of F. We have $x \in X \subseteq \mathrm{var}(C_1' \cap C_2')$ and $y \in Y \subseteq \mathrm{var}(C_1' \bigtriangleup C_2')$. Thus xy is an edge of G_F. Since B is a vertex cover of G_F, either x or y must belong to B. This contradicts the fact that $\mathrm{var}(F[\tau]) \cap B = \emptyset$. Whence it follows that B^* is indeed a strong CLU-backdoor set of F. □

From Theorem 2 and the previous two lemmas we get immediately the main result of this section.

Theorem 3. *Given a formula with n variables together with its obstruction graph and an integer k, in time $O(1.273^k + nk)$ we can find a strong* CLU*-backdoor set of F of size at most k, or decide that the size of every deletion* CLU*-backdoor set of F exceeds k.*

8 Comparison with Other Parameters

In this section we introduce a general framework for comparing parameters that allow fixed-parameter algorithms for #SAT. Here we consider as a parameter any computable function p that assigns to each formula F a non-negative integer $p(F)$. We assume that the parameter is invariant under changing the names of variables.

The following three parameters arise from the considerations of this paper. We denote by $\mathrm{str}_{\mathrm{CLU}}(F)$ the size of a smallest strong backdoor set of a formula F with respect to CLU, and we denote by $\mathrm{del}_{\mathrm{CLU}}(F)$ the size of a smallest deletion backdoor set of F with respect to CLU. The *clustering-width* $\mathrm{clu}(F)$ of F is the size of a smallest vertex cover of the obstruction graph of F. Consequently, HIT is the class of formulas with clustering-width 0. From Lemmas 1 and 6 we know that for every formula F the following holds:

$$\mathrm{str}_{\mathrm{CLU}}(F) \leq \mathrm{clu}(F) \leq \mathrm{del}_{\mathrm{CLU}}(F). \tag{1}$$

For a parameter p we consider the following generic parameterized problem.

#SAT(p)
Instance: A formula F and a non-negative integer k such that $p(F) \leq k$.
Parameter: The integer k.
Question: What is the total number of models of F? (I.e., what is the number $\#(F)$?)

The definition of fixed-parameter tractability carries over from decision problems to counting problems in a natural way. Flum and Grohe [9] provide a framework of intractability of parameterized counting problems.

Note that the above formulation of #SAT(p) is a "promise problem" in the sense that we only need to consider instances (F, k) for which we can take as granted that $p(F) \leq k$ holds. However, for most parameters p considered in the sequel for which #SAT(p) is fixed-parameter tractable, deciding whether $p(F) \leq k$ actually holds is also fixed-parameter tractable with respect to the parameter k. An exception is the parameter $\mathrm{del}_{\mathrm{CLU}}$; however, also in that case we do not depend on the promise as will be discussed below.

By Theorem 2, deciding whether $\mathrm{clu}(F) \leq k$ is fixed-parameter tractable; if $\mathrm{clu}(F) \leq k$, then it is also fixed-parameter tractable to produce a strong CLU-backdoor set B of F of size at most k. We then compute $\#(F)$ as the sum of $\#(F[\tau])$ over all truth assignments $\tau : B \to \{0, 1\}$. Whence we have the following corollary to Theorem 2.

Corollary 1. *The problem* #SAT(clu) *is fixed-parameter tractable.*

Note that the algorithm outlined above also checks whether the promise $\mathrm{clu}(F) \leq k$ is true. Furthermore, from (1) it follows that every instance (F, k) of #SAT($\mathrm{del}_{\mathrm{CLU}}$) is also an instance of #SAT(clu). Whence Corollary 1 also implies fixed-parameter tractability of #SAT($\mathrm{del}_{\mathrm{CLU}}$).

Corollary 2. *The problem* #SAT($\mathrm{del}_{\mathrm{CLU}}$) *is fixed-parameter tractable.*

Although we do not know whether DELETION $\mathcal{C}$-BACKDOOR is fixed-parameter tractable, we emphasize that the algorithm for Corollary 2 will not produce an incorrect solution, even if the promise $\mathrm{del}_{\mathrm{CLU}}(F) \leq k$ does not hold. Consider F and k with $\mathrm{del}_{\mathrm{CLU}}(F) > k$. The algorithm checks whether $\mathrm{clu}(F) \leq k$. If $\mathrm{clu}(F) \leq k$, then the algorithm outputs the correct solution #SAT(F). If, however, $\mathrm{clu}(F) > k$, then we know by (1) that also $\mathrm{del}_{\mathrm{CLU}}(F) > k$, hence the algorithm can reject the input.

8.1 Treewidth, Branchwidth, and Clique- Width

Several parameters are defined in terms of the following directed and undirected graphs associated with a formula F. The *primal graph* $P(F)$ is the graph whose vertices are the variables of F, and where two variables x and y are joined by an edge if and only if F contains a clause C with $x, y \in \mathrm{var}(C)$. The *incidence graph* $I(F)$ is the bipartite graph where one vertex class consists of the variables of F, the other vertex class consists of the clauses of F; a variable x and a clause C are joined by an edge if and only if $x \in \mathrm{var}(C)$. The *directed* or *signed* incidence graph $I_d(F)$ arises from $I(F)$ by orienting edges from C to x if $x \in C$, and from x to C if $\overline{x} \in C$. The *underlying graph* G_D of a directed graph D is the undirected graph obtained from D by "forgetting" the orientation of edges and by identifying possible parallel edges. Thus $I(F)$ is the underlying graph of $I_d(F)$. For an undirected graph G we consider its treewidth $\mathrm{tw}(G)$, its branchwidth $\mathrm{bw}(G)$, and its clique-width $\mathrm{cwd}(G)$; clique-width is also defined for directed graphs. For definitions of these graph parameters we refer the reader to related work [2,6,5,1,13,25]. By means of primal, incidence and directed incidence graphs, these graph parameters apply to formulas as follows: For a formula F we call $\mathrm{tw}(F) = \mathrm{tw}(P(F))$ the *primal treewidth* of F, $\mathrm{tw}^*(F) = \mathrm{tw}(I(F))$ the *incidence treewidth* of F, $\mathrm{bw}(F) = \mathrm{bw}(P(F))$ the *branchwidth* of F, $\mathrm{cwd}(F) = \mathrm{cwd}(I_d(F))$ the *clique-width* of F.

For two formula parameters p and q we say that p *dominates* q if there is a computable function f such that $p(F) \leq f(q(F))$ holds for all formulas F. We say that p and q are *incomparable* if neither p dominates q nor q dominates p. Note that if #SAT(p) is fixed-parameter tractable and p dominates q, then also #SAT(q) is fixed-parameter tractable. From known results it follows that clique-width dominates incidence treewidth, and that, in turn, incidence treewidth dominates primal treewidth and branchwidth [25]. Whence, clique-width can be considered as the most general parameter considered so far. Fischer, Makowsky, and Ravve [8] show that #SAT(cwd) is fixed-parameter tractable, combining an earlier result of Courcelle, Makowsky, and Rotics [5] and a recent result of Oum and Seymour [21]. By the above relationships among the various parameters, this result also implies the fixed-parameter tractability of #SAT(tw^*), #SAT(tw), and #SAT(bw):

Theorem 4. *The problems* #SAT(cwd)*,* #SAT(tw^*)*,* #SAT(tw)*, and* #SAT(bw)*, are fixed-parameter tractable.*

The question arises how our new parameter, the clustering-width, is related to the other parameters. Does any of the above parameters dominate clustering-width, or does clustering-width dominate any of the other parameters? We will show that the answer to both questions is 'no': clustering-width is *incomparable* with any of the other parameters.

Lemma 8. *The class* HIT *has unbounded clique-width.*

Proof. Let $n \geq 3$ be an integer and let G denote an $n \times n$ grid. That is, G is a bipartite graph with n^2 vertices $v_{i,j}$, $i, j \in \{1, \ldots, n\}$, where two vertices $v_{i,j}$ and $v_{i',j'}$ are joined by an edge if and only if either $i = i'$ and $|j - j'| = 1$, or $|i - i'| = 1$ and $j = j'$. Let V_1, V_2 be a bipartition of the vertex set of G. We obtain a formula F with $I(F) = G$ by considering vertices in V_1 as variables and putting $F = \{\, N(v_{i,j}) : v_{i,j} \in V_2 \,\}$; here $N(v_{i,j})$ denotes the set of neighbors of $v_{i,j}$ in G.

Consider a directed graph D whose underlying graph is the complete graph K_m for $m = |V_2|$. We construct the hitting formula F_D as described at the beginning of Sect. 5; we assume that F and F_D do not share variables. Observe that $|F_D| = m$; thus we can write $F = \{C_1, \ldots, C_m\}$ and $F_D = \{C_{1,D}, \ldots, C_{m,D}\}$, ordering the clauses arbitrarily.

Let H be the formula $\{C_1 \cup C_{1,D}, \ldots, C_m \cup C_{m,D}\}$. Clearly H is a hitting formula since F_D is a hitting formula. Golumbic and Rotics [10] show that the clique-width of $n \times n$ grids, $n \geq 3$, is exactly $n + 1$, hence $\mathrm{cwd}(G) = n + 1$. Note that $I(F) = G$ is isomorphic to a vertex-induced subgraph of $I(H)$; this implies that $\mathrm{cwd}(H) \geq \mathrm{cwd}(G) = n + 1$ (see Courcelle and Olariu [6]). Moreover, also noted by Courcelle and Olariu, the clique-width of a directed graph is at least as large as the clique-width of its underlying graph; hence we have $\mathrm{cwd}(I_d(H)) \geq \mathrm{cwd}(I(H)) \geq \mathrm{cwd}(I(F)) = \mathrm{cwd}(G) = n+1$. We conclude that for every positive integer n there exists a hitting formula H with $\mathrm{cwd}(H) > n$. □

Lemma 9. *The class of formulas with primal treewidth* 1 *has unbounded clustering-width.*

Proof. Let $\mathcal{C}$ denote the class of formulas with primal treewidth 1. Let n be an even positive integer and consider the formula

$$F = \{\{x_0, x_1\}, \{x_1, x_2\}, \ldots, \{x_{n-1}, x_n\}\}.$$

The primal graph of F is a path. Since paths have treewidth 1, $F \in \mathcal{C}$ follows.

For every $i = 1, \ldots, n-1$, the formula F contains the overlap obstruction $\{\{x_{i-1}, x_i\}, \{x_i, x_{i+1}\}\}$ with the corresponding deletion pair $\{\{x_i\}, \{x_{i-1}, x_{i+1}\}\}$. There are no clash obstructions. The obstruction graph is therefore a path P on the vertices $x_1, \ldots, x_n$. Any vertex cover of P contains at least $n/2$ vertices, hence $\mathrm{clu}(F) \geq n/2$ follows.

As we can choose arbitrarily large n, $\mathcal{C}$ has unbounded clustering-width. □

In view of the relationships omong the parameters cwd, tw*, tw, and bw stated above, the last two lemmas imply the following result.

Theorem 5. *The parameters* cwd, tw*, tw, *and* bw, *are all incomparable with clustering-width.*

References

1. F. Bacchus, S. Dalmao, and T. Pitassi. Algorithms and complexity results for #SAT and Bayesian Inference. In *44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, 340–351, 2003.
2. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoret. Comput. Sci.*, 209(1-2):1–45, 1998.
3. J. Chen, I. A. Kanj, and G. Xia. Simplicity is beauty: Improved upper bounds for vertex cover. Technical Report TR05-008, DePaul University, Chicago IL, 2005.
4. B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000.
5. B. Courcelle, J. A. Makowsky, and U. Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discr. Appl. Math.*, 108(1-2):23–52, 2001.
6. B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discr. Appl. Math.*, 101(1-3):77–114, 2000.
7. R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer Verlag, 1999.
8. E. Fischer, J. A. Makowsky, and E. R. Ravve. Counting truth assignments of formulas of bounded tree-width or clique-width. *Discr. Appl. Math.* to appear.
9. J. Flum and M. Grohe. The parameterized complexity of counting problems. *SIAM J. Comput.*, 33(4):892–922, 2004.

10. M. C. Golumbic and U. Rotics. On the clique-width of some perfect graph classes. *Internat. J. Found. Comput. Sci.*, 11(3):423–443, 2000. Selected papers from the Workshop on Graph-Theoretical Aspects of Computer Science (WG 99), Part 1 (Ascona).
11. J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-modeled data clustering: fixed-parameter algorithms for clique generation. *Theory Comput. Syst.*, 38(4):373–392, 2005.
12. G. Gottlob, F. Scarcello, and M. Sideri. Fixed-parameter complexity in AI and nonmonotonic reasoning. *Artificial Intelligence*, 138(1-2):55–86, 2002.
13. G. Gottlob and S. Szeider. Fixed-parameter algorithms for artificial intelligence, constraint satisfaction, and database problems. Submitted, April 2006.
14. Y. Interian. Backdoor sets for random 3-SAT. In *Informal Proceedings of SAT 2003*, 231–238, 2003.
15. K. Iwama. CNF-satisfiability test by counting and polynomial average time. *SIAM J. Comput.*, 18(2):385–391, 1989.
16. P. Kilby, J. K. Slaney, S. Thiébaux, and T. Walsh. Backbones and backdoors in satisfiability. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference (AAAI 2005)*, 1368–1373, 2005.
17. H. Kleine Büning and X. Zhao. Satisfiable formulas closed under replacement. In *Proceedings for the Workshop on Theory and Applications of Satisfiability*, volume 9 of *Electronic Notes in Discrete Mathematics*. Elsevier Science Publishers, North-Holland, 2001.
18. I. Lynce and J. P. Marques-Silva. Hidden structure in unsatisfiable random 3-SAT: An empirical study. In *16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004)*, 246–251. IEEE Computer Society, 2004.
19. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.
20. N. Nishimura, P. Ragde, and S. Szeider. Detecting backdoor sets with respect to Horn and binary clauses. In *Informal Proceedings of SAT 2004*, 96–103, 2004.
21. S. Oum and P. Seymour. Approximating clique-width and branch-width. *J. Combin. Theory, Ser. B*, to appear.
22. N. Robertson and P. D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *J. Combin. Theory Ser. B*, 52(2):153–190, 1991.
23. D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.
24. Y. Ruan, H. A. Kautz, and E. Horvitz. The backdoor key: A path to understanding problem hardness. In *Proceedings of the 19th National Conference on Artificial Intelligence, 16th Conference on Innovative Applications of Artificial Intelligence*, 124–130. AAAI Press / The MIT Press, 2004.
25. S. Szeider. On fixed-parameter tractable parameterizations of SAT. In *Theory and Applications of Satisfiability, 6th International Conference, SAT 2003, Selected and Revised Papers*, *LNCS*, vol. 2919, 188–202. Springer, 2004.
26. S. Szeider. Backdoor sets for DLL subsolvers. *Journal of Automated Reasoning*, 2005. In press.
27. L. G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8(2):189–201, 1979.
28. R. Williams, C. Gomes, and B. Selman. On the connections between backdoors, restarts, and heavy-tailedness in combinatorial search. In *Informal Proceedings of SAT 2003*, 222–230, 2003.