Community Structure Inspired Algorithms for SAT and #SAT

Robert Ganian(✉) and Stefan Szeider

Algorithms and Complexity Group, TU Wien, Vienna, Austria
rganian@gmail.com, stefan@szeider.net

Abstract. We introduce h-modularity, a structural parameter of CNF formulas, and present algorithms that render the decision problem SAT and the model counting problem #SAT fixed-parameter tractable when parameterized by h-modularity. The new parameter is defined in terms of a partition of clauses of the given CNF formula into strongly interconnected communities which are sparsely interconnected with each other. Each community forms a hitting formula, whereas the interconnections between communities form a graph of small treewidth. Our algorithms first identify the community structure and then use it for an efficient solution of SAT and #SAT, respectively. We further show that h-modularity is incomparable with known parameters under which SAT or #SAT is fixed-parameter tractable.

1 Introduction

Large networks often exhibit a certain structure, where nodes form strongly interconnected communities which are sparsely connected with each other; to what extent a network exhibits such a structure can be measured by its *modularity* [17–19,31]. Recently the community structure and modularity of practical SAT instances has been empirically studied, revealing an interesting correlation between the modularity and the solving time of state-of-the art SAT solvers. Interestingly, learnt clauses tend to lie within communities and learnt clauses of low Literal Block Distance (LBD) are shared by few communities [1,20]. These findings contribute towards a better understanding of the spectacular performance of today's SAT solvers on practical instances, which is generally not well understood and remains a challenge for the research community [29].

However, the presence of a community structure with low modularity is not a *guarantee* for an instance to be easy; instead, the correlation between modularity and solving time is of statistical nature. In fact, it is not difficult to show that SAT remains NP-hard for highly modular instances. More specifically, given any SAT formula F, one can use a padding process (i.e., the addition of multiple variable-disjoint dense satisfiable subformulas) to create an equisatisfiable formula F' whose size is linear in F and whose modularity can be better than any arbitrarily fixed threshold.

Supported by the Austrian Science Fund (FWF), project P26696.

M. Heule and S. Weaver (Eds.): SAT 2015, LNCS 9340, pp. 223–237, 2015.
DOI: 10.1007/978-3-319-24318-4_17

In this paper we propose the notion of *h-modularity* for SAT instances that provides a worst-case performance guarantee for SAT decision. The h-modularity of a SAT instance is an integer-valued parameter, where instances with small h-modularity can provably be solved quickly. More precisely, we propose an algorithm that, given a SAT instance F of input length ℓ and h-modularity k, decides the satisfiability of F in time $f(k)\ell^2$, where f is singly exponential function in the parameter k. In other words, SAT is *fixed-parameter tractably* (FPT) in the parameter h-modularity. We also provide an FPT algorithm for propositional model counting (i.e., #SAT) parameterized by h-modularity. The parameter dependency is single-exponential for SAT and double-exponential for #SAT.

Our parameter is defined based on the partition of the set of clauses into subsets, which we call *h-communities*. Each h-community forms a strongly interconnected set of clauses. This is ensured by the requirement that any two clauses of an h-community clash in at least one variable (i.e., h-communities are so-called "hitting formulas" [11–13,22]). Furthermore, the h-communities are sparsely interconnected with each other, which is ensured by the requirement that a certain graph which represents the interaction between h-communities has small treewidth as well as h-communities are of small degree (graphs of small treewidth are sparse [14,24]). A formal definition of h-modularity is given in Section 3. We show that h-modularity is incomparable with the parameters signed clique-width and clustering-width, hence h-modularity is not dominated by well-known parameters that admit fixed-parameter tractability of SAT or #SAT. As a consequence, our parameter pushes the frontiers of tractability for SAT and exploits a type of structure not accessible to known FPT algorithms.

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 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 $\overline{x} \in C$. Similarly, for a formula F we write $var(F) = \bigcup_{C \in F} var(C)$. The *length* of a formula F is defined as $\sum_{C \in F} |C|$.

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 $var(C) \cap var(D) \neq \emptyset$ (i.e., if C and D clash or overlap), and the degree $\mathbf{deg}(C)$ of C in a formula F is the number of clauses $D \in F$ adjacent to C. 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. The dual graph allows us to use standard graph terminology, such as *neighborhood* and *edge-disjoint paths*, when speaking about a formula.

We will also use the *primal graph* of a formula F, specifically in the proof of Theorem 3. The primal graph of F is the graph whose vertices are variables of

F and where two variables a, b are adjacent iff there exists a clause C such that $a, b \in C$.

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 $\mathit{var}(F[\tau]) \cap X = \emptyset$ 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 $\tau : \mathit{var}(F) \to \{0,1\}$ that satisfies F is a *model* of F. We denote by $\#(F)$ the number of models of F. A formula F is *satisfiable* if $\#(F) > 0$.

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 [7,21].

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 $O(f(k)n^c)$ where f is a computable function and c is a constant independent of k. The function f is called the *parameter dependence.*

2.3 Hitting Formulas

A *hitting formula* is a CNF formula with the property that any two of its clauses clash (see [11,12,22]). The same notion for DNF formulas is termed orthogonality [5]. The following result makes hitting formulas particularly attractive in the context of SAT and #SAT.

Fact 1 ([10]). *A hitting formula F with n variables has exactly $2^n - \sum_{C \in F} 2^{n-|C|}$ models.*

The following observation will be implicitly used in several of our proofs.

Fact 2. *Let F be a hitting formula, and let F' be obtained from F by an arbitrary sequence of clause deletions and restrictions under truth assignments. Then F' is also a hitting formula.*

2.4 Treewidth

Let G be a simple, undirected, finite graph with vertex set $V = V(G)$ and edge set $E = E(G)$. For standard graph-theoretic notions not defined here, we refer to [6]. 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 $v \in V$, the set $\{\, i \in I \mid v \in X_i \,\}$ induces a (connected) subtree in T with at least one node.

The *width* of a tree decomposition is $\max_{i \in I} |X_i| - 1$. The *treewidth* [14,23] of G is the minimum width taken over all tree decompositions of G and it is denoted by $\mathbf{tw}(G)$.

Fact 3 ([3]). *There exists an algorithm which, given a graph G and an integer k, runs in time $2^{k^{\mathcal{O}(1)}} \cdot (|V(G)| + |E(G)|)$, and either outputs a tree decomposition of G of width at most k or correctly determines that $\mathbf{tw}(G) > k$.*

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 X_i such that $Z \subseteq X_i$ [14]. Furthermore, an n-vertex graph of treewidth k is sparse and has $\mathcal{O}(nk)$ edges [14,24].

3 h-Communities and h-Modularity

Let F be a formula. We call a hitting formula $H \subseteq F$ a *hitting community* (or *h-community* in brief) in F. The *degree* $\mathbf{deg}(H)$ of an h-community H is the number of edges in the dual graph of F between a clause in H and a clause outside of H. A *hitting community structure* (or *h-structure* in brief) $\mathcal{P}$ is a partitioning of F into h-communities, and the degree $\mathbf{deg}(\mathcal{P})$ of $\mathcal{P}$ is $\max\{\, \mathbf{deg}(H) \mid H \in \mathcal{P} \,\}$.

To measure the treewidth of an h-structure $\mathcal{P}$, we construct a *community graph* G as follows. The vertices of G are the h-communities in $\mathcal{P}$, and two vertices A, B in G are joined by an edge if and only if there exist clauses $C \in A$ and $D \in B$ which are adjacent. Then we let $\mathbf{tw}(\mathcal{P}) = \mathbf{tw}(G)$.

We define the *h-modularity* of an h-structure $\mathcal{P}$ as the maximum over $\mathbf{deg}(\mathcal{P})$ and $\mathbf{tw}(\mathcal{P})$. The h-modularity $\mathbf{h\text{-}mod}(F)$ of a formula F is then defined as the minimum $\mathbf{h\text{-}mod}(\mathcal{P})$ over all h-structures $\mathcal{P}$ of F.

Observe that this definition ensures that clauses in individual h-communities are strongly interconnected (since they form hitting formulas), but each h-community is only sparsely connected to other h-communities (due to the community graph having small treewidth and degree). At the same time we will prove that, unlike modularity, h-modularity is a parameter that guarantees the existence of structure which can be algorithmically exploited to establish the fixed-parameter tractability of SAT and #SAT.

Example: Consider the formula $F = \{xy\overline{a}$, $\overline{x}ya$, $x\overline{y}$, $\overline{xy}$, abc, $\overline{b}$, $c\overline{d}ef$, $d\overline{e}$, $f\overline{g}\overline{h}$, $h\overline{i}$, $i\overline{j}$, $jkl\overline{mn}$, $u\overline{v}g\overline{k}lm\overline{n}$, $uv\overline{l}$, $\overline{u}v\}$. Figure 1 (left) then illustrates the dual graph of F with the indicated partition $\mathcal{P} = \{H_1, \ldots, H_6\}$ of F into h-communities $H_1 = \{xy\overline{a}$, $\overline{x}ya$, $x\overline{y}$, $\overline{xy}\}$, $H_2 = \{abc$, $\overline{b}\}$, $H_3 = \{c\overline{d}ef$, $d\overline{e}\}$, $H_4 = \{f\overline{g}\overline{h}$, $h\overline{i}\}$, $H_5 = \{i\overline{j}$, $jkl\overline{mn}\}$, and $H_6 = \{u\overline{v}g\overline{k}lm\overline{n}$, $uv\overline{l}, \overline{u}v\}$. Figure 1 (right) shows the community graph of $\mathcal{P}$; it is easy to verify that this graph has treewidth 2 [14] (observe, for instance, that the deletion of a single vertex turns it into a tree).

The h-communities H_1 and H_3 have degree 2, and all other h-communities have degree 3. Therefore the h-modularity of F is at most $\max(3, 2) = 3$.

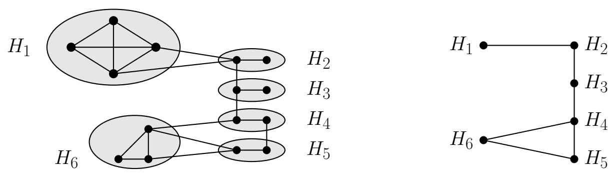

Fig. 1. The dual graph (left) and community graph (right) of the formula F and the h-structure $\mathcal{P}$.

An h-structure $\mathcal{P}$ of F is called a *witness* of $\mathbf{h\text{-}mod}(F) \le k$ if $\mathbf{h\text{-}mod}(\mathcal{P}) \le k$. Given an h-structure $\mathcal{P}$ of F and a subformula $F' \subseteq F$, we denote by $\mathcal{P}[F']$ the h-structure induced by $\mathcal{P}$ on F'; observe that $\mathbf{h\text{-}mod}(\mathcal{P}[F']) \le \mathbf{h\text{-}mod}(\mathcal{P})$.

We introduce some additional notation which will be useful later, always w.r.t. a fixed h-structure. A clause $C \in H$ is a *bridge clause* if there exists a clause outside of H adjacent to C. A variable x is a *bridge variable* if it occurs in a clause in one h-community and at least one other clause in another h-community. Notice that every clause containing a bridge variable is a bridge clause, and that h-structures of low h-modularity can still contain a large number of bridge variables, even in a single h-community.

We can now formalize the parameterized problems we are solving and present our main results.

#SAT[**h-mod**]
Instance: A formula F of length ℓ and an integer $k \ge 0$.
Task: Either compute the number of models of F, or correctly determine that $\mathbf{h\text{-}mod}(F) > k$.
Parameter: k.

The problem SAT[**h-mod**] is then defined analogously to #SAT[**h-mod**], with the distinction that the task is only to determine whether the number of models is non-zero (in which case we say that F is satisfiable).

Theorem 1. #SAT[**h-mod**] *and* SAT[**h-mod**] *are fixed parameter tractable.*

Our approach for proving Theorem 1 can be separated into two main tasks: first, we compute an h-structure $\mathcal{P}$ of small h-modularity, and then we use $\mathcal{P}$ to solve the problem. Our techniques to achieve this are discussed in detail in the following two sections. We remark that the parameter dependence is single-exponential for our SAT algorithm and double-exponential for our #SAT algorithm.

Before proceeding, we make a short digression comparing the new notion of h-modularity 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 $X(F) \le f(Y(F))$ [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. In the following, we show that h-modularity is incomparable with the *signed clique-width* (the clique-width of the signed incidence graph [4,28]) and with *clustering-width* (the smallest number of variables whose deletion results in a variable-disjoint union of hitting formulas) [22]. We remark that the former claim implies that h-modularity 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, h-modularity is also not dominated by signed rank-width [9], which both dominates and is dominated by signed clique-width.

Proposition 1. *The following claims hold.*

1. *Signed clique-width and h-modularity are incomparable.*
2. *Clustering-width and h-modularity are incomparable.*

Proof. We prove both 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 $\mathbf{clu}(F)$ denote its signed clique-width and clustering width, respectively. Our proof does not require a formal definition of these parameters, as we refer to known properties of these notions.

Let $\mathbb{N}$ be the set of positive integers, and let us choose an arbitrary $i \in \mathbb{N}$. For the first claim, it is known that already the class of all hitting formulas has unbounded $\mathbf{scw}$ [22]. In particular, this means that there exists a hitting formula F_1 such that $\mathbf{scw}(F_1) \geq i$. Recall that, since F_1 is a hitting formula, clearly $\mathbf{h\text{-}mod}(F_1) = 0$.

Conversely, consider the following formula $F_2 = \{C, C_1, \ldots, C_{i+2}\}$. The formula contains variables $x_1, \ldots x_{i+2}$, and each variable x_j occurs (either positively or negatively) in clause C and C_j. Then the incidence graph of F_2 is a tree and hence has treewidth 1. Since signed clique-width dominates the treewidth of the incidence graph, it follows that there exists a constant c independent of i such that $\mathbf{scw}(F_2) \leq c$ (in particular, one can check from the definition of $\mathbf{scw}$ that $c \leq 2$). On the other hand, the degree of any h-community H containing C is at least $i+1$, and hence $\mathbf{h\text{-}mod}(F_2) \geq i+1$.

We proceed similarly for the second claim; let $i \in \mathbb{N}$. Let F_1'' be a hitting formula, let F_1' be constructed by adding a new variable z into an arbitrary clause in F_1'' and adding a clause Z containing only z (both occurrences can either be positive or negative). Observe that $\mathbf{clu}(F_1') = \mathbf{h\text{-}mod}(F_1') = 1$. Let F_1 then contain $i+2$ disjoint copies of F_1'; clearly, $\mathbf{clu}(F_1) = i+2$. However, since the h-modularity of a formula is equal to the maximum h-modularity over all of its connected components, it holds that $\mathbf{h\text{-}mod}(F_1) = 1$.

Conversely, let F_2' and F_2'' be variable-disjoint hitting formulas containing at least $i+2$ clauses each, and let F_2 be obtained from a disjoint union of F_2' and F_2'' by adding a variable z which occurs (either positively or negatively) in $\lfloor i/2 \rfloor$ clauses in F_2' and in $\lfloor i/2 \rfloor$ clauses in F_2''. While F_2 is not a hitting formula, deleting z results in two variable-disjoint hitting formulas and hence $\mathbf{clu}(F_2) = 1$. On the other hand, the three inclusion-maximal h-communities in F_2 are F_2', F_2''

and possibly the set of clauses where z occurs; each of these have a degree which is greater than i. Consequently, it holds that $\mathbf{h\text{-}mod}(F_2) \geq i + 1$. $\square$

4 Finding h-Structures

Our approach for finding h-structures of small h-modularity consists of two steps. Generally speaking, we introduce a preprocessing procedure which we exhaustively apply until all clauses have a sufficiently small degree (Lemma 1), and once the degree of all clauses is sufficiently small we compute a tree decomposition of the dual graph and use it to find a suitable h-structure (Lemma 2). The result is an FPT-approximation algorithm [16]. One of the technical obstacles we have to overcome is that the preprocessing procedure given by Lemma 1 only guarantees the preservation of h-modularity up to a certain bound. This bound then represents an additional constraint on the approximation algorithm presented in Lemma 2.

Lemma 1. *There exists an algorithm which, given $q \in \mathbb{N}$ and a formula F of length ℓ containing a clause C such that $\mathbf{deg}(C) > 3q + 2$, runs in time $\mathcal{O}(\ell^2)$ and either correctly determines that $\mathbf{h\text{-}mod}(F) > q$, or outputs a strictly smaller subformula F' with the following property: if $\mathbf{h\text{-}mod}(F') \leq q$, then $\mathbf{h\text{-}mod}(F) = \mathbf{h\text{-}mod}(F')$. Furthermore, a witness $\mathcal{P}$ of $\mathbf{h\text{-}mod}(F)$ can be computed from F, F' and a witness $\mathcal{P}'$ of $\mathbf{h\text{-}mod}(F') \leq q$ in linear time.*

Proof. Let Z_0 be the set containing C and all clauses which are neighbors of C, let Z_1 be the subset of Z_0 containing clauses which have a neighbor outside of Z_0, and let $Z = Z_0 \setminus Z_1$. Let W be the subset of Z containing clauses which have at least $q + 2$ neighbors in Z. We now make a series of tests:

1. if $Z_1 > q$, then $\mathbf{h\text{-}mod}(F) > q$;
2. if $|W| < q + 3$, then $\mathbf{h\text{-}mod}(F) > q$;
3. if W is not a hitting formula, then $\mathbf{h\text{-}mod}(F) > q$;
4. if Z contains a clause which clashes with exactly $|W| - 1$ clauses in W, then $\mathbf{h\text{-}mod}(F) > q$;
5. let $B \in W$ be a clause with no neighbors outside W; if no such B exists, then $\mathbf{h\text{-}mod}(F) > q$.

Otherwise we set $F' = F \setminus B$.

We prove correctness. Observe that if $|Z_1| > q$ then there exists no $\mathcal{P}$ of h-modularity at most q. Indeed, for each neighbor D of Z_1 outside of Z_0, it holds that D and C cannot be in the same h-community, since they are not adjacent. Hence each element of Z_1 increases the degree of the h-community containing C by at least 1; either due to the edge between C and that element, or the edge between D and that element. Hence we can assume that $|Z| \geq 2q + 3$.

For the second test, observe that if $|W| < q + 3$ then there exists no $\mathcal{P}$ of h-modularity at most q. Indeed, since the number of neighbors of C in Z is at least $2q + 2$, at least $q + 2$ of these neighbors must be in the same h-community as C if $\mathbf{h\text{-}mod}(\mathcal{P}) \leq q$. This implies that at least $q+2$ of these neighbors would have to be pairwise-adjacent, and in particular would each have at least $q+2$ neighbors in Z. Then W necessarily must contain C and at least $q+2$ neighbors of C.

For the third test, if W is not a hitting formula, then any h-structure $\mathcal{P}$ of h-modularity at most q would need to partition W into (subsets of) at least two h-structures; let H_C be the hypothetical h-community containing C, and let $D \in W \setminus H_C$. Since D has $q+2$ neighbors in Z, there are at least $q+2$ edge-disjoint paths between D and C, and each of these paths contributes at least 1 to the degree of H_C. But then it follows that $\mathbf{deg}(H_C) \geq q+2$, which would contradict $\mathbf{h\text{-}mod}(\mathcal{P}) \leq q$, and hence W must be a hitting formula. Observe that this argument also implies that every clause in W is in fact adjacent to every other clause in W, and that every $\mathcal{P}$ of h-modularity at most q must contain an h-community H_C which contains W.

For the fourth test, assume there exists a clause D which clashes with exactly $|W|-1$ clauses in W. Consider any witness $\mathcal{P}$ of $\mathbf{h\text{-}mod}(F) \leq q$, and let H_C be the h-community containing C. Since $D \notin H_C$ and there are at least $q+1$ edge-disjoint paths between D and C, the existence of D would imply that $\mathbf{deg}(H_C) \geq q+1$.

For the fifth test, recall that for any clause $Q \in Z \setminus W$ it holds that $W \cup \{Q\}$ cannot be a hitting formula because Q cannot be adjacent to every clause in W. Hence every clause in W with a neighbor outside of W contributes at least 1 to the degree of any h-community containing W. Together with $|W| > q+2$ this implies that if no clause B exists, then $\mathbf{h\text{-}mod}(F) > q$.

Finally, assume there exists a clause $B \in W$ with no neighbors outside of W and let $F' = F \setminus B$. If $\mathbf{h\text{-}mod}(F') > q$ then the lemma already holds, so assume there exists a witness $\mathcal{P}'$ of $\mathbf{h\text{-}mod}(F') \leq q$. Let $W' = W \setminus B$. Observe that W' must be contained in a single h-community $H' \in \mathcal{P}'$, since otherwise the fact that each clause of W' is adjacent to every other clause of W' would contradict the degree bound given by $\mathbf{h\text{-}mod}(\mathcal{P}') \leq q$. Then let $\mathcal{P}$ be obtained from $\mathcal{P}'$ by adding B to H'. Observe that there cannot exist a clause $D \in H'$ such that D and B do not clash; since D clashes with every other clause in W, it follows that D would clash with $|W|-1$ clauses in W. Hence $B \cup H'$ is still an h-community. Furthermore, by our choice of B it holds that B contains no neighbors outside of W', and hence $\mathbf{deg}(H') = \mathbf{deg}(H' \cup \{B\})$ and in turn $\mathbf{deg}(\mathcal{P}') = \mathbf{deg}(\mathcal{P})$.

Finally, observe that, if we are given a witness $\mathcal{P}'$ of $\mathbf{h\text{-}mod}(F') \leq q$, we can construct a witness of $\mathbf{h\text{-}mod}(F)$ by adding B back into the unique h-community in $\mathcal{P}$ containing the neighbors of B (i.e., W'). □

Lemma 2. *There exists an algorithm which, given $k \in \mathbb{N}$ and a formula F of length ℓ such that $\mathbf{deg}(F) \leq 12k^2+2$, runs in time $2^{k^{\mathcal{O}(1)}} \cdot \ell$, and either outputs an h-structure $\mathcal{P}$ of F such that $\mathbf{h\text{-}mod}(\mathcal{P}) \leq k^2+k$, or correctly determines that $\mathbf{h\text{-}mod}(F) > k$.*

Proof. We first test whether the treewidth of the dual graph G of F is at most $k \cdot (12k^2+3)$; if not, then $\mathbf{h\text{-}mod}(F) > k$, and if yes, we compute a tree decomposition of F. This can be achieved in time at most $2^{k^{\mathcal{O}(1)}} \cdot \ell$ by Fact 3. Next, we enumerate every inclusion-maximal clique in G of cardinality at least $k+2$ in time $\mathcal{O}(k^3) \cdot \ell$ by a simple traversal of the tree decomposition. Let L be the set of all such cliques. For each clique $K \in L$ we test whether K is a hitting formula and whether $\mathbf{deg}(K) \leq k$; if not, then $\mathbf{h\text{-}mod}(F) > k$. For each pair of cliques $K_1, K_2 \in L$ we test that they are pairwise disjoint; if not, then $\mathbf{h\text{-}mod}(F) > k$. Let G' be the graph obtained from G by contracting each clique in L into a single vertex; that is, each $K \in L$ is replaced by a vertex adjacent to all neighbors of K. We test that $\mathbf{deg}(G') \leq 2k$ and $\mathbf{tw}(G') \leq k^2 + k$; if not, then $\mathbf{h\text{-}mod}(F) > k$. Finally, let $\mathcal{P}'$ be the vertex set of G'. Then $\mathcal{P}'$ is an h-structure witnessing $\mathbf{h\text{-}mod}(F) \leq k^2 + k$.

We prove correctness. First, assume for a contradiction that $\mathbf{tw}(G) > k \cdot (12k^2 + 3)$ and that there exists a witness $\mathcal{P}'$ of $\mathbf{h\text{-}mod}(F) \leq k$. Since $\mathbf{deg}(F) \leq 12k^2 + 2$, every h-community in $\mathcal{P}'$ must have size at most $12k^2 + 3$. Let (β, T) be a width-k tree decomposition of the community graph of $\mathcal{P}'$, and let β' be obtained by replacing each h-community $H \in \mathcal{P}'$ with $\bigcup_{C \in H} C$. Then (β', T) is a tree decomposition of G of width at most $k \cdot (12k^2 + 3)$, contradicting our assumptions.

Next, assume that there exists a clique $K \in L$ which is not a hitting formula. Then any hypothetical h-structure $\mathcal{P}$ of F must partition K into several h-communities. Let $C, D \in K$ and $H \in \mathcal{P}$ be such that $C \in H$ and $D \notin H$. Since there exist $k+1$ edge-disjoint paths between C and D, this implies that $\mathbf{deg}(H) \geq k+1$ and hence $\mathbf{h\text{-}mod}(\mathcal{P}) > k$.

Similarly, assume that there exist inclusion-maximal cliques $K_1, K_2 \in L$ which intersect in some clause C. Then any hypothetical h-structure $\mathcal{P}$ must contain an h-community H containing C, and there must exist a clause $D \in K_1 \cup K_2$ such that $D \notin H$. As in the previous case, this gives rise to at least $k+1$ edge-disjoint paths between C and D and hence $\mathbf{h\text{-}mod}(\mathcal{P}) > k$. In particular, we conclude that each element of L must form an h-community in any hypothetical witness of $\mathbf{h\text{-}mod}(F) \leq k$. This in turn implies that if there exists an h-community $K \in L$ of degree at least $k+1$, then $\mathbf{h\text{-}mod}(F) > k$.

We proceed by considering the graph G'. Assume it contains a vertex v of degree at least $2k+1$. If v is a clause in F, then at most k neighbors of v can form an h-community with v (since we have contracted all cliques of cardinality at least $k+2$). This means that at least $k+1$ neighbors of v would contribute to the degree of the h-community containing v, which guarantees $\mathbf{h\text{-}mod}(F) > k$. On the other hand, if v is an element of L, then we already know that v itself must be an h-community in any witness of $\mathbf{h\text{-}mod}(F) \leq k$, and hence v having more than k neighbors also implies $\mathbf{h\text{-}mod}(F) > k$.

Next, consider the case $\mathbf{tw}(G') > k^2 + k$. Observe that each hitting subformula of F not contained in L contains at most $k+1$ clauses. Consider a width-k tree decomposition (β, T) of the community graph Q of a hypothetical witness of $\mathbf{h\text{-}mod}(F) \leq k$. By replacing, in β, each h-community $H \in V(Q) \setminus L$ with the set of clauses contained in H, we would obtain a tree decomposition of G' of width at most $k \cdot (k+1)$, contradicting our assumption. Hence we conclude that $\mathbf{h\text{-}mod}(F) > k$.

Finally, we summarize why $\mathcal{P}'$ is indeed an h-structure of G such that $\mathbf{h\text{-}mod}(\mathcal{P}') \leq k^2 + k$. The fact that $\mathcal{P}'$ is an h-structure follows by construction; indeed, each element in $\mathcal{P}'$ is either a single clause, or an element of L which is guaranteed to be a hitting formula. Regarding the h-modularity of $\mathcal{P}'$, recall that G' is the community graph of $\mathcal{P}'$ and that $\mathbf{tw}(G') \leq k^2 + k$. As for the degree bound, each vertex v in G' is either a clause C in F, which means that $\mathbf{deg}(v) \leq 2k$, or an element of K, in which case we have already tested that $\mathbf{deg}(v) \leq k$. ☐

Theorem 2. *There exists an algorithm which, given $k \in \mathbb{N}$ and a formula F of length ℓ, runs in time $\mathcal{O}(\ell^3) + 2^{k^{\mathcal{O}(1)}} \cdot \ell$, and either outputs an h-structure $\mathcal{P}$ of F such that* $\mathbf{h\text{-}mod}(\mathcal{P}) \leq k^2 + k$, *or correctly determines that* $\mathbf{h\text{-}mod}(F) > k$.

Proof. We begin by exhaustively applying Lemma 1 on F for $q = 4k^2$; let us denote the resulting formula F'. Then we apply Lemma 2 on F' to find an h-structure $\mathcal{P}'$ of F' such that $\mathbf{h\text{-}mod}(\mathcal{P}') \leq k^2 + k \leq q$. Finally, we use Lemma 1 to convert $\mathcal{P}'$ into an h-structure $\mathcal{P}$ of F. Correctness follows from the correctness of Lemmas 1 and 2. ☐

5 Using h-Structures

With Theorem 2 in hand, we proceed to show how the identified h-structure of small h-modularity can be used to obtain fixed-parameter tractability of SAT and #SAT. The general strategy is to replace each h-community by a suitable object that represents all the satisfying assignments of this h-community. This way, variables only appearing in a single h-community are eliminated. In case of SAT, we represent an h-community by a set of clauses over the bridge variables of the h-community, and in the case of #SAT, we represent an h-community by a so-called valued constraint. This way, we reduce the problems SAT and #SAT parameterized by h-modularity to certain problems (SAT and SumProd, respectively) parameterized by primal treewidth. For solving the latter problems we can use known algorithms.

For making this general strategy work, we have to overcome the difficulty that the number of bridge variables of a single h-community can be arbitrarily large even when the input formula has small h-modularity. In the case of SAT we can handle this by replacing the input formula with a satisfiability-equivalent subformula using a known construction. This approach does not work for #SAT since this replacement does not preserve the number of models. However, by replacing equivalence classes of variables that appear in the same way in all clauses by 3-valued variables (which represent the three possibilities that all variables in the module are set to true, all are set to false, or some are set to true and some to false, respectively), we can reduce the number of variables for a single valued constraint so that we can make our overall strategy work.

We begin with the conceptually simpler case of SAT. Our solution relies on the following folklore result.

Fact 4 (**[26]**). *There exists an algorithm which takes as input a formula F of length ℓ and a tree decomposition of the primal graph of F of width k, runs in time $2^{\mathcal{O}(k)} \cdot \ell^2$, and determines whether F is satisfiable.*

Theorem 3. *Given a formula F' of length ℓ and an h-structure $\mathcal{P}'$ of F', we can decide whether F' is satisfiable in time $2^{\mathcal{O}(\mathbf{h\text{-}mod}(\mathcal{P}')^2)} \cdot \ell^2$.*

Proof. Our algorithm has three steps. First, we compute an equisatisfiable subformula F of F' where F has the following property: for every nonempty set X of variables of F there are at least $|X|+1$ clauses C of F such that some variable in X occurs in C. Formulas with this property are called 1-*expanding* or *matching-lean*, and it is known that for any formula F' of length ℓ, an equisatisfiable 1-expanding subformula F can be computed in time $\mathcal{O}(\ell^{3/2})$ [8,15,27]. We set $\mathcal{P} = \mathcal{P}'[F]$ and $k = \mathbf{h\text{-}mod}(\mathcal{P}')$; note that $\mathbf{h\text{-}mod}(\mathcal{P}) = \mathbf{h\text{-}mod}(\mathcal{P}'[F]) \leq \mathbf{h\text{-}mod}(\mathcal{P}') = k$. Observe that since each $H \in \mathcal{P}$ satisfies $\mathbf{deg}(H) \leq k$, it follows that the number of bridge variables which occur in any clause in H is upper-bounded by k.

For the second step, we construct a formula I as follows. The variable set of I consists of all the bridge variables of $\mathcal{P}$. For each h-community $H \in \mathcal{P}$ containing bridge variables $X_H = \{x_1, \ldots, x_p\}$ and for each assignment α of variables in X_H, we test whether α satisfies H; if it does not, we add the clause C_α over X_α into I, where C_α is the unique clause which is not satisfied by α.

For the final third step, we compute a tree decomposition of the primal graph of I with width at most k^2+k by Fact 3, and then decide whether I is satisfiable by Fact 4. If it is, we output "YES", and otherwise we output "NO". The rest of the proof is dedicated to verifying the bound on the treewidth of I and arguing correctness.

We argue that the treewidth of the primal graph of I at most k^2+k. Let (β, T) be a tree decomposition of the community graph G of $\mathcal{P}$ of width at most k. Consider the tree decomposition (γ, T) obtained from (β, T) by replacing each h-community H in β by X_H. Since F is 1-expanding and the variables of X_H only appear in at most $k+1$ clauses of F due to the degree bound, the cardinality of each X_H is upper-bounded by $k+1$. Consequently, the cardinality of each element in γ is at most k^2+k.

Next, we show that (γ, T) is indeed a tree decomposition of the primal graph of I. For every edge ab in this graph, there exists at least one clause $C \in H$ which contains both variable a and variable b in its scope, and hence a, b are both bridge variables for H, which in turn means that a, b will both be present in every element of γ which used to contain H; this proves that the first property of tree decompositions is satisfied. For every bridge variable a, let $\mathcal{D}_a$ denote the set of h-communities which contain a. Since each pair of h-communities containing a are adjacent in the community graph of $\mathcal{P}$, $\mathcal{D}_a$ forms a clique in the community graph of $\mathcal{P}$ and hence there must exist an element θ_a of β which contains every h-community in $\mathcal{D}_a$. Since a occurs in an element of γ if and only if this originated from an element of β containing an h-community in $\mathcal{D}_a$, and since all h-communities in $\mathcal{D}_a$ occur in θ_a, we conclude that the nodes of

T containing a are connected in (γ, T); this proves that the second property of tree decompositions is satisfied.

Finally, we argue that I is satisfiable if and only if F is satisfiable. Let τ_I be a satisfying assignment for I, and consider the assignment τ which assigns each bridge variable in F based on τ_I. The resulting instance $F[\tau]$ consists of variable-disjoint h-communities. Furthermore, by the construction of each constraint in I, it holds that each h-commnunity in $F[\tau]$ is satisfiable, and hence both $F[\tau]$ and F are satisfiable. On the other hand, let τ_F be a satisfying assignment for F, and consider the restriction τ of τ_F to the set of bridge variables. Then applying τ on F once again results in a satisfiable formula $F[\tau]$ consisting of variable-disjoint h-communities. Furthermore, since each such h-community is satisfiable, it follows that τ also satisfies every clause in I. □

Our next goal is to show how h-structures of low h-modularity can be used to solve #SAT. To this end, we will make use of a reduction to the SumProd (Sum of Products) problem [2], sometimes also called Valued #CSP [30], which can be viewed as a generalization of the Constraint Satisfaction problem. An instance I of SumProd is a triple $(V, D, \mathcal{C})$, where V is a finite set of *variables*, D is a finite set of *domain values*, and $\mathcal{C}$ is a finite set of *valued constraints*. Each valued constraint C in $\mathcal{C}$ is a tuple (S_C, f_C), where S_C, the *constraint scope*, is a non-empty sequence $s_1, s_2, \ldots, s_r$ of distinct variables of V, and f_C, the *cost function*, is a function from D^r to $\mathbb{N} \cup \{0\}$.

An *assignment* is a mapping $\psi : V \to D$. Each assignment ψ results in a cost, $f_C(\psi)$, being assigned to each constraint C, where $f_C(\psi) = f_C((\psi(s_1), \psi(s_2), \ldots, \psi(s_r)))$. The task in the SumProd problem is to compute the value $\mathbf{cost}(I)$, defined as the sum over all assignments of the products of cost functions for that assignment. In other words, $\mathbf{cost}(I) = \sum_{\psi : V \to D} \prod_{C \in \mathcal{C}} f_C(\psi)$.

The *primal graph* G of a SumProd instance I is defined as follows. The vertices of G are the variables of I, and two vertices a, b of G are adjacent if and only if there exists a constraint whose scope contains both a and b. The *primal treewidth* of I, denoted $\mathbf{ptw}(I)$, is the treewidth of the primal graph of I. The crucial property which we exploit is that primal treewidth allows a straightforward dynamic programming FPT algorithm for SumProd over a fixed and finite domain D. The following fact assumes that arithmetic operations can be carried out in polynomial time in the number of variables.

Fact 5 ([2]). *Let D be a fixed set. There exists an algorithm which takes as input an n-variable instance $I = (V, D, \mathcal{C})$ of* SumProd *and a tree decomposition of the primal graph of I of width k, runs in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$, and correctly outputs $\mathbf{cost}(I)$.*

Lemma 3. *There exists an algorithm which, given a formula F of length ℓ and an h-structure $\mathcal{P}$ of F, runs in time $\mathcal{O}(3^{\mathbf{h\text{-}mod}(\mathcal{P})} \cdot \ell^{\mathcal{O}(1)})$, and computes an instance $I = (V, D, \mathcal{C})$ of* SumProd *such that $\mathbf{ptw}(I) \leq 2^{\mathcal{O}(\mathbf{h\text{-}mod}(\mathcal{P}))}$, $D = \{0, 1, mix\}$, $|V| \leq \ell$ and $\mathbf{cost}(I)$ is the number of models of F.*

Proof (Sketch). Our goal is to capture the contribution of an h-community H to the total number of models of F by using only a small number of variables in

I; specifically, the number of these variables should depend only on $\mathbf{h\text{-}mod}(\mathcal{P})$. Unlike in Theorem 3, here we cannot directly use 1-expanding subformulas, since these do not preserve the number of models. So instead we group bridge variables into equivalence classes, where two bridge variables are in the same equivalence class iff they occur in the same way in the same clauses; crucially, the number of equivalence classes which intersect with each H is bounded by a function of $\mathbf{h\text{-}mod}(\mathcal{P})$. Furthermore, every "mixed" assignment (mapping at least one variable to 0 and at least one to 1) of an equivalence class satisfies the same clauses as any other mixed assignment of that equivalence class, allowing us to aggregate all such assignments without loss of information. Then we construct our instance I so that each of its variables represents one equivalence class, and each constraint represents one h-community. An assignment ψ of I then corresponds to determining whether all bridge variables of F in each equivalence class are assigned to 0, to 1, or mix.

The cost function is then constructed so as to capture the contribution of each h-community to the total number of models. However, since many assignments in F can be aggregated into a single assignment in I due to the mix value, the cost function also needs to reflect this. To this end, each equivalence class is assigned (arbitrarily) to some valued constraint C and whenever that equivalence class is mapped to mix, f_C is increased by a factor corresponding to the number of assignments in F aggregated into this mixed assignment.

The desired running time follows by showing that equivalence classes can be computed in at most $\mathcal{O}(\ell^3)$ time and that the number of equivalence classes which occur in the same h-community is upper-bounded by $3^{\mathbf{h\text{-}mod}(\mathcal{P})+1}$. The lemma then follows from the following two claims, whose proofs are omitted in this version: (i) $\mathbf{ptw}(I) \leq 2^{\mathcal{O}(\mathbf{h\text{-}mod}(\mathcal{P}))}$, and (ii) $\mathbf{cost}(I)$ is the number of models of F. □

Theorem 4. *Given a formula F of length ℓ and an h-structure $\mathcal{P}$ of F, we can count the number of models of F in time $2^{2^{\mathcal{O}(\mathbf{h\text{-}mod}(\mathcal{P}))}} \cdot \ell^{\mathcal{O}(1)}$.*

Proof. Let $k = \mathbf{h\text{-}mod}(\mathcal{P})$. We apply Lemma 3 to obtain an instance $I = (V, D, \mathcal{C})$ of SumProd such that $\mathbf{ptw}(I) \leq 2^{\mathcal{O}(k)}$ and $\mathbf{cost}(I)$ is the number of models of F. Next, we compute a tree decomposition of the primal graph of I of width $2^{\mathcal{O}(k)}$: either by observing that the algorithm of Lemma 3 implicitly also computes such a tree decomposition of I, or in time $2^{2^{\mathcal{O}(k)}} \cdot \ell$ by Fact 3. Finally, we use Fact 5 to solve I in time $2^{2^{\mathcal{O}(k)}} \cdot \ell^{\mathcal{O}(1)}$. □

Proof (of Theorem 1). Let F be the given CNF formula and k the parameter. First we apply Theorem 2 to either find an h-structure $\mathcal{P}$ of F of h-modularity at most $k^2 + k$, or correctly determine that $\mathbf{h\text{-}mod}(F) > k$. To decide whether F is satisfiable, we now use Theorem 3. This establishes that SAT[$\mathbf{h\text{-}mod}(F)$] is fixed parameter tractable. To compute the number of models of F, we use Theorem 4. This establishes the fixed parameter tractability of #SAT[$\mathbf{h\text{-}mod}(F)$] and concludes the proof. □

6 Concluding Notes

We have introduced the notion of an h-community structure in CNF formulas and the associated parameter h-modularity. Furthermore, we have shown that it is fixed-parameter tractable to find a suitable h-community structure and to use it to solve the problems SAT and #SAT, all parameterized by the h-modularity (Theorems 2, 3, and 4, respectively). Since the h-modularity is small for formulas where other known parameters can be arbitrarily large (Proposition 1), our FPT results provide worst-case performance guarantees for instances that are not accessible by known methods. Our results give rise to the question of how the notion of h-community structure can be further generalized, for example by using a suitably defined property for the communities that generalizes hitting formulas. This way, we hope that ultimately one can build bridges between empirically observed problem hardness and theoretical worst case upper bounds.

References

1. Ansótegui, C., Bonet, M.L., Giráldez-Cru, J., Levy, J.: The fractal dimension of SAT formulas. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014. LNCS, vol. 8562, pp. 107–121. Springer, Heidelberg (2014)
2. Bacchus, F., Dalmao, S., Pitassi, T.: Solving #SAT and Bayesian inference with backtracking search. J. Artif. Intell. Res. **34**, 391–442 (2009)
3. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput. **25**(6), 1305–1317 (1996)
4. 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)
5. Crama, Y., Hammer, P.L.: Boolean functions. Encyclopedia of Mathematics and its Applications, vol. 142. Cambridge University Press, Cambridge (2011). Theory, algorithms, and applications
6. Diestel, R.: Graph Theory. Graduate Texts in Mathematics, vol. 173, 4th edn. Springer Verlag, New York (2010)
7. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science. Springer, New York (1999)
8. Fleischner, H., Kullmann, O., Szeider, S.: Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. Theoretical Computer Science **289**(1), 503–516 (2002)
9. Ganian, R., Hlinený, P., Obdrzálek, J.: Better algorithms for satisfiability problems for formulas of bounded rank-width. Fund. Inform. **123**(1), 59–76 (2013)
10. Iwama, K.: CNF-satisfiability test by counting and polynomial average time. SIAM J. Comput. **18**(2), 385–391 (1989)
11. Büning, H.K., 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, chapter 11, pp. 339–401. IOS Press (2009)
12. Büning, H.K., 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)

13. Büning, K.H., Zhao, X.: On the structure of some classes of minimal unsatisfiable formulas. Discr. Appl. Math. **130**(2), 185–207 (2003)
14. Kloks, T.: Treewidth: Computations and Approximations. Springer Verlag, Berlin (1994)
15. Kullmann, O.: Lean clause-sets: Generalizations of minimally unsatisfiable clause-sets. Discr. Appl. Math. **130**(2), 209–249 (2003)
16. Marx, D.: Parameterized complexity and approximation algorithms. The Computer Journal **51**(1), 60–78 (2008)
17. Newman, M.E.J.: The structure and function of complex networks. SIAM Review **45**(2), 167–256 (2003)
18. Newman, M.E.J.: Modularity and community structure in networks. Proceedings of the National Academy of Sciences **103**(23), 8577–8582 (2006)
19. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E **69**(2), 026113 (2004)
20. Newsham, Z., Ganesh, V., Fischmeister, S., Audemard, G., Simon, L.: Impact of community structure on SAT solver performance. In: Sinz, C., Egly, U. (eds.) SAT 2014. LNCS, vol. 8561, pp. 252–268. Springer, Heidelberg (2014)
21. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, Oxford (2006)
22. Nishimura, N., Ragde, P., Szeider, S.: Solving #SAT using vertex covers. Acta Informatica **44**(7–8), 509–523 (2007)
23. Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. J. Algorithms **7**(3), 309–322 (1986)
24. Rose, D.J.: On simple characterizations of k-trees. Discrete Math. **7**, 317–322 (1974)
25. Samer, M., Szeider, S.: Fixed-parameter tractability. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, chapter 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. Szeider, S.: Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. J. of Computer and System Sciences **69**(4), 656–674 (2004)
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)
29. Vardi, M.Y.: Boolean satisfiability: theory and engineering. Communications of the ACM **57**(3), 5 (2014)
30. Živný, S.: The Complexity of Valued Constraint Satisfaction Problems. Cognitive Technologies. Springer (2012)
31. Zhang, W., Pan, G., Wu, Z., Li, S.: Online community detection for large complex networks. In: Rossi, F. (eds.) Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013, Beijing, China, August 3–9, 2013. IJCAI/AAAI (2013)