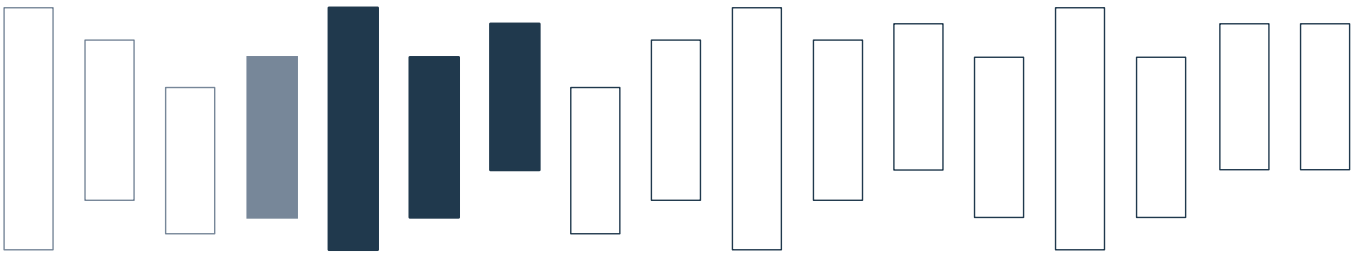
Technical Report AC-TR-17-013

December 2017

New Width Parameters for Model Counting

Robert Ganian and Stefan Szeider



This is the authors' copy of a paper that appeared in S. Gaspers and T. Walsh (Eds.): SAT 2017, LNCS 10491, pp. 38–52, 2017. DOI: 10.1007/978-3-319-66263-3_3

www.ac.tuwien.ac.at/tr

New Width Parameters for Model Counting

Robert Ganian( ) and Stefan Szeider

Algorithms and Complexity Group, TU Wien, Vienna, Austria
{ganian,sz}@ac.tuwien.ac.at

Abstract. We study the parameterized complexity of the propositional model counting problem #SAT for CNF formulas. As the parameter we consider the treewidth of the following two graphs associated with CNF formulas: the consensus graph and the conflict graph. Both graphs have as vertices the clauses of the formula; in the consensus graph two clauses are adjacent if they do not contain a complementary pair of literals, while in the conflict graph two clauses are adjacent if they do contain a complementary pair of literals. We show that #SAT is fixed-parameter tractable for the treewidth of the consensus graph but W[1]-hard for the treewidth of the conflict graph. We also compare the new parameters with known parameters under which #SAT is fixed-parameter tractable.

1 Introduction

Propositional model counting (#SAT) is the problem of determining the number of models (satisfying truth assignments) of a given propositional formula in conjunctive normal form (CNF). This problem arises in several areas of artificial intelligence, in particular in the context of probabilistic reasoning [1,23]. The problem is well-known to be #P-complete [29], and remains #P-hard even for monotone 2CNF formulas and Horn 2CNF formulas. Thus, in contrast to the decision problem SAT, restricting the syntax of instances does not lead to tractability.

An alternative to restricting the syntax is to impose *structural restrictions* on the input formulas. Structural restrictions can be applied in terms of certain parameters (invariants) of *graphical models*, i.e., of certain graphs associated with CNF formulas. Among the most frequently used graphical models are *primal graphs* (sometimes called *variable interaction graphs* or VIGs), *dual graphs*, and *incidence graphs* (see Fig. 1 for definitions and examples).

The most widely studied and prominent graph parameter is *treewidth*, which was introduced by Robertson and Seymour in their Graph Minors Project. Small treewidth indicates that a graph resembles a tree in a certain sense (e.g., trees have treewidth 1, cyles have treewidth 2, cliques on $k+1$ vertices have treewidth k). Many otherwise NP-hard graph problems are solvable in polynomial time for graphs of bounded treewidth. It is generally believed that many practically

Supported by the Austrian Science Fund (FWF), project P26696. Robert Ganian is also affiliated with FI MU, Brno, Czech Republic.

S. Gaspers and T. Walsh (Eds.): SAT 2017, LNCS 10491, pp. 38–52, 2017.
DOI: 10.1007/978-3-319-66263-3_3

relevant problems actually do have low treewidth [2]. Treewidth is based on certain decompositions of graphs, called tree decompositions, where sets of vertices ("bags") of a graph are arranged at the nodes of a tree such that certain conditions are satisfied (see Sect. 2.3). If a graph has treewidth k then it admits a tree decomposition of *width* k, i.e., a tree decomposition where all bags have size at most $k + 1$.

Depending on whether we consider the treewidth of the primal, dual, or incidence graph of a given CNF formula, we speak of the primal, dual, or incidence treewidth of the formula, respectively. It is known that the number of models of a CNF formula of size L with primal, dual, or incidence treewidth k can be computed in time $f(k)L^c$ for a computable function f and a constant c which is independent of k; in other words, #SAT is *fixed-parameter tractable* parameterized by primal, dual, or incidence treewidth (see, e.g., [26]).

1.1 Contribution

In this paper we consider the treewidth of two further graphical models: the *consensus graph* and the *conflict graph* (see, e.g., [10,18,27]), giving rise to the parameters *consensus treewidth* and *conflict treewidth*, respectively. Both graphs have as their vertices the clauses of the formula. In the consensus graph two clauses are adjacent if they do not contain a complementary pair of literals; in the conflict graph, two clauses are adjacent if they do contain a complementary pair of literals (see Fig. 1 for examples). Here, we study the parameterized complexity of #SAT with respect to the new parameters and provide a comparison to known parameters under which #SAT is fixed-parameter tractable.

Our main result regarding consensus treewidth is a novel fixed-parameter algorithm for model counting (Theorem 1). The algorithm is based on dynamic programming along a tree decomposition of the consensus graph. This result is particularly interesting as none of the known parameters under which #SAT is fixed-parameter tractable dominates consensus treewidth, in the sense that there are instances of small consensus treewidth where all the other parameters can be arbitrarily large (Proposition 1). Hence consensus treewidth pushes the state-of-the-art for fixed-parameter tractability of #SAT further, and moreover does so via a parameter that forms a natural counterpart to the already established primal, dual and incidence treewidth parameters. We also note that the presented fixed-parameter algorithm generalizes the polynomial-time algorithm on hitting formulas (see Fact 1 below).

This positive result is complemented by our results on conflict treewidth. First we observe that when considering the conflict treewidth one needs to restrict the scope to formulas without pure literals: recall that #SAT remains #P-complete for monotone 2-CNF formulas, and the conflict graph of such formulas is edge-less and therefore of treewidth 0. We show that conflict treewidth in its general form does not provide a parameter under which #SAT is fixed-parameter tractable, even for formulas without pure literals (subject to the well-established complexity theoretic assumption $\mathsf{W}[1] \neq \mathrm{FPT}$ [8]). In fact, we show

that already the decision problem SAT for formulas without pure literals is W[1]-hard when parameterized by conflict treewidth, or even by a weaker parameter, the size of a smallest vertex cover of the conflict graph (Proposition 2). However, if we bound in addition also the width of clauses (i.e., the number of literals in clauses), then #SAT becomes fixed-parameter tractable for formulas without pure literals. This result, however, does not add anything new to the complexity landscape, as we show that the incidence treewidth of a formula without pure literals is upper bounded by a function of conflict treewidth and clause width (Proposition 3).

2 Preliminaries

The set of natural numbers (that is, positive integers) will be denoted by $\mathbb{N}$. For $i \in \mathbb{N}$ we write $[i]$ to denote the set $\{1, \ldots, i\}$.

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 of clauses.

For a literal $l = \overline{x}$ we write $\overline{l} = x$; for a clause C we set $\overline{C} = \{\, \overline{l} \mid l \in C \,\}$. For a clause C, $var(C)$ denotes the set of variables x with $x \in C$ or $\overline{x} \in C$, and the *width* of C is $|var(C)|$. Similarly, for a formula F we write $var(F) = \bigcup_{C \in F} var(C)$. The *length* of a formula F is the total number of literals it contains, i.e., $\sum_{C \in F} |var(C)|$. 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. Two clauses C, D are *adjacent* if $var(C) \cap var(D) \neq \emptyset$. A variable is *pure* if it only occurs as either a positive literal or as a negative literal; the literals of a pure variable are then called pure literals.

The *dual graph* of a formula F is the graph whose vertices are clauses of F and whose edges are defined by the adjacency relation of clauses. We will also make references to the *primal graph* and the *incidence graph* of a formula F. The former is the graph whose vertices are the variables of F and where two variables a, b are adjacent iff there exists a clause C such that $a, b \in var(C)$, while the latter is the graph whose vertices are the variables and clauses of F and where two vertices a, b are adjacent iff a is a clause and $b \in var(a)$ (see Fig. 1 for an illustration).

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 $var(F[\tau]) \cap X = \emptyset$ holds for every assignment $\tau : X \to \{0, 1\}$ and every formula F. An assignment $\tau : X \to \{0, 1\}$ *satisfies* a formula F if $F[\tau] = \emptyset$.

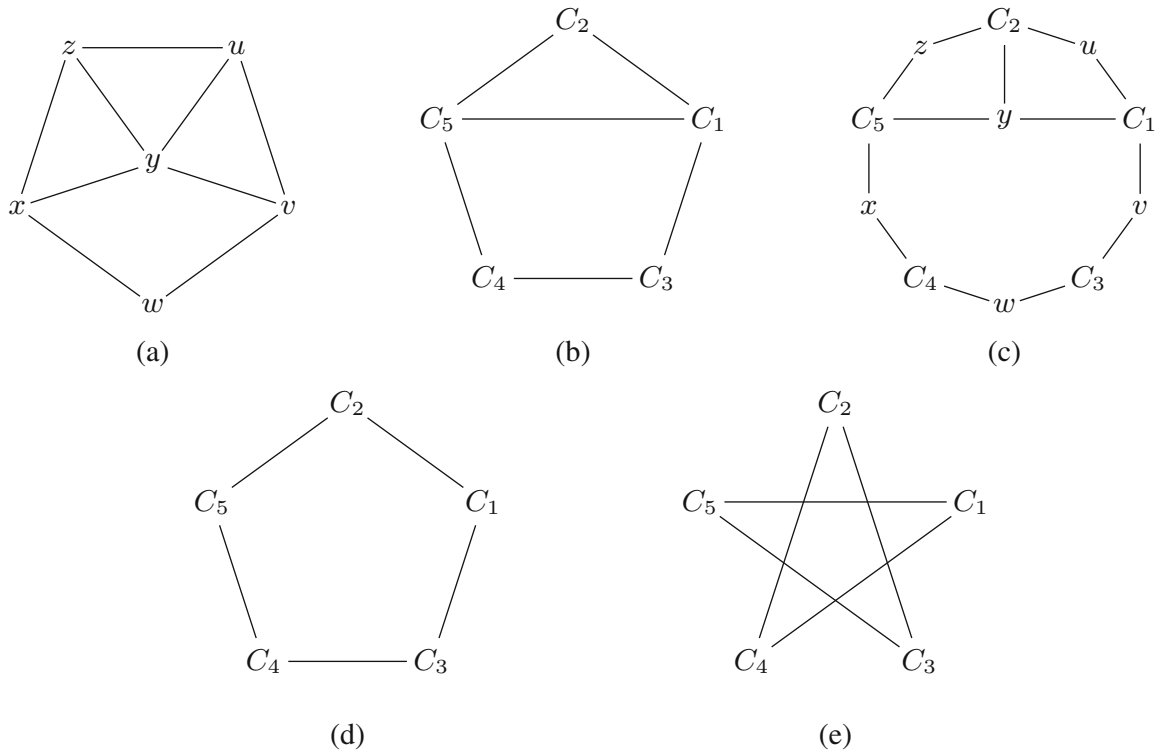


Fig. 1. The primal graph (a), dual graph (b), incidence graph (c), conflict graph (d) and consensus graph (e) of the formula $\{C_1, \ldots, C_5\}$ with $C_1 = \{u, \overline{v}, y\}$, $C_2 = \{\overline{u}, z, \overline{y}\}$, $C_3 = \{v, \overline{w}\}$, $C_4 = \{w, \overline{x}\}$, $C_5 = \{x, y, \overline{z}\}$. (a) The *primal graph* has as vertices the variables of the given formula, two variables are joined by an edge if they occur together in a clause. (b) The *dual graph* has as vertices the clauses, two clauses are joined by an edge if they share a variable. (c) The *incidence graph* is a bipartite graph where one vertex class consists of the clauses and the other consists of the variables; a clause and a variable are joined by an edge if the variable occurs in the clause. (d) The *conflict graph* has as vertices the clauses of the formula, two clauses are joined by an edge if they do contain a complementary pair of literals. (e) The *consensus graph* has as vertices the clauses of the formula, two clauses are joined by an edge if they do not contain a complementary pair of literals.

A truth assignment $\tau : var(F) \to \{0, 1\}$ that satisfies F is a *model* of F. We denote by $\#(F)$ the number of models of F. A formula F is *satisfiable* if $\#(F) > 0$. In the SAT problem, we are given a formula F and the task is to determine whether F is satisfiable. In the #SAT problem, we are also given a formula F and the task is to compute $\#(F)$.

A *hitting formula* is a CNF formula with the property that any two of its clauses clash (see [14,15,20]). The following result makes SAT and #SAT easy for hitting formulas.

Fact 1 ([13]). *A hitting formula F with n variables has exactly $2^n - \sum_{C \in F} 2^{n-|C|}$ models.*

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 [8,19].

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* (FPT) if instances (I, k) of size n (with respect to some reasonable encoding) can be solved in time $f(k)n^c$ where f is a computable function and c is a constant independent of k. The function f is called the *parameter dependence*.

To obtain our lower bounds, we will need the notion of a parameterized reduction. Let L_1, L_2 be parameterized problems. A *parameterized reduction* (or fpt-reduction) from L_1 to L_2 is a mapping P from instances of L_1 to instances of L_2 such that

1. $(x, k) \in L_1$ iff $P(x, k) \in L_2$,
2. the mapping can be computed by a fixed-parameter algorithm w.r.t. parameter k, and
3. there is a computable function g such that $k' \leq g(k)$, where $(x', k') = P(x, k)$.

The class W[1] captures parameterized intractability and contains all parameterized decision problems that are fpt-reducible to Multicolored Clique (defined below) [8]. Showing W[1]-hardness for a problem rules out the existence of a fixed-parameter algorithm unless the Exponential Time Hypothesis fails.

Multicolored Clique
Instance: A k-partite graph $G = (V, E)$ with a partition $V_1, \ldots, V_k$ of V.
Parameter: The integer k.
Question: Are there vertices $v_1, \ldots, v_k$ such that $v_i \in V_i$ and $\{v_i, v_j\} \in E$ for all i and j with $1 \leq i < j \leq k$ (i.e. the subgraph of G induced by $\{v_1, \ldots, v_k\}$ is a clique of size k)?

2.3 Treewidth

Let G be a simple, undirected, finite graph with vertex set $V = V(G)$ and edge set $E = E(G)$. A *tree decomposition* of G is a pair $(T, \{B_i : i \in I\})$ where $B_i \subseteq V$, T is a tree, and $I = V(T)$ such that:

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

The *width* of a tree decomposition is $\max_{i \in I} |B_i| - 1$. The *treewidth* [16,22] of G is the minimum width taken over all tree decompositions of G and it is denoted by $\mathbf{tw}(G)$. We call the elements of I *nodes* and B_i *bags*. As an example, consider

the graphs depicted in Fig. 1: graphs (b), (d), (e) have treewidth 2, while graphs (a) and (c) have treewidth 3.

While it is possible to compute the treewidth exactly using a fixed-parameter algorithm [3], the asymptotically best running time is achieved by using the recent state-of-the-art 5-approximation algorithm of Bodlaender et al. [4].

Fact 2 ([4]). *There exists an algorithm which, given an n-vertex graph G and an integer k, in time $2^{\mathcal{O}(k)} \cdot n$ either outputs a tree decomposition of G of width at most $5k+4$ and $\mathcal{O}(n)$ nodes, or correctly determines that $\mathbf{tw}(G) > k$.*

For other standard graph-theoretic notions not defined here, we refer to [7]. It is well known that, for every clique over $Z \subseteq V(G)$ in G, it holds that every tree decomposition of G contains an element B_i such that $Z \subseteq B_i$ [16]. Furthermore, if i separates a node j from another node l in T, then B_i separates $B_j \setminus B_i$ from $B_l \setminus B_i$ in G [16]; this *inseparability property* will be useful in some of our later proofs..

A tree decomposition $(T, \mathcal{B})$ of a graph G is *nice* if the following conditions hold:

1. T is rooted at a node r.
2. Every node of T has at most two children.
3. If a node t of T has two children t_1 and t_2, then $B_t = B_{t_1} = B_{t_2}$; in that case we call t a *join node*.
4. If a node t of T has exactly one child t', then exactly one of the following holds:
   (a) $|B_t| = |B_{t'}| + 1$ and $B_{t'} \subset B_t$; in that case we call t an *introduce node*.
   (b) $|B_t| = |B_{t'}| - 1$ and $B_t \subset B_{t'}$; in that case we call t a *forget node*.
5. If a node t of T is a leaf, then $|B_t| = 1$; we call these *leaf nodes*.

The main advantage of nice tree decompositions is that they allow the design of much more transparent dynamic programming algorithms, since one only needs to deal with four very specific types of nodes. It is well known (and easy to see) that for every fixed k, given a tree decomposition of a graph $G = (V, E)$ of width at most k and with $\mathcal{O}(|V|)$ nodes, one can construct in linear time a nice tree decomposition of G with $\mathcal{O}(|V|)$ nodes and width at most k [5]. We say that a vertex v was *forgotten* below a node $t \in V(T)$ if the subtree rooted at t contains a (forget) node s with a child s' such that $B_{s'} \setminus B_s = \{v\}$.

Finally, we summarize known algorithms for SAT and #SAT when parameterized by the treewidth of the three natural graph representations discussed in previous Subsect. 2.1; we note that the original results assumed that a tree decomposition is supplied as part of the input, and we can obtain one using Fact 2 (even while retaining the running time bounds).

Fact 3 ([26]). #SAT *is FPT when parameterized by the treewidth of any of the following graphical models of the formula: the incidence graph, the primal graph, or the dual graph.*

3 Consensus Treewidth

Recall that the consensus graph of a CNF formula F is the graph G whose vertices are the clauses of F and which contains an edge ab iff clauses a and b do not clash. Observe that the consensus graph of a hitting formula is edgeless. The consensus treewidth of F, denoted $\mathbf{contw}(F)$, is then the treewidth of its consensus graph.

Before proceeding to the algorithm, we make a short digression comparing the new notion of consensus treewidth to established parameters for SAT. We say that parameter X *dominates* parameter Y if there exists a computable function f such that for each formula F we have $X(F) \leq f(Y(F))$ [25]. In particular, if X dominates Y and SAT is FPT parameterized by X, then SAT is FPT parameterized by Y [25]. We say that two parameters are *incomparable* if neither dominates the other. We note that in our comparison, we only consider parameters which are known to give rise to fixed-parameter algorithms for SAT (i.e., not *incidence cliue-width* [21]) and can be used without requiring additional information from an oracle (i.e., not *PS-width* [24]).

In the following, we show that consensus treewidth is incomparable with the *signed clique-width* [6,28] (the clique-width of the signed incidence graph; we note that a decomposition for signed clique-width can be approximated by using signed rank-width [11]), with *clustering-width* [20] (the smallest number of variables whose deletion results in a variable-disjoint union of hitting formulas) and with *h-modularity* [12] (a structural parameter inspired by the community structure of SAT instances). We remark that the former claim implies that consensus treewidth is not dominated by the treewidth of neither the incidence nor the primal graph, since these parameters are dominated by signed clique-width [28]. Furthermore, consensus treewidth is also not dominated by signed rank-width [11], which both dominates and is dominated by signed clique-width.

Proposition 1. *The following claims hold.*

1. *Signed clique-width and consensus treewidth are incomparable.*
2. *Clustering-width and consensus treewidth are incomparable.*
3. *H-modularity and consensus treewidth are incomparable.*

Proof. We prove these claims by showing that there exist classes of formulas such that each formula in the class has one parameter bounded while the other parameter can grow arbitrarily. For a formula F, let $\mathbf{scw}(F)$ and $\mathbf{clw}(F)$ denote its signed clique-width and clustering width, respectively.

Let us choose an arbitrary positive integer $i \in \mathbb{N}$. For the first claim, it is known that already the class of all hitting formulas has unbounded **scw** [20]. In particular, this means that there exists a hitting formula F_1 such that $\mathbf{scw}(F_1) \geq i$. Observe that the consensus graph of F_1 is edgeless, and hence $\mathbf{contw}(F_1) = 0$.

Conversely, consider the following formula $F_2 = \{c_1, \ldots, c_i\}$. The formula contains variables $x_1, \ldots x_i$, and each variable x_ℓ occurs only in clause c_ℓ. Since the incidence graph of F_2 is just a matching, its signed clique-width is bounded by a constant (in particular, it will be 2). However, the consensus graph of F_2 is

a complete graph on i vertices, and it is known that such graphs have treewidth precisely $i-1$, hence $\mathbf{contw}(F_2) = i-1$.

We proceed similarly for the second and third claims; in fact, we can use a single construction to deal with both h-modularity and clustering width. Let us once again fix some $i \in \mathbb{N}$, let F_1' be the union of two variable-disjoint hitting formulas each containing i clauses. Both h-modularity and clustering width have a value of 0 for variable-disjoint hitting formulas. However, the consensus graph of F_1' is a complete bipartite graph with each side containing precisely i vertices, and it is well-known that such graphs have treewidth i; hence, $\mathbf{contw}(F_1') = i$.

Conversely, consider the formula F_2' over variable sets $Y = \{y_1, \ldots, y_i\}$ and $X = \{x_1, \ldots, x_i\}$. For each subset α of X, we will add two clauses to F_2':

- c_α contains α as positive literals and $X \setminus \alpha$ as negative literals;
- c_α^y contains α as positive literals, $X \setminus \alpha$ as negative literals, and all variables in Y as positive literals.

We observe that for each α, clause c_α^y clashes with all other clauses except for c_α (and vice-versa for c_α). This implies that the consensus graph of F_2' is a matching, and hence $\mathbf{contw}(F_2') = 1$. On the other hand, note that for each distinct pair of subsets $\alpha, \beta \subseteq X$, the clauses c_α, c_β, c_α^y, c_β^y form a formula which is not a variable-disjoint union of hitting formulas. However, deleting a subset of X from F_2' will only resolve this obstruction for choices of α and β which differed in X; for instance, even if we deleted all of X except for a single variable (w.l.o.g. say x_1), the resulting formula would still not be a disjoint union of hitting formulas (it would contain clauses $\{x_1\} \cup Y, \{x_1\}, \{\overline{x_1}\} \cup Y, \{\overline{x_1}\}$). Similarly, deleting any proper subset $Y' \subset Y$ will also clearly not result in a disjoint union of hitting formulas (it would, in fact, not change the consensus graph at all), and the same goes for any combination of deleting Y' along with a proper subset of X. Hence we conclude that $\mathbf{clw}(F_2') \geq i$.

Finally, we argue that F_2' has h-modularity at least i. We note that we will not need the definition of h-modularity to do so, as it will sufficent to follow the proof of Lemma 1 in the paper [12] which provides a suitable lower-bound for h-modularity. In particular, closely following that proof, let us fix $q = i$ and a clause $c \in F_2'$. Then:

1. the set Z_0 defined in the proof will be equal to F_2';
2. the set Z_1 defined in the proof will be empty;
3. the set Z defined in the proof will be equal to F_2';
4. the set W defined in the proof will be equal to F_2';
5. since W is not a hitting formula, by point 3 of the proof it holds that F_2' has h-modularity greater than $q = i$.

The above general constructions show that for any choice of i, one can produce formulas with a gap of at least i between consensus treewidth and any of the three other measures under consideration. □

Next, we proceed to our main algorithmic result. Our algorithm will in certain cases invoke the previously known algorithm [26] for #SAT parameterized

by dual treewidth as a subroutine, and so we provide the full statement of its runtime below. We note that the runtime of that algorithm depends on the time required to multiply two n-bit integers, denoted δ.

Fact 4 ([26]). *Given a nice tree decomposition $(T, \mathcal{B})$ of the dual graph of a formula F, #SAT can be solved in time $2^k(k\ell + \delta)N$, where N is the number of nodes of T, k is its width, and ℓ is the maximum width of a clause in F.*

In the literature there exist several algorithms for multiplying two n-bit integers; we refer the interested reader to Knuth's in-depth overview [17]. One of the most prominent of these algorithms is due to Schönhage and Strassen [17] and runs in time $\mathcal{O}(n \log n \log\log n)$. Thus, we can assume that $\delta = \mathcal{O}(n \log n \log\log n)$, where n is the number of variables of the given CNF formula. Recently, Fürer [9] presented an even faster algorithm. If arithmetic operations are assumed to have constant runtime, that is, $\delta = \mathcal{O}(1)$, then we obtain an upper bound on the runtime of $2^{\mathcal{O}(k)} \cdot L^2$.

Theorem 1. *#SAT can be solved in time $2^{\mathcal{O}(k)} \cdot L(L+\delta)$, where L is the length of the formula and k is the consensus treewidth.*

Proof. Let F be an input formula over n variables, and let G be its consensus graph. Let $(T, \mathcal{B})$ be a nice tree decomposition of G of width at most $5k + 4$; recall that such $(T, \mathcal{B})$ can be computed in time $2^{\mathcal{O}(k)}$ by Fact 2. For brevity, we will use the following terminology: for a node t with bag B_t and a clause set $X \subseteq B_t$, we say that an assignment is X^t-validating if it satisfies all clauses in X but does not satisfy any clause in $B_t \setminus X$. For instance, if $X = \emptyset$ then a X^t-validating assignment cannot satisfy any clause in B_t, while if $X = B_t$ then a X^t-validating assignment must satisfy every clause in B_t.

Consider the following leaf-to-root dynamic programming algorithm $\mathcal{A}$ on T. At each bag B_t associated with a node t of T, $\mathcal{A}$ will compute two mappings ϕ_t^+, $\phi_t^\sim$, each of which maps each $X \subseteq B_t$ to an integer between 0 and 2^d. These mappings will be used to store the number of X^t-validating assignments of $var(F)$ under an additional restriction:

- in ϕ_t^+, we count only assignments which satisfy all clauses that were already forgotten below t, and
- in $\phi_t^\sim$, we count only assignments which invalidate at least one clause that was already forgotten below t.

Since we assume that the root r of a nice tree decomposition is an empty bag, the total number of satisfying assignments of F is equal to $\phi_r^+(\emptyset)$. The purpose of also keeping records for $\phi_t^\sim$ will become clear during the algorithm; in particular, they will be needed to correctly determine the records for ϕ_t^+ at certain stages.

At each node t, let σ_t be the set of clauses which were forgotten below t; for example, $\sigma_r = F$ and $\sigma_\ell = \emptyset$ for each leaf ℓ of T. We now proceed to explain how $\mathcal{A}$ computes the mappings ϕ_t^+, $\phi_t^\sim$ at each node t of T, starting from the leaves, along with arguing correctness of the performed operations.

1. *Leaf nodes.* Since σ_t is empty, $\phi_t^{\sim}$ will map each subset of B_t to 0. As for ϕ_t^{+}, we observe that there are precisely $2^{n-|c|}$ many assignments which invalidate a clause $c \in B_t$. Hence we correctly set $\phi_t^{+}(c) = 2^{n-|c|}$ and $\phi_t^{+}(\emptyset) = 2^n - 2^{n-|c|}$.
2. *Forget nodes.* Let t be a forget node with child p and let $B_p \setminus B_t = \{c\}$. We begin by observing that the number of X^t-validating assignments which satisfy all clauses in σ_t is precisely equal to the number of $(X \cup \{c\})^p$-validating assignments which satisfy all clauses in σ_p. In other words, for each $X \subseteq B_t$ we correctly set $\phi_t^{+}(X) = \phi_p^{+}(X \cup \{c\})$.
   On the other hand, X^t-validating assignments which do not satisfy at least one clause in σ_t are partitioned into the following mutually exclusive cases:
   (a) $(X \cup \{c\})^p$-validating assignments which do not satisfy at least one clause in σ_p;
   (b) X^p-validating assignments which do not satisfy at least one clause in σ_p;
   (c) X^p-validating assignments which satisfy all clauses in σ_p.
   Hence, we correctly set $\phi_t^{\sim}(X) = \phi_p^{\sim}(X \cup \{c\}) + \phi_p^{\sim}(X) + \phi_p^{+}(X)$.
3. *Join nodes.* Let t be a join node with children p, q. Recall that $\sigma_p \cap \sigma_q = \emptyset$ and $\sigma_t = \sigma_p \cup \sigma_q$ due to the properties of tree decompositions. Furthermore, an assignment satisfies all clauses in σ_t if and only if it satisfies all clauses in both σ_p and σ_q. In other words, X^t-validating assignments which do not satisfy at least one clause in σ_t are partitioned into the following mutually exclusive cases (recall that $B_p = B_q$ by the definition of join nodes):
   (a) X^p-validating assignments which do not satisfy at least one clause in σ_p but satisfy all clauses in σ_q;
   (b) X^p-validating assignments which do not satisfy at least one clause in σ_q but satisfy all clauses in σ_p;
   (c) X^p-validating assignments which invalidate at least one clause in σ_p and also at least one clause in σ_q.
   Recall that B_t is a separator between σ_p and σ_q, which means that every clause in σ_p clashes with ever clause in σ_q. That in turn implies that the number of assignments covered by point 3c must be equal to 0: every assignment that does not satisfy at least one clause in one of σ_p, σ_q must satisfy all clauses in the other set. Since we now know that every assignment which does not satisfy a clause in σ_p must satisfy all clauses in σ_q and vice-versa, we can correctly set $\phi_t^{\sim}(X) = \phi_q^{\sim}(X) + \phi_p^{\sim}(X)$. Finally, to compute $\phi_t^{+}(X)$ we can subtract $\phi_t^{\sim}(X)$ from the total number of X^t-validating assignments (which is equal to the sum of $\phi_p^{+}(X)$ and $\phi_p^{\sim}(X)$ and hence is known to us), i.e., we set $\phi_t^{+}(X) = \phi_p^{+}(X) + \phi_p^{\sim}(X) - \phi_t^{\sim}(X)$.

4. *Introduce nodes.* Let t be an introduce node with child p and let $B_t = B_p \cup \{c\}$. For each $X \subseteq B_p$, we consider two cases and proceed accordingly. On one hand, if $\phi_p^{\sim}(X) = 0$ (i.e., there exists no X^p-validating assignment invalidating at least one clause in σ_p), then clearly $\phi_p^{\sim}(X) = \phi_t^{\sim}(X) + \phi_t^{\sim}(X \cup \{c\}) = 0$ and in particular $\phi_t^{\sim}(X) = \phi_t^{\sim}(X \cup \{c\}) = 0$. On the other hand, assume $\phi_p^{\sim}(X) > 0$ and consider a X^p-validating assignment α which invalidates at least one clause in σ_p. Since c clashes with all clauses in σ_p, it follows that α must satisfy c. Consequently, we correctly set $\phi_t^{\sim}(X) = \phi_p^{\sim}(X \cup c)$ and

$\phi_t^{\sim}(X) = 0$. Since each subset of B_t is a subset of $B_p \cup \{c\}$, it follows that using the above rules $\mathcal{A}$ has computed the mapping $\phi_t^{\sim}$ for all $X' \subseteq B_t$.
The last remaining step is to compute $\phi_t^{+}(X')$ for each $X' \subseteq B_t$. In order to do so, we will first use Fact 4 to compute the number $s_{X'}$ of all X'^t-validating assignments of F. Since we are now interested in assignments which must invalidate all clauses in $B_t \setminus X'$, we can construct the subformula F' from F by

(a) removing all clauses except for those in X', i.e., $F' := X'$, and
(b) assigning all variables which occur in $B_t \setminus X'$ in order to invalidate clauses outside of X'. Formally, for each clause $c \in B_t \setminus X'$, we apply the partial assignment $x \mapsto 0$ whenever $x \in c$ and the partial assignment $x \mapsto 1$ whenever $\overline{x} \in c$. If a contradiction arises for some variable, then we know that there exists no X'-validating assignment and hence set $s_{X'} = 0$.

Clearly, F' can be constructed in time $\mathcal{O}(L)$ and satisfies $\#F' = s_{X'}$. Furthermore, since F' contains at most k clauses, we can construct a trivial nice tree decomposition of $F'$of width at most k containing at most $2k+1$ nodes in linear time by first consecutively introducing all of its nodes and then consecutively forgetting them. With this decomposition in hand, we invoke Fact 4 to compute $\#F'$ in time at most $2^k(kL+\delta)(2k+1)$, i.e., $2^{\mathcal{O}(k)} \cdot (L+\delta)$. Once we compute $s_{X'}$, we use the fact that $s_{X'} = \phi_t^{\sim}(X) + \phi_t^{+}(X)$ and correctly set $\phi_t^{+}(X) = s_{X'} - \phi_t^{\sim}(X)$.

Observe that the time requirements for performing the above-specified operations at individual nodes of T are dominated by the time requirements for processing introduce nodes, upper-bounded by $2^k \cdot (L + 2^{\mathcal{O}(k)} \cdot (L + \delta)) = 2^{\mathcal{O}(k)} \cdot (L+\delta)$. Furthermore, a nice tree decomposition with at most $\mathcal{O}(L)$ nodes and width at most $5k + 4$ can be obtained in time $2^{\mathcal{O}(k)} \cdot L$ by Fact 2. Hence we conclude that it is possible to compute $\phi_r^{+}(\emptyset) = \#(F)$ in time at most $2^{\mathcal{O}(k)} \cdot L(L + \delta)$. The correctness of the whole algorithm follows from the correctness of computing the mappings $\phi_t^{\sim}$ and ϕ_t^{+} at each node t in T. □

4 Conflict Treewidth

The algorithmic applications of the consensus graph, as detailed above, gives rise to a natural follow-up question: what can we say about its natural counterpart, the *conflict graph*? Recall that the conflict graph of a CNF formula F is the graph G whose vertices are the clauses of F and which contains an edge ab if and only if clauses a and b clash. Observe that the conflict graph of a hitting formula is a complete graph, and that the conflict graph is the complement graph of the consensus graph. The conflict treewidth of F is then the treewidth of its conflict graph.

Since the conflict graph is a subgraph of the dual graph, conflict treewidth can be much (and in fact arbitrarily) smaller than the dual treewidth. However, unlike the case of dual treewidth, we will show that SAT does not admit a fixed-parameter algorithm parameterized by conflict treewidth (unless W[1] ≠ FPT).

Proposition 2. SAT *is W[1]-hard when parameterized by conflict treewidth. Furthermore,* SAT *remains W[1]-hard when parameterized by the size of a minimum vertex cover of the conflict graph, even for instances without pure literals.*

Proof. We provide a parameterized reduction from Multicolored Clique. Given an instance G of Multicolored Clique over vertex set $V = V_1 \cup \cdots \cup V_k$, we construct a formula F over the variable set V (i.e., each vertex in G is a variable in F). We add the following clauses to F (observe that F contains no pure literals):

1. for each $i \in [k]$, we add one clause containing one positive literal of each variable $x \in V_i$;
2. for each $i \in [k]$ and each distinct $x, y \in V_i$, we add one clause $\{\overline{x}, \overline{y}\}$;
3. for each non-edge between distinct vertices x, y in G, we add one clause $\{\overline{x}, \overline{y}\}$.

F can clearly be constructed from G in polynomial time. The intuition behind the construction is the following: variables set to true correspond to the vertices of a multicolored clique, clauses in groups 1 and 2 enforce the selection of a single vertex from each color class, and the remaining clauses ensure that the result is a clique.

To formally prove that the reduction is correct, consider a solution X to G, and consider the assignment α which sets variables in X to true and all other variables to false. Since X contains precisely one vertex from each color class V_i, α clearly satisfies all clauses in groups 1 and 2. Now consider any clause in group 3, and observe that it can only be invalidated if both of its variables are set to true. However, since X is a clique it must hold that for each pair of distinct variables $x, y \in C$ we'll never have a clause in group 3 between x and y, and hence in particular each such clause will always contain at least one variable that is set to false and that therefore satisfies it.

On the other hand, consider a satisfying assignments α for F. Then clauses in group 1 ensure that at least one variable is set to true in each color class, and clauses in group 2 ensure that at most one variable is set to true in each color class. Finally, clauses in group 3 prevent α from setting two variables to true if they are the endpoints of a non-edge in G. Consequently, the variables set to true by α must form a solution to the multicolored clique instance G.

Finally, we argue that the parameter values are bounded by k, as claimed by the hardness result. Observe that all literals in clause groups 2 and 3 are negative, which means that whenever two clauses clash, at least one of them must be in group 1. Furthermore, recall that there are precisely k clauses in group 1. Hence the clauses in group 1 form a vertex cover of size k in the conflict graph of F. It is well known (and easy to verify) that the vertex cover is an upper bound on the treewidth of a graph. □

Observe that Proposition 2 implies that there exist instances where the conflict treewidth is arbitrarily smaller than the incidence treewidth (since SAT is known to be FPT when parameterized by the latter). On the other hand, we can show that in the case of formulas of bounded clause width and without

pure literals, conflict treewidth (denoted **conflict-tw**) is dominated by incidence treewidth.

Proposition 3. *For any formula F with clauses of width at most d and without pure literals, it holds that* $\mathbf{itw}(F) \leq (d+1) \cdot (\mathbf{conflict\text{-}tw}(F)+1)$.

Proof. Let G be the conflict graph of F and $(T, \mathcal{B})$ be a tree decomposition of G of width k. Consider the structure $(T, \mathcal{B}')$ obtained as follows: for each $B_i \in \mathcal{B}$, we create a set B'_i in $\mathcal{B}'$ where $B'_i = B_i \cup \{ x \mid \exists c \in B_i : x \in var(c) \}$. Informally, the set $\mathcal{B}'$ is obtained by extending the bags in $(T, \mathcal{B})$ by the variables that occur in the clauses of that bag. We claim that $(T, \mathcal{B}')$ is a tree decomposition of the incidence graph G' of F.

Towards proving this claim, first observe that T is still a tree and each $B'_i \in \mathcal{B}'$ is a subset of $V(G')$. Furthermore, for any edge ab of G' between a clause a and variable b, it must hold that $a \in B_i$ for some $B_i \in \mathcal{B}$. By construction, B'_i must then contain both a and b and so condition 1 of the definition of tree decompositions is satisfied. As for condition 2, assume first for a contradiction that some vertex $v \in G'$ is not contained in any bag of $(T, \mathcal{B}')$. This clearly cannot happen if v is a clause, and so v must be a variable; furthermore, since F contains no pure literals, v must occur in at least two clauses.

It remains to show that all bags containing v induce a connected subtree of T. So, let us assume once more for a contradiction that this is not the case. By construction of $(T, \mathcal{B}')$ this implies that $(T, \mathcal{B})$ must contain a node t such that B_t separates some set of clauses containing v, say X_1, from all remaining clauses containing v, say X_2. Next, observe that $X_1 \cup X_2$ forms a complete bipartite graph in G: indeed, one side consists of all clauses containing v as a literal, while the other side consists of all clauses containing $\overline{v}$. But these two facts together contradict the inseparability property of tree decompositions: $X_1 \cup X_2$ induce a connected subgraph of G', and yet they are supposedly separated by B_t which does not intersect $X_1 \cup X_2$. Hence we conclude that no such node B_t exists and that the bags containing v indeed induce a connected subtree of T.

We conclude the proof by observing that the size of each bag $B'_i \in \mathcal{B}'$ is equal to $d+1$ times $|B_i|$, since we added at most d extra vertices for each vertex in B_i. □

As a consequence of Proposition 3, restricted to formulas of bounded clause width, #SAT is FPT when parameterized by conflict treewidth, since in this case the parameter is dominated by incidence treewidth [26]. We note that the domination is strict: for each $i \in \mathbb{N}$ there exists a formula F_i of clause width 2 and without pure literals such that $\mathbf{itw}(F_i) = 1$ and $\mathbf{contw}(F_i) \geq i$. Indeed, one such example is the formula $F_i = \{\{y, x_1\}, \{\overline{x_1}\}, \{y, x_2\}, \{\overline{x_2}\}, \ldots, \{y, x_i\}, \{\overline{x_i}\}\} \cup \{\{\overline{y}, z_1\}, \{\overline{z_1}\}, \{\overline{y}, z_2\}, \{\overline{z_2}\}, \ldots, \{\overline{y}, z_i\}, \{\overline{z_i}\}\}$.

5 Concluding Remarks

We have considered two natural graphical models of CNF formulas and established whether #SAT is fixed-parameter tractable parameterized by their

treewidth or not. The introduced notion of consensus treewidth generalizes and, in some sense, builds upon the classical #SAT algorithm on hitting formulas [13], and as such may be efficient in cases where other structural parameters fail. Our results show that it is worthwhile to consider further graphical models in addition to the already established ones such as primal, dual, and incidence graphs.

References

1. Bacchus, F., Dalmao, S., Pitassi, T.: Algorithms and complexity results for #SAT and Bayesian inference. In: 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2003), pp. 340–351 (2003)
2. Bodlaender, H.L.: A tourist guide through treewidth. Acta Cybernetica **11**, 1–21 (1993)
3. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput. **25**(6), 1305–1317 (1996)
4. Bodlaender, H.L., Drange, P.G., Dregi, M.S., Fomin, F.V., Lokshtanov, D., Pilipczuk, M.: A c^k n 5-approximation algorithm for treewidth. SIAM J. Comput. **45**(2), 317–378 (2016)
5. Bodlaender, H.L., Kloks, T.: Efficient and constructive algorithms for the pathwidth and treewidth of graphs. J. Algorithms **21**(2), 358–402 (1996)
6. Courcelle, B., Makowsky, J.A., Rotics, U.: On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. Discr. Appl. Math. **108**(1–2), 23–52 (2001)
7. Diestel, R.: Graph Theory. Graduate Texts in Mathematics, vol. 173, 4th edn. Springer, New York (2010)
8. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science. Springer, London (2013)
9. Fürer, M.: Faster integer multiplication. SIAM J. Comput. **39**(3), 979–1005 (2009)
10. Galesi, N., Kullmann, O.: Polynomial time SAT decision, hypergraph transversals and the hermitian rank. In: Hoos, H.H., Mitchell, D.G. (eds.) SAT 2004. LNCS, vol. 3542, pp. 89–104. Springer, Heidelberg (2005). doi:10.1007/11527695_8
11. Ganian, R., Hlinený, P., Obdrzálek, J.: Better algorithms for satisfiability problems for formulas of bounded rank-width. Fund. Inform. **123**(1), 59–76 (2013)
12. Ganian, R., Szeider, S.: Community structure inspired algorithms for SAT and #SAT. In: Heule, M., Weaver, S. (eds.) SAT 2015. LNCS, vol. 9340, pp. 223–237. Springer, Cham (2015). doi:10.1007/978-3-319-24318-4_17
13. Iwama, K.: CNF-satisfiability test by counting and polynomial average time. SIAM J. Comput. **18**(2), 385–391 (1989)
14. Kleine Büning, H., Kullmann, O.: Minimal unsatisfiability and autarkies. In: Biere, A., Heule, M.J.H., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, chap. 11, pp. 339–401. IOS Press (2009)
15. Kleine Büning, H., Zhao, X.: Satisfiable formulas closed under replacement. In: Kautz, H., Selman, B. (eds.) Proceedings for the Workshop on Theory and Applications of Satisfiability. Electronic Notes in Discrete Mathematics, vol. 9. Elsevier Science Publishers, North-Holland (2001)
16. Kloks, T.: Treewidth: Computations and Approximations. Springer, Berlin (1994)
17. Knuth, D.E.: How fast can we multiply? In: The Art of Computer Programming. Seminumerical Algorithms, 3rd edn., vol. 2, chap. 4.3.3, pp. 294–318. Addison-Wesley (1998)

18. Kullmann, O.: The combinatorics of conflicts between clauses. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 426–440. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24605-3_32
19. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, Oxford (2006)
20. Nishimura, N., Ragde, P., Szeider, S.: Solving #SAT using vertex covers. Acta Informatica **44**(7–8), 509–523 (2007)
21. Ordyniak, S., Paulusma, D., Szeider, S.: Satisfiability of acyclic and almost acyclic CNF formulas. Theor. Comput. Sci. **481**, 85–99 (2013)
22. Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. J. Algorithms **7**(3), 309–322 (1986)
23. Roth, D.: On the hardness of approximate reasoning. Artif. Intell. **82**(1–2), 273–302 (1996)
24. Sæther, S.H., Telle, J.A., Vatshelle, M.: Solving #SAT and MAXSAT by dynamic programming. J. Artif. Intell. Res. **54**, 59–82 (2015)
25. Samer, M., Szeider, S.: Fixed-parameter tractability. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, chap. 13, pp. 425–454. IOS Press (2009)
26. Samer, M., Szeider, S.: Algorithms for propositional model counting. J. Discrete Algorithms **8**(1), 50–64 (2010)
27. Scheder, D., Zumstein, P.: How many conflicts does it need to be unsatisfiable? In: Kleine Büning, H., Zhao, X. (eds.) SAT 2008. LNCS, vol. 4996, pp. 246–256. Springer, Heidelberg (2008). doi:10.1007/978-3-540-79719-7_23
28. Szeider, S.: On fixed-parameter tractable parameterizations of SAT. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 188–202. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24605-3_15
29. Valiant, L.G.: The complexity of computing the permanent. Theoret. Comput. Sci. **8**(2), 189–201 (1979)