On Weighted Maximum Model Counting: Complexity and Fragments

Max Bannach
Advanced Concepts Team
European Space Agency
Noordwijk, The Netherlands
https://orcid.org/0000-0002-6475-5512

Markus Hecher
Computer Science and Artificial Intelligence Lab
Massachusetts Institute of Technology
Cambridge, USA
https://orcid.org/0000-0003-0131-6771

***Abstract*—Maximum model counting (MAX#∃SAT) is a recently introduced extension of projected model counting (#∃SAT) that maximizes over a set of variables X the number of assignments over a set Y that can be extended to a satisfying assignment over Z. It is known that MAX#∃SAT also generalizes weighted #∃SAT and MAXSAT if weights are introduced to the problem. However, for the latter a non-trivial gadget is needed. We propose a more generic weighting scheme that evaluates a *fitness term* and a *probability term* simultaneously. In this setting, MAX#∃SAT extends weighted MAXSAT and #∃SAT without the need of gadgets. As MAXSAT is the canonical problem of *cost-optimal reasoning* and #∃SAT can be seen as canonical problem of *probabilistic reasoning*, MAX#∃SAT with the proposed weighting scheme naturally fills the role as canonical problem for *cost-optimal probabilistic reasoning*.**

We study the problem from a complexity-theoretic point of view for unary weights and prove that the decision version is $\mathrm{D}_2^{\mathrm{P}}$-complete. We then focus on structural parameters and provide an ETH lower bound with respect to the inputs treewidth, as well as a treewidth-aware reduction from MAX#∃SAT to MAX#SAT.

***Index Terms*—max-sat, sharp-sat, treewidth**

I. Introduction

We consider *weighted* propositional formulas in *conjunctive normal form* (called WCNFs) like

$$\varphi = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_3 \vee x_4) \wedge (\neg x_3 \vee \neg x_4)$$

as *set of sets of literals*

$$\big\{\{x_1, x_2\}, \{\neg x_1, \neg x_2\}, \{x_3, x_4\}, \{\neg x_3, \neg x_4\}\big\}$$

with weights on the *literals* $w\colon \mathrm{lits}(\varphi) \to \mathbb{N}$.[1] Let us denote for a set of variables X with $\beta \sqsubseteq X$ an *assignment* over X, i.e., a set with $|\beta| = |X|$ and $|\beta \cap \{x, \neg x\}| = 1$ for all $x \in X$. The *satisfiability* problem (SAT) is the fundamental problem of many reasoning tasks that asks whether there is an assignment over the variables of φ that satisfies all clauses, i.e., whether there is a $\beta \sqsubseteq \mathrm{vars}(\varphi)$ with $\beta \cap c \neq \emptyset$ for every $c \in \varphi$.

Many reasoning tasks are build on qualitative extensions of the satisfiability problem. Prominently, in *cost-optimal* reasoning we do not just seek a model of φ, but we are looking for a *good* one. In the *weighted partial maximum satisfiability* problem (MAXSAT) we search the solution for:[2]

$$\max(\varphi) \coloneqq \max_{\substack{\beta \sqsubseteq \mathrm{vars}(\varphi) \\ \beta \models \varphi}} \sum_{\ell \in \beta} w(\ell).$$

A broad range of discrete optimization problems can be encoded into MAXSAT with industrial applications [25] like timetabling [23], planning [8], software analysis [27], and data analysis [4]. For a running example in this article, we consider the *minimum cut problem,* see Figure 1.

Another common extension of SAT is the *weighted model counting* problem, which we call simply #SAT as we consider only weighted formulas throughout this paper. The goal of this problem is to compute the following value:

$$\#(\varphi) \coloneqq \sum_{\substack{\beta \sqsubseteq \mathrm{vars}(\varphi) \\ \beta \models \varphi}} \prod_{\ell \in \beta} w(\ell).$$

The value $\#(\varphi)$ corresponds to the *probability* of satisfying φ with a random assignment that picks literals with probabilities according to their weight. Therefore, #SAT is the canonical problem of *probabilistic reasoning* with applications in explainable artificial intelligence [24], [32], reliability [10], and verification [12]. In industrial applications, it is convenient to work with the *projected* version #∃SAT, in which the input $\varphi(X, Y)$ is defined over two sets of variables. The task is to compute the following, where $\beta \models^* \varphi$ denotes that the partial assignment β can be extended to a satisfying assignment:

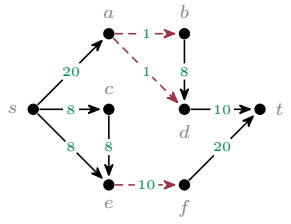
$$\#(\varphi, X) \coloneqq \sum_{\substack{\beta \sqsubseteq X \\ \beta \models^* \varphi}} \prod_{\ell \in \beta} w(\ell).$$

Figure 2 contains an illustrating example of this definition. Both, MAXSAT and #SAT, succinctly describe semiring operations over exponentially many items, namely a *max-of-sums* or a *sum-of-products,* respectively. Fremont et al. [16] proposed a generalization of both problems as *max-of-(sum-of-products)*:

This research was funded by the Austrian Science Fund (FWF), grants J 4656 and P 32830, the Society for Research Funding in Lower Austria (GFF, Gesellschaft für Forschungsförderung NÖ) grant ExzF-0004, as well as the Vienna Science and Technology Fund (WWTF) grant ICT19-065.

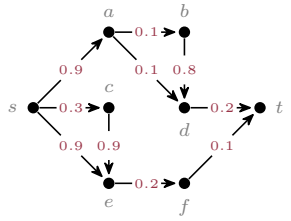
[1] Note that $\mathbb{N}$ is enough the express *bounded-precision* floating-point numbers, as we can multiple these simply with a large enough number.

[2] One can also define the problem with weights on the clauses and search for an assignment that maximizes the sum of the weights of the satisfied clauses. It is well-known that both definitions are equivalent.



$$\begin{aligned}&\text{min-cut}(G,s,t) = -\max\big((r_s) \wedge (\neg r_t)\\ &\wedge (r_s \wedge \neg d_{sa} \to r_a) \wedge (r_s \wedge \neg d_{sc} \to r_c)\\ &\wedge (r_s \wedge \neg d_{se} \to r_e) \wedge (r_a \wedge \neg d_{ab} \to r_b) \wedge (r_a \wedge \neg d_{ad} \to r_d)\\ &\wedge (r_b \wedge \neg d_{bd} \to r_d) \wedge (r_d \wedge \neg d_{dt} \to r_t) \wedge (r_c \wedge \neg d_{ce} \to r_e)\\ &\wedge (r_e \wedge \neg d_{ef} \to r_f) \wedge (r_f \wedge \neg d_{ft} \to r_t)\big) = 12\end{aligned}$$

Fig. 1. Given a directed graph G with weights on the edges and two vertices $s, t \in V(G)$. We seek the set $S \subseteq E(G)$ that minimizes $\sum_{e \in S} \text{cost}(e)$ such that $G \setminus S$ does not contain an s-t-path. The problem can elegantly be encoded into MAXSAT by introducing for every $v \in V(G)$ a variable r_v (for v is **r**eachable) with $w(r_v) = w(\neg r_v) = 0$, and for every arc $uv \in E(G)$ a variable d_{uv} (for **d**elete uv) with $w(d_{uv}) = -\text{cost}(uv)$ and $w(\neg d_{uv}) = 0$. Then $-\max\big((r_s) \wedge (\neg r_t) \wedge \bigwedge_{v \in V(G)} \bigwedge_{vw \in E(G)} (r_v \wedge \neg d_{vw} \to r_w)\big)$ is precisely the value of the minimum s-t-cut. The figure shows an instance with an optimal solution of size 12 (highlighted in red) and the instantiation of the formula.



$$\begin{aligned}&\Pr[\text{there is no } s\text{-}t\text{-path}] = \#\Big(\\ &\qquad (r_s) \wedge (\neg r_t) \wedge \bigwedge_{v \in V(G)} \bigwedge_{vw \in E(G)} (r_v \wedge \neg p_{vw} \to r_w),\\ &\qquad \{p_{vw} \mid vw \in E(G)\}\\ &\Big) = 0.622.\end{aligned}$$

Fig. 2. A #∃SAT task related to the minimum cut problem that occurs in reliability applications is to compute the *probability* that a network, in which every edge $vw \in E(G)$ may randomly fail (i.e., disappear), contains no s-t-path. We can use almost the same encoding with $w(p_{uv}) = \Pr[uv \text{ fails}]$ and $w(\neg p_{uv}) = 1 - \Pr[uv \text{ fails}]$ (weights of r_v are not relevant due to the projection). The figure shows the network with probabilities assigned to edges.

Definition 1 (The Maximum Model Counting Problem)**.** *Let $\varphi(X, Y, Z)$ be a formula over three distinct sets of variables. The* maximum model counting *problem asks to find the partial assignment $\beta \sqsubseteq X$ that maximizes the projected model count $\#(\varphi|\beta, Y)$, were $\varphi|\beta$ is the formula* conditioned *under β in which we delete all $c \in \varphi$ with $c \cap \beta \neq \emptyset$ and remove all literals ℓ with $\neg\ell \in \beta$ from the remaining clauses.*

Fremont et al. considered the problem without weights and provided a (relatively easy but non-trivial) reduction from MAXSAT (*with* weights) to the maximum model counting problem (*without* weights) [16]. Audemard et al. observed that the definition extends to a version in which the inner sum-of-products uses weights [3] and, indeed, Definition 1, as stated above, allows weights by our definition of $\#(\varphi, X)$. Maximum model counting, thus, generalizes the canonical problems of cost-optimal and probabilistic reasoning. Implementations of restricted fragments thereof are available [21], [22]. However, we are convinced that Definition 1 lacks two properties to be a canonical problem for *cost-optimal probabilistic reasoning*:

1) while MAX#∃SAT trivially generalizes #∃SAT (just set $X = \emptyset$)[3], a non-trivial gadget-construction is needed to encode and solve MAXSAT;
2) it is *not* possible to optimize costs and probabilities simultaneously.

We propose a slightly more generic weighting scheme that takes a weight over X into account (called the *fitness term*). In distinction, we call the inner model count the *probability term*, as probabilities are the most common use case.[4] Technically, we argue for a *max-of-(sum-plus-(sum-of-products)):*

$$\max\#\exists(\varphi, X, Y) = \max_{\substack{\alpha \sqsubseteq X\\ \alpha \models^* \varphi}} \Big(\underbrace{\sum_{\ell \in \alpha} w(\ell)}_{\textit{fitness term}} + \underbrace{\sum_{\substack{\beta \sqsubseteq Y\\ \alpha \cup \beta \models^* \varphi}} \prod_{\ell \in \beta} w(\ell)}_{\textit{probability term}} \Big).$$

We denote with MAX#∃SAT the problem of computing the value $\max\#\exists(\varphi) := \max\#\exists(\varphi, X, Y)$ of a weighted formula $\varphi(X, Y, Z)$. There are some fine technicalities of this definition worth mentioning. First observe that the definition interleaves two semiring operations, namely a *max-of-sums* and a *sum-of-products*. The neutral elements of these semirings are $-\infty$ and 0, i.e., $\max_\emptyset = -\infty$ and $\sum_\emptyset = 0$. The second detail is that the empty set has exactly one subset, namely $\emptyset$. Hence, the outer maximum produces $-\infty$ if, and only if, the formula is *unsatisfiable* (in this case, there is no $\alpha \sqsubseteq X$ with $\alpha \models^* \varphi$). In contrast, if $X = \emptyset$, there is an $\alpha \sqsubseteq \emptyset$ that can be extended to a model of φ (namely $\alpha = \emptyset$) and, hence, the expression simplifies to:

$$\max\#\exists(\varphi, \emptyset, Y) = \begin{cases} -\infty & \text{if } \varphi \text{ is unsatisfiable;}\\ \sum_{\substack{\beta \sqsubseteq Y\\ \beta \models^* \varphi}} \prod_{\ell \in \beta} w(\ell) & \text{else.}\end{cases}$$

Similarly, the product over the empty set is defined to be 1, i.e., $\prod_\emptyset = 1$ and, thus, we obtain:

$$\max\#\exists(\varphi, X, \emptyset) = \begin{cases} -\infty & \text{if } \varphi \text{ is unsatisfiable;}\\ \max\limits_{\substack{\alpha \sqsubseteq X\\ \alpha \models^* \varphi}} \sum_{\ell \in \alpha} w(\ell) \; + 1 & \text{else;}\end{cases}$$

The table in Figure 3 contains an overview of all possible special cases. The case of $Z = \emptyset$ does not change these simplifications but removes the projection. We, thus, call this

[3]There is exactly one assignment over the empty set, which is empty.

[4]The definition, however, is general and allows non-stochastic applications.

Fig. 3. Special cases of the maximum model counting problem, that is, simplifications of $\max\#\exists(\varphi, X, Y)$ if $X = \emptyset$ or $Y = \emptyset$. The table contains a "✓" in the X or Y column if the corresponding set of variables is empty in $\varphi(X, Y, Z)$. The columns SAT and UNSAT contain the corresponding simplification of the expression $\max\#\exists(\varphi)$ in dependence on the satisfiability of φ. Whether or not Z is empty has no effect.

X	Y	SAT	UNSAT
		$\max\limits_{\substack{\alpha \sqsubseteq X \\ \alpha \models^* \varphi}} \left(\sum\limits_{\ell \in \alpha} w(\ell) + \sum\limits_{\substack{\beta \sqsubseteq Y \\ \alpha \cup \beta \models^* \varphi}} \prod\limits_{\ell \in \beta} w(\ell) \right)$	$-\infty$
✓		$\sum\limits_{\substack{\beta \sqsubseteq Y \\ \beta \models^* \varphi}} \prod\limits_{\ell \in \beta} w(\ell)$	$-\infty$
	✓	$\max\limits_{\substack{\alpha \sqsubseteq X \\ \alpha \models^* \varphi}} \sum\limits_{\ell \in \alpha} w(\ell) \quad + 1$	$-\infty$
✓	✓	1	$-\infty$

case MAX#SAT (without "∃").[5] Our definition of MAX#∃SAT resolves both of the raised issues: The first because MAX#∃SAT trivially generalizes MAXSAT (set $Y = Z = \emptyset$), #∃SAT (set $X = \emptyset$), and #SAT (set $X = Z = \emptyset$). The second because we optimize over the fitness and probability terms simultaneously. Figure 4 illustrates the definition using our running example.

Despite its elegance and broad application range, the maximum model counting problem is not well understood. Previous work on the topic, namely by Fremont et al. [16] and Audemard et al. [3], focused on applications, practical considerations, and implementations. But to the best of our knowledge, no systematic study on the theoretical foundations of the problem has been carried out so far – which is exactly what this article provides.

Contribution I: A Complexity-Theoretic Analysis of the Maximum Model Counting Problem with Unary Weights

We study the computational complexity of natural decision versions of MAX#∃SAT. First, we prove the following quadchotomy for *simple* WCNFs $\varphi(X, Y, Z)$, in which we have $w(x) = 1$ and $w(\neg x) = 0$ for all $x \in X$, and $w(y) = w(\neg y) = 1$ for $y \in Y$.

Theorem 1 (Quadchotomy Theorem)**.** *Let $\varphi(X, Y, Z)$ be a simple* WCNF*. Then deciding $\max\#\exists(\varphi) = k$ for $k \in \mathbb{N}$ given in unary is:*

- $\mathrm{D}_2^{\mathrm{P}}$*-complete;*
- $\mathrm{D}_2^{\mathrm{P}}$*-complete, if X or Y, Z are empty;*
- $\mathrm{D}_2^{\mathrm{P}} \cap C_=^{\mathrm{P}}$ *and* US*-hard, if X and Z are empty;*
- NP*-complete ($k = 1$) /* coNP*-complete ($k = -\infty$), if X and Y are empty.*

Corollary 1. *Let $\varphi(X, Y, Z)$ be a simple* WCNF*. Then, deciding whether $\max\#\exists(\varphi) = k$ for $k \in \mathbb{N}$ is* NP*-complete if X and Z are empty, or if Y is empty.*

Corollary 2. *Let $\varphi(X, Y, Z)$ be a simple* WCNF*. Then, deciding $\max\#\exists(\varphi) \leq k$ for $k \in \mathbb{N}$ is* coNP*-complete, even if X and Z are empty. If Y is empty, the problem is* coNP*-complete.*

Proof. This is the co-problem of the problem in the proof of the previous task, as we can ask $\leq k$ via asking $\geq k + 1$ and then inverting the answer. □

[5] Fremont et al. called the problem of Definition 1 MAX#SAT, but we think that this name hides the fact that a projection appears internally.

Using these computational insights, we obtain the (decision) complexity of computing MAX#∃SAT for simple WCNFs. This renders the computation of MAX#∃SAT even slightly harder.

Theorem 2. *Let $\varphi(X, Y, Z)$ and $\psi(X', Y', Z')$ be simple* WCNF*s and $k \in \mathbb{N}$ be given in unary. Then, deciding whether we have*

$$k \geq \max\#\exists(\varphi) > \max\#\exists(\psi)$$

is Θ_2^{P}-complete, even if we have $X = X' = \emptyset$, $Z = Z' = \emptyset$, $X = Z' = \emptyset$, $Z = X' = \emptyset$, or $Y = Y' = Z = Z' = \emptyset$.

Contribution II: A Complexity Trichotomy Under the Exponential Time Hypothesis (ETH) and Structure-Aware Compilations

Given the high complexity of MAX#∃SAT, we study the hardness for structural measures, focusing on the prominent parameter treewidth. Indeed, structure is crucial for practical counting algorithms [14], as a winner of model counting competitions has observed [20].

Theorem 3 (Trichotomy Theorem)**.** *Let $\varphi(X, Y, Z)$ be a* WCNF *and assume* ETH *holds. Then $\max\#\exists(\varphi)$ cannot be computed in time*

- $2^{2^{2^{o(\mathrm{tw}(\varphi))}}} \cdot \mathrm{poly}(|\varphi|)$*;*
- $2^{2^{o(\mathrm{tw}(\varphi))}} \cdot \mathrm{poly}(|\varphi|)$ *if either X or Z are empty;*
- $2^{o(\mathrm{tw}(\varphi))} \cdot \mathrm{poly}(|\varphi|)$ *if X and Z are empty or if Y is empty.*

We supplement the lower bounds established in the Trichotomy Theorem with an algorithm that compiles MAX#∃SAT into MAX#SAT. This reduction is *structure-aware* in the sense that it preserves the treewidth optimally with respect to the bounds of the previous theorem.

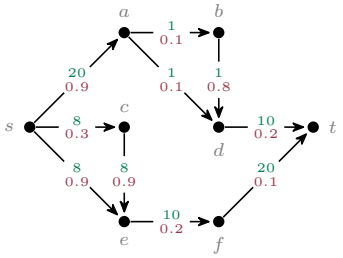
Theorem 4 (Structure-Aware Compilation)**.** *There is an algorithm that, on input of a* WCNF *$\varphi(X, Y, Z)$ and a width-k tree decomposition of φ, outputs in time $2^{2k+3} \cdot (|\operatorname{vars}(\varphi)| + |\varphi|)$ a* WCNF *$\psi(X, Y')$ and a width-2^{k+3} tree decomposition of ψ such that $\max\#\exists(\varphi) = \max\#\exists(\psi)$.*

We also provide an *upper bound* that we obtain based on an involved dynamic program over a given tree decomposition of the input's primal graph. In detail, we show:

Theorem 5. *There is an algorithm that, on input of a simple* WCNF *$\varphi(X, Y, \emptyset)$ and a width-k tree decomposition of φ, solves* MAX#SAT *in time $2^{2^{O(k)}} |\operatorname{vars}(\varphi)|$.*

Corollary 3. *There is an algorithm that, on input of a simple* WCNF *$\varphi(X, Y, Z)$ and a width-k tree decomposition of φ, computes $\max\#\exists(\varphi)$ in time $2^{2^{2^{O(k)}} |\operatorname{vars}(\varphi)|}$.*

We stress that Theorem 5 is stated for *simple* WCNFs, i.e., we assume $w(x) = 1$ and $w(\neg x) = 0$ for all $x \in X$ and $w(y) = w(\neg y) = 1$ for all $y \in Y$. While this *strengthens* the results for lower bounds, other weights are, of course, *desirable for upper bounds*. We will later discuss possibilities to generalize the dynamic program.



$$\max\#\Big((r_s) \wedge (\neg r_t) \wedge \bigwedge_{v \in V(G)} \bigwedge_{vw \in E(G)} (r_v \wedge \neg d_{uv} \wedge \neg p_{vw} \to r_w), \{d_{vw} \mid vw \in E(G)\}, \{p_{vw} \mid vw \in E(G)\} \Big).$$

Fig. 4. We compute the $S \subseteq E(G)$ that maximizes $-\sum_{e \in S} \mathrm{cost}(e) + \Pr[\text{there is no } s\text{-}t\text{-path}]$. Of course, we face a scaling issue as the probability term is in $[0,1]$ while the fitness depends on the costs. However, we can add scaling factors $\gamma_1, \gamma_2 \in \mathbb{N}$ and optimize instead the objective $-\gamma_1 \sum_{e \in S} \mathrm{cost}(e) + \gamma_2 \Pr[\text{there is no } s\text{-}t\text{-path}]$ by simply multiplying the weights of all d_{uv} by γ_1 and by adding a fresh variable x to Y with $w(x) = \gamma_2$ and $w(\neg x) = 0$.

Fig. 5. Survey of complexity results on computing $\max\#\exists(\varphi)$ for given WCNFs φ, ψ. The value $\max\#\exists(\varphi)$ for $\varphi(X,Y,Z)$ involves computing over all X-assignments, the maximum number (weight) of Y-assignments that can be extended by a Z-assignment, whose result satisfies φ. The value k is given in unary, which better captures the essence of the problem, as for binary representations, already a single restricted call reaches the polynomial hierarchy [28].

Problem on φ	$X,Y,Z \neq \emptyset$	$X{=}\emptyset$ or $Y{=}Z{=}\emptyset$	$X{=}Z{=}\emptyset$	$X{=}Y{=}\emptyset$	Result
$\max\#\exists(\varphi) = k$	$\mathrm{D}_2^{\mathrm{P}}$-c	$\mathrm{D}_2^{\mathrm{P}}$-c	$(\mathrm{D}_2^{\mathrm{P}} \cap C_=^{\mathrm{P}}) \cap$ US-h	NP-c ($k{=}1$) / coNP-c ($k{=}-\infty$)	Theorem 1
$\max\#\exists(\varphi) \geq k$	NP-c	NP-c	NP-c	NP-c	Corollary 1
$\max\#\exists(\varphi) \leq k$	coNP-c	coNP-c	coNP-c	coNP-c	Corollary 2
$k \geq \max\#\exists(\varphi) > \max\#\exists(\psi)$	Θ_2^{P}-c	Θ_2^{P}-c	Θ_2^{P}	$\mathrm{D}_2^{\mathrm{P}}$-c	Thm. 2, Cor. 4

Structure of this Article

We provide essential preliminaries in Section II and start by proving the Quadchotomy Theorem and Theorem 2 in Section III. We continue with ETH lower bounds and discuss the Trichotomy Theorem, Theorem 4, and Theorem 5 in Section IV. We conclude in Section V. The table in Figure 5 provides an overview of our results.

II. Preliminaries

We assume the that reader is familiar with propositional logic and refer to the standard textbooks [5], [19]. The notations we use are stated within the introduction. We also assume basic knowledge of graph theory, for which we suggest the textbook by Diestel [9]. The *primal graph* G_φ of a WCNF is the graph with vertex set $V(G_\varphi) = \mathrm{vars}(\varphi)$ and edge set $E(G_\varphi) = \{\{u,v\} \mid u \text{ and } v \text{ appear together in a clause of } \varphi\}$. We are interested in formulas that are *structured* in the sense that their primal graph has small treewidth:

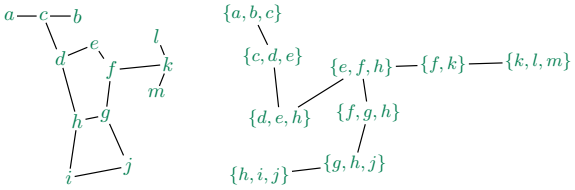
Definition 2 (Tree Decomposition)**.** *A* tree decomposition *of a graph G is a* rooted tree T *together with a mapping* $\mathrm{bag}\colon V(T) \to 2^{V(G)}$ *that satisfies the following properties:*

Connectedness *For every $v \in V(G)$ the induced subgraph $T[\{t \in V(T) \mid v \in \mathrm{bag}(t)\}]$ is a nonempty directed tree.*

Covering *For every $\{u,v\} \in E(G)$ there is a $t \in V(T)$ with $\{u,v\} \subseteq \mathrm{bag}(t)$.*

The *width* of (T, bag) is $\max_{t \in V(T)} |\mathrm{bag}(t)| - 1$, and the treewidth $\mathrm{tw}(G)$ of a graph G is the minimum width of any tree decomposition of G. To ease the design of algorithms over tree decompositions, we assume that tree decompositions are *nice* [7], i.e., that every internal node t is either an *introduce* or *forget* bag with exactly one child t' and $\mathrm{bag}(t) = \mathrm{bag}(t') \cup \{v\}$ or $\mathrm{bag}(t) = \mathrm{bag}(t') \setminus \{v\}$ for some $v \in V(G)$, respectively; or a *join* bag with children t' and t'' with $\mathrm{bag}(t) = \mathrm{bag}(t') = \mathrm{bag}(t'')$. A tree decomposition of a WCNF is a tree decomposition of G_φ, and we define $\mathrm{tw}(\varphi) := \mathrm{tw}(G_\varphi)$. In this case, we assume the presence of a mapping $\mathrm{label}\colon V(T) \to 2^\varphi$ that maps nodes t to sets of clauses c with $\mathrm{vars}(c) \subseteq \mathrm{bag}(t)$ such that every clause of φ appears in at least one label.

Example 1. *The treewidth of the Orion constellation (as graph on the left) is at least two since it contains a cycle. It is also at most two by the tree decomposition on the right:*



For a background in complexity theory we refer to Arora and Barak [2] and use NP, coNP, $\Sigma_2^{\mathrm{P}} = \mathrm{NP}^{\mathrm{NP}}$, $\Pi_2^{\mathrm{P}} = \mathrm{coNP}^{\mathrm{NP}}$, $\mathrm{D}_2^{\mathrm{P}} = \{L \cap L' \mid L \in \mathrm{NP}, L' \in \mathrm{coNP}\}$, $\Theta_2^{\mathrm{P}} = \mathrm{P}^{\mathrm{NP}[\log]} \subseteq \Delta_2^{\mathrm{P}} = \mathrm{P}^{\mathrm{NP}}$ in their usual meaning. The class of problems solvable by an NP machine such that the answer is "yes" iff there is exactly one accepting path is called US with $\mathrm{coNP} \subseteq \mathrm{US}$ [6]. The class $C_=^{\mathrm{P}}$ contains decision problems solvable by an NP machine such that on any yes-instance the number of accepting paths equals the number of rejecting paths [13], [30].

III. Complexity-Theoretic Aspects

To study the complexity of MAX#∃SAT we argue along the lines of the decision problem. We thereby query the result against a given value k in unary, as it is known that already very restricted cases are among the hardest counting problems. Indeed, already the problem with $X = Z = \emptyset$ is PP-complete for binary weights [1], [30], and a single of these calls reaches the polynomial hierarchy [28]. Consequently, the (function) problem of computing $\max\#\exists(\varphi)$ for a WCNF φ is hard

for PH under polynomial-time Turing reductions [28] and the problem is contained in FPSPACE (similarly to the PSPACE algorithm for quantified Boolean formulas [2, Page 83]).

However, we prove that already asking whether the resulting value $\max\#\exists(\varphi)$ is $\{\leq, =, \geq\}$ a polynomially bounded value k (given in unary), is a challenging task. Indeed, this study leads to an interesting complexity landscape of the problem ranging over various complexity classes. We prove the Quadchotomy Theorem, Theorem 1, via the following lemmas.

Lemma 1. *Deciding whether* $\max\#\exists(\varphi){=}k$ *for a simple* WCNF $\varphi(X, Y, Z)$ *and* $k \in \mathbb{N}$ *is* $\mathrm{D}_2^{\mathrm{P}}$*-complete.*

Proof. We first show membership, i.e., we show that we can decide whether $\max\#\exists(\varphi){=}k$ in $\mathrm{D}_2^{\mathrm{P}}$. To that end, we need to construct two propositional formulas such that the first formula is satisfiable and the second is unsatisfiable iff $\max\#\exists(\varphi){=}k$.

The idea of the proof is to construct a formula $\varphi_{\geq\rho}$ that is satisfiable if, and only if, $\max\#\exists(\varphi) \geq \rho$. Then $\max\#\exists(\varphi){=}k$ holds iff $\varphi_{\geq k}$ is satisfiable and $\varphi_{\geq k+1}$ is not.

To define $\varphi_{\geq\rho}$, we make ρ copies of φ such that we keep the X variables but obtain fresh sets Y_i and Z_i for the i-th copy ($1 \leq i \leq \rho$). Then, we add clauses (plain implications) specifying that the satisfying assignment over Y_i (viewed as a binary number) has to be smaller or equal to the satisfying assignment over Y_{i+1}. This construction allows us to track (up to) ρ satisfying assignments.

Using auxiliary variables α_{i+1}, we encode whether the $(i+1)$-th assignment over Y_{i+1} is equivalent to the one over Y_i (α_1 is a negative fact by adding the clause $\neg\alpha_1$). We can encode the multiplication of weights of literals up to ρ for every assignment over Y_i, as this is possible in TC^0 [17]. The result of bit-wise anding with $\neg\alpha_i$ shall then be stored using auxiliary variables β_i^0, ..., $\beta_i^{\lceil\log(\rho)\rceil+1}$, which also foresees an overflow bit. Finally, we sum up all these weights using bit-wise plus arithmetic, and express that the resulting binary digit must be larger than ρ (which is possible in AC^0). This construction then results in the propositional formula

$$\varphi_{\geq\rho} = \psi(X \cup Y_1 \cup \ldots \cup Y_k \cup Z).$$

As claimed, this formula is satisfiable iff $\max\#\exists(\varphi) \geq \rho$.

To show hardness, we reduce from the $\mathrm{D}_2^{\mathrm{P}}$-complete problem of deciding whether a given formula ψ_1 is satisfiable while another formula ψ_2 is unsatisfiable. By renaming variables if necessary, we can assume $\mathrm{vars}(\psi_1) \cap \mathrm{vars}(\psi_2) = \emptyset$.

For the reduction, we will construct an instance $\varphi(X, Y, Z)$ such that $\max\#\exists(\varphi) = 2$ iff ψ_1 is satisfiable and ψ_2 is not. We obtain φ as "union" of ψ_1 and ψ_2 using two fresh auxiliary variables a and s. Initially, we set $\varphi = \psi_1 \wedge \psi_2$, and use s to "switch" between ψ_1 and ψ_2. More precisely, for every clause $(l_1 \vee \ldots \vee l_k)$ in ψ_1 we create a clause $(\neg s \vee l_1 \vee \ldots \vee l_k)$ and for every $(l_1 \vee \ldots \vee l_k)$ in ψ_2 we create $(s \vee l_1 \vee \ldots \vee l_k)$. Furthermore, we add the clause $(\neg s \to a)$.

To conclude the construction, we set

$$X = \mathrm{vars}(\psi_1) \cup \mathrm{vars}(\psi_2), \quad Y = \{a, s\}, \quad \text{and} \quad Z = \emptyset.$$

All variables in X obtain a weight of 0, while all variables in Y have 1. Observe that for $\beta \sqsubseteq \mathrm{vars}(\psi_1) \cup \mathrm{vars}(\psi_2)$ with $\beta \not\models \psi_1$, we need to set $s = 0$ and, thus, $a = 1$, to extend β to a model of φ. Therefore, there is only *one* extension of β over Y to satisfy φ. If β is a model of ψ_1 and ψ_2, the variables s and a are only constrained by $(\neg s \to a)$ and, hence, there are *three* extensions. On the other hand, if $\beta \models \psi_1$ and $\beta \not\models \psi_2$, we force $s = 1$, but a is free and, hence, *two* extensions of β satisfy φ. Since we *maximize* overall assignments to the variables in X, we have that ψ_1 is satisfiable and ψ_2 is unsatisfiable if, and only if, $\max\#\exists(\varphi) = 2$. □

Lemma 2. *Let* $\varphi(X, Y, Z)$ *be a given simple* WCNF*. Deciding* $\max\#\exists(\varphi) = k$ *for* $k \in \mathbb{N}$ ...

- *is in* $\mathrm{D}_2^{\mathrm{P}} \cap C_=^{\mathrm{P}}$ *and* US*-hard if* $X = Z = \emptyset$*;*
- *remains* $\mathrm{D}_2^{\mathrm{P}}$*-hard if* $Y = Z = \emptyset$*.*

Proof. The proof of membership in $\mathrm{D}_2^{\mathrm{P}}$ for $X = Z = \emptyset$ works similarly to Lemma 1. It is also known that the problem is in the counting complexity class $C_=^{\mathrm{P}}$ for unbounded k (i.e., even if k is given in binary) [30].

For the special case $k = 1$ (and $X = Z = \emptyset$), the problem asks whether a given propositional formula has exactly one satisfying assignment, which is the canonical problem of *unique polynomial time*. Hence, the problem is US-hard.

In the case of the lemma with $Y = Z = \emptyset$, we are left with MAXSAT. We follow the line of argument as in the hardness proof of Lemma 1, so we reduce from the $\mathrm{D}_2^{\mathrm{P}}$-complete problem of deciding whether a given formula ψ_1 is satisfiable while another given formula ψ_2 is unsatisfiable.

We construct $\varphi(X, Y, Z)$ such that $\max\#\exists(\varphi){=}1$ iff ψ_1 is satisfiable and ψ_2 is not. We obtain φ again as "union" of ψ_1 and ψ_2, but using *three* fresh auxiliary variables s_1, s_2, and a this time. Initially, we set $\varphi = \psi_1 \wedge \psi_2$ and use s_1 and s_2 to deactivate the clauses of ψ_1 or ψ_2, respectively. More precisely, for every clause $(l_1 \vee \ldots \vee l_k)$ in ψ_1 we create a clause $(\neg s_1 \vee l_1 \vee \ldots \vee l_k)$ and for every $(l_1 \vee \ldots \vee l_k)$ in ψ_2 we create $(\neg s_2 \vee l_1 \vee \ldots \vee l_k)$. We also bound the variable a to s_2 by adding the clause $(s_2 \leftrightarrow a)$. Finally, we set

$$X = \mathrm{vars}(\psi_1) \cup \mathrm{vars}(\psi_2) \cup \{s_1, s_2, a\},\ Y = \emptyset, \text{ and } Z = \emptyset.$$

All variables in $\{s_1, s_2, a\}$ obtain weight 1, while all variables in $X \setminus \{s_1, s_2, a\}$ have 0. Then, the maximum value any assignment of φ can have is 3 (setting $s_1 = s_2 = a = 1$), which requires an assignment that satisfies ψ_1 *and* ψ_2. On the other hand, an assignment that satisfies just ψ_2 has weight 2 ($s_1 = 0$ and $s_2 = a = 1$ due to $s_2 \leftrightarrow a$); an assignment just satisfying ψ_1 has weight 1 ($s_1 = 0$ and $s_2 = a = 0$); and an assignment that does neither satisfies ψ_1 or ψ_2 has weight 0. We conclude that $\max\#\exists(\varphi) = 1$ if, and only if, ψ_1 is satisfiable and ψ_2 is unsatisfiable. □

The result allows us to prove Theorem 2, i.e., that testing $k \geq \max\#\exists(\varphi) > \max\#\exists(\psi)$ is Θ_2^{P}-complete.

Proof of Theorem 2. We first prove membership, i.e., we show that $\max\#\exists(\varphi) > \max\#\exists(\psi)$ can be decided in Θ_2^{P}.

The idea is to use the formula $\varphi_{\geq\rho}$ from Lemma 1 and perform a binary search. In detail, we perform a binary search on the interval $[0, k]$ (recall that $k \in \mathbb{N}$ is given in unary in the statement of the theorem) and, for the current $k' \in [0, k]$, we test whether $\max\#\exists(\varphi) \geq k'$ and $\max\#\exists(\psi) \geq k'$. If it is true for φ but not for ψ, we found a certificate that we deal with a yes-instance; if it is the opposite, we certified that we deal with a no-instance. In the remaining cases (both queries are positive or both negative), we simply continue the binary search. This procedure requires $O(\log k)$ calls to an NP-oracle since every question translates to a SAT-question using Lemma 1. Therefore, the decision can be done in Θ_2^{P}.

To establish hardness, we reduce from the PARITY(SAT) problem, which is well-known to be Θ_2^{P}-complete [11], [31]. The input for this problem is a sequence of CNFs $\varphi_1, \ldots, \varphi_n$ with $\mathrm{vars}(\varphi_i) \cap \mathrm{vars}(\varphi_j) = \emptyset$ for $1 \leq i < j \leq n$. The question is whether there is an *odd index* $i \in \{1, \ldots, n-1\}$ such that all $\varphi_1, \ldots, \varphi_i$ are satisfiable while all $\varphi_{i+1}, \ldots, \varphi_n$ are unsatisfiable. Without loss of generality, we can assume that no φ_i is a tautology and that n is even; the latter can be ensured by adding a trivial invalid formula at the end of the sequence. We can also assume that φ_1 is valid by adding two trivially valid formulas at the beginning of the sequence.

For the reduction, we set $k = n - 1$ and construct two formulas φ and ψ with $k > \max\#\exists(\varphi)$ and $k > \max\#\exists(\psi)$ such that $\max\#\exists(\varphi) > \max\#\exists(\psi)$ if, and only if, there is an odd index $i \in \{1, \ldots, n-1\}$ for which all $\varphi_1 \ldots, \varphi_i$ are satisfiable, while all $\varphi_{i+1} \ldots, \varphi_n$ are not. The idea is to construct ψ such that $\max\#\exists(\psi) = |\{\varphi_i \mid \varphi_i \in \text{SAT}\}|$ is the number of satisfiable instances. In contrast, φ will have an "advantage" in the first block of satisfiable instances of the sequence, which only helps in yes-instances. Let us first describe ψ, for which we need auxiliary variables $S = \{s_1, \ldots, s_n\}$ and $A = \{a_1, \ldots, a_n\}$. We denote with $\mathrm{tseitin}(\gamma)$ the CNF encoding of any formula γ due to Tseitin [29] and let ψ be:

$$\psi \coloneqq \bigwedge_{1 \leq i \leq n} \mathrm{tseitin}(s_i \leftrightarrow \varphi_i) \wedge \bigwedge_{1 \leq i \leq n} (a_i \rightarrow s_i) \wedge \bigvee_{1 \leq i \leq n} (a_i) \wedge \bigwedge_{1 \leq i < j \leq n} (\neg a_i \vee \neg a_j).$$

We force to set *exactly* one a_i variable to true. Furthermore, if a_i is set to true, then s_i is forced to be true and, thus, φ_i must be satisfiable. Setting $w(a_i) = w(\neg a_i) = 1$ for all $a_i \in A$ and $w(x) = w(\neg x) = 0$ for all $x \notin A$ we obtain:

$$\begin{aligned}&\max\#\exists\big(\psi(\emptyset, A, \mathrm{vars}(\psi) \setminus A)\big)\\ =&\max\#\exists\big(\psi(\mathrm{vars}(\psi) \setminus A), A, \emptyset\big) = |\{\varphi_i \mid \varphi_i \in \text{SAT}\}|.\end{aligned}$$

We build φ based on ψ, but set $w(a_i) = 2$ and $w(\neg a_i) = 1$:

$$\varphi \coloneqq \psi \wedge \bigwedge_{2 \leq i \leq n} (s_i \rightarrow s_{i-1}) \wedge \bigwedge_{\substack{2 \leq i \leq n \\ i \equiv 0 \pmod 2}} (\neg a_i).$$

Due to the last part of the formula $(\neg a_i)$, we can only set a_i on odd positions to true, and due to $(s_i \rightarrow s_{i-1})$, we can only select an a_i if all formulas φ_j with $j \in \{1, \ldots, i\}$ are satisfiable. Hence, if r is the *maximum* odd index such that all φ_j with $j \in \{1, \ldots, r\}$ are satisfiable, we get:

$$\begin{aligned}&\max\#\exists\big(\varphi(\emptyset, A, \mathrm{vars}(\psi) \setminus A)\big)\\ =&\max\#\exists\big(\varphi(\mathrm{vars}(\psi) \setminus A, A, \emptyset)\big) = r + 1.\end{aligned}$$

Consequently, if $\varphi_1, \ldots, \varphi_n$ is a yes-instance, we have:

$$\begin{aligned}\max\#\exists(\varphi) &= |\{\varphi_i \mid \varphi_i \in \text{SAT}\}| + 1\\ &> |\{\varphi_i \mid \varphi_i \in \text{SAT}\}| = \max\#\exists(\psi).\end{aligned}$$

On the other hand, in a no-instance we clearly have:

$$\max\#\exists(\varphi) \leq \max\#\exists(\psi).$$

This construction establishes hardness even if $X = X' = \emptyset$, $Z = Z' = \emptyset$, $X = Z' = \emptyset$, or $Z = X' = \emptyset$. For the remaining case of the theorem, i.e., $Y = Y' = \emptyset$, we need to "move" the logic from counting to optimization. This can easily be achieved by removing the "select at most one a_i" part from ψ. Let the resulting formula be ψ' and let φ' be the formula obtained by replacing ψ with ψ' in φ. If we set $w(\neg a_i) = 0$ for all $a_i \in A$ and $w(x) = w(\neg x) = 0$ for all $x \notin A$, we get:

$$\begin{aligned}&\max\#\exists\big(\varphi'(\mathrm{vars}(\varphi'), \emptyset, \emptyset)\big)\\ >\ &\max\#\exists\big(\psi'(\mathrm{vars}(\psi'), \emptyset, \emptyset)\big)\end{aligned}$$

if, and only if, the input is a yes-instance. □

One interesting case is not handled by Theorem 2, namely $X = X' = Y = Y' = \emptyset$, i.e., that only Z and Z' are non-empty. In this case, we only deal with propositional satisfiability problems and, hence, have to check whether φ is satisfiable while ψ is not. This is exactly the canonical complete problem for $\mathrm{D}_2^{\mathrm{P}}$.

Corollary 4. *Deciding $k \geq \max\#\exists(\varphi) > \max\#\exists(\psi)$ for a $k \in \mathbb{N}$ and two simple* WCNF*s $\varphi(\emptyset, \emptyset, Z)$ and $\psi(\emptyset, \emptyset, Z')$ is complete for $\mathrm{D}_2^{\mathrm{P}}$.*

IV. ETH Lower Bounds and Structure-Aware Compilations

The results of the last section illustrate that computing $\max\#\exists(\varphi)$ is a difficult problem. We may hope that structured instances allow for faster algorithms, i.e., that we can solve the problem efficiently on instances of *small treewidth.* In a sense, we answer this in the affirmative by providing an algorithm that computes $\max\#\exists(\varphi)$ in time $f(k) \cdot |\varphi|$ for some computable function $f\colon \mathbb{N} \to \mathbb{N}$ if a width-k tree decomposition of G_φ is given. That is, we prove that MAX#∃SAT is *fixed-parameter tractable* for the parameter *treewidth*. Unfortunately, the function f in our algorithm is *triple-exponential,* and we show that this is optimal assuming the exponential time hypothesis (ETH) [18]. On the other hand, the various complexities explored in the last section are mirrored in the parameterized world: If one or two of the variable sets are empty, we end up with a double- or single-exponential runtime. We first prove the lower bounds:

Proof of the Trichotomy Theorem. We show the result for the most involved case where X, Y, and Z are not empty. The

remaining cases work analogously; see also Lemma 1. The proof is by a reduction from the validity problem of a QBF of the form $\varphi'(U,V,W) = \exists U.\forall V.\exists W.\varphi$, which under ETH can*not* be solved in $2^{2^{2^{o(\mathrm{tw}(\varphi))}}} \cdot \mathrm{poly}(|U \cup V \cup W|)$ [15].

We can decide φ' by computing $\max\#\exists(\varphi)$ such that φ' is valid if, and only if, $\max\#\exists(\varphi) = 2^{|V|}$. Note that here $2^{|V|}$ is a number given in binary, which is in contrast to the decision problems studied in Section III. However, this statement focuses on the *computation* of the resulting value $\max\#\exists(\varphi)$. Consequently, the lower bound immediately carries over to computing $\max\#\exists(\varphi)$. □

A. Eliminating Projection via a Structure-Aware Reduction

We complement the lower bounds with a compilation from MAX#∃SAT to MAX#SAT (without "∃", i.e., without the projection). The reduction is *structure-aware* in the sense that it obtains a tree decomposition of the input formula and produces a decomposition of the output of a width matching Theorem 3. To establish Theorem 4, we first describe the translation from φ to ψ, prove soundness in Lemma 3, the bound on the treewidth in Lemma 4, and the claimed runtime in Lemma 5.

Let in the following $\varphi(X,Y,Z)$ be the given WCNF, and let $\mathcal{T} = (T, \mathrm{bag}, \mathrm{label})$ be a labeled width-k tree decomposition of G_φ. Without loss of generality, we assume $|\operatorname{label}(t)| \leq 1$ and $|\operatorname{children}(t)| \leq 2$ for all $t \in V(T)$. We introduce variables S of the form sat_c^α for every $c \in \varphi$ and every assignment $\alpha \sqsubseteq \mathrm{bag}(\mathrm{label}^{-1}(c)) \cap Z$ indicating that c is satisfied in its bag:

$$\bigwedge_{c \in \psi} \bigwedge_{\substack{\alpha \sqsubseteq \mathrm{bag}(\mathrm{label}^{-1}(c)) \cap Z \\ \{c\}|\alpha \neq \emptyset}} \left[\mathrm{sat}_c^\alpha \leftrightarrow \bigvee_{\lambda \in \{c\}|\alpha} \lambda \right], \tag{6}$$

$$\bigwedge_{c \in \psi} \bigwedge_{\substack{\alpha \sqsubseteq \mathrm{bag}(\mathrm{label}^{-1}(c)) \cap Z \\ \{c\}|\alpha = \emptyset}} \left[\mathrm{sat}_c^\alpha \right]. \tag{7}$$

We need variables $\mathrm{sat}_{\leq t}^\alpha$ and $\mathrm{sat}_{<t,t'}^\alpha$ to propagate information along the tree decomposition:

// There is a clause satisfying the bag and clauses below are satisfied as well:

$$\bigwedge_{t \in V(T)} \bigwedge_{\alpha \sqsubseteq \mathrm{bag}(t) \cap Z} \left[\mathrm{sat}_{\leq t}^\alpha \leftrightarrow \bigwedge_{c \in \mathrm{label}(t)} \mathrm{sat}_c^\alpha \wedge \bigwedge_{t' \in \mathrm{children}(t)} \mathrm{sat}_{<t,t'}^\alpha \right], \tag{8}$$

// Propagate satisfiability:

$$\bigwedge_{t \in V(T)} \bigwedge_{\alpha \sqsubseteq \mathrm{bag}(t) \cap Z} \bigwedge_{t' \in \mathrm{children}(t)} \left[\mathrm{sat}_{<t,t'}^\alpha \leftrightarrow \bigvee_{\substack{\beta \sqsubseteq \mathrm{bag}(t') \cap Z \\ \beta \cap \mathrm{lits}(\mathrm{bag}(t)) = \alpha \cap \mathrm{lits}\,\mathrm{bag}(t')}} \mathrm{sat}_{\leq t'}^\beta \right]. \tag{9}$$

To ensure that there is an assignment over Z that satisfies all clauses, we add constraints for the root:

$$\bigvee_{\alpha \sqsubseteq \mathrm{bag}(\mathrm{root}(T)) \cap Z} \mathrm{sat}_{\leq \mathrm{root}(T)}^\alpha. \tag{10}$$

Observe that the presented formulas can be converted into CNF, even without introducing additional auxiliary variables. Indeed, the shown formulas are essentially equivalences over variables of the form sat_c^α such that for any fixed assignment over variables in $X \cup Y$, those variables in S are pinned down.

Lemma 3 (Soundness)**.** *Let $\varphi(X,Y,Z)$ be a given* WCNF *and $(T, \mathrm{bag}, \mathrm{label})$ be a tree decomposition of G_φ. Equations (6)–(10) yield a formula $\psi(X,Y')$ with $Y' = Y \cup S$ such that any assignment $\alpha \sqsubseteq X \cup Y$ extends to a model of φ iff a (single) extension of α is a model of ψ.*

Proof. The insight is that, by construction and by the properties of a tree decomposition, Equations (6)–(10) ensure that any assignment α over $X \cup Y$ can be extended to a model of φ if, and only if, α can be extended to a model of ψ.

Let α be the given assignment over $X \cup Y$. If there is an extension β over Z such that $\alpha \cup \beta$ is a model of φ, there is a unique extension β' over S such that $\alpha \cup \beta'$ is a model of ψ because Equations (6)–(10) are bi-implications. Due to the existence of β, we know from the correctness of dynamic programming for satisfiability [26] that also Equation (10) evaluates to true. For the other direction, suppose there is an extension β' over S of α such that $\alpha \cup \beta'$ is a model of ψ, and note that β' captures all satisfying extensions over Z of α. We construct an extension β by inspecting compatible $sat_{\leq t}^\gamma$ variables from the root node downwards towards the leaves of T. Going top-down, we pick a variable that is compatible with the variable for the parent node. Then we unify the corresponding assignments over Z, which yields β. □

Lemma 4 (Treewidth-Awareness)**.** *The reduction over Equations (6)–(10) on a* WCNF *$\varphi(X,Y,Z)$ and a width-k tree decomposition of G_φ, outputs a formula ψ with $\mathrm{tw}(\psi) \leq 2^{k+3}$.*

Proof. Let (T, bag) be a tree decomposition of G_φ of width k. By copying nodes if needed, we obtain a labeled tree decomposition $\mathcal{T} = (T, \mathrm{bag}, \mathrm{label})$ with $|\operatorname{children}(t)| \leq 2$ for every $t \in V(T)$. We construct $\mathcal{T}' = (T, \mathrm{bag}')$ for ψ (given through Equations (6)–(10)) by defining for every $t \in V(T)$:

$$\begin{aligned} \mathrm{bag}'(t) = (\mathrm{bag}(t) \cap (X \cup Y)) &\cup \{\mathrm{sat}_{\leq t}^\alpha \mid \alpha \sqsubseteq \mathrm{bag}(t) \cap Z\} \\ &\cup \{\mathrm{sat}_c^\alpha \mid \alpha \sqsubseteq \mathrm{bag}(t) \cap Z, c \in \mathrm{label}(t)\} \\ &\cup \big\{ \mathrm{sat}_{\leq t'}^\beta, \mathrm{sat}_{<t,t'}^\alpha \mid t' \in \mathrm{children}(t), \alpha \sqsubseteq \mathrm{bag}(t) \cap Z, \\ &\beta \sqsubseteq \mathrm{bag}(t') \cap Z, \alpha \cap \mathrm{lits}(\mathrm{bag}(t')) = \beta \cap \mathrm{lits}(\mathrm{bag}(t)) \big\}. \end{aligned}$$

By construction, $|\,\mathrm{bag}'(t)| \leq k + 2^k + 2^k + 4 \cdot 2^k \leq 2^{k+3}$. □

Lemma 5 (Runtime)**.** *The reduction over Equations (6)–(10) on a* WCNF *$\varphi(X,Y,Z)$ and a width-k tree decomposition $\mathcal{T}$ of G_φ runs in time $2^{2k+3} \cdot (|\operatorname{vars}(\varphi)| + |\varphi|)$.*

Proof. The runtime follows from analyzing the effort required for Equations (6)–(10). The $2k$ superscript is due to the $(2^k)^2$ effort behind Equation (9). □

B. An ETH-Tight Parameterized Algorithm for Max#SAT

The reduction of the previous subsection eliminates projection. We may ask whether we can apply a similar idea to eliminate counting or maximization. Unfortunately, this seems to be difficult due to the weights. However, the approach can be extended to the case in which we are only interested in those assignments over X for which every extension over Y is a satisfying assignment. We can create a MAXSAT instance ψ by encoding universality over Y by replacing the disjunctions

Listing 1: Dynamic Programming Algorithm Max#SAT$(t, \varphi, \text{bag}, \langle \tau_1, \ldots, \tau_{|\text{children}(t)|} \rangle)$.

In: Decomposition node t, WCNF $\varphi(X, Y)$, bag, child tables $\tau_1, \ldots, \tau_{|\text{children}(t)|}$. **Out:** Table for node t.

1 **if** t *is an (empty) leaf node* **then return** $\{\langle \emptyset, 0, \{\langle \emptyset, 1 \rangle\}\rangle\}$

2 **else if** t *is an introduce node introducing variable* $v \in \text{bag}(t)$ **then**
return $\{\langle J, f, \mathcal{J} \rangle \mid \langle I, f, \mathcal{I} \rangle \in \tau_1, J \in \{I^+_v \cap \text{lits}(X), I^+_v \cap \text{lits}(X)\}, |\mathcal{J}| > 0,$
$\mathcal{J} = \{\langle J', f' \rangle \mid \langle I', f' \rangle \in \mathcal{I}, J' \in \{I'^+_v \cap \text{lits}(Y), I'^+_v \cap \text{lits}(Y)\}, (J \cup J') \models \varphi_t\}\}$

3 **else if** t *is a forget node forgetting* $v \in X \setminus \text{bag}(t)$ **then**
return KeepMax $(\{\langle I^-_v, f + w(I \cap \{v, \neg v\}), \{\langle I'^-_v, f' \rangle \mid \langle I', f' \rangle \in \mathcal{I}\}\rangle \quad | \langle I, f, \mathcal{I} \rangle \in \tau_1\})$

4 **else if** t *is a forget node forgetting* $v \in Y \setminus \text{bag}(t)$ **then**
return KeepMax $(\{\langle I^-_v, f, \{\langle I'^-_v, \Sigma_{\langle J, g \rangle \in \mathcal{I} : J^-_v = I'^-_v} g \rangle \mid \langle I', f' \rangle \in \mathcal{I}\}\rangle \quad | \langle I, f, \mathcal{I} \rangle \in \tau_1\})$

5 **else if** t *is a join node* **then**
return KeepMax $(\{\langle I, f_1 + f_2, \mathcal{J} \rangle, |\mathcal{J}| > 0, \mathcal{J} = \{\langle I', f'_1 \cdot f'_2 \rangle \mid \langle I', f'_1 \rangle \in \mathcal{I}_1, \langle I', f'_2 \rangle \in \mathcal{I}_2\} \quad | \langle I, f_1, \mathcal{I}_1 \rangle \in \tau_1, \langle I, f_2, \mathcal{I}_2 \rangle \in \tau_2\})$

For a set S, a variable v, and a literal ℓ, we let $S^+_\ell = S \cup \{\ell\}$ and $S^-_v = S \setminus \{v, \neg v\}$.

in Equations (9) and (10) by conjunctions. The goal is to simulate $\delta = \sum_{\beta \sqsubseteq Y} \prod_{\ell \in \beta} w(\ell)$ via soft variables. To this end, let B be the binary representation of δ. We add auxiliary variables p_i for positions i in B that are set to 1. For every such i, we introduce i additional auxiliary variables $b^1_i, \ldots, b^i_i$ to create weight 2^i. Whenever p_i is 0, we pin down the value of the b^j_i variables, i.e., we add $\neg p_i \to \neg b^j_i$ for every $1 \leq j \leq i$. We require clauses enforcing that exactly one of the p_i's is set to true (with a treewidth increase of 3). Unfortunately, it is unclear how this approach generalizes, as both the weight and the final result depend on the assignment over X.

C. An ETH-Tight Algorithm for Max#∃SAT

We develop a dynamic program over a tree decomposition to obtain matching upper bounds. Especially challenging are the cases for forget and join nodes. To ensure these precise guarantees, we must not have identical rows (with different weights). However, while keeping the maximum weight would suffice, we do not know which of the partial assignments over Y sustains. Depending on which of these rows participate in a satisfying assignment, this might result in different maxima. Consequently, our approach proceeds in *two computing stages*. First, we compute partial assignments in a bottom-up traversal (in post-order along the tree decomposition), which results in a table of rows. Then, we remove those rows that do not participate in a satisfying assignment. This is crucial, as we must not have identical sets of assignments with different weights. Afterwards, we can correctly determine the weights of individual rows in a *second bottom-up tree traversal*.

We design an algorithm for MAX#SAT, which generalizes to MAX#∃SAT using the encoding from the previous section. Let $\varphi(X, Y, \emptyset)$ be a WCNF formula and $\mathcal{T} = (T, \text{bag})$ be a tree decomposition of G_φ. Algorithm 1 is a dynamic program that can be used for both stages. As data structure for nodes t in T, we maintain sets of tuples, where the first elements I are assignments restricted to variables in $Y \cap \text{bag}(t)$. Further, f is a fitness value (integer). Then, the third tuple component is a set $\mathcal{I}$ of tuples $\langle I', f' \rangle$ with I' being again an assignment restricted to $X \cap \text{bag}(t)$ and f' being a fitness value.

For the first stage, the function KeepMax in Algorithm 1 is the identity function, and we do not need to compute fitness values f (second tuple positions), which are highlighted in green (dark green) respectively. The algorithm maintains tuples comprising a partial model I over X and a set $\mathcal{I}$ of partial models over Y. Whenever a variable is introduced, Line 2 decides on the truth value and keeps satisfying assignments. Lines 3 and 4 project these assignments to $\text{bag}(t)$ for forget nodes; Line 5 merges assignments for join nodes.

In between the stages, we purge tables in a bottom-up traversal for every node t in T: We remove from $\mathcal{I}$ (with $\langle I, \mathcal{I} \rangle$ being contained in the table of t) those elements that do not lead to a (succeeding) element in the table for the parent of t. These non-succeeding elements e for t can be determined by passing them one by one as singletons $\{\langle I, \{e\} \rangle\}$ for τ_1 (τ_2) and checking whether the result is empty. Let τ_t be the table for t after purging. Given the τ_t, we can define the function KeepMax used in Algorithm 1 (second stage). The function for a table τ at a node t then only keeps those elements in τ that (after shaving off fitness values f from τ) appear in τ_t. If there are several elements in τ appearing in τ_t after shaving off fitness values, we only keep one among these, namely the one yielding the maximum sum over f and fitness values f'. This algorithm establishes Theorem 5.

D. Simulating Positive Weights

Algorithm 1 operates on simple WCNFs, in contrast to the generic definition of MAX#SAT. The issue is Lines 4, where we multiply fitness value g by $w(J \cap \{v, \neg v\})$. Due to this product, decisions to keep the maximum in Lines 3, 4, or 5 might be wrong. In fact, in case of multiplications, Lines 3, 4, or 5 must not decide on a maximum row in case of duplicates, as these might not form the global maxima. However, we can simulate positive integer weights that are powers of 2.

Corollary 5. *There is an algorithm that solves* MAX#∃SAT *for a* WCNF $\varphi(X, Y, Z)$ *with natural weights over* X *and power-of-2 weights over* Y *in time* $\exp(3, \mathcal{O}(\text{tw}(\varphi))) \cdot |\text{vars}(\varphi)|$.

Corollary 6. MAX#∃SAT *on* WCNF $\varphi(X, Y, Z)$ *with natural weights over* X *and pos. integer weights* $\leq w$ *over* Y *can be solved in time* $\exp(3, \mathcal{O}(\max(\log(w), \text{tw}(\varphi)))) \cdot |\text{vars}(\varphi)|$.

V. Conclusion

We identified MAX#∃SAT with a weighting scheme over X (the *fitness* term) *and* over Y (the *probability* term) as

the canonical problem of *cost-optimal probabilistic reasoning*. We investigated the complexity of the problem and proved with the Quadchotomy Theorem that versions are complete for classes beyond NP and coNP. This complexity landscape of MAX#∃SAT is mirrored in the parameterized setting via the Trichotomy Theorem, in which we established ETH-tight bounds containing single, double, or triple exponential dependencies on the input's treewidth. The table in Figure 5 provides an overview of all our results.

References

[1] Shyan Akmal and Ryan Williams. MAJORITY-3SAT (and Related Problems) in Polynomial Time. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1033–1043. IEEE, 2021.

[2] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[3] Gilles Audemard, Jean-Marie Lagniez, and Marie Miceli. A New Exact Solver for (Weighted) Max#SAT. In *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, pages 28:1–28:20, 2022.

[4] Jeremias Berg, Antti Hyttinen, and Matti Järvisalo. Applications of MaxSAT in Data Analysis. In *Proceedings of Pragmatics of SAT 2015, Austin, Texas, USA, September 23, 2015 / Pragmatics of SAT 2018, Oxford, UK, July 7, 2018*, pages 50–64, 2018.

[5] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability, Second Edition*. IOS Press, 2021.

[6] Andreas Blass and Yuri Gurevich. On the Unique Satisfiability Problem. *Inf. Control.*, 55(1-3):80–88, 1982.

[7] Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996.

[8] Blai Bonet, Guillem Francès, and Hector Geffner. Learning Features and Abstract Actions for Computing Generalized Plans. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 2703–2710, 2019.

[9] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[10] Leonardo Dueñas-Osorio, Kuldeep S. Meel, Roger Paredes, and Moshe Y. Vardi. Counting-based reliability estimation for power-transmission grids. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4488–4494, 2017.

[11] Thomas Eiter and Georg Gottlob. The complexity class theta$_2^{\mathrm{p}}$: Recent results and applications in AI and modal logic. In *Fundamentals of Computation Theory, 11th International Symposium, FCT '97, Kraków, Poland, September 1-3, 1997, Procs.*, volume 1279 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 1997.

[12] Linus Feiten, Matthias Sauer, Tobias Schubert, Alexander Czutro, Eberhard Böhl, Ilia Polian, and Bernd Becker. #sat-based vulnerability analysis of security components - A case study. In *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFT 2012, Austin, TX, USA, October 3-5, 2012*, pages 49–54, 2012.

[13] Stephen A. Fenner, Frederic Green, Steven Homer, and Randall Pruim. Determining Acceptance Possibility for a Quantum Computation is Hard for the Polynomial Hierarchy. *Electron. Colloquium Comput. Complex.*, TR99-003, 1999.

[14] Johannes K. Fichte, Markus Hecher, and Florim Hamiti. The Model Counting Competition 2020. *ACM Journal of Experimental Algorithmics*, 26(1):1–26, 2021.

[15] Johannes K. Fichte, Markus Hecher, and Andreas Pfandler. Lower Bounds for QBFs of Bounded Treewidth. In *35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 410–424. ACM, 2020.

[16] Daniel J. Fremont, Markus N. Rabe, and Sanjit A. Seshia. Maximum Model Counting. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3885–3892, 2017.

[17] William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.*, 65(4):695–716, 2002.

[18] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[19] Donald E. Knuth. *The Art of Computer Programming*, volume 4, Fascicle 6. Addison-Wesley, 2016.

[20] Tuukka Korhonen and Matti Järvisalo. Integrating Tree Decompositions into Decision Heuristics of Propositional Model Counters (Short Paper). In *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, pages 8:1–8:11, 2021.

[21] Nian-Ze Lee, Yen-Shi Wang, and Jie-Hong R. Jiang. Solving stochastic boolean satisfiability under random-exist quantification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 688–694, 2017.

[22] Nian-Ze Lee, Yen-Shi Wang, and Jie-Hong R. Jiang. Solving exist-random quantified stochastic boolean satisfiability via clause selection. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 1339–1345, 2018.

[23] Alexandre Lemos, Pedro T. Monteiro, and Inês Lynce. Minimal Perturbation in University Timetabling with Maximum Satisfiability. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 17th International Conference, CPAIOR 2020, Vienna, Austria, September 21-24, 2020, Proceedings*, pages 317–333, 2020.

[24] Nina Narodytska, Aditya A. Shrotri, Kuldeep S. Meel, Alexey Ignatiev, and João Marques-Silva. Assessing heuristic machine learning explanations with model counting. In *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, pages 267–278, 2019.

[25] Marek Piotrów. UWrMaxSat: Efficient Solver for MaxSAT and Pseudo-Boolean Problems. In *32nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2020, Baltimore, MD, USA, November 9-11, 2020*, pages 132–136. IEEE, 2020.

[26] Marko Samer and Stefan Szeider. Algorithms for propositional model counting. *Journal of Discrete Algorithms*, 8(1):50–64, 2010.

[27] Xujie Si, Xin Zhang, Radu Grigore, and Mayur Naik. Maximum Satisfiability in Software Analysis: Applications and Techniques. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, pages 68–94, 2017.

[28] Seinosuke Toda. PP is as Hard as the Polynomial-Time Hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[29] G. S. Tseitin. On the complexity of derivation in propositional calculus. In *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970*, pages 466–483. Springer Berlin Heidelberg, 1983.

[30] Ito Tsuyoshi. Discussion on stackexchange: Comparing Solution Counts among Propositional Formulas and PP-Completeness. https://tinyurl.com/27m6r3tq. Accessed: 2024-03-15.

[31] Klaus W. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theor. Comput. Sci.*, 51:53–80, 1987.

[32] Stephan Wäldchen, Jan MacDonald, Sascha Hauch, and Gitta Kutyniok. The Computational Complexity of Understanding Binary Classifier Decisions. *J. Artif. Intell. Res.*, 70:351–387, 2021.