Exploiting Treewidth for Projected Model Counting and its Limits*

Johannes K. Fichte, Markus Hecher, Michael Morak, and Stefan Woltran

Institute of Logic and Computation, TU Wien, Vienna, Austria
lastname@dbai.tuwien.ac.at

Abstract. In this paper, we introduce a novel algorithm to solve *projected model counting* (PMC). PMC asks to count solutions of a Boolean formula with respect to a given set of *projected variables*, where multiple solutions that are identical when restricted to the projected variables count as only one solution. Our algorithm exploits small treewidth of the primal graph of the input instance. It runs in time $\mathcal{O}(2^{2^{k+4}}n^2)$ where k is the treewidth and n is the input size of the instance. In other words, we obtain that the problem PMC is fixed-parameter tractable when parameterized by treewidth. Further, we take the exponential time hypothesis (ETH) into consideration and establish lower bounds of bounded treewidth algorithms for PMC, yielding asymptotically tight runtime bounds of our algorithm.

Keywords: Parameterized Algorithms, Tree Decompositions, Multi-Pass Dynamic Programming, Projected Model Counting, Propositional Logic

1 Introduction

A problem that has been used to solve a large variety of real-world questions is the *model counting problem* (#Sat) [2, 11, 14, 16, 33, 37, 40, 42, 45]. It asks to compute the number of solutions of a Boolean formula [24] and is theoretically of high worst-case complexity (#· P-complete [43, 38]). Lately, both #Sat and its approximate version have received renewed attention in theory and practice [9, 16, 31, 39]. A concept that allows very natural abstractions of data and query results is projection. Projection has wide applications in databases [1] and declarative problem modeling. The problem *projected model counting* (PMC) asks to count solutions of a Boolean formula with respect to a given set of *projected variables*, where multiple solutions that are identical when restricted to the projected variables count as only one solution. If all variables of the formula are projected variables, then PMC is the #Sat problem and if there are no projected variables then it is simply the Sat problem. Projected variables allow for solving problems where one needs to introduce auxiliary variables, in particular, if these variables are functionally independent of the variables of interest, in the problem encoding, e.g., [21, 23].

* The work has been supported by the Austrian Science Fund (FWF), Grants Y698 and P26696, and the German Science Fund (DFG), Grant HO 1294/11-1. The first two authors are also affiliated with the University of Potsdam, Germany.

When we consider the computational complexity of PMC it turns out that under standard assumptions the problem is even harder than #Sat, more precisely, complete for the class $\#\cdot$ NP [17]. Even though there is a PMC solver [3] and an ASP solver that implements projected enumeration [22], PMC has received very little attention in parameterized algorithmics so far. Parameterized algorithms [12, 15, 20, 34] tackle computationally hard problems by directly exploiting certain structural properties (parameter) of the input instance to solve the problem faster, preferably in polynomial-time for a fixed parameter value. In this paper, we consider the treewidth of graphs associated with the given input formula as parameter, namely the primal graph [41]. Roughly speaking, small *treewidth* of a graph measures its tree-likeness and sparsity. Treewidth is defined in terms of *tree decompositions (TDs)*, which are arrangements of graphs into trees. When we take advantage of small treewidth, we usually take a TD and evaluate the considered problem in parts, via *dynamic programming (DP)* on the TD.

New Contributions.

1. We introduce a novel algorithm to *solve projected model counting (*PMC*)* in time $\mathcal{O}(2^{2^{k+4}} n^2)$ where k is the treewidth of the primal graph of the instance and n is the size of the input instance. Similar to recent DP algorithms for problems on the second level of the polynomial hierarchy [19], our algorithm traverses the given tree decomposition multiple times (multi-pass). In the first traversal, we run a dynamic programming algorithm on tree decompositions to solve Sat [41]. In a second traversal, we construct equivalence classes on top of the previous computation to obtain model counts with respect to the projected variables by exploiting combinatorial properties of intersections.
2. We establish that our *runtime bounds are asymptotically tight under the exponential time hypothesis (ETH)* [28] using a recent result by Lampis and Mitsou [32], who established lower bounds for the problem $\exists\forall$-Sat assuming ETH. Intuitively, ETH states a complexity theoretical lower bound on how fast satisfiability problems can be solved. More precisely, one *cannot* solve 3-Sat in time $2^{s \cdot n} \cdot n^{\mathcal{O}(1)}$ for some $s > 0$ and number n of variables.

2 Preliminaries

For a set X, let 2^X be the *power set of* X consisting of all subsets Y with $\emptyset \subseteq Y \subseteq X$. Recall the well-known combinatorial inclusion-exclusion principle [25], which states that for two finite sets A and B it is true that $|A \cup B| = |A| + |B| - |A \cap B|$. Later, we need a generalized version for arbitrary many sets. Given for some integer n a family of finite sets X_1, X_2, ..., X_n, the number of elements in the union over all sets is $|\bigcup_{j=1}^{n} X_j| = \sum_{I \subseteq \{1,\ldots,n\}, I \neq \emptyset} (-1)^{|I|-1} |\bigcap_{i \in I} X_i|$.

Satisfiability. A literal is a (Boolean) variable x or its negation $\neg x$. A *clause* is a finite set of literals, interpreted as the disjunction of these literals. A *(CNF) formula* is a finite set of clauses, interpreted as the conjunction of its clauses. A

3-CNF has clauses of length at most 3. Let F be a formula. A *sub-formula* S of F is a subset $S \subseteq F$ of F. For a clause $c \in F$, we let $\mathrm{var}(c)$ consist of all variables that occur in c and $\mathrm{var}(F) := \bigcup_{c \in F} \mathrm{var}(c)$. A (partial) *assignment* is a mapping $\alpha : \mathrm{var}(F) \to \{0,1\}$. For $x \in \mathrm{var}(F)$, we define $\alpha(\neg x) := 1 - \alpha(x)$. The formula F *under the assignment* $\alpha \in 2^{\mathrm{var}(F)}$ is the formula $F_{|\alpha}$ obtained from F by removing all clauses c containing a literal set to 1 by α and removing from the remaining clauses all literals set to 0 by α. An assignment α is *satisfying* if $F_{|\alpha} = \emptyset$ and F is *satisfiable* if there is a satisfying assignment α. Let V be a set of variables. An *interpretation* is a set $J \subseteq V$ and its induced assignment $\alpha_{J,V}$ of J with respect to V is defined as follows $\alpha_{J,V} := \{v \mapsto 1 \mid v \in J \cap V\} \cup \{v \mapsto 0 \mid v \in V \setminus J\}$. We simply write α_J for $\alpha_{J,V}$ if $V = \mathrm{var}(F)$. An interpretation J is a *model* of F, denoted by $J \vDash F$, if its induced assignment α_J is satisfying. Given a formula F; the problem Sat asks whether F is satisfiable and the problem #Sat asks to output the number of models of F, i.e., $|S|$ where S is the set of all models of F.

Projected Model Counting. An instance of the projected model counting problem is a pair (F, P) where F is a (CNF) formula and P is a set of Boolean variables such that $P \subseteq \mathrm{var}(F)$. We call the set P *projection variables* of the instance. The *projected model count* of a formula F with respect to P is the number of total assignments α to variables in P such that the formula $F_{|\alpha}$ under α is satisfiable. The *projected model counting problem (*PMC*)* [3] asks to output the projected model count of F, i.e., $|\{M \cap P \mid M \in S\}|$ where S is the set of all models of F.

Example 1. Consider formula $F := \{\overbrace{\neg a \vee b \vee p_1}^{c_1}, \overbrace{a \vee \neg b \vee \neg p_1}^{c_2}, \overbrace{a \vee p_2}^{c_3}, \overbrace{a \vee \neg p_2}^{c_4}\}$ and set $P := \{p_1, p_2\}$ of projection variables. The models of formula F are $\{a, b\}$, $\{a, p_1\}$, $\{a, b, p_1\}$,$\{a, b, p_2\}$, $\{a, p_1, p_2\}$, and $\{a, b, p_1, p_2\}$. However, projected to the set P, we only have models $\emptyset$, $\{p_1\}$, $\{p_2\}$, and $\{p_1, p_2\}$. Hence, the model count of F is 6 whereas the projected model count of instance (F, P) is 4. ■

Computational Complexity. We assume familiarity with standard notions in computational complexity [35] and use counting complexity classes as defined by Hesaspaandra and Vollmer [27]. For parameterized complexity, we refer to standard texts [12, 15, 20, 34]. Let Σ and Σ' be some finite alphabets. We call $I \in \Sigma^*$ an *instance* and $\|I\|$ denotes the size of I. Let $L \subseteq \Sigma^* \times \mathbb{N}$ and $L' \subseteq \Sigma'^* \times \mathbb{N}$ be two parameterized problems. An *fpt-reduction* r from L to L' is a many-to-one reduction from $\Sigma^* \times \mathbb{N}$ to $\Sigma'^* \times \mathbb{N}$ such that for all $I \in \Sigma^*$ we have $(I, k) \in L$ if and only if $r(I, k) = (I', k') \in L'$ such that $k' \leq g(k)$ for a fixed computable function $g : \mathbb{N} \to \mathbb{N}$, and there is a computable function f and a constant c such that r is computable in time $O(f(k)\|I\|^c)$ [20]. A *witness function* is a function $\mathcal{W}\colon \Sigma^* \to 2^{\Sigma'^*}$ that maps an instance $I \in \Sigma^*$ to a finite subset of Σ'^*. We call the set $\mathcal{W}(I)$ the *witnesses*. A *parameterized counting problem* $L : \Sigma^* \times \mathbb{N}_0 \to \mathbb{N}_0$ is a function that maps a given instance $I \in \Sigma^*$ and an integer $k \in \mathbb{N}$ to the cardinality of its witnesses $|\mathcal{W}(I)|$. We call k the *parameter*. The *exponential time hypothesis* (ETH) states that the (decision) problem Sat on 3-CNF formulas *cannot* be solved in time $2^{s \cdot n} \cdot n^{\mathcal{O}(1)}$ for some $s > 0$ where n is the number of variables [28].

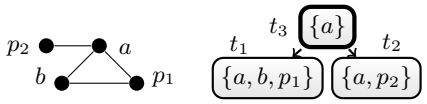

Fig. 1. Primal graph P_F of F from Example 2 (left) with a TD $\mathcal{T}$ of graph P_F (right).

Tree Decompositions and Treewidth. For basic terminology on graphs, we refer to standard texts [13, 8]. For a tree $T = (N, A, n)$ with root n and a node $t \in N$, we let children(t, T) be the sequence of all nodes t' in arbitrarily but fixed order, which have an edge $(t, t') \in A$. Let $G = (V, E)$ be a graph. A *tree decomposition (TD)* of graph G is a pair $\mathcal{T} = (T, \chi)$ where $T = (N, A, n)$ is a rooted tree, $n \in N$ the root, and χ a mapping that assigns to each node $t \in N$ a set $\chi(t) \subseteq V$, called a *bag*, such that the following conditions hold: (i) $V = \bigcup_{t \in N} \chi(t)$ and $E \subseteq \bigcup_{t \in N} \{uv \mid u, v \in \chi(t)\}$; (ii) for each $r, s, t \in N$ such that s lies on the path from r to t, we have $\chi(r) \cap \chi(t) \subseteq \chi(s)$. Then, width$(\mathcal{T}) := \max_{t \in N} |\chi(t)| - 1$. The *treewidth* $tw(G)$ of G is the minimum width$(\mathcal{T})$ over all tree decompositions $\mathcal{T}$ of G. For arbitrary but fixed $w \geq 1$, it is feasible in linear time to decide if a graph has treewidth at most w and, if so, to compute a tree decomposition of width w [5]. In order to simplify case distinctions in the algorithms, we always use so-called nice tree decompositions, which can be computed in linear time without increasing the width [7] and are defined as follows. For a node $t \in N$, we say that type(t) is *leaf* if children$(t, T) = \langle\rangle$; *join* if children$(t, T) = \langle t', t'' \rangle$ where $\chi(t) = \chi(t') = \chi(t'') \neq \emptyset$; *int* ("introduce") if children$(t, T) = \langle t' \rangle$, $\chi(t') \subseteq \chi(t)$ and $|\chi(t)| = |\chi(t')| + 1$; *rem* ("removal") if children$(t, T) = \langle t' \rangle$, $\chi(t') \supseteq \chi(t)$ and $|\chi(t')| = |\chi(t)| + 1$. If for every node $t \in N$, type$(t) \in \{leaf, join, int, rem\}$ and bags of leaf nodes and the root are empty, then the TD is called *nice*.

3 Dynamic Programming on TDs for SAT

Before we introduce our algorithm, we need some notations for dynamic programming on tree decompositions and recall how to solve the decision problem Sat by exploiting small treewidth.

Graph Representation of Sat *Formulas.* In order to use tree decompositions for satisfiability problems, we need a dedicated graph representation of the given formula F. The *primal graph* P_F of F has as vertices the variables of F and two variables are joined by an edge if they occur together in a clause of F. Further, we define some auxiliary notation. For a given node t of a tree decomposition (T, χ) of the primal graph, we let $F_t := \{c \mid c \in F, \text{var}(c) \subseteq \chi(t)\}$, i.e., clauses entirely covered by $\chi(t)$. The set $F_{\leq t}$ denotes the union over F_s for all descendant nodes $s \in N$ of t. In the following, we sometimes simply write *tree decomposition of formula* F or *treewidth of* F and omit the actual graph representation of F.

Example 2. Consider formula F from Example 1. The primal graph P_F of formula F and a tree decomposition $\mathcal{T}$ of P_F are depicted in Figure 1. Intuitively, $\mathcal{T}$ allows to evaluate formula F in parts. When evaluating $F_{\leq t_3}$, we split into $F_{\leq t_1} = \{c_1, c_2\}$ and $F_{\leq t_2} = \{c_3, c_4\}$, respectively. ■

Dynamic Programming on TDs. Algorithms that solve Sat or #Sat [41] in linear time for input formulas of bounded treewidth proceed by dynamic programming along the tree decomposition (in post-order) where at each node t of the tree information is gathered [6] in a table τ_t. A *table* τ is a set of rows, where a *row* $\boldsymbol{u} \in \tau$ is a sequence of fixed length. Tables are derived by an algorithm, which we therefore call *table algorithm* $\mathbb{A}$. The actual length, content, and meaning of the rows depend on the algorithm $\mathbb{A}$ that derives tables. Therefore, we often explicitly state $\mathbb{A}$-*row* if rows of this *type* are syntactically used for table algorithm $\mathbb{A}$ and similar $\mathbb{A}$-*table* for tables. For sake of comprehension, we specify the rows before presenting the actual table algorithm for manipulating tables. The rows used by a table algorithm $\mathbb{SAT}$ have in common that the first position of these rows manipulated by $\mathbb{SAT}$ consists of an interpretation. The remaining positions of the row depend on the considered table algorithm. For each sequence $\boldsymbol{u} \in \tau$, we write $I(\boldsymbol{u})$ to address the interpretation (first) part of the sequence $\boldsymbol{u}$. Further, for a given positive integer i, we denote by $\boldsymbol{u}_{(i)}$ the i-th element of row $\boldsymbol{u}$ and define $\tau_{(i)}$ as $\tau_{(i)} := \{\boldsymbol{u}_{(i)} \mid \boldsymbol{u} \in \tau\}$.

Then, the dynamic programming approach for propositional satisfiability performs the following steps:

1. Construct the primal graph P_F of F.
2. Compute a tree decomposition (T, χ) of P_F, where $T = (N, \cdot, n)$.
3. Run $\mathtt{DP}_{\mathbb{SAT}}$ (see Listing 1), which executes a table algorithm $\mathbb{SAT}$ for every node t in post-order of the nodes in N, and returns $\mathbb{SAT}$-Comp mapping every node t to its table. $\mathbb{SAT}$ takes as input[1] bag $\chi(t)$, sub-formula F_t, and tables Child-Tabs previously computed at children of t and outputs a table τ_t.
4. Print the result by interpreting the table for root n of T.

Listing 2 presents table algorithm $\mathbb{SAT}$ that uses the primal graph representation. We provide only brief intuition, for details we refer to the original source [41]. The main idea is to store in table τ_t only interpretations that are a model of sub-formula $F_{\leq t}$ when restricted to bag $\chi(t)$. Table algorithm $\mathbb{SAT}$ transforms at node t certain row combinations of the tables (Child-Tabs) of child nodes of t into rows of table τ_t. The transformation depends on a case where variable a is added or not added to an interpretation (*int*), removed from an interpretation (*rem*), or where coinciding interpretations are required (*join*). In the end, an interpretation $I(\boldsymbol{u})$ from a row $\boldsymbol{u}$ of the table τ_n at the root n proves that there is a supset $J \supseteq I(\boldsymbol{u})$ that is a model of $F = F_{\leq n}$, and hence that the formula is satisfiable. Example 3 lists selected tables when running algorithm $\mathtt{DP}_{\mathbb{SAT}}$.

Example 3. Consider formula F from Example 2. Figure 2 illustrates a tree decomposition $\mathcal{T}' = (\cdot, \chi)$ of the primal graph of F and tables τ_1, …, τ_{12} that are obtained during the execution of $\mathtt{DP}_{\mathbb{SAT}}((F, \cdot), \mathcal{T}', \cdot)$. We assume that each row in a table τ_t is identified by a number, i.e., row i corresponds to $\boldsymbol{u_{t.i}} = \langle J_{t.i} \rangle$.

[1] Actually, $\mathbb{SAT}$ takes in addition as input *PP*-Tabs, which contains a mapping of nodes of the tree decomposition to tables, i.e., tables of the previous pass. Later, we use this for a second traversal to pass results ($\mathbb{SAT}$-Comp) from the first traversal to the table algorithm $\mathbb{PROJ}$ for projected model counting in the second traversal.

Listing 1: Algorithm $\mathtt{DP}_{\mathbb{A}}((F,P),\mathcal{T},\mathit{PP\text{-}Tabs})$ for DP on TD $\mathcal{T}$ [18].

In: Table algorithm $\mathbb{A}$, TD $\mathcal{T}=(T,\chi)$ of F s.t. $T=(N,\cdot,n)$, tables *PP*-Tabs.
Out: Table $\mathbb{A}$-Comp, which maps each TD node $t\in N$ to some computed table τ_t.

```
for iterate t in post-order(T,n) do
    Child-Tabs := ⟨𝔸-Comp[t_1], ..., 𝔸-Comp[t_ℓ]⟩ where children(t,T) = ⟨t_1, ..., t_ℓ⟩
    𝔸-Comp[t] ← 𝔸(t, χ(t), F_t, P ∩ χ(t), Child-Tabs, PP-Tabs)
return 𝔸-Comp
```

Listing 2: Table algorithm $\mathbb{SAT}(t,\chi_t,F_t,\cdot,\text{Child-Tabs},\cdot)$ [41].

In: Node t, bag χ_t, clauses F_t, sequence Child-Tabs of tables. **Out:** Table τ_t.

```
if type(t) = leaf then τ_t ← {⟨∅⟩}
else if type(t) = int, a ∈ χ_t is introduced, and Child-Tabs = ⟨τ'⟩ then
    τ_t ← {⟨K⟩ | ⟨J⟩ ∈ τ', K ∈ {J, J ∪ {a}}, K ⊨ F_t}
else if type(t) = rem, a ∉ χ_t is removed, and Child-Tabs = ⟨τ'⟩ then
    τ_t ← {⟨J \ {a}⟩ | ⟨J⟩ ∈ τ'}
else if type(t) = join, and Child-Tabs = ⟨τ', τ''⟩ then
    τ_t ← {⟨J⟩ | ⟨J⟩ ∈ τ', ⟨J⟩ ∈ τ''}
return τ_t
```

Table $\tau_1=\{\langle\emptyset\rangle\}$ as $\text{type}(t_1)=\mathit{leaf}$. Since $\text{type}(t_2)=\mathit{int}$, we construct table τ_2 from τ_1 by taking $J_{1.i}$ and $J_{1.i}\cup\{a\}$ for each $\langle J_{1.i}\rangle\in\tau_1$. Then, t_3 introduces p_1 and t_4 introduces b. $F_{t_1}=F_{t_2}=F_{t_3}=\emptyset$, but since $\chi(t_4)\subseteq\text{var}(c_1)$ we have $F_{t_4}=\{c_1,c_2\}$ for t_4. In consequence, for each $J_{4.i}$ of table τ_4, we have $\{c_1,c_2\}\vDash J_{4.i}$ since $\mathbb{SAT}$ enforces satisfiability of F_t in node t. Since $\text{type}(t_5)=\mathit{rem}$, we remove variable p_1 from all elements in τ_4 to construct τ_5. Note that we have already seen all rules where p_1 occurs and hence p_1 can no longer affect interpretations during the remaining traversal. We similarly create $\tau_6=\{\langle\emptyset\rangle,\langle a\rangle\}$ and $\tau_{10}=\{\langle a\rangle\}$. Since $\text{type}(t_{11})=\mathit{join}$, we build table τ_{11} by taking the intersection of τ_6 and τ_{10}. Intuitively, this combines interpretations agreeing on a. By definition (primal graph and TDs), for every $c\in F$, variables $\text{var}(c)$ occur together in at least one common bag. Hence, $F=F_{\leq t_{12}}$ and since $\tau_{12}=\{\langle\emptyset\rangle\}$, we can reconstruct for example model $\{a,b,p_2\}=J_{11.1}\cup J_{5.4}\cup J_{9.2}$ of F using highlighted (yellow) rows in Figure 2. On the other hand, if F was unsatisfiable, τ_{12} would be empty ($\emptyset$). ■

The following definition simplifies the presentation. At a node t and for a row $\boldsymbol{u}$ of the table $\mathbb{SAT}$-Comp$[t]$, it yields the rows in the tables of the children of t that were involved in computing row $\boldsymbol{u}$ by algorithm $\mathbb{SAT}$.

Definition 1 (c.f., [19]). *Let F be a formula, $\mathcal{T}=(T,\chi)$ be a tree decomposition of F, t be a node of T that has ℓ children, and $\tau_1,\ldots,\tau_\ell$ be the $\mathbb{SAT}$-tables computed by $\mathtt{DP}_{\mathbb{SAT}}((F,\cdot),\mathcal{T},\cdot)$ where $\text{children}(t,T)=\langle t_1,\ldots,t_\ell\rangle$. Given a sequence $\boldsymbol{s}=\langle s_1,\ldots,s_\ell\rangle$, we let $\langle\!\{\boldsymbol{s}\}\!\rangle:=\langle\{s_1\},\ldots,\{s_\ell\}\rangle$, for technical reasons.*

For a given $\mathbb{SAT}$-row $\boldsymbol{u}$, we define the originating $\mathbb{SAT}$-rows of $\boldsymbol{u}$ in node t by $\mathbb{SAT}\text{-origins}(t,\boldsymbol{u}):=\{\boldsymbol{s}\mid\boldsymbol{s}\in\tau_1\times\cdots\times\tau_\ell,\tau=\mathbb{SAT}(t,\chi(t),F_t,\cdot,\langle\!\{\boldsymbol{s}\}\!\rangle,\cdot),\boldsymbol{u}\in\tau\}$. We extend this to a $\mathbb{SAT}$-table σ by $\mathbb{SAT}\text{-origins}(t,\sigma):=\bigcup_{\boldsymbol{u}\in\sigma}\mathbb{SAT}\text{-origins}(t,\boldsymbol{u})$.

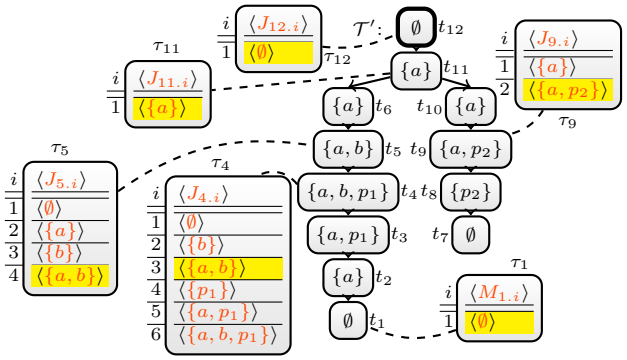

Fig. 2. Selected tables obtained by algorithm $\mathtt{DP}_{\mathbb{PRIM}}$ on tree decomposition $\mathcal{T}'$.

Remark 1. An actual implementation would not compute origins, but store and reuse them without side-effects to worst-case complexity during tree traversal.

Example 4. Consider formula F, tree decomposition $\mathcal{T}' = (T, \chi)$, and tables $\tau_1, \ldots, \tau_{12}$ from Example 3. We focus on $\boldsymbol{u_{1.1}} = \langle J_{1.1} \rangle = \langle \emptyset \rangle$ of table τ_1 of the leaf t_1. The row $\boldsymbol{u_{1.1}}$ has no preceding row, since $\text{type}(t_1) = \textit{leaf}$. Hence, we have $\mathbb{SAT}\text{-origins}(t_1, \boldsymbol{u_{1.1}}) = \{\langle\rangle\}$. The origins of row $\boldsymbol{u_{5.1}}$ of table τ_5 are given by $\mathbb{SAT}\text{-origins}(t_5, \boldsymbol{u_{5.1}})$, which correspond to the preceding rows in table t_4 that lead to row $\boldsymbol{u_{5.1}}$ of table τ_5 when running algorithm $\mathbb{SAT}$, i.e., $\mathbb{SAT}\text{-origins}(t_5, \boldsymbol{u_{5.1}}) = \{\langle \boldsymbol{u_{4.1}} \rangle, \langle \boldsymbol{u_{4.4}} \rangle\}$. Observe that $\mathbb{SAT}\text{-origins}(t_i, \boldsymbol{u}) = \emptyset$ for any row $\boldsymbol{u} \notin \tau_i$. For node t_{11} of type *join* and row $\boldsymbol{u_{11.1}}$, we obtain $\mathbb{SAT}\text{-origins}(t_{11}, \boldsymbol{u_{11.1}}) = \{\langle \boldsymbol{u_{6.2}}, \boldsymbol{u_{10.1}} \rangle\}$ (see Example 3). More general, when using algorithm $\mathbb{SAT}$, at a node t of type *join* with table τ we have $\mathbb{SAT}\text{-origins}(t, \boldsymbol{u}) = \{\langle \boldsymbol{u}, \boldsymbol{u} \rangle\}$ for row $\boldsymbol{u} \in \tau$. ■

Definition 1 talked about a top-down direction for rows and their origins. In addition, we need definitions to talk about a recursive version of these origins from a node t down to the leafs, mainly to state properties for our algorithms.

Definition 2. *Let F be a formula, $\mathcal{T} = (T, \chi)$ be a tree decomposition with $T = (N, \cdot, n)$, $t \in N$, $\mathbb{SAT}$-Comp$[t']$ be obtained by $\mathtt{DP}_{\mathbb{SAT}}((F, \cdot), \mathcal{T}, \cdot)$ for each node t' of the* induced sub-tree $T[t]$ rooted at t*, and $\boldsymbol{u}$ be a row of $\mathbb{SAT}$-Comp$[t]$.*

An extension below t *is a set of pairs where a pair consists of a node t' of $T[t]$ and a row $\boldsymbol{v}$ of $\mathbb{SAT}$-Comp$[t']$ and the cardinality of the set equals the number of nodes in the sub-tree $T[t]$. We define the family of* extensions below t *recursively as follows. If t is of type leaf, then* $\text{Ext}_{\leq t}(\boldsymbol{u}) := \{\{\langle t, \boldsymbol{u} \rangle\}\}$*; otherwise* $\text{Ext}_{\leq t}(\boldsymbol{u}) := \bigcup_{\boldsymbol{v} \in \mathbb{SAT}\text{-origins}(t, \boldsymbol{u})} \big\{\{\langle t, \boldsymbol{u} \rangle\} \cup X_1 \cup \ldots \cup X_\ell \mid X_i \in \text{Ext}_{\leq t_i}(\boldsymbol{v}_{(i)})\big\}$ *for the ℓ children $t_1, \ldots, t_\ell$ of t. We extend this notation for a $\mathbb{SAT}$-table σ by* $\text{Ext}_{\leq t}(\sigma) := \bigcup_{\boldsymbol{u} \in \sigma} \text{Ext}_{\leq t}(\boldsymbol{u})$*. Further, we let* $\text{Exts} := \text{Ext}_{\leq n}(\mathbb{SAT}\text{-Comp}[n])$.

If we would construct all extensions below the root n, it allows us to also obtain all models of a formula F. To this end, we state the following definition.

Definition 3. *Let F be a formula, $\mathcal{T} = (T, \chi)$ be a tree decomposition of F, t be a node of T, and $\sigma \subseteq \mathbb{SAT}$-Comp$[t]$ be a set of $\mathbb{SAT}$-rows that have been computed*

by $\mathtt{DP}_{\mathbb{SAT}}((F,\cdot),\mathcal{T},\cdot)$ *at* t*. We define the* satisfiable extensions below t *for* σ *by* $\mathrm{SatExt}_{\leq t}(\sigma) := \bigcup_{\boldsymbol{u}\in\sigma}\{X \mid X \in \mathrm{Ext}_{\leq t}(\boldsymbol{u}), X \subseteq Y, Y \in \mathrm{Exts}\}$.

Observation 1. *Let* F *be a formula,* $\mathcal{T}$ *be a tree decomposition with root* n *of* F*. Then,* $\mathrm{SatExt}_{\leq n}(\mathbb{SAT}\text{-}Comp[t]) = \mathrm{Exts}$.

Next, we define an auxiliary notation that gives us a way to reconstruct interpretations from families of extensions.

Definition 4. *Let* (F,P) *be an instance of* PMC*,* $\mathcal{T} = (T,\chi)$ *be a tree decomposition of* F*,* t *be a node of* T*. Further, let* E *be a family of extensions below* t*, and* P *be a set of projection variables. We define the* set $I(E)$ of interpretations *of* E *by* $I(E) := \big\{\bigcup_{\langle\cdot,\boldsymbol{u}\rangle\in X} I(\boldsymbol{u}) \mid X \in E\big\}$ *and the set* $I_P(E)$ *of* projected interpretations *by* $I_P(E) := \big\{\bigcup_{\langle\cdot,\boldsymbol{u}\rangle\in X} I(\boldsymbol{u}) \cap P \mid X \in E\big\}$.

Example 5. Consider again formula F and tree decomposition $\mathcal{T}'$ with root n of F from Example 3. Let $X = \{\langle t_{12},\langle\emptyset\rangle\rangle, \langle t_{11},\langle\{a\}\rangle\rangle, \langle t_6,\langle\{a\}\rangle\rangle, \langle t_5,\langle\{a,b\}\rangle\rangle, \langle t_4,\langle\{a,b\}\rangle\rangle, \langle t_3,\langle\{a\}\rangle\rangle, \langle t_2,\langle\{a\}\rangle\rangle, \langle t_1,\langle\emptyset\rangle\rangle, \langle t_{10},\langle\{a\}\rangle\rangle, \langle t_9,\langle\{a,p_2\}\rangle\rangle, \langle t_8,\langle\{p_2\}\rangle\rangle, \langle t_7,\langle\emptyset\rangle\rangle\}$ be an extension below n. Observe that $X \in \mathrm{Exts}$ and that Figure 2 highlights those rows of tables for nodes $t_{12}, t_{11}, t_9, t_5, t_4$ and t_1 that also occur in X (in yellow). Further, $I(\{X\}) = \{a,b,p_2\}$ computes the corresponding model of X, and $I_P(\{X\}) = \{p_2\}$ derives the projected model of X. $I(\mathrm{Exts})$ refers to the set of models of F, whereas $I_P(\mathrm{Exts})$ is the set of projected models of F. ■

4 Counting Projected Models by Dynamic Programming

In this section, we introduce the dynamic programming algorithm $\mathtt{PCNT}_{\mathbb{SAT}}$ to solve the projected model counting problem (PMC) for Boolean formulas. Our algorithm traverses the tree decomposition twice following a multi-pass dynamic programming paradigm [19]. Similar to the previous section, we construct a graph representation and heuristically compute a tree decomposition of this graph. Then, we run $\mathtt{DP}_{\mathbb{SAT}}$ (see Listing 1) in Step 3a as first traversal. Step 3a can also be seen as a preprocessing step for projected model counting, from which we immediately know whether the problem has a solution. Afterwards we remove all rows from the $\mathbb{SAT}$-tables which cannot be extended to a solution for the SAT problem (*"Purge non-solutions"*). In other words, we keep only rows $\boldsymbol{u}$ in table $\mathbb{SAT}$-Comp$[t]$ at node t if its interpretation $I(\boldsymbol{u})$ can be extended to a model of F, more formally, $(t,\boldsymbol{u}) \in X$ for some $X \in \mathrm{SatExt}_{\leq t}(\mathbb{SAT}\text{-Comp}[t])$. Thereby, we avoid redundancies and can simplify the description of our next step, since we then only have to consider (parts of) models. In Step 3b ($\mathtt{DP}_{\mathbb{PROJ}}$), we traverse the tree decomposition a second time to count projections of interpretations of rows in $\mathbb{SAT}$-tables. In the following, we only describe the table algorithm $\mathbb{PROJ}$, since the traversal in $\mathtt{DP}_{\mathbb{PROJ}}$ is the same as before. For $\mathbb{PROJ}$, a row at a node t is a pair $\langle\sigma,c\rangle$ where σ is a $\mathbb{SAT}$-table, in particular, a subset of $\mathbb{SAT}$-Comp$[t]$ computed by $\mathtt{DP}_{\mathbb{SAT}}$, and c is a non-negative integer. In fact, we store in integer c a count that expresses the number of "all-overlapping" solutions (ipmc), whereas in the end we aim for the projected model count (pmc), clarified in the following.

Definition 5. *Let F be a formula, $\mathcal{T} = (T, \chi)$ be a tree decomposition of F, t be a node of T, $\sigma \subseteq \mathbb{SAT}$-Comp$[t]$ be a set of $\mathbb{SAT}$-rows that have been computed by $\mathtt{DP}_{\mathbb{SAT}}((F, \cdot), \mathcal{T}, \cdot)$ at node t in T. Then, the* projected model count $\mathrm{pmc}_{\leq t}(\sigma)$ *of σ below t is the size of the union over projected interpretations of the satisfiable extensions of σ below t, formally,* $\mathrm{pmc}_{\leq t}(\sigma) := |\bigcup_{\boldsymbol{u} \in \sigma} I_P(\mathrm{SatExt}_{\leq t}(\{\boldsymbol{u}\}))|$.

The intersection projected model count $\mathrm{ipmc}_{\leq t}(\sigma)$ *of σ below t is the size of the intersection over projected interpretations of the satisfiable extensions of σ below t, i.e.,* $\mathrm{ipmc}_{\leq t}(\sigma) := |\bigcap_{\boldsymbol{u} \in \sigma} I_P(\mathrm{SatExt}_{\leq t}(\{\boldsymbol{u}\}))|$.

The next definitions provide central notions for grouping rows of tables according to the given projection of variables.

Definition 6. *Let (F, P) be an instance of* PMC *and σ be a $\mathbb{SAT}$-table. We define the relation $=_{\mathrm{P}} \subseteq \sigma \times \sigma$ to consider equivalent rows with respect to the projection of its interpretations by $=_{\mathrm{P}} := \{(\boldsymbol{u}, \boldsymbol{v}) \mid \boldsymbol{u}, \boldsymbol{v} \in \sigma, I(\boldsymbol{u}) \cap P = I(\boldsymbol{v}) \cap P\}$.*

Observation 2. *The relation $=_{\mathrm{P}}$ is an equivalence relation.*

Definition 7. *Let τ be a $\mathbb{SAT}$-table and $\boldsymbol{u}$ be a row of τ. The relation $=_{\mathrm{P}}$ induces equivalence classes $[\boldsymbol{u}]_P$ on the $\mathbb{SAT}$-table τ in the usual way, i.e., $[\boldsymbol{u}]_P = \{\boldsymbol{v} \mid \boldsymbol{v} =_{\mathrm{P}} \boldsymbol{u}, \boldsymbol{v} \in \tau\}$ [44]. We denote by* $\mathrm{buckets}_P(\tau)$ *the set of equivalence classes of τ, i.e.,* $\mathrm{buckets}_P(\tau) := (\tau / =_{\mathrm{P}}) = \{[\boldsymbol{u}]_P \mid \boldsymbol{u} \in \tau\}$. *Further, we define the set* $\text{sub-buckets}_P(\tau)$ *of all sub-equivalence classes of τ by* $\text{sub-buckets}_P(\tau) := \{S \mid \emptyset \subsetneq S \subseteq B, B \in \mathrm{buckets}_P(\tau)\}$.

Example 6. Consider again formula F and set P of projection variables from Example 1 and tree decomposition $\mathcal{T}' = (T, \chi)$ and $\mathbb{SAT}$-table τ_4 from Figure 2. We have $\boldsymbol{u_{4.1}} =_P \boldsymbol{u_{4.2}}$ and $\boldsymbol{u_{4.4}} =_P \boldsymbol{u_{4.5}}$. We obtain the set $\tau_4 / =_{\mathrm{P}}$ of equivalence classes of τ_4 by $\mathrm{buckets}_P(\sigma) = \{\{\boldsymbol{u_{4.1}}, \boldsymbol{u_{4.2}}, \boldsymbol{u_{4.3}}\}, \{\boldsymbol{u_{4.4}}, \boldsymbol{u_{4.5}}, \boldsymbol{u_{4.6}}\}\}$. ■

Since $\mathbb{PROJ}$ stores a counter in $\mathbb{PROJ}$-tables together with a $\mathbb{SAT}$-table, we need an auxiliary definition that given $\mathbb{SAT}$-table allows us to select the respective counts from a $\mathbb{PROJ}$-table. Later, we use the definition in the context of looking up the already computed projected counts for tables of *children* of a given node.

Definition 8. *Given a $\mathbb{PROJ}$-table ι and a $\mathbb{SAT}$-table σ we define the* stored ipmc *for all rows of σ in ι by* $\text{s-ipmc}(\iota, \sigma) := \sum_{\langle \sigma, c \rangle \in \iota} c$. *Later, we apply this to rows from several origins. Therefore, for a sequence s of $\mathbb{PROJ}$-tables of length ℓ and a set O of sequences of $\mathbb{SAT}$-rows where each sequence is of length ℓ, we let* $\text{s-ipmc}(s, O) = \prod_{i \in \{1, \ldots, \ell\}} \text{s-ipmc}(s_{(i)}, O_{(i)})$.

When computing s-ipmc in Definition 8, we select the i-th position of the sequence together with sets of the i-th position from the set of sequences. We need this somewhat technical construction, since later at node t we apply this definition to $\mathbb{PROJ}$-tables of children of t and origins of subsets of $\mathbb{SAT}$-tables. There, we may simply have several children if the node is of type *join* and hence we need to select from the right children.

Now, we are in position to give a core definition for our algorithm that solves PMC. Intuitively, when we are at a node t in the Algorithm $\mathtt{DP}_{\mathbb{PROJ}}$ we already

computed all tables $\mathbb{SAT}$-Comp by $\mathtt{DP}_{\mathbb{SAT}}$ according to Step 3a, purged non-solutions, and computed $\mathbb{PROJ}$-Comp$[t']$ for all nodes t' below t and in particular the $\mathbb{PROJ}$-tables Child-Tabs of the children of t. Then, we compute the projected model count of a subset σ of the $\mathbb{SAT}$-rows in $\mathbb{SAT}$-Comp$[t]$, which we formalize in the following definition, by applying the generalized inclusion-exclusion principle to the stored projected model count of origins.

Definition 9. *Let (F, P) be an instance of* PMC*, $\mathcal{T} = (T, \chi)$ be a tree decomposition of F, $\mathbb{SAT}$-Comp$[s]$ be the $\mathbb{SAT}$-tables computed by $\mathtt{DP}_{\mathbb{SAT}}((F, \cdot), \mathcal{T}, \cdot)$ for every node s of T. Further, let t be a node of T with ℓ children, Child-Tabs $= \langle \mathbb{PROJ}\text{-Comp}[t_1], \ldots, \mathbb{PROJ}\text{-Comp}[t_\ell] \rangle$ be the sequence of $\mathbb{PROJ}$-tables computed by $\mathtt{DP}_{\mathbb{PROJ}}((F, P), \mathcal{T}, \mathbb{SAT}\text{-Comp})$ where* children$(t, T) = \langle t_1, \ldots, t_\ell \rangle$*, and $\sigma \subseteq \mathbb{SAT}$-Comp$[t]$ be a table. We define the* (inductive) projected model count *of σ:*

$$\text{pmc}(t, \sigma, \textit{Child-Tabs}) := \sum_{\emptyset \subsetneq O \subseteq \mathbb{SAT}\text{-origins}(t,\sigma)} (-1)^{(|O|-1)} \cdot \quad \text{s-ipmc}(\textit{Child-Tabs}, O).$$

Vaguely speaking, pmc determines the $\mathbb{SAT}$-origins of the set σ of rows, goes over all subsets of these origins and looks up the stored counts (s-ipmc) in the $\mathbb{PROJ}$-tables of the children of t. Example 7 provides an idea on how to compute the projected model count of tables of our running example using pmc.

Example 7. The function defined in Definition 9 allows us to compute the projected count for a given $\mathbb{SAT}$-table. Therefore, consider again formula F and tree decomposition $\mathcal{T}'$ from Example 2 and Figure 2. Say we want to compute the projected count pmc$(t_5, \{\boldsymbol{u_{5.4}}\}, \text{Child-Tabs})$ where Child-Tabs $:= \big\{\langle\{\boldsymbol{u_{4.3}}\}, 1\rangle, \langle\{\boldsymbol{u_{4.6}}\}, 1\rangle\big\}$ for row $\boldsymbol{u_{5.4}}$ of table τ_5. Note that t_5 has $\ell = 1$ child nodes $\langle t_4 \rangle$ and therefore the product of Definition 8 consists of only one factor. Observe that $\mathbb{SAT}$-origins$(t_5, \boldsymbol{u_{5.4}}) = \{\langle \boldsymbol{u_{4.3}} \rangle, \langle \boldsymbol{u_{4.6}} \rangle\}$. Since the rows $\boldsymbol{u_{4.3}}$ and $\boldsymbol{u_{4.6}}$ do not occur in the same $\mathbb{SAT}$-table of Child-Tabs, only the value of s-ipmc for the two singleton origin sets $\{\langle \boldsymbol{u_{4.3}} \rangle\}$ and $\{\langle \boldsymbol{u_{4.6}} \rangle\}$ is non-zero; for the remaining set of origins we have zero. Hence, we obtain pmc$(t_5, \{\boldsymbol{u_{5.4}}\}, \text{Child-Tabs}) = 2$. ■

Before we present algorithm $\mathbb{PROJ}$ (Listing 3), we give a definition that allows us at a certain node t to compute the intersection pmc for a given $\mathbb{SAT}$-table σ by computing the pmc (using stored ipmc values from $\mathbb{PROJ}$-tables for children of t), and subtracting and adding ipmc values for subsets $\emptyset \subsetneq \rho \subsetneq \sigma$ accordingly.

Definition 10. *Let $\mathcal{T} = (T, \cdot)$ be a tree decomposition, t be a node of T, ρ be $\mathbb{SAT}$-table, and Child-Tabs be a sequence of tables. Then, we define the* (recursive) ipmc *of σ as follows:*

$$\text{ipmc}(t, \sigma, \textit{Child-Tabs}) := \begin{cases} 1, \ \textit{if } \text{type}(t) = \textit{leaf}, \\ \big|\, \text{pmc}(t, \sigma, \textit{Child-Tabs}) + \\ \quad \sum_{\emptyset \subsetneq \rho \subsetneq \sigma} (-1)^{|\rho|} \cdot \text{ipmc}(t, \rho, \textit{Child-Tabs}) \big|, \textit{otherwise.} \end{cases}$$

In other words, if a node is of type *leaf* the ipmc is one, since by definition of a tree decomposition the bags of nodes of type *leaf* contain only one projected interpretation (the empty set). Otherwise, using Definition 9, we are able to

Listing 3: Table algorithm $\mathbb{PROJ}(t, \cdot, \cdot, P, \text{Child-Tabs}, \mathbb{SAT}\text{-Comp})$.

In: Node t, set P of projection variables, Child-Tabs, and $\mathbb{SAT}$-Comp.
Out: Table ι_t consisting of pairs $\langle \sigma, c \rangle$, where $\sigma \subseteq \mathbb{SAT}\text{-Comp}[t]$ and $c \in \mathbb{N}$.
1 $\iota_t \leftarrow \big\{ \langle \sigma, \text{ipmc}(t, \sigma, \text{Child-Tabs}) \rangle \big| \, \sigma \in \text{sub-buckets}_P(\mathbb{SAT}\text{-Comp}[t]) \big\}$
2 **return** ι_t

compute the ipmc for a given $\mathbb{SAT}$-table σ, which is by construction the same as $\text{ipmc}_{\leq t}(\sigma)$ (c.f. proof of Theorem 3 later). In more detail, we want to compute for a $\mathbb{SAT}$-table σ its ipmc that represents "all-overlapping" counts of σ with respect to set P of projection variables, that is, $\text{ipmc}_{\leq t}(\sigma)$. Therefore, for ipmc, we rearrange the inclusion-exclusion principle. To this end, we take pmc, which computes the "non-overlapping" count of σ with respect to P, by once more exploiting the inclusion-exclusion principle on $\mathbb{SAT}$-origins of σ (as already discussed) such that we count every projected model only once. Then we have to alternately subtract and add ipmc values for strict subsets ρ of σ, accordingly.

Finally, Listing 3 presents table algorithm $\mathbb{PROJ}$, which stores for given node t a $\mathbb{PROJ}$-table consisting of every sub-bucket of the given table $\mathbb{SAT}$-Comp$[t]$ together with its ipmc (as presented above).

Example 8. Recall instance (F, P), tree decomposition $\mathcal{T}'$, and tables τ_1, ..., τ_{12} from Example 1, 2, and Figure 2. Figure 3 depicts selected tables of $\iota_1, \ldots \iota_{12}$ obtained after running $\mathtt{DP}_{\mathbb{PROJ}}$ for counting projected interpretations. We assume numbered rows, i.e., row i in table ι_t corresponds to $\boldsymbol{v_{t.i}} = \langle \sigma_{t.i}, c_{t.i} \rangle$. Note that for some nodes t, there are rows among different $\mathbb{SAT}$-tables that occur in $\text{Ext}_{\leq t}$, but not in $\text{SatExt}_{\leq t}$. These rows are removed during purging. In fact, rows $\boldsymbol{u_{4.1}}, \boldsymbol{u_{4.2}}$, and $\boldsymbol{u_{4.4}}$ do not occur in table ι_4. Observe that purging is a crucial trick here that avoids to correct stored counters c by backtracking whenever a certain row of a table has no succeeding row in the parent table.

Next, we discuss selected rows obtained by $\mathtt{DP}_{\mathbb{PROJ}}((F, P), \mathcal{T}', \mathbb{SAT}\text{-Comp})$. Tables ι_1, ..., ι_{12} that are computed at the respective nodes of the tree decomposition are shown in Figure 3. Since $\text{type}(t_1) = \textit{leaf}$, we have $\iota_1 = \langle \{\langle \emptyset \rangle\}, 1 \rangle$. Intuitively, up to node t_1 the $\mathbb{SAT}$-row $\langle \emptyset \rangle$ belongs to 1 bucket. Node t_2 introduces variable a, which results in table $\iota_2 := \big\{ \langle \{\langle \{a\} \rangle\}, 1 \rangle \big\}$. Note that the $\mathbb{SAT}$-row $\langle \emptyset \rangle$ is subject to purging. Node t_3 introduces p_1 and node t_4 introduces b. Node t_5 removes projected variable p_1. The row $\boldsymbol{v_{5.2}}$ of $\mathbb{PROJ}$-table ι_5 has already been discussed in Example 7 and row $\boldsymbol{v_{5.1}}$ works similar. For row $\boldsymbol{v_{5.3}}$ we compute the count $\text{ipmc}(t_5, \{\boldsymbol{u_{5.2}}, \boldsymbol{u_{5.4}}\}, \langle \iota_4 \rangle)$ by means of pmc. Therefore, take for ρ the sets $\{\boldsymbol{u_{5.2}}\}$, $\{\boldsymbol{u_{5.4}}\}$, and $\{\boldsymbol{u_{5.2}}, \boldsymbol{u_{5.4}}\}$. For the singleton sets, we simply have $\text{pmc}(t_5, \{\boldsymbol{u_{5.2}}\}, \langle \iota_4 \rangle) = \text{ipmc}(t_5, \{\boldsymbol{u_{5.2}}\}, \langle \iota_4 \rangle) = c_{5.1} = 1$ and $\text{pmc}(t_5, \{\boldsymbol{u_{5.4}}\}, \langle \iota_4 \rangle) = \text{ipmc}(t_5, \{\boldsymbol{u_{5.4}}\}, \langle \iota_4 \rangle) = c_{5.2} = 2$. To compute $\text{pmc}(t_5, \{\boldsymbol{u_{5.2}}, \boldsymbol{u_{5.4}}\}, \langle \iota_4 \rangle)$ following Definition 9, take for O the sets $\{\boldsymbol{u_{4.5}}\}$, $\{\boldsymbol{u_{4.3}}\}$, and $\{\boldsymbol{u_{4.6}}\}$ into account, since all other non-empty subsets of origins of $\boldsymbol{u_{5.2}}$ and $\boldsymbol{u_{5.4}}$ in ι_4 do not occur in ι_4. Then, we take the sum over the values $\text{s-ipmc}(\langle t_4 \rangle, \{\langle \boldsymbol{u_{4.5}} \rangle\}) = 1$, $\text{s-ipmc}(\langle t_4 \rangle, \{\langle \boldsymbol{u_{4.3}} \rangle\}) = 1$, and $\text{s-ipmc}(\langle t_4 \rangle, \{\langle \boldsymbol{u_{4.6}} \rangle\}) = 1$; and subtract $\text{s-ipmc}(\langle t_4 \rangle, \{\langle \boldsymbol{u_{4.5}} \rangle, \langle \boldsymbol{u_{4.6}} \rangle\}) = 1$. Hence, $\text{pmc}(t_5, \{\boldsymbol{u_{5.2}}, \boldsymbol{u_{5.4}}\}, \langle \iota_4 \rangle) = 2$. In order to compute $\text{ipmc}(t_5, \{\boldsymbol{u_{5.2}}, \boldsymbol{u_{5.4}}\}, \langle \iota_4 \rangle) = |\, \text{pmc}(t_5, \{\boldsymbol{u_{5.2}}, \boldsymbol{u_{5.4}}\},$

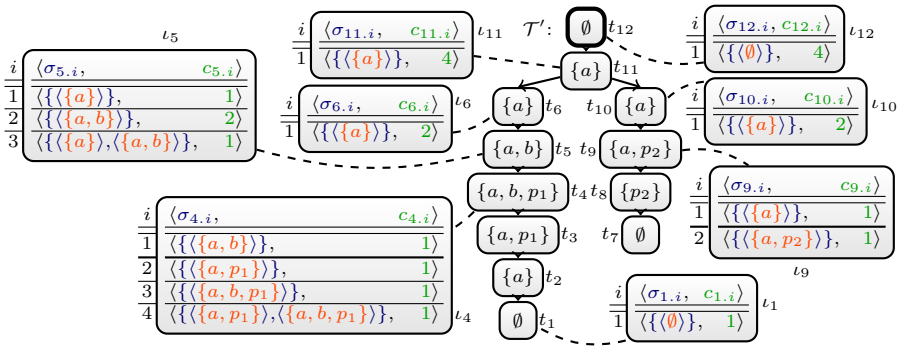

Fig. 3. Selected tables obtained by $\mathtt{DP}_{\mathbb{PROJ}}$ on TD $\mathcal{T}'$ using $\mathtt{DP}_{\mathbb{SAT}}$ (c.f., Figure 2).

$\langle \iota_4 \rangle) - \text{ipmc}(t_5, \{\boldsymbol{u_{5.2}}\}, \langle \iota_4 \rangle) - \text{ipmc}(t_5, \{\boldsymbol{u_{5.4}}\}, \langle \iota_4 \rangle)| = |2 - 1 - 2| = |-1| = 1$. Hence, $c_{5.3} = 1$ represents the number of projected models, both rows $\boldsymbol{u_{5.2}}$ and $\boldsymbol{u_{5.4}}$ have in common. We then use it for table t_6.

For node t_{11} of type *join* one simply in addition multiplies stored s-ipmc values for $\mathbb{SAT}$-rows in the two children of t_{11} accordingly (see Definition 8). In the end, the projected model count of F corresponds to s-ipmc$(\iota_{12}, \cdot) = 4$. ■

5 Runtime (Upper and Lower Bounds)

In this section, we first present asymptotic upper bounds on the runtime of our Algorithm $\mathtt{DP}_{\mathbb{PROJ}}$. For the analysis, we assume $\gamma(n)$ to be the costs for multiplying two n-bit integers, which can be achieved in time $n \cdot \log n \cdot \log \log n$ [30, 26].

Then, our main result is a lower bound that establishes that there cannot be an algorithm that solves PMC in time that is only single exponential in the treewidth and polynomial in the size of the formula unless the exponential time hypothesis (ETH) fails. This result establishes that there *cannot* be an algorithm exploiting treewidth that is *asymptotically better* than our presented algorithm, although one can likely improve on the analysis and give a better algorithm.

Theorem 1. *Given a* PMC *instance* (F, P) *and a tree decomposition* $\mathcal{T} = (T, \chi)$ *of* F *of width* k *with* g *nodes. Algorithm* $\mathtt{DP}_{\mathbb{PROJ}}$ *runs in time* $\mathcal{O}(2^{2^{k+4}} \cdot \gamma(\|F\|) \cdot g)$.

Proof. Let $d = k + 1$ be maximum bag size of $\mathcal{T}$. For each node t of T, we consider table $\tau = \mathbb{SAT}\text{-Comp}[t]$ which has been computed by $\mathtt{DP}_{\mathbb{SAT}}$ [41]. The table τ has at most 2^d rows. In the worst case we store in $\iota = \mathbb{PROJ}\text{-Comp}[t]$ each subset $\sigma \subseteq \tau$ together with exactly one counter. Hence, we have 2^{2^d} many rows in ι. In order to compute ipmc for σ, we consider every subset $\rho \subseteq \sigma$ and compute pmc. Since $|\sigma| \leq 2^d$, we have at most 2^{2^d} many subsets ρ of σ. For computing pmc, there could be each subset of the origins of ρ for each child table, which are less than $2^{2^{d+1}} \cdot 2^{2^{d+1}}$ (join and remove case). In total, we obtain a runtime bound of $\mathcal{O}(2^{2^d} \cdot 2^{2^d} \cdot 2^{2^{d+1}} \cdot 2^{2^{d+1}} \cdot \gamma(\|F\|)) \subseteq \mathcal{O}(2^{2^{d+3}} \cdot \gamma(\|F\|))$ since

we also need multiplication of counters. Then, we apply this to every node t of the tree decomposition, which results in running time $\mathcal{O}(2^{2^{d+3}} \cdot \gamma(\|F\|) \cdot g)$. □

Corollary 1. *Given an instance (F, P) of* PMC *where F has treewidth k. Algorithm* $\texttt{PCNT}_{\mathbb{SAT}}$ *runs in time $\mathcal{O}(2^{2^{k+4}} \cdot \gamma(\|F\|) \cdot \|F\|)$.*

Proof. We compute in time $2^{\mathcal{O}(k^3)} \cdot |V|$ a tree decomposition $\mathcal{T}'$ of width at most k [5] of primal graph P_F. Then, we run a decision version of the algorithm $\texttt{DP}_{\mathbb{SAT}}$ by Samer and Szeider [41] in time $\mathcal{O}(2^k \cdot \gamma(\|F\|) \cdot \|F\|)$. Then, we again traverse the decomposition, thereby keeping rows that have a satisfying extension ("purging"), in time $\mathcal{O}(2^k \cdot \|F\|)$. Finally, we run $\texttt{DP}_{\mathbb{PROJ}}$ and obtain the claim by Theorem 1 and since $\mathcal{T}'$ has linearly many nodes [5]. □

The next results also establish the lower bounds for our worst-cases.

Theorem 2. *Unless ETH fails,* PMC *cannot be solved in time $2^{2^{o(k)}} \cdot \|F\|^{o(k)}$ for a given instance (F, P) where k is the treewidth of the primal graph of F.*

Proof. Assume for proof by contradiction that there is such an algorithm. We show that this contradicts a recent result [32], which states that one cannot decide the validity of a QBF [4, 29] $Q = \exists V_1.\forall V_2.E$ in time $2^{2^{o(k)}} \cdot \|E\|^{o(k)}$ under ETH. Given an instance (Q, k) of $\exists\forall$-Sat when parameterized by the treewidth k of E, we provide a reduction to an instance $(\forall V_1.\exists V_2.E', k)$ of $\forall\exists$-Sat where $E' \equiv \neg E$ and E' is in CNF. Observe that the primal graphs of E and E' are isomorphic and therefore have the same treewidth k [32]. Then, given an instance $(\forall V_1.\exists V_2.E', k)$ of $\forall\exists$-Sat when parameterized by the treewidth k, we provide a reduction to an instance $((F, P, n), k)$ of decision version PMC-exactly-n of PMC such that $F = E$, $P = V_1$, and the number n of solutions is exactly $2^{|V_1|}$. The reduction gives a yes instance $((F, P, n), k)$ of PMC-exactly-n if and only if $(\forall V_1.\exists V_2.E', k)$ is a yes instance of $\forall\exists$-Sat. The reduction is also an fpt-reduction, since the treewidth of F is exactly k. □

Corollary 2. *Given an instance (F, P) of* PMC *where F has treewidth k. Then, Algorithm* $\texttt{PCNT}_{\mathbb{SAT}}$ *runs in time $2^{2^{\Theta(k)}} \cdot \gamma(\|F\|) \cdot \|F\|$.*

6 Correctness of the Algorithm

In the following, we state definitions required for the correctness proofs of our algorithm $\mathbb{PROJ}$. In the end, we only store rows that are restricted to the bag content to maintain runtime bounds. Similar to related work [18, 41], we proceed in two steps. First, we define properties of so-called $\mathbb{PROJ}$*-solutions up to t*, and then restrict these to $\mathbb{PROJ}$*-row solutions* at t.

For the following statements, we assume that we have given an arbitrary instance (F, P) of PMC and a tree decomposition $\mathcal{T} = (T, \chi)$ of formula F, where $T = (N, A, n)$, node $n \in N$ is the root and $\mathcal{T}$ is of width k. Moreover, for every $t \in N$ of tree decomposition $\mathcal{T}$, we let $\mathbb{SAT}$-Comp$[t]$ be the tables that have been computed by running algorithm $\texttt{DP}_{\mathbb{SAT}}$ for the dedicated input. Analogously, let $\mathbb{PROJ}$-Comp$[t]$ be the tables computed by running $\texttt{DP}_{\mathbb{PROJ}}$ for the input.

Definition 11. *Let $\emptyset \subsetneq \sigma \subseteq \mathbb{SAT}$-Comp[t] be a table with $\sigma \in$ sub-buckets$_P(\mathbb{SAT}$-Comp[t]). We define a $\mathbb{PROJ}$-solution* up to t *to be the sequence $\langle\hat{\sigma}\rangle = \langle \text{SatExt}_{\leq t}(\sigma)\rangle$.*

Next, we recall that we can reconstruct all models from the tables.

Proposition 1. $I(\text{SatExt}_{\leq n}(\mathbb{SAT}\text{-}Comp[n])) = I(\text{Exts}) = \{J \in 2^{\text{var}(F)} | J \vDash F\}$.

Proof (Idea). We use a construction similar to Samer and Szeider [41] and Pichler, Rümmele, and Woltran [36, Fig. 1], where we simply collect preceding rows. □

Before we present equivalence results between $\text{ipmc}_{\leq t}(\ldots)$ and the recursive version $\text{ipmc}(t, \ldots)$ (Definition 10) used during the computation of $\texttt{DP}_{\mathbb{PROJ}}$, recall that $\text{ipmc}_{\leq t}$ and $\text{pmc}_{\leq t}$ (Definition 5) are key to compute the projected model count. The following corollary states that computing $\text{ipmc}_{\leq n}$ at the root n actually suffices to compute the projected model count $\text{pmc}_{\leq n}$ of the formula.

Corollary 3. $\text{ipmc}_{\leq n}(\mathbb{SAT}\text{-}Comp[n]) = \text{pmc}_{\leq n}(\mathbb{SAT}\text{-}Comp[n]) =$
$|I_P(\text{SatExt}_{\leq n}(\mathbb{SAT}\text{-}Comp[n]))| = |I_P(\text{Exts})| = |\{J \cap P \mid J \in 2^{\text{var}(F)}, J \vDash F\}|$

Proof. The corollary immediately follows from Proposition 1 and the observation that $|\mathbb{SAT}\text{-Comp}[n]| \leq 1$ by properties of algorithm $\mathbb{SAT}$ and since $\chi(n) = \emptyset$. □

The following lemma establishes that the $\mathbb{PROJ}$-solutions up to root n of a given tree decomposition solve the PMC problem.

Lemma 1. *The value $c = \sum_{\langle\hat{\sigma}\rangle \text{ is a } \mathbb{PROJ}\text{-solution up to } n} |I_P(\hat{\sigma})|$ if and only if c is the projected model count of F with respect to the set P of projection variables.*

Proof. Assume that $c = \sum_{\langle\hat{\sigma}\rangle \text{ is a } \mathbb{PROJ}\text{-solution up to } n} |I_P(\hat{\sigma})|$. Observe that there can be at most one projected solution up to n, since $\chi(n) = \emptyset$. If $c = 0$, then $\mathbb{SAT}$-Comp$[n]$ contains no rows. Hence, F has no models, c.f., Proposition 1, and obviously also no models projected to P. Consequently, c is the projected model count of F. If $c > 0$ we have by Corollary 3 that c is equivalent to the projected model count of F with respect to P. We proceed similar in the if direction. □

In the following, we provide for a given node t and a given $\mathbb{PROJ}$-solution up to t, the definition of a $\mathbb{PROJ}$-row solution at t.

Definition 12. *Let $t, t' \in N$ be nodes of a given tree decomposition $\mathcal{T}$, and $\hat{\sigma}$ be a $\mathbb{PROJ}$-solution up to t. Then, we define* the local table for t' *as* $\text{local}(t', \hat{\sigma}) := \{\langle \boldsymbol{u}\rangle | \langle t', \boldsymbol{u}\rangle \in \hat{\sigma}\}$*, and if $t = t'$, the* $\mathbb{PROJ}$-row solution at t *by* $\langle \text{local}(t, \hat{\sigma}), |I_P(\hat{\sigma})|\rangle$.

Observation 3. *Let $\langle\hat{\sigma}\rangle$ be a $\mathbb{PROJ}$-solution up to a node $t \in N$. There is exactly one corresponding $\mathbb{PROJ}$-row solution $\langle \text{local}(t, \hat{\sigma}), |I_P(\hat{\sigma})|\rangle$ at t.*

Vice versa, let $\langle\sigma, c\rangle$ be a $\mathbb{PROJ}$-row solution at t for some integer c. Then, there is exactly one corresponding $\mathbb{PROJ}$-solution $\langle \text{SatExt}_{\leq t}(\sigma)\rangle$ up to t.

We need to ensure that storing $\mathbb{PROJ}$-row solutions at a node suffices to solve the PMC problem, which is necessary to obtain runtime bounds (c.f. Corollary 1).

Lemma 2. *Let $t \in N$ be a node of the tree decomposition $\mathcal{T}$. There is a $\mathbb{PROJ}$-row solution at root n if and only if the projected model count of F is larger than 0.*

Proof. ("$\Longrightarrow$"): Let $\langle \sigma, c \rangle$ be a $\mathbb{PROJ}$-row solution at root n where σ is a $\mathbb{SAT}$-table and c is a positive integer. Then, by Definition 12, there also exists a corresponding $\mathbb{PROJ}$-solution $\langle \hat{\sigma} \rangle$ up to n such that $\sigma = \text{local}(n, \hat{\sigma})$ and $c = |I_P(\hat{\sigma})|$. Moreover, since $\chi(n) = \emptyset$, we have $|\mathbb{SAT}\text{-Comp}[n]| = 1$. Then, by Definition 11, $\hat{\sigma} = \mathbb{SAT}\text{-Comp}[n]$. By Corollary 3, we have $c = |I_P(\mathbb{SAT}\text{-Comp}[n])|$. Finally, the claim follows. ("$\Longleftarrow$"): Similar to the only-if direction. □

Observation 4. *Let $X_1, \ldots, X_n$ be finite sets. The number $|\bigcap_{i \in X} X_i|$ is given by $|\bigcap_{i \in X} X_i| = \big||\bigcup_{j=1}^{n} X_j| + \sum_{\emptyset \subsetneq I \subsetneq X} (-1)^{|I|} |\bigcap_{i \in I} X_i|\big|$.*

Lemma 3. *Let $t \in N$ be a node of the tree decomposition $\mathcal{T}$ with* $\text{children}(t, T) = \langle t_1, \ldots, t_\ell \rangle$ *and let $\langle \sigma, \cdot \rangle$ be a $\mathbb{PROJ}$-row solution at t. Then,*

1. $\text{ipmc}(t, \sigma, \langle \mathbb{PROJ}\text{-}\mathit{Comp}[t_1], \ldots, \mathbb{PROJ}\text{-}\mathit{Comp}[t_\ell] \rangle) = \text{ipmc}_{\leq t}(\sigma)$
2. *If* $\text{type}(t) \neq$ *leaf:* $\text{pmc}(t, \sigma, \langle \mathbb{PROJ}\text{-}\mathit{Comp}[t_1], \ldots, \mathbb{PROJ}\text{-}\mathit{Comp}[t_\ell] \rangle) = \text{pmc}_{\leq t}(\sigma)$.

Proof (Sketch). We prove the statement by simultaneous induction. ("Induction Hypothesis"): Lemma 3 holds for the nodes in $\text{children}(t, T)$ and also for node t, but on strict subsets $\rho \subsetneq \sigma$. ("Base Cases"): Let $\text{type}(t) =$ *leaf*. By definition, $\text{ipmc}(t, \emptyset, \langle\rangle) = \text{ipmc}_{\leq t}(\emptyset) = 1$. Recall that for pmc the equivalence does not hold for leaves, but we use a node t that has a node $t' \in N$ with $\text{type}(t') =$ *leaf* as child for the base case. Observe that by definition t has exactly one child. Then, we have $\text{pmc}(t, \sigma, \langle \mathbb{PROJ}\text{-Comp}[t'] \rangle) = \sum_{\emptyset \subsetneq O \subseteq \mathbb{SAT}\text{-origins}(t,\sigma)} (-1)^{(|O|-1)}$. $\text{s-ipmc}(\langle \mathbb{SAT}\text{-Comp}[t'] \rangle, O) = |\bigcup_{\boldsymbol{u} \in \sigma} I_P(\text{SatExt}_{\leq t}(\{\boldsymbol{u}\}))| = \text{pmc}_{\leq t}(\sigma) = 1$ for $\mathbb{PROJ}$-row solution $\langle \sigma, \cdot \rangle$ at t. ("Induction Step"): We proceed by case distinction. Assume that $\text{type}(t) =$ *int*. Let $a \in (\chi(t) \setminus \chi(t'))$ be the introduced variable. We have two cases. Assume Case (i): a also belongs to $(\text{var}(F) \setminus P)$, i.e., a is not a projection variable. Let $\langle \sigma, c \rangle$ be a $\mathbb{PROJ}$-row solution at t for some integer c. By construction of algorithm $\mathbb{SAT}$ there are many rows in the table $\mathbb{SAT}\text{-Comp}[t]$ for one row in the table $\mathbb{SAT}\text{-Comp}[t']$, more precisely, $|\,\text{buckets}_P(\sigma)| = 1$. As a result, $\text{pmc}_{\leq t}(\sigma) = \text{pmc}_{\leq t'}(\mathbb{SAT}\text{-origins}(t, \sigma))$ by applying Observation 3. We apply the inclusion-exclusion principle on every subset ρ of the origins of σ in the definition of pmc and by induction hypothesis we know that $\text{ipmc}(t', \rho, \langle \mathbb{PROJ}\text{-Comp}[t'] \rangle) = \text{ipmc}_{\leq t'}(\rho)$, therefore, $\text{s-ipmc}(\mathbb{PROJ}\text{-Comp}[t'], \rho) = \text{ipmc}_{\leq t'}(\rho)$. This concludes Case (i) for pmc. The induction step for ipmc works similar by applying Observation 4 and comparing corresponding $\mathbb{PROJ}$-solutions up to t or t', respectively. Further, for showing the lemma for ipmc, one has to additionally apply the hypothesis for node t, but on strict subsets $\emptyset \subsetneq \rho \subsetneq \sigma$ of σ. Assume that we have Case (ii): a also belongs to P, i.e., a is a projection variable. This is a special case of Case (i) since $|\,\text{buckets}_P(\sigma)| = 1$. Similarly, for join and remove nodes. □

Lemma 4 (Soundness). *Let $t \in N$ be a node of the tree decomposition $\mathcal{T}$ with* $\text{children}(t, T) = \langle t_1, \ldots, t_\ell \rangle$. *Then, each row $\langle \tau, c \rangle$ at node t obtained by $\mathbb{PROJ}$ is a $\mathbb{PROJ}$-row solution for t.*

Proof (Idea). Observe that Listing 3 computes a row for each sub-bucket $\sigma \in \text{sub-buckets}_P(\mathbb{SAT}\text{-Comp}[t])$. The resulting row $\langle \sigma, c \rangle$ obtained by ipmc is indeed a $\mathbb{PROJ}$-row solution for t according to Lemma 3. □

Lemma 5 (Completeness). *Let $t \in N$ be a node of tree decomposition $\mathcal{T}$ where* $\text{children}(t, T) = \langle t_1, \ldots, t_\ell \rangle$ *and* $\text{type}(t) \neq$ *leaf. Given a $\mathbb{PROJ}$-row solution $\langle \sigma, c \rangle$ at t. Then, there is $\langle C_1, \ldots, C_\ell \rangle$ where each C_i is a set of $\mathbb{PROJ}$-row solutions at t_i with $\sigma = \mathbb{PROJ}(t, \cdot, \cdot, P, \langle C_1, \ldots, C_\ell \rangle, \mathbb{SAT}\text{-}Comp)$.*

Proof (Idea). Since $\langle \sigma, c \rangle$ is a $\mathbb{PROJ}$-row solution for t, there is by Definition 12 a corresponding $\mathbb{PROJ}$-solution $\langle \hat{\sigma} \rangle$ up to t such that $\text{local}(t, \hat{\sigma}) = \sigma$. We proceed again by case distinction. Assume $\text{type}(t) = \mathit{int}$ and $t' = t_1$. Then we define $\hat{\sigma'} := \{(t', \hat{\rho}) \mid (t', \hat{\rho}) \in \sigma, t \neq t'\}$. Then, for each subset $\emptyset \subsetneq \rho \subseteq \text{local}(t', \hat{\sigma'})$, we define $\langle \rho, |I_P(\text{SatExt}_{\leq t}(\rho))| \rangle$ in accordance with Definition 12. By Observation 3, we have that $\langle \rho, |I_P(\text{SatExt}_{\leq t}(\rho))| \rangle$ is a $\mathbb{SAT}$-row solution at t'. Since we defined $\mathbb{PROJ}$-row solutions for t' for all respective $\mathbb{PROJ}$-solutions up to t', we encountered every $\mathbb{PROJ}$-row solution for t' required for deriving $\langle \sigma, c \rangle$ via $\mathbb{PROJ}$ (c.f. Definitions 10 and 9). Similarly, for remove and join nodes. □

Theorem 3. *The algorithm $\texttt{DP}_{\mathbb{PROJ}}$ is correct. More precisely, $\texttt{DP}_{\mathbb{PROJ}}((F, P), \mathcal{T}, \mathbb{SAT}\text{-}Comp)$ returns tables $\mathbb{PROJ}$-Comp such that $c = \text{s-ipmc}(\mathbb{SAT}\text{-}Comp[n], \cdot)$ is the projected model count of F with respect to the set P of projection variables.*

Proof. By Lemma 4 we have soundness for every node $t \in N$ and hence only valid rows as output of table algorithm $\mathbb{PROJ}$ when traversing the tree decomposition in post-order up to the root n. By Lemma 2 we know that the projected model count c of F is larger than zero if and only if there exists a certain $\mathbb{PROJ}$-row solution for n. This $\mathbb{PROJ}$-row solution at node n is of the form $\langle \{\langle \emptyset, \ldots \rangle\}, c \rangle$. If there is no $\mathbb{PROJ}$-row solution at node n, then $\mathbb{SAT}\text{-Comp}[n] = \emptyset$ since the table algorithm $\mathbb{SAT}$ is correct (c.f. Proposition 1). Consequently, we have $c = 0$. Therefore, $c = \text{s-ipmc}(\mathbb{SAT}\text{-Comp}[n], \cdot)$ is the pmc of F w.r.t. P in both cases.

Next, we establish completeness by induction starting from root n. Let therefore, $\langle \hat{\sigma} \rangle$ be the $\mathbb{PROJ}$-solution up to n, where for each row in $\boldsymbol{u} \in \hat{\sigma}$, $I(\boldsymbol{u})$ corresponds to a model of F. By Definition 12, we know that for n we can construct a $\mathbb{PROJ}$-row solution at n of the form $\langle \{\langle \emptyset, \ldots \rangle\}, c \rangle$ for $\hat{\sigma}$. We already established the induction step in Lemma 5. Finally, we stop at the leaves. □

Corollary 4. *The algorithm $\texttt{PCNT}_{\mathbb{SAT}}$ is correct, i.e., $\texttt{PCNT}_{\mathbb{SAT}}$ solves* PMC.

Proof. The result follows, since $\texttt{PCNT}_{\mathbb{SAT}}$ consists of pass $\texttt{DP}_{\mathbb{SAT}}$, a purging step and $\texttt{DP}_{\mathbb{PROJ}}$. For correctness of $\texttt{DP}_{\mathbb{SAT}}$ we refer to other sources [18, 41]. By Proposition 1, "purging" neither destroys soundness nor completeness of $\texttt{DP}_{\mathbb{SAT}}$. □

7 Conclusions

We introduced a dynamic programming algorithm to solve projected model counting (PMC) by exploiting the structural parameter treewidth. Our algorithm is asymptotically optimal under the exponential time hypothesis (ETH). Its runtime is double exponential in the treewidth of the primal graph of the instance and polynomial in the size of the input instance. We believe that our results can also be extended to another graph representation, namely the incidence graph. Our approach is very general and might be applicable to a wide range of other hard combinatorial problems, such as projection for ASP [18] and QBF [10].

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases: The Logical Level. Addison-Wesley, Boston, MA, USA, 1st edn. (1995)
2. Abramson, B., Brown, J., Edwards, W., Murphy, A., Winkler, R.L.: Hailfinder: A Bayesian system for forecasting severe weather. International Journal of Forecasting 12(1), 57–71 (1996)
3. Aziz, R.A., Chu, G., Muise, C., Stuckey, P.: #(∃)SAT: Projected Model Counting. In: Heule, M., Weaver, S. (eds.) Proceedings of the 18th International Conference on Theory and Applications of Satisfiability Testing (SAT'15). pp. 121–137. Springer Verlag, Austin, TX, USA (Sep 2015)
4. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press, Amsterdam, Netherlands (Feb 2009)
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., Kloks, T.: Efficient and constructive algorithms for the pathwidth and treewidth of graphs. J. Algorithms 21(2), 358–402 (1996)
7. Bodlaender, H.L., Koster, A.M.C.A.: Combinatorial optimization on graphs of bounded treewidth. The Computer Journal 51(3), 255–269 (2008)
8. Bondy, J.A., Murty, U.S.R.: Graph theory, Graduate Texts in Mathematics, vol. 244. Springer Verlag, New York, USA (2008)
9. Chakraborty, S., Meel, K.S., Vardi, M.Y.: Improving approximate counting for probabilistic inference: From linear to logarithmic SAT solver calls. In: Kambhampati, S. (ed.) Proceedings of 25th International Joint Conference on Artificial Intelligence (IJCAI'16). pp. 3569–3576. The AAAI Press, New York City, NY, USA (Jul 2016)
10. Charwat, G., Woltran, S.: Dynamic programming-based QBF solving. In: Lonsing, F., Seidl, M. (eds.) Proceedings of the 4th International Workshop on Quantified Boolean Formulas (QBF'16). vol. 1719, pp. 27–40. CEUR Workshop Proceedings (CEUR-WS.org) (2016), co-located with 19th International Conference on Theory and Applications of Satisfiability Testing (SAT'16)
11. Choi, A., Van den Broeck, G., Darwiche, A.: Tractable learning for structured probability spaces: A case study in learning preference distributions. In: Yang, Q. (ed.) Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI'15). The AAAI Press (2015)
12. Cygan, M., Fomin, F.V., Kowalik, Ł., Lokshtanov, D., Dániel Marx, M.P., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer Verlag (2015)
13. Diestel, R.: Graph Theory, 4th Edition, Graduate Texts in Mathematics, vol. 173. Springer Verlag (2012)
14. Domshlak, C., Hoffmann, J.: Probabilistic planning via heuristic forward search and weighted model counting. J. Artif. Intell. Res. 30, 565–620 (2007)
15. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science, Springer Verlag, London, UK (2013)
16. Dueñas-Osorio, L., Meel, K.S., Paredes, R., Vardi, M.Y.: Counting-based reliability estimation for power-transmission grids. In: Singh, S.P., Markovitch, S. (eds.) Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI'17). pp. 4488–4494. The AAAI Press, San Francisco, CA, USA (Feb 2017)
17. Durand, A., Hermann, M., Kolaitis, P.G.: Subtractive reductions and complete problems for counting complexity classes. Theoretical Computer Science 340(3), 496–513 (2005)

18. Fichte, J.K., Hecher, M., Morak, M., Woltran, S.: Answer set solving with bounded treewidth revisited. In: Balduccini, M., Janhunen, T. (eds.) Proceedings of the 14th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'17). Lecture Notes in Computer Science, vol. 10377, pp. 132–145. Springer Verlag, Espoo, Finland (Jul 2017)
19. Fichte, J.K., Hecher, M., Morak, M., Woltran, S.: DynASP2.5: Dynamic programming on tree decompositions in action. In: Lokshtanov, D., Nishimura, N. (eds.) Proceedings of the 12th International Symposium on Parameterized and Exact Computation (IPEC'17). Dagstuhl Publishing (2017)
20. Flum, J., Grohe, M.: Parameterized Complexity Theory, Theoretical Computer Science, vol. XIV. Springer Verlag, Berlin (2006)
21. Gebser, M., Schaub, T., Thiele, S., Veber, P.: Detecting inconsistencies in large biological networks with answer set programming. Theory Pract. Log. Program. 11(2-3), 323–360 (2011)
22. Gebser, M., Kaufmann, B., Schaub, T.: Solution enumeration for projected boolean search problems. In: van Hoeve, W.J., Hooker, J.N. (eds.) Proceedings of the 6th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'09). Lecture Notes in Computer Science, vol. 5547, pp. 71–86. Springer Verlag, Berlin (2009)
23. Ginsberg, M.L., Parkes, A.J., Roy, A.: Supermodels and robustness. In: Rich, C., Mostow, J. (eds.) Proceedings of the 15th National Conference on Artificial Intelligence and 10th Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI'98). pp. 334–339. The AAAI Press, Madison, Wisconsin, USA (Jul 1998)
24. Gomes, C.P., Sabharwal, A., Selman, B.: Chapter 20: Model counting. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, pp. 633–654. IOS Press, Amsterdam, Netherlands (Feb 2009)
25. Graham, R.L., Grötschel, M., Lovász, L.: Handbook of Combinatorics, vol. I. Elsevier Science Publishers, North-Holland (1995)
26. Harvey, D., van der Hoeven, J., Lecerf, G.: Even faster integer multiplication. J. Complexity 36, 1–30 (2016)
27. Hemaspaandra, L.A., Vollmer, H.: The satanic notations: Counting classes beyond #P and other definitional adventures. SIGACT News 26(1), 2–13 (Mar 1995)
28. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. of Computer and System Sciences 63(4), 512–530 (2001)
29. Kleine Büning, H., Lettman, T.: Propositional logic: deduction and algorithms. Cambridge University Press, Cambridge, New York, NY, USA (1999)
30. Knuth, D.E.: How fast can we multiply? In: The Art of Computer Programming, Seminumerical Algorithms, vol. 2, chap. 4.3.3, pp. 294–318. Addison-Wesley, 3 edn. (1998)
31. Lagniez, J.M., Marquis, P.: An improved decision-DNNF compiler. In: Sierra, C. (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI'17). The AAAI Press (2017)
32. Lampis, M., Mitsou, V.: Treewidth with a quantifier alternation revisited. In: Lokshtanov, D., Nishimura, N. (eds.) Proceedings of the 12th International Symposium on Parameterized and Exact Computation (IPEC'17). Dagstuhl Publishing (2017)
33. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)

34. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms, Oxford Lecture Series in Mathematics and its Applications, vol. 31. Oxford University Press, New York, NY, USA (2006)
35. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1994)
36. Pichler, R., Rümmele, S., Woltran, S.: Counting and enumeration problems with bounded treewidth. In: Clarke, E.M., Voronkov, A. (eds.) Proceedings of the 16th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'10). Lecture Notes in Computer Science, vol. 6355, pp. 387–404. Springer Verlag (2010)
37. Pourret, O., Naim, P., Bruce, M.: Bayesian Networks - A Practical Guide to Applications. John Wiley & Sons (2008)
38. Roth, D.: On the hardness of approximate reasoning. Artificial Intelligence 82(1–2) (1996)
39. Sæther, S.H., Telle, J.A., Vatshelle, M.: Solving #SAT and MAXSAT by dynamic programming. J. Artif. Intell. Res. 54, 59–82 (2015)
40. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A Bayesian approach to filtering junk e-mail. In: Joachims, T. (ed.) Proceedings of the AAAI-98 Workshop on Learning for Text Categorization. vol. 62, pp. 98–105 (1998)
41. Samer, M., Szeider, S.: Algorithms for propositional model counting. J. Discrete Algorithms 8(1), 50—64 (2010)
42. Sang, T., Beame, P., Kautz, H.: Performing Bayesian inference by weighted model counting. In: Veloso, M.M., Kambhampati, S. (eds.) Proceedings of the 29th National Conference on Artificial Intelligence (AAAI'05). The AAAI Press (2005)
43. Valiant, L.: The complexity of enumeration and reliability problems. SIAM J. Comput. 8(3), 410–421 (1979)
44. Wilder, R.L.: Introduction to the Foundations of Mathematics. John Wiley & Sons, 2nd edition edn. (1965)
45. Xue, Y., Choi, A., Darwiche, A.: Basing decisions on sentences in decision diagrams. In: Hoffmann, J., Selman, B. (eds.) Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI'12). The AAAI Press, Toronto, ON, Canada (2012)