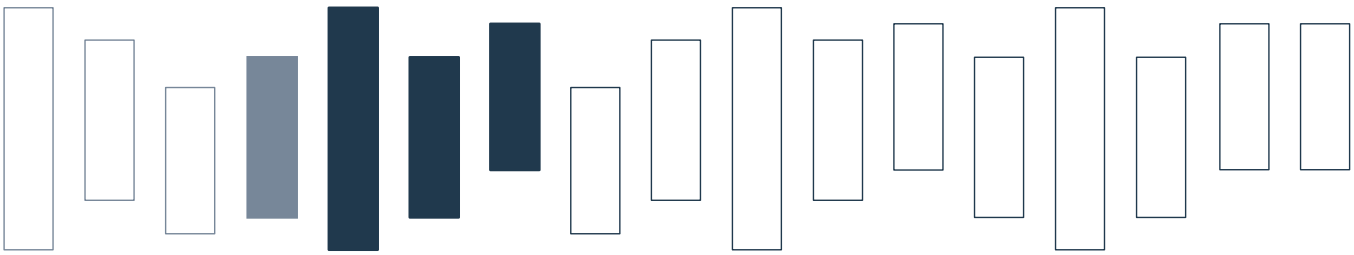
Technical Report AC-TR-20-006

June 2020

A Faster Algorithm for Propositional Model Counting Parameterized by Incidence Treewidth

Friedrich Slivovsky and Stefan Szeider



This is the authors' copy of a paper that will appear in the proceedings of SAT'20, the 23rd International Conference on Theory and Applications of Satisfiability Testing.

www.ac.tuwien.ac.at/tr

A Faster Algorithm for Propositional Model Counting Parameterized by Incidence Treewidth*

Friedrich Slivovsky and Stefan Szeider

TU Wien, Vienna, Austria
{fs,sz}@ac.tuwien.ac.at

Abstract. The propositional model counting problem (#SAT) is known to be fixed-parameter-tractable (FPT) when parameterized by the width k of a given tree decomposition of the incidence graph. The running time of the fastest known FPT algorithm contains the exponential factor of 4^k. We improve this factor to 2^k by utilizing fast algorithms for computing the zeta transform and covering product of functions representing partial model counts, thereby achieving the same running time as FPT algorithms that are parameterized by the less general treewidth of the primal graph. Our new algorithm is asymptotically optimal unless the Strong Exponential Time Hypothesis (SETH) fails.

1 Introduction

Propositional model counting (#SAT) is the problem of determining the number of satisfying truth assignments of a given propositional formula. The problem arises in several areas of AI, in particular in the context of probabilistic reasoning [2,15]. #SAT is #P-complete [18], even for 2-CNF Horn formulas, and it is NP-hard to approximate the number of models of a formula with n variables within $2^{n^{1-\varepsilon}}$, for any $\varepsilon > 0$ [15].

Since syntactic restrictions do not make the problem tractable, research generally focused on structural restrictions in terms of certain graphs associated with the input formula, which is often assumed to be in CNF. Popular graphical models are the primal graph (vertices are the variables, two variables are adjacent if they appear together in a clause), the dual graph (vertices are the clauses, two clauses are adjacent if they share a variable), and the incidence graph (vertices are variables and clauses, a variable and a clause are adjacent if the variable occurs in the clause). The structural complexity of a graph can be restricted in terms of the fundamental graph invariant *treewidth* [14] By taking the treewidth of the primal, dual, or incidence graph one obtains the *primal treewidth*, the *dual treewidth*, and the *incidence treewidth* of the formula, respectively. If we consider CNF formulas for which any of the three parameters is bounded by a

* Supported by the Austrian Science Fund (FWF) under grant P32441 and the Vienna Science and Technology Fund (WWTF) under grants ICT19-060 and ICT19-065.

constant, the number of models can be computed in polynomial time. Indeed, the order of the polynomial is independent of the treewidth bound, and so #SAT is fixed-parameter tractable (FPT) when parameterized by primal, dual or incidence treewidth.

Incidence treewidth is considered the most general parameter among the three, as any formula of primal or dual treewidth k has incidence treewidth at most $k+1$. However, one can easily construct formulas of constant incidence treewidth and arbitrarily large primal and dual treewidth. Known model counting algorithms based on incidence treewidth have to pay for this generality with a significant larger running time: Whereas the number of models for formulas of primal or dual treewidth k can be counted in time[1] $O^*(2^k)$, the best know algorithm for formulas of incidence treewidth k takes time $O^*(4^k)$.[2] This discrepancy cannot be accounted for by a loose worst-case analysis, but is caused by the actual size of the dynamic programming tables constructed by the algorithms.

In this paper, we show that algebraic techniques can be used to bring down the running time to $O^*(2^k)$. Specifically, we prove that the most time-consuming steps can be expressed as *zeta transforms* and *covering products* of functions obtained from partial model counts. Since there are fast algorithms for computing these operations [3], we obtain the desired speedup.

2 Preliminaries

Treewidth. Let $G = (V(G), E(G))$ be a graph, $T = (V(T), E(T))$ be a tree, and χ be a labeling of the vertices of T by sets of vertices of G. We refer to the vertices of T as "nodes" to avoid confusion with the vertices of G. The tuple (T, χ) is a *tree decomposition* of G if the following three conditions hold:

1. For every $v \in V(G)$ there exists a node $t \in V(T)$ such that $v \in \chi(t)$.
2. For every $vw \in E(G)$ there exists a node $t \in V(T)$ such that $v, w \in \chi(t)$.
3. For any three nodes $t_1, t_2, t_3 \in V(T)$, if t_2 lies on the path from t_1 to t_3, then $\chi(t_1) \cap \chi(t_3) \subseteq \chi(t_2)$.

The *width* of a tree decomposition (T, χ) is defined by $\max_{t \in V(T)} |\chi(t)| - 1$. The *treewidth* $tw(G)$ of a graph G is the minimum width over all its tree decompositions. For constant k, there exists a linear-time algorithm that checks whether a given graph has treewidth at most k and, if so, outputs a tree decomposition of minimum width [5]. However, the huge constant factor in the runtime of this algorithm makes it practically infeasible. For our purposes, it suffices to obtain tree decompositions of small but not necessarily minimal width. There exist

[1] The O^* notation omits factors that are polynomial in the input size [19].

[2] Alternatively, one can convert a formula with incidence treewidth k into a 3-CNF formula that has the same number of models and dual treewidth at most $3(k+1)$, or an equisatisfiable 3-CNF formula of primal treewidth at most $3(k+1)$ [17]. Applying one of these transformations followed by an algorithm for the corresponding width parameter results in an overall running time of $O^*(8^k)$.

several powerful tree decomposition heuristics that construct tree decompositions of small width for many cases that are relevant in practice [4,12], and the single-exponential FPT algorithm by Bodlander et al. [6] produces a factor-5 approximation of treewidth.

In this paper we also consider a particular type of tree decompositions. The triple (T, χ, r) is a *nice tree decomposition* of G if (T, χ) is a tree decomposition, the tree T is rooted at node r, and the following three conditions hold [11]:

1. Every node of T has at most two children.
2. If a node t of T has two children t_1 and t_2, then $\chi(t) = \chi(t_1) = \chi(t_2)$; in that case we call t a *join node*.
3. If a node t of T has exactly one child t', then one of the following holds:
   (a) $|\chi(t)| = |\chi(t')| + 1$ and $\chi(t') \subset \chi(t)$; in that case we call t an *introduce node*.
   (b) $|\chi(t)| = |\chi(t')| - 1$ and $\chi(t) \subset \chi(t')$; in that case we call t a *forget node*.

It is known that one can transform efficiently any tree decomposition of width k of a graph with n vertices into a nice tree decomposition of width at most k and at most $4n$ nodes [11, Lemma 13.1.3].

Propositional Formulas. We consider propositional formulas F in conjunctive normal form (CNF) represented as set of clauses. Each clause in F is a finite set of *literals*, and a literal is a negated or unnegated propositional *variable*. For a clause C we denote by $var(C)$ the set of variables that occur (negated or unnegated) in C; for a formula F we put $var(F) = \bigcup_{C \in F} var(C)$. The *size* of a clause is its cardinality. A *truth assignment* is a mapping $\tau : X \to \{0, 1\}$ defined on some set X of variables. We extend τ to literals by setting $\tau(\neg x) = 1 - \tau(x)$ for $x \in X$. A truth assignment $\tau : X \to \{0, 1\}$ *satisfies* a clause C if for some variable $x \in var(C) \cap X$ we have $x \in C$ and $\tau(x) = 1$, or $\neg x \in C$ and $\tau(x) = 0$. An assignment satisfies a set F of clauses if it satisfies every clause in F. For a formula F, we call a truth assignment $\tau : var(F) \to \{0, 1\}$ a *model* of F if τ satisfies F. We denote the number of models of F by $\#(F)$. The *propositional model counting problem* #SAT is the problem of computing $\#(F)$ for a given propositional formula F in CNF.

Incidence Treewidth. The *incidence graph* $G^*(F)$ of a CNF formula F is the bipartite graph with vertex set $F \cup var(F)$; a variable x and a clause C are joined by an edge if and only if $x \in var(C)$. The *incidence treewidth* $tw^*(F)$ of a CNF formula F is the treewidth of its incidence graph, that is $tw^*(F) = tw(G^*(F))$.

Definition 1 (Zeta and Möbius Transforms). *Let V be a finite set and let $f : 2^V \to \mathbb{Z}$ be a function. The* zeta transform *ζf of f is defined as*

$$(\zeta f)(X) = \sum_{Y \subseteq X} f(Y), \tag{1}$$

and the Möbius transform *μf of f is given by*

$$(\mu f)(X) = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} f(Y). \tag{2}$$

Theorem 1 (Kennes [10]). *Let V be an k-element set and let $f : 2^V \to \mathbb{Z}$ be a function. All values of ζf and μf can be computed using $O(2^k k)$ arithmetic operations.*

Definition 2. *The* covering product *of two functions $f, g : 2^V \to \mathbb{Z}$ is a function $(f *_c g) : 2^V \to \mathbb{Z}$ such that for every $Y \subseteq V$,*

$$(f *_c g)(Y) = \sum_{A \cup B = Y} f(A)g(B). \tag{3}$$

The covering product can be computed using zeta and Möbius transforms by applying the following two results (see Aigner [1]).

Lemma 1. *Given functions $f, g : 2^V \to \mathbb{Z}$, the zeta transform of the covering product of f and g is the pointwise product of the zeta-transformed arguments. That is, for each $X \subseteq V$*

$$\zeta(f *_c g)(X) = (\zeta f(X))(\zeta g(X)).$$

Theorem 2 (Inversion formula). *Let $f : 2^V \to \mathbb{Z}$. Then for every $X \subseteq V$*

$$f(X) = (\mu\zeta f)(X) = (\zeta\mu f)(X).$$

3 Faster Model Counting for Incidence Treewidth

Samer and Szeider presented an algorithm for #SAT with a running time of $O^*(4^k)$ [16], where k is the width of a given tree decomposition of the incidence graph. In this section, we are going to show how to improve this to $O^*(2^k)$.

Their algorithm proceeds by bottom-up dynamic programming on a nice tree decomposition, maintaining tables that contain partial solution counts for each node. For the remainder of this section, let F be an arbitrary but fixed CNF formula, and let (T, χ, r) be a nice tree-decomposition of the incidence graph $G^*(F)$ that has width k. For each node t of T, let T_t denote the subtree of T rooted at t, and let $V_t = \bigcup_{t' \in V(T_t)} \chi(t')$ denote the set of vertices appearing in bags of T_t. Further, let F_t denote the set of clauses in V_t, and let X_t denote the set of all variables in V_t. We also use the shorthands $\chi_c(t) = \chi(t) \cap F$ and $\chi_v(t) = \chi(t) \cap var(F)$ for the set of clauses and the set of variables in $\chi(t)$, respectively. Let t be a node of T. For each assignment $\alpha : \chi_v(t) \to \{0, 1\}$ and subset $A \subseteq \chi_c(t)$, we define $N(t, \alpha, A)$ as the set of assignments $\tau : X_t \to \{0, 1\}$ for which the following two conditions hold:

1. $\tau(x) = \alpha(x)$ for all variables $x \in \chi_v(t)$.
2. A is exactly the set of clauses in F_t that are not satisfied by τ.

We represent the values of $n(t, \alpha, A) = |N(t, \alpha, A)|$ for all $\alpha : \chi_v(t) \to \{0, 1\}$ and $A \subseteq \chi_c(t)$ by a table M_t with $|\chi(t)| + 1$ columns and $2^{|\chi(t)|}$ rows. The first $|\chi(t)|$ columns of M_t contain Boolean values encoding $\alpha(x)$ for variables

$x \in \chi_v(t)$, and membership of C in A for clauses $C \in \chi_c(t)$. The last entry of each row contains the integer $n(t, \alpha, A)$.

Samer and Szeider showed that the entries of the table M_t can be efficiently computed for each node t. More specifically, they showed how M_t can be obtained for leaf nodes t, and how M_t can be computed from the tables for the child nodes of introduce, forget, and join nodes t. Since the running time for join and variable introduce nodes are the bottleneck of the algorithm, we summarize the results concerning correctness and running time for the remaining node types as follows.

Lemma 2 (Samer and Szeider [16]). *If $t \in T$ is a leaf node, or a forget or clause introduce node with child t' such that $M_{t'}$ has already been computed, the table M_t can be obtained in time $2^k|\varphi|^{O(1)}$.*

The table entries for a join node can be computed as a sum of products from tables of its child nodes.

Lemma 3 (Samer and Szeider [16]). *Let $t \in T$ be a* join node *with children t_1, t_2. For each assignment $\alpha : \chi_v(t) \to \{0,1\}$ and set $A \subseteq \chi_c(t)$ we have*

$$n(t, \alpha, A) = \sum_{\substack{A_1, A_2 \subseteq \chi_c(t), \\ A_1 \cap A_2 = A}} n(t_1, \alpha, A_1)\, n(t_2, \alpha, A_2). \tag{4}$$

A straight-forward algorithm for computing this sum requires an arithmetic operation for each pair (A_1, A_2) where $A_1 \subseteq \chi_c(t_1)$, $A_2 \subseteq \chi_c(t_2)$, and thus $2^k 2^k = 4^k$ operations in the worst case. By using a fast algorithm for the covering product, we can significantly reduce the number of arithmetic operations and thus the running time. The key observation is that the sum of products in (4) can be readily expressed as a covering product (3).

Lemma 4. *Let t be a* join node *of T with children t_1, t_2 and let $\alpha : \chi_v(t) \to \{0,1\}$ be a truth assignment. For $i \in \{1,2\}$ let $f_i : 2^{\chi_c(t_i)} \to \mathbb{Z}$ be the function given by $f_i(A) := n(t_i, \alpha, \chi_c(t) \setminus A)$. Then $n(t, \alpha, \chi_c(t) \setminus A) = (f_1 *_c f_2)(A)$ for each subset $A \subseteq \chi_c(t)$.*

Proof. For $S \subseteq \chi_c(t)$, let $S^c = \chi_c(t) \setminus S$. We have

$$\begin{aligned}(f_1 *_c f_2)(A) &= \sum_{\substack{A_1, A_2 \subseteq \chi_c(t) \\ A_1 \cup A_2 = A}} f_1(A_1)\, f_2(A_2) \\ &= \sum_{\substack{A_1, A_2 \subseteq \chi_c(t) \\ A_1^c \cap A_2^c = A^c}} n(t_1, \alpha, A_1^c)\, n(t_2, \alpha, A_2^c) = n(t, \alpha, A^c).\end{aligned}$$

□

For variable introduce nodes the table entry for each assignment and subset of clauses can be computed as a sum over table entries of the child table.

Lemma 5 (Samer and Szeider [16]). *Let t be an* introduce node *with child t' such that $\chi(t) = \chi(t') \cup \{x\}$ for a variable x. Then, for each truth assignment $\alpha : \chi_v(t') \to \{0,1\}$ and set $A \subseteq \chi_c(t)$, we have*

$$n(t, \alpha \cup \{(x,0)\}, A) = \begin{cases} 0 & \text{if } \neg x \in C \text{ for some } C \in A; \\ \sum_{B' \subseteq B} n(t', \alpha, A \cup B') & \text{otherwise, where } B = \{C \in \chi_c(t) \mid \neg x \in C\}; \end{cases} \tag{5}$$

$$n(t, \alpha \cup \{(x,1)\}, A) = \begin{cases} 0 & \text{if } x \in C \text{ for some } C \in A; \\ \sum_{B' \subseteq B} n(t', \alpha, A \cup B') & \text{otherwise, where } B = \{C \in \chi_c(t) \mid x \in C\}. \end{cases} \tag{6}$$

A simple approach is to go through all 2^k assignments α and subsets A and, if necessary, compute the sums in (5) and (6). Since there could be up to 2^k subsets to sum over, this again requires 4^k arithmetic operations in the worst case. The following lemma observes that we can instead use the zeta transform (1).

Lemma 6. *Let t be an* introduce node *with child t' such that $\chi(t) = \chi(t') \cup \{x\}$ for a variable x. Define $f(S) = n(t', \alpha, S)$ for $S \subseteq \chi_c(t')$. Then, for each truth assignment $\alpha : \chi_v(t') \to \{0,1\}$ and set $A \subseteq \chi_c(t)$, we have*

$$n(t, \alpha \cup \{(x,0)\}, A) = \begin{cases} 0 & \text{if } \neg x \in C \text{ for some } C \in A; \\ (\zeta f)(A \cup B) - (\zeta f)(A) & \text{otherwise, where } B = \{C \in \chi_c(t) \mid \neg x \in C\}; \end{cases} \tag{7}$$

$$n(t, \alpha \cup \{(x,1)\}, A) = \begin{cases} 0 & \text{if } x \in C \text{ for some } C \in A; \\ (\zeta f)(A \cup B) - (\zeta f)(A) & \text{otherwise, where } B = \{C \in \chi_c(t) \mid x \in C\}. \end{cases} \tag{8}$$

Proof. We can rewrite the sums in (5) and (6) as

$$\sum_{A \subseteq S \subseteq A \cup B} f(S) = \sum_{S \subseteq A \cup B} f(S) - \sum_{S \subseteq A} f(S) = (\zeta f)(A \cup B) - (\zeta f)(A).$$

□

By Theorem 1, the zeta and Möbius transform of a function $f : 2^V \to \mathbb{Z}$ can be computed using $O(2^k k)$ arithmetic operations, where $k = |V|$ is the size of the underlying set. In conjunction with Lemma 1 and Lemma 2, this lets us compute the covering product of two functions $f, g : 2^V \to \mathbb{Z}$ with $O(2^k k)$ operations.

How this translates into running time depends on the choice of computational model. Since the model count of a formula can be exponential in the number of variables, it is unrealistic to assume that arithmetic operations can be performed in constant time. Instead, we adopt a random access machine model where two n-bit integers can be added, subtracted, and compared in time $O(n)$, and multiplied in time $O(n \log n)$ [8]. For the purposes of proving a bound of $O^*(2^k)$, it is sufficient to show that the bit size of integers obtained as intermediate results while computing the zeta and Möbius transforms is polynomially bounded by

the number of variables in the input formula. To verify that this is the case, we present the dynamic programming algorithms used to efficiently compute these transforms [3], following the presentation by Fomin and Kratsch [7].

Theorem 3. *Let V be a k-element set and let $f : 2^V \to \mathbb{Z}$ be a function that can be evaluated in time $O(1)$ and whose range is contained in the interval $(-2^N, 2^N)$. All values of ζf and μf can be computed in time $2^k(k+N)^{O(1)}$.*

Proof. Let $V = \{1, \ldots, k\}$. We compute intermediate values

$$\zeta_j(X) = \sum_{Y \subseteq X \cap \{1,\ldots,j\}} f(Y \cup (X \cap \{j+1, \ldots, k\})),$$

for $j = 0, \ldots, k$. Note that $\zeta_k(X) = (\zeta f)(X)$. The values ζ_j can be computed as

$$\zeta_j(X) = \begin{cases} \zeta_{j-1}(X) & \text{when } j \notin X, \\ \zeta_{j-1}(X) + \zeta_{j-1}(X \setminus \{j\}) & \text{when } j \in X. \end{cases}$$

For the Möbius transform, we compute intermediate values

$$\mu_j(X) = \sum_{Y \subseteq X \cap \{1,\ldots,j\}} (-1)^{|(X \cap \{1,\ldots,j\}) \setminus Y|} f(Y \cup (X \cap \{j+1, \ldots, k\})).$$

Again we have $\mu_k(X) = (\mu f)(X)$, and the values μ_j can be computed as

$$\mu_j(X) = \begin{cases} \mu_{j-1}(X) & \text{when } j \notin X, \\ \mu_{j-1}(X) - \mu_{j-1}(X \setminus \{j\}) & \text{when } j \in X. \end{cases}$$

In both cases, this requires k arithmetic operations for each set $X \subseteq V$, and the intermediate values are contained in the interval $(-2^k 2^N, 2^k 2^N)$. □

Corollary 1. *Let V be an k-element set and let $f, g : 2^V \to \mathbb{Z}$ be functions that can be evaluated in time $O(1)$ and whose range is contained in the interval $(-2^N, 2^N)$. All values of $f *_c g$ can be computed in time $2^k(k+N)^{O(1)}$.*

Having obtained these bounds on the time required to compute the zeta transform and the covering product, we can now state improved time bounds for obtaining the entries of the tables M_t of join and variable introduce nodes t.

Lemma 7. *The table M_t for a join node $t \in T$ can be computed in time $O^*(2^k)$ given the tables M_{t_1} and M_{t_2} of its child nodes t_1 and t_2.*

Proof. Let $p = |\chi_v(t)|$ and $q = |\chi_c(t)|$. For each assignment $\alpha : \chi_v(t) \to \{0, 1\}$, we perform the following steps. For $i \in \{1, 2\}$, we first modify the table M_{t_i} by flipping the values encoding membership of a clause C in the set A so as to obtain a table M'_i containing the values $f_i(A) := n(t_i, \alpha, \chi_c(t) \setminus A)$ for each $A \subseteq \chi_c(t)$. Clearly, this can be done in time $O^*(2^p)$. By Lemma 4, the values $(f_1 *_c f_2)(A)$ correspond to $n(t, \alpha, \chi_c(t) \setminus A)$. Each entry of the tables M_{t_i} represents a partial model count that cannot exceed 2^n, so we can compute all values of the covering product $f_1 *_c f_2$ in time $2^q(q+n)^{O(1)}$ by Corollary 1, where n is the number of variables. Since $q \leq n$ this is in $O^*(2^q)$. There are at most 2^p assignments $\alpha : \chi_v(t) \to \{0, 1\}$, so the overall running time is in $O^*(2^k)$. □

Lemma 8. *The table M_t for a variable introduce node $t \in T$ can be computed in time $O^*(2^k)$ given the table $M_{t'}$ of its child node t'.*

Proof. As before, let $p = |\chi_v(t')|$ and $q = |\chi_c(t')|$. For each truth assignment $\alpha : \chi_v(t') \to \{0,1\}$, we proceed as follows. We compute the value of the zeta transform $(\zeta f)(A)$ for all subsets $A \subseteq \chi_c(t')$. Again, each entry of $M_{t'}$ represents a partial model count that is bounded by 2^n, so we can do this in time $2^q(q+n)^{O(1)}$ by Theorem 1. Since $q \leq n$ this is in $O^*(2^q)$. Then, we iterate over all $A \subseteq \chi_c(t')$ and set the entries $M_t(\alpha, A)$ based on (7) and (8), using the values of ζf. This can again be done in time $O^*(2^q)$ and is correct by Lemma 6. The number of assignments $\alpha : \chi_v(t) \to \{0,1\}$ is 2^p, so the overall running time is in $O^*(2^k)$. □

Theorem 4. *Given a CNF formula F and a nice tree decomposition of $G^*(F)$, we can compute $\#(F)$ in time $O^*(2^k)$, where k is the width of the decomposition.*

Proof. Let (T, χ, r) be a nice tree decomposition of the incidence graph of F. We compute the tables M_t for all nodes t of T, starting from the leaf nodes of T. By Lemma 2, Lemma 7, and Lemma 8, each table can be computed in time $O^*(2^k)$. We can compute $\#(F) = \sum_{\alpha:\chi_v(r)\to\{0,1\}} n(r, \alpha, \emptyset)$ at the root r. □

4 Discussion

The space requirements of the algorithm remain unchanged by the proposed improvements. Each table M_t has at most 2^{k+1} entries, and each entry requires up to n bits. By keeping as few tables in memory as possible and discarding tables whenever they are no longer needed, no more than $\lfloor 1 + \log_2(N+1) \rfloor$ tables need to be stored in memory at any point, where N is the number of nodes in the tree decomposition [16, Proposition 3].

Let $s_r := \inf\{\, \delta \mid$ there exists an $O^*(2^{\delta n})$ algorithm that decides the satisfiability of n-variable r-CNF formulas with parameter $n \,\}$ and let $s_\infty := \lim_{r\to\infty} s_r$. Impagliazzo et al. [9] introduced the Strong Exponential Time Hypothesis (SETH), which states that $s_\infty = 1$. SETH has served as a very useful hypothesis for establishing tight bounds on the running time for NP-hard problems [13]. For instance, an immediate consequence of the SETH is that the satisfiability of an n-variable CNF formula cannot be solved in time $O^*((2-\varepsilon))^n)$ for any $\varepsilon > 0$. However, for the incidence graph of an n-variable CNF formula $F = \{C_1, \ldots, C_m\}$ we can always give a tree decomposition (T, χ) of width n (recall that the width of a tree decomposition is the size of its largest bag minus one) by taking as T a star with center t and leaves $t_1, \ldots, t_m$, and by putting $\chi(t) = var(F)$ and $\chi(t_i) = var(F) \cup \{C_i\}$, for $1 \leq i \leq n$. Thus, if the bound in Theorem 4 could be improved from $O^*(2^k)$ to $O^*((2-\varepsilon)^k)$, we would have an $O^*((2-\varepsilon)^n)$ SAT-algorithm, and hence a contradiction to the SETH. We can, therefore, conclude that Theorem 4 is tight under the SETH.

Acknowledgements

We thank Andreas Björklund for the suggestion of using covering products to improve the running time of SAT algorithms for instances of bounded incidence treewidth.

References

1. Aigner, M.: Combinatorial theory. Springer Science & Business Media (2012)
2. 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'03). pp. 340–351 (2003)
3. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Fourier meets möbius: fast subset convolution. In: Johnson, D.S., Feige, U. (eds.) Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007. pp. 67–74. Assoc. Comput. Mach., New York (2007)
4. Bodlaender, H.L.: Discovering treewidth. In: Proceedings of the 31st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'05). Lecture Notes in Computer Science, vol. 3381, pp. 1–16. Springer Verlag (2005)
5. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput. 25(6), 1305–1317 (1996)
6. 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)
7. Fomin, F.V., Kratsch, D.: Exact Exponential Algorithms. Springer Verlag (2010)
8. Harvey, D., Van Der Hoeven, J.: Integer multiplication in time $O(n \log n)$. HAL archives ouvertes (hal-02070778) (2019)
9. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. of Computer and System Sciences 63(4), 512–530 (2001)
10. Kennes, R.: Computational aspects of the mobius transformation of graphs. IEEE Trans. Systems, Man, and Cybernetics 22(2), 201–223 (1992)
11. Kloks, T.: Treewidth: Computations and Approximations. Springer Verlag, Berlin (1994)
12. Koster, A.M.C.A., Bodlaender, H.L., van Hoesel, S.P.M.: Treewidth: Computational experiments. Electronic Notes in Discrete Mathematics 8 (2001)
13. Lokshtanov, D., Marx, D., Saurabh, S.: Lower bounds based on the exponential time hypothesis. Bulletin of the European Association for Theoretical Computer Science 105, 41–72 (2011)
14. Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. J. Algorithms 7(3), 309–322 (1986)
15. Roth, D.: On the hardness of approximate reasoning. Artificial Intelligence 82(1-2), 273–302 (1996)
16. Samer, M., Szeider, S.: Algorithms for propositional model counting. J. Discrete Algorithms 8(1), 50–64 (2010)
17. Samer, M., Szeider, S.: Constraint satisfaction with bounded treewidth revisited. J. of Computer and System Sciences 76(2), 103–114 (2010)
18. Valiant, L.G.: The complexity of computing the permanent. Theoretical Computer Science 8(2), 189–201 (1979)

19. Woeginger, G.J.: Exact algorithms for NP-hard problems: A survey. In: Combinatorial Optimization - Eureka, You Shrink!, Papers Dedicated to Jack Edmonds, 5th International Workshop, Aussois, France, March 5-9, 2001, Revised Papers. Lecture Notes in Computer Science, vol. 2570, pp. 185–208 (2003)