

Extending CDCL-Based Model Enumeration with Weights

Giuseppe Spallitta^{1,*}, Moshe Vardi¹

¹Rice University, Houston, Texas, United States of America

Abstract

In this work we investigate *Weighted Model Enumeration* (WME): given a Boolean formula and a weight function over its satisfying assignments, enumerate models while accounting for their weights. This setting supports weight-driven queries, such as producing the top- k models or all models above a threshold. While related to AllSAT, Weighted Model Counting, and MaxSAT, these paradigms do not treat selective enumeration under weights as a native solver task. We present CDCL-based algorithms for WME that integrate weight propagation, weight-based pruning, and *weight-aware* conflict analysis into both chronological and non-chronological backtracking frameworks. Chronological backtracking exploits implicit blocking to keep the clause database compact, thereby reducing the memory footprint and enabling efficient propagation. In contrast, non-chronological backtracking with clause learning supports explicit blocking and restarts. We show that both approaches are feasible and complementary, highlighting trade-offs in pruning effectiveness with weights and clarifying when each performs best.

Keywords

AllSAT, Weighted Model Enumeration, Weighted Conflict Analysis

1. Introduction

In many artificial-intelligence applications, including probabilistic inference [1], structured decision-making [2], fault diagnosis [3], and interpretable machine learning [4], the objective is not merely to obtain a satisfying assignment of a Boolean formula, but to identify a set of "high-quality" models. Such models often correspond to the most probable explanations under a weight function. Classical formulations such as Maximum a Posteriori (MAP) [5] and Most Probable Explanation (MPE) [6, 7] capture this idea by seeking the single highest-scoring assignment. Yet in practice, many applications require richer outputs: multiple explanations, such as top- k weighted models or all models above a weight threshold, are indispensable for robust decision support and interpretability. This need has long been recognized in database theory through top- k query processing [8, 9] and in probabilistic inference through top- k explanation tasks [10].

Existing CDCL-based reasoning tools fall short of providing native support for weight-aware enumeration of Boolean formulae. AllSAT solvers [11, 12, 13, 14] enumerate indiscriminately, ignoring weights entirely. Weighted Model Counting (WMC) [15, 16, 17, 18, 19] computes global aggregates but provides no access to individual weighted models. MaxSAT solvers [20, 21, 22] return a single maximum-weight solution, but lack mechanisms for ranking or filtering multiple alternatives. The 2020 MaxSAT Evaluation introduced a dedicated track for top- k enumeration [23], highlighting both the recognized importance and the unsolved nature of top- k reasoning. Thus, none of these approaches natively supports selective enumeration under weight constraints. As a result, a natural and practically important task remains unexplored: *Weighted Model Enumeration* (WME).

This work formalizes and develops a CDCL-native framework for WME. Although the problem is naturally connected to both AllSAT and MaxSAT, addressing it efficiently requires solver mechanisms that go beyond a straightforward combination of the two. Beyond the definition, we introduce weight-aware propagation and pruning rules that enable the solver to prune infeasible branches early, and weight-conflict clauses that integrate weight reasoning into CDCL's clause-learning algorithms. These

17th Pragmatics of SAT International Workshop 2026 (PoS'26)

*Corresponding author.

✉ gs81@rice.edu (G. Spallitta); vardi@rice.edu (M. Vardi)

ORCID 0000-0002-4321-4995 (G. Spallitta); 0000-0002-0661-5773 (M. Vardi)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

mechanisms reshape the solver’s behavior, enabling both threshold-based and top- k enumeration to be performed natively.

A central issue, then, is how weight conflicts are resolved. In classical SAT solving, non-chronological backtracking and state-of-the-art CDCL techniques are considered indispensable for efficiently scanning the search space [24], and their benefits have been discussed for enumeration [11]. Yet their advantages may not carry over when weight thresholds guide the search. Conversely, chronological backtracking, recently shown to be effective for model enumeration [14, 25], could align more naturally with WME. This duality raises the question of which backtracking approach is better suited for weight-aware enumeration, motivating the development and empirical comparison of both chronological and non-chronological CDCL frameworks for WME.

1.1. Contributions

- We formally introduce *Weighted Model Enumeration*, a generalization of classical reasoning problems, such as AllSAT, WMC, and MaxSAT, supporting the enumeration of weighted satisfying assignments, possibly under weight-based criteria.
- We introduce weight-aware propagation and pruning within CDCL-based enumeration, enabling early detection of infeasible branches.
- We develop two CDCL frameworks for WME: a *chronological* variant (implicit blocking clauses, low memory footprint, order-sensitive) and a *non-chronological* variant (explicit blocking clauses, restart-friendly).
- We implement both designs and study their behavior across thresholds and weight distributions, highlighting regimes where one approach outperforms the other.

1.2. Related Work

AllSAT solvers such as `clasp` [13], the set of tools BC, NBC and BDD, all three presented in [11], and, more recently, AllSATCC [12] and TabularAllSAT [14, 25] enumerate *all* satisfying assignments without regard to weight. Introducing weighted and filtered results based on weight constraints must be done post hoc, which is computationally prohibitive in large solution spaces.

WMC tools such as Ganak [26, 27], D4 [17], Cachet [15], miniC2D [16], ADDMC [28], DPMC [18], sharpSAT-TD [29] and algebraic extensions [19] compute aggregate weights over all models. These are central in probabilistic inference but do not enumerate or rank individual assignments.

MaxSAT solvers, including MaxHS [20], Open-WBO [30], MPE-SAT [6], and the newer Open-WBO-Inc [22] and TT-Open-WBO-Inc [21], optimize for a maximum-weight assignment, typically by combining SAT reasoning with core-guided, hitting-set, or branch-and-bound optimization techniques. In particular, branch-and-bound MaxSAT solvers such as MaxCDCL combine clause learning with lower-bound reasoning over soft clauses [31]. Their soft-conflict analysis is closely related in spirit to our weight-conflict analysis: both mechanisms derive clauses from the observation that a partial assignment cannot yield a sufficiently good solution. There is also a close connection to weight-aware branching and polarity-selection heuristics for MaxSAT. For instance, the anytime weighted MaxSAT approach WMB introduces polarity and variable-selection heuristics designed to quickly find high-quality assignments [32]. Although these works heavily inspire the ideas developed in this manuscript, we emphasize that retrieving multiple high-quality models is not a native operation for MaxSAT solvers; it is typically simulated via repeated MaxSAT calls, which can scale poorly in practice. The MaxSAT Evaluation’s experimental top- k track [23] further highlights that ranked extraction is not a standard MaxSAT objective, as it primarily evaluates correctness (i.e., number and ordering of solutions, possibly with approximation) rather than efficiency. Leading approaches (e.g., RC2 [33] and MaxHS [20]) explicitly rely on iterative re-optimization with blocking clauses rather than integrated enumeration and often limit preprocessing, which hinders scalability when extracting multiple solutions.

An alternative approach to target WME is to compile the formula into a tractable representation from knowledge compilation [34] (e.g., d-DNNF and related decision-diagram formalisms) and then enumerate

solutions in ranked order. Recent work started investigating the idea of ranked enumeration on circuits under restricted scoring schemes [35, 36]. This direction could be effective when compilation is feasible and results in a compact representation. Our focus in this manuscript, however, is different: we target anytime, solver-native top- k /threshold enumeration directly within CDCL-style search. This is already a natural fit in the propositional setting, since the weight bounds can be maintained online during the search and used immediately for pruning. The choice is also motivated by how naturally it carries over to SMT. In that setting, a purely propositional compiled representation is insufficient: an assignment may satisfy the Boolean abstraction yet still be theory-inconsistent. Therefore, a compilation-based approach would need to account not only for propositional structure but also for the effect of theory reasoning and theory-pruned branches, which is technically nontrivial [37]. By contrast, CDCL($\mathcal{T}$) [38] already combines Boolean search with online theory consistency checking, making the integration of weight reasoning into the search loop more straightforward.

At last, a related line of work is the dynamic-programming optimization tool DPO [39]. DPO targets optimization on hybrid constraint systems (including XORs) by combining SAT reasoning with dynamic programming over the input problem structure. While DPO supports computing an optimal high-weight model, its optimization framework does not allow a trivial modification to enumerate multiple high-weight satisfying models.

2. Weighted Model Enumeration

Let F be a CNF Boolean formula over variables $\mathcal{V} = \{A_1, \dots, A_n\}$. A total assignment $\eta : \mathcal{V} \rightarrow \{\perp, \top\}$ satisfies F if all clauses of F evaluate to true under η . A partial assignment μ assigns only a subset of the variables in $\mathcal{V}$. We represent assignments as sets of literals, e.g. $\mu = \{A_1, \neg A_2, A_3\}$, or as conjunctions of literals, e.g. $\mu = (A_1 \wedge \neg A_2 \wedge A_3)$, using the two representations interchangeably. In the CDCL loop, μ denotes the current trail, i.e., an ordered partial assignment partitioned into decision levels. Let ε denote the empty trail. We define a *weight function* $w : \mathcal{L} \rightarrow \mathbb{Q}_{>0}$ over the literal set $\mathcal{L} = \{A_i, \neg A_i \mid A_i \in \mathcal{V}\}$. Finally, let $\text{vars}(\mu) = \{\text{var}(\ell) \mid \ell \in \mu\}$, where $\text{var}(\ell)$ denotes the Boolean variable underlying literal ℓ .

Definition 2.1 (Weighted Model Enumeration). Given a Boolean formula F over variables $\mathcal{V}$ and a literal-level weight function $w : \mathcal{L} \rightarrow \mathbb{Q}_{>0}$, the *Weighted Model Enumeration (WME)* problem asks to compute the set of weighted satisfying assignments:

$$\mathcal{M}_w(F) = \{(\eta, w(\eta)) \mid \eta \models F\} \quad \text{where} \quad w(\eta) = \prod_{\ell \in \eta} w(\ell).$$

The goal is to enumerate every model η satisfying F together with its aggregated weight $w(\eta)$. The weighted model enumeration problem generalizes:

- **ALLSAT**: when $w(\ell) = 1$ for all ℓ , we get the unweighted set of satisfying assignments;
- **WMC**: when only the scalar quantity $\sum_{\eta \models F} w(\eta)$ is returned, rather than the full set of weighted models.

To illustrate the WME task, consider a simple weighted CNF instance over Boolean variables A_1 , A_2 , and A_3 :

$F = (A_1 \vee A_2) \wedge A_3$		A_1	A_2	A_3
	$w(\ell)$	0.6	0.8	0.5
	$w(\neg \ell)$	0.4	0.2	0.5

Table 1 lists all 8 possible assignments, indicating which ones satisfy F , and reports their corresponding weights for those that do.

In addition to the baseline WME task, we define two constrained variants:

A_1	A_2	A_3	Sat. F	Weight		A_1	A_2	A_3	Sat. F	Weight
0	0	0	×	–		1	0	0	×	–
0	0	1	×	–		1	0	1	✓	$0.6 \cdot 0.2 \cdot 0.5 = 0.06$
0	1	0	×	–		1	1	0	×	–
0	1	1	✓	$0.4 \cdot 0.8 \cdot 0.5 = 0.16$		1	1	1	✓	$0.6 \cdot 0.8 \cdot 0.5 = 0.24$

Table 1

All possible assignments of $F = (A_1 \vee A_2) \wedge A_3$ and their associated weights.

Feature	AllSAT	WMC	MaxSAT	WME
Model enumeration	✓	×	×	✓
Weight support	×	✓	✓	✓
Top- k /threshold enumeration	×	×	$\sim \sqrt{2}$	✓

Table 2

Comparison of WME against similar SAT-based paradigms.

- **Threshold WME:** Given a threshold θ , enumerate all models $\eta \models F$ such that $w(\eta) \geq \theta$ ¹.
- **Top- k WME:** Enumerate the k models with the highest weights among all models in F .

These two modes are general enough to express a wide range of practical requirements, including top- k explanation, utility-based model extraction, and probabilistic inference.

Remark 2.2. Throughout this work, we adopt the multiplicative aggregation of weights,

$$w(\eta) = \prod_{\ell \in \eta} w(\ell),$$

as commonly used in Weighted Model Counting and probabilistic reasoning. While the examples in this paper use weights in $(0, 1]$, in line with the probabilistic intuition common in WMC, the upper-bound reasoning developed later applies over the entire positive range. For numerical stability, one may equivalently work in the log domain by replacing products with sums. This is often preferable in floating-point implementations, as repeated multiplication of many positive weights can easily cause underflow, overflow, or loss of precision.

2.1. Comparison to Existing Approaches

Table 2 positions Weighted Model Enumeration with respect to the main SAT-based paradigms it relates to, namely AllSAT, Weighted Model Counting, and MaxSAT. While these approaches share individual aspects with WME (such as model enumeration, weight handling, or optimization objectives), none of them combine these features within a single solver framework.

AllSAT [11, 12, 25] provides complete model enumeration but lacks any notion of weights or ranking, making selective exploration infeasible without exhaustive enumeration. WMC [28, 18, 19] supports weights but collapses all models into a single aggregate quantity, offering no access to individual solutions. MaxSAT [30, 21] optimizes for a single best model and does not support ranked or threshold-based enumeration without repeated solver invocations.

In contrast, WME integrates model enumeration, weight-aware search, and weight-based pruning directly into a CDCL-style solver. This enables native support for top- k and threshold-based enumeration with incremental output, allowing entire regions of the search space to be pruned based on weight bounds when needed.

¹We focus on lower-bound thresholds $w(\eta) \geq \theta$ in this paper. Other relational operators (e.g., $>$, $\leq$, $<$) can be handled analogously by reversing the comparison and/or using corresponding upper/lower residual bounds.

²Some MaxSAT extensions can retrieve multiple near-optimal models via repeated invocations, but this remains an indirect use of MaxSAT rather than a native ranked-enumeration operation.

3. Integrating Weight Reasoning into CDCL-Based Frameworks

Weighted Model Enumeration extends standard CDCL-based enumeration by incorporating Boolean search with weight-based reasoning and threshold constraints, while preserving the overall structure of the CDCL loop. In particular, Boolean constraint propagation and Boolean clause learning remain unchanged.

The key extension is the integration of *weight-based conflict analysis* into the CDCL loop. In addition to detecting logical conflicts, the solver continuously evaluates whether the current partial assignment can still satisfy an active weight bound θ , which may represent either a fixed threshold or a dynamically updated cutoff (e.g., the current k -th best score in top- k enumeration). If the maximum achievable weight of a branch falls below θ , the branch is declared infeasible, and a weight conflict is analyzed, resulting in a learned clause that prunes the corresponding region of the search space. Conceptually, this mechanism plays a role analogous to theory-conflict detection in lazy SMT solvers [40], but operates over simple weight bounds rather than complex theory constraints.

3.1. Weight Propagation and Pruning

WME extends CDCL search with a numerical dimension: besides logical consistency, the solver must track whether the current partial assignment can still be extended to a model whose weight reaches the active threshold θ . The goal is to detect as early as possible when this is no longer feasible. This process, called *weight-based pruning*, complements standard Boolean propagation and provides the condition that triggers weight conflicts. Each literal ℓ is associated with a real-valued weight $w(\ell)$. For every Boolean variable A_i , we define:

$$best(A_i) = \max\{w(A_i), w(\neg A_i)\}$$

For each partial assignment μ , the solver maintains two quantities: the partial weight $w(\mu)$ on the assigned variables, and an optimistic residual bound $I_{\max}(\mu)$ on the unassigned variables:

$$I_{\max}(\mu) := \prod_{A \in \mathcal{V} \setminus \text{vars}(\mu)} best(A).$$

Algorithm 1 maintains these quantities incrementally. At initialization (lines 1–2), $w(\mu)$ is set to 1, and $I_{\max}(\mu)$ is the product of all $best(A_i)$, corresponding to the assumption that unassigned variables will take their highest-weight polarity. When assigning a literal ℓ (lines 3–6), the solver multiplies $w(\mu)$ by $w(\ell)$ and removes the corresponding $best(A_i)$ contribution from $I_{\max}(\mu)$. Unassigning a literal (lines 7–10) reverses both updates. Both operations preserve the invariant

$$\forall \eta \supseteq \mu : w(\eta) \leq w(\mu) \cdot I_{\max}(\mu).$$

That is, $w(\mu) \cdot I_{\max}(\mu)$ is an upper bound on the weight achievable by any completion of μ . For brevity, we write $best(\ell)$ to denote $best(\text{var}(\ell))$.

The pruning test (lines 11–14) is invoked after Boolean propagation. If $w(\mu) \cdot I_{\max}(\mu) < \theta$, even an ideal extension cannot reach the lower bound. In this case, we prune the branch, and a weight conflict can be reported³.

Proposition 3.1 (Weight pruning soundness). *Let μ be any partial trail reachable by the main CDCL loop, and let $w(\mu)$ and $I_{\max}(\mu)$ be maintained as in Algorithm 1. Then:*

$$\forall \eta \supseteq \mu : w(\eta) \leq w(\mu) \cdot I_{\max}(\mu).$$

*In particular, if $WEIGHTCONFLICT(\mu)$ returns **true**, i.e., $w(\mu) \cdot I_{\max}(\mu) < \theta$, then no completion $\eta \supseteq \mu$ can satisfy $w(\eta) \geq \theta$.*

³A symmetric test based on the pessimistic bound $I_{\min}$ could prune branches that exceed an upper bound, though, for clarity, only the lower-bound variant is shown here.

Algorithm 1 WEIGHT-STATE MAINTENANCE AND PRUNING

Require: Current assignment μ

- 1: $w(\mu) \leftarrow 1$ $\triangleright$ partial assignment weight
 - 2: $I_{\max}(\mu) \leftarrow \prod_{A \in \mathcal{V}} \text{best}(A)$ $\triangleright$ optimistic residual

 - 3: **function** ASSIGN(μ, ℓ)
 - 4: $\mu.\text{append}(\ell)$
 - 5: $w(\mu) \leftarrow w(\mu) \cdot w(\ell)$
 - 6: $I_{\max}(\mu) \leftarrow I_{\max}(\mu) / \text{best}(\ell)$

 - 7: **function** UNASSIGN(μ, ℓ)
 - 8: $\mu.\text{remove}(\ell)$
 - 9: $w(\mu) \leftarrow w(\mu) / w(\ell)$
 - 10: $I_{\max}(\mu) \leftarrow I_{\max}(\mu) \cdot \text{best}(\ell)$

 - 11: **function** WEIGHTCONFLICT(μ)
 - 12: **if** $w(\mu) \cdot I_{\max}(\mu) < \theta$ **then**
 - 13: **return true** $\triangleright$ prune branch
 - 14: **return false**
-

PROOF. We prove that $w(\mu) \cdot I_{\max}(\mu)$ is always an upper bound on the weight of any extension of the current trail μ .

Initially $\mu = \varepsilon$, $w(\mu) = 1$, and $I_{\max}(\mu) = \prod_{A \in \mathcal{V}} \text{best}(A)$, so the bound holds trivially. Whenever a literal ℓ of variable A is assigned, Algorithm 1 updates $w(\mu) \leftarrow w(\mu) \cdot w(\ell)$ and $I_{\max}(\mu) \leftarrow I_{\max}(\mu) / \text{best}(\ell)$, thus maintaining:

$$w(\mu) = \prod_{\ell \in \mu} w(\ell) \quad \text{and} \quad I_{\max}(\mu) = \prod_{B \in \mathcal{V} \setminus \text{vars}(\mu)} \text{best}(B).$$

Let $\eta \supseteq \mu$ be any completion. For every unassigned variable B , the literal ℓ_B chosen in η satisfies $w(\ell_B) \leq \text{best}(B)$ by definition. Thus:

$$w(\eta) = \prod_{\ell \in \mu} w(\ell) \cdot \prod_{B \in \mathcal{V} \setminus \text{vars}(\mu)} w(\ell_B) \leq \prod_{\ell \in \mu} w(\ell) \cdot \prod_{B \in \mathcal{V} \setminus \text{vars}(\mu)} \text{best}(B) = w(\mu) \cdot I_{\max}(\mu).$$

Hence $w(\mu) \cdot I_{\max}(\mu)$ is a valid upper bound for all extensions of μ , and if $w(\mu) \cdot I_{\max}(\mu) < \theta$ then $w(\eta) < \theta$ for every $\eta \supseteq \mu$, establishing the correctness of pruning.

Example 3.2. Consider the following weighted instance, where ϕ is a subformula over the set of variables $B_1, \dots, B_n$:

$$F = (A_1 \vee \phi) \wedge A_2.$$

	A_1	A_2	$B_i \in \phi$
$w(\ell)$	0.6	0.3	$\beta_i \in (0, 1)$
$w(\neg \ell)$	0.4	0.7	$1 - \beta_i$

Suppose we enforce a lower-bound threshold $\theta = 0.2$. Initially,

$$I_{\max}(\mu) = \max(0.4, 0.6) \cdot \max(0.7, 0.3) \cdot \underbrace{\prod_{i=1}^n \text{best}(B_i)}_{\alpha} = 0.42 \alpha. \quad (1)$$

Algorithm 2 ANALYZEWEIGHTCONFLICT(μ)

Require: Current trail μ ; threshold θ

- 1: $S \leftarrow \text{GREEDYCONFLICTSET}(\mu, \theta)$
 - 2: $C_w \leftarrow \bigvee_{\ell \in S} \neg \ell$
 - 3: $\text{LEARN}(C_w)$ $\triangleright$ add learned weight-conflict clause
 - 4: $\text{BACKJUMP}(C_w)$ $\triangleright$ backjump per CDCL policy
-

Notice that due to the weight definition, α is guaranteed to be in the range $(0, 1)$. Now suppose the trail is $\mu = \{\neg A_1, A_2\}$. Then $w(\mu) = 0.4 \cdot 0.3 = 0.12$. We divide $\text{best}(A_1)$ and $\text{best}(A_2)$ from $I_{\max}(\mu)$, leaving the residual:

$$I_{\max}(\mu) = \frac{I_{\max}(\mu)}{\text{best}(A_1) \cdot \text{best}(A_2)} = \frac{0.6 \cdot 0.7 \cdot \alpha}{0.6 \cdot 0.7} = \alpha$$

. Hence, for every extension of the partial assignment (including the optimistic completion):

$$w(\mu) \cdot I_{\max}(\mu) = 0.12 \cdot \alpha \leq 0.12 < \theta = 0.2,$$

Therefore, this branch can be pruned early, avoiding the assignment of any variables in ϕ .

3.2. Weight Conflict Analysis

The previous section defines how a partial trail maintains an upper bound on the achievable model weight and how branches that cannot reach the target threshold θ are pruned. When such pruning occurs, we interpret it as a *weight conflict*, analogous to a Boolean conflict in standard CDCL. The goal of weight-conflict analysis is to extract a short explanation from the current trail and learn a clause that prevents revisiting the same infeasible region.

Definition 3.3 (Weight conflict). Given a CNF formula F , a threshold θ , and a partial trail μ , we say that μ induces a **weight conflict** if every total extension $\eta \supseteq \mu$ fails to reach the threshold, i.e.,

$$\forall \eta \supseteq \mu : w(\eta) < \theta.$$

By Proposition 3.1, $w(\mu) \cdot I_{\max}(\mu) < \theta$ provides an efficiently checkable sufficient condition: if even the optimistic upper bound on any extension of μ is below θ , the branch can be pruned.

When a weight conflict is detected (line 11 of Algorithm 1), the solver triggers weight-conflict analysis (Algorithm 2). The procedure first extracts a *weight-conflict set* $S \subseteq \mu$ and then learns the corresponding *weight-conflict clause*

$$C_w := \bigvee_{\ell \in S} \neg \ell,$$

which blocks the simultaneous reoccurrence of all literals in S .

The extraction of the weight conflict set is shown in Algorithm 3. We use a greedy extraction strategy because it is easy to integrate, incurs little overhead, and provides a practical trade-off between clause compactness and extraction cost. Literals in the current trail μ are sorted by ascending weight $w(\ell)$ (line 4). The procedure then greedily builds a set $S \subseteq \mu$ (lines 5–10), while maintaining (i) the partial weight $w(S) = \prod_{\ell \in S} w(\ell)$ and (ii) an optimistic residual bound $I_{\max}(S)$ for the variables not yet fixed by S . As soon as $w(S) \cdot I_{\max}(S) < \theta$ (line 9), the set S is used to generate C_w .

While stronger conflict-set extraction procedures could be considered, e.g., by minimizing the learned weight-conflict clause, we use a lightweight greedy strategy to keep the extraction cost negligible within the CDCL loop, following prior work that prioritizes low-overhead reasoning during search [14]. More compact or cost-aware weight-conflict explanations are orthogonal to our framework and left as future work.

Algorithm 3 GREEDYCONFLICTSET(μ, θ)

Require: Current trail μ ; threshold θ

```
1:  $S \leftarrow \emptyset$ 
2:  $w(S) \leftarrow 1$ 
3:  $I_{\max}(S) \leftarrow \prod_{A \in \mathcal{V}} \text{best}(A)$ 
4:  $L \leftarrow$  literals of  $\mu$  sorted by ascending  $w(\ell)$ 
5: for each  $\ell \in L$  do
6:    $S \leftarrow S \cup \{\ell\}$ 
7:    $w(S) \leftarrow w(S) \cdot w(\ell)$ 
8:    $I_{\max}(S) \leftarrow I_{\max}(S) / \text{best}(\ell)$ 
9:   if  $w(S) \cdot I_{\max}(S) < \theta$  then
10:     return  $S$ 
11: return  $S$ 
```

▷ All literals in μ needed

Proposition 3.4 (Soundness of greedy weight-conflict learning). *Let μ be a trail such that WEIGHTCONFLICT(μ) holds. Let $S \subseteq \mu$ be the set returned by GREEDYCONFLICTSET(μ, θ), and let C_w be the learned weight-conflict clause. Assume that when the procedure returns S it satisfies $w(S) \cdot I_{\max}(S) < \theta$, where $w(S) = \prod_{\ell \in S} w(\ell)$ and $I_{\max}(S) = \prod_{A \in \mathcal{V} \setminus \text{vars}(S)} \text{best}(A)$ are maintained as in Algorithm 3. Then every total model η of F with $w(\eta) \geq \theta$ satisfies C_w . Consequently, learning C_w cannot eliminate any model of weight at least θ .*

PROOF. Let S be the set returned by GREEDYCONFLICTSET. By assumption, at return time the procedure has $w(S) \cdot I_{\max}(S) < \theta$.

We first show that this implies that every completion of S is below the threshold. Let $\eta \supseteq S$ be any total assignment containing all literals in S . For each variable $A \notin \text{vars}(S)$, let ℓ_A be the literal on A chosen by η . By definition of *best* we have $w(\ell_A) \leq \text{best}(A)$, hence

$$w(\eta) = \left(\prod_{\ell \in S} w(\ell) \right) \cdot \left(\prod_{A \in \mathcal{V} \setminus \text{vars}(S)} w(\ell_A) \right) \leq w(S) \cdot \left(\prod_{A \in \mathcal{V} \setminus \text{vars}(S)} \text{best}(A) \right) = w(S) \cdot I_{\max}(S) < \theta.$$

Therefore:

$$\forall \eta \supseteq S : w(\eta) < \theta. \quad (2)$$

Now consider any total assignment η of F that falsifies $C_w = \bigvee_{\ell \in S} \neg \ell$. Falsifying C_w means ℓ is true in η for all $\ell \in S$, hence $S \subseteq \eta$. Applying Equation 2 returns $w(\eta) < \theta$.

Thus, no assignment of weight at least θ can falsify C_w . Equivalently, every model η with $w(\eta) \geq \theta$ satisfies C_w , so learning C_w cannot eliminate any model of weight at least θ .

Example 3.5. Consider the following weighted CNF instance:

$F = (A_1 \vee A_2 \vee \neg A_3).$		A_1	A_2	A_3
	$w(\ell)$	0.6	0.8	0.7
	$w(\neg \ell)$	0.4	0.2	0.3

Suppose the lower-bound threshold is $\theta = 0.2$ and the current trail is $\mu = \{\neg A_1, A_2, \neg A_3\}$. The partial weight is $w(\mu) = 0.4 \cdot 0.8 \cdot 0.3 = 0.096$. Since $w(\mu) < \theta$, this branch is pruned. The greedy strategy identifies the literal with the lowest contribution (here, $\neg A_3$) as the main cause of the underweight condition (since $w(\neg A_3) \cdot \text{best}(A_1) \cdot \text{best}(A_2) = 0.144 < 0.2$), returning the weight conflict set $S = \{\neg A_3\}$. The corresponding weight conflict blocking clause $C_w = A_3$ is then learned.

Algorithm 4 RESIDUAL-AWARE BACKTRACK(μ, θ)

Require: Current trail μ ; maintained $w(\mu), I_{\max}(\mu)$; threshold θ

```
1: while  $\mu$  contains a literal at decision level  $> 0$  do
2:    $\ell \leftarrow \mu.\text{pop}()$ 
3:    $w(\mu) \leftarrow w(\mu)/w(\ell)$ 
4:    $I_{\max}(\mu) \leftarrow I_{\max}(\mu) \cdot \text{best}(\ell)$ 
5:   if  $w(\mu) \cdot I_{\max}(\mu) > \theta$  then
6:     return  $\mu$   $\triangleright$  resume search from this trail and decision level
7: return  $\mu$   $\triangleright$  return the decision level-0 trail and stop enumeration
```

3.3. Residual-Aware Backtracking for Top- k Enumeration

In top- k enumeration, whenever we find a satisfying assignment whose weight enters the current top- k , we tighten the active threshold θ accordingly. After such an update, continuing the search from the last decision level is often pointless: the current trail μ may already be so constrained that, even under the most optimistic completion of the remaining variables, no model strictly above θ can be reached.

Algorithm 4 uses the same optimistic bound from Algorithm 1 to determine how far to backtrack immediately after updating θ . Starting from the current trail μ , the procedure repeatedly removes the most recent literal, updating $w(\mu)$ and $I_{\max}(\mu)$ according to the incremental rules of Algorithm 1. After each removal, the quantity $w(\mu) \cdot I_{\max}(\mu)$ remains an upper bound on the weight of any completion of the current trail. Any trail satisfying $w(\mu) \cdot I_{\max}(\mu) \leq \theta$ cannot lead to an improved model, so we continue backtracking.

The procedure stops as soon as $w(\mu) \cdot I_{\max}(\mu) > \theta$, because some completion may still improve on the current cutoff. Thus, the solver can resume the search from the corresponding decision level by exploring the alternative branch. If the procedure backtracks to decision level 0 and the bound is still not above θ , then no model of weight greater than θ exists, and top- k enumeration can terminate.

Example 3.6. Consider the following weighted CNF instance:

$$F = (A_1 \vee A_2) \wedge A_3.$$

	A_1	A_2	A_3
$w(\ell)$	0.6	0.8	0.7
$w(\neg\ell)$	0.4	0.2	0.3

The assignment $\eta = \{\neg A_1, A_2, A_3\}$ satisfies F with $w(\eta) = 0.4 \cdot 0.8 \cdot 0.7 = 0.224$. We assume the CDCL trail has the same order. Assuming we are looking for the top-1 solution, the threshold θ is updated accordingly.

Removing A_3 with $\neg A_1, A_2$ fixed results in:

$$(A_3 = \top) : 0.4 \cdot 0.8 \cdot 0.7 = 0.224; \quad (A_3 = \perp) : 0.4 \cdot 0.8 \cdot 0.3 = 0.096.$$

No extension of the partial assignment can be a new top-1 solution, so we can prune A_3 .

Removing A_2 with only $\neg A_1$ fixed results in:

$$\begin{aligned} (A_2 = \top, A_3 = \top) : 0.4 \cdot 0.8 \cdot 0.7 &= 0.224; & (A_2 = \perp, A_3 = \top) : 0.4 \cdot 0.2 \cdot 0.7 &= 0.056; \\ (A_2 = \top, A_3 = \perp) : 0.4 \cdot 0.8 \cdot 0.3 &= 0.096; & (A_2 = \perp, A_3 = \perp) : 0.4 \cdot 0.2 \cdot 0.3 &= 0.024. \end{aligned}$$

None is above θ , so A_2 can be safely removed. At this point, the remaining trail is $\mu = \{\neg A_1\}$. Its optimistic residual bound is $I_{\max}(\mu) = 0.6 \cdot 0.8 \cdot 0.7 = 0.336$, which is $> \theta$, so there is a chance that an assignment with a higher score satisfying F does exist. The solver backtracks and flips A_1 .

3.4. Weight-based Variable Priority for Top- k Enumeration

Many real-world problems are not naturally given in CNF and are first encoded using structure-preserving encodings such as Tseitin [41] or Plaisted–Greenbaum [42]. The newly introduced variables in these encodings (and, more generally, any variable whose two polarities contribute the same weight) are often *weight-irrelevant*: they do not affect model ranking under a weight-based threshold.

Let $\mathcal{V}$ be the full variable set. We partition $\mathcal{V}$ into *weight-relevant* variables W_r and *weight-irrelevant* variables W_i . Accordingly, any assignment η can be decomposed as $\eta = \eta_r \cup \eta_i$ with η_r over W_r and η_i over W_i . The contribution of W_i is constant, denoted θ_i , so we maintain a lower-bound threshold θ_r on the restricted score over W_r only. The branching heuristic prioritizes variables in W_r so that weight-relevant variables are assigned first.

For any assignment η_r over W_r , the assignment weight is defined such that:

$$\forall \eta_i \text{ over } W_i : \quad w(\eta_r \wedge \eta_i) = w(\eta_r) \cdot \theta_i$$

Once all variables in W_r have been assigned, if $w(\eta_r) < \theta_r$, then no extension over W_i can reach the current threshold. During partial search over W_r , the same reasoning applies using an optimistic residual bound over the still-unassigned variables in W_r .

If all variables in W_r are assigned and $w(\eta_r) \geq \theta_r$, the solver checks extendability to a total satisfying assignment including variables in W_i . When this succeeds, θ_r is updated accordingly, and residual-aware backtracking is applied with respect to the last decision on a variable in W_r ; assignments on W_i made afterward cannot improve the score. Conversely, if the optimistic bound over the remaining variables in W_r drops below θ_r , then variables in W_i are skipped, and weight-based conflict analysis is triggered.

Example 3.7. Consider the following weighted CNF instance:

$$F = (A_1 \vee A_2) \wedge A_3 \wedge (\neg Z_1 \vee A_1) \wedge (\neg Z_2 \vee A_2),$$

$$W_r = \{A_1, A_2, A_3\}, \quad W_i = \{Z_1, Z_2\}.$$

	A_1	A_2	A_3	Z_1	Z_2
$w(\ell)$	0.9	0.8	0.7	0.5	0.5
$w(\neg \ell)$	0.1	0.2	0.3	0.5	0.5

Since both polarities of each Z_j carry the same weight, every assignment on W_i contributes the same factor $\theta_i = 0.25$, so W_i is weight-irrelevant. We maintain a lower-bound threshold θ_r over W_r and prioritize decisions on W_r . Assume $\theta_r = 0.1$.

Case 1: Suppose the trail on W_r is $\eta_r = \{A_1, \neg A_2, A_3\}$. Then

$$w(\eta_r) = 0.9 \cdot 0.2 \cdot 0.7 = 0.126.$$

Since $w(\eta_r) \geq \theta_r$, we check extendability to a total model satisfying F (e.g., $Z_1 = Z_2 = \perp$ works). Any completion over W_i preserves the relative ranking of two assignments over W_r , so if $w(\mu_{1_r}) > w(\mu_{2_r})$ then

$$w(\eta_1) = w(\eta_{1_r}) \cdot \theta_i > w(\eta_{2_r}) \cdot \theta_i = w(\eta_2).$$

After validating satisfiability, θ_r is updated with the full score and residual-aware backtracking resumes from the last decision in W_r ; assignments to (Z_1, Z_2) are irrelevant for the score.

Case 2: Suppose instead we reach $\eta'_r = \{\neg A_1, \neg A_2, A_3\}$, giving

$$w(\eta'_r) = 0.1 \cdot 0.2 \cdot 0.7 = 0.014.$$

Since $w(\eta'_r) < \theta_r$, the solver can skip assigning Z_1, Z_2 and trigger a weight conflict.

In the top- k setting, two total assignments may share the same total weight such that: (i) they have the same assignment on W_r , and (ii) different assignments on W_i . To avoid pruning such tie-equivalent solutions too early, we activate the optimization only after the first k satisfying assignments have been collected. From that point onward, only assignments whose best achievable score is strictly better than the current k -th score can affect the maintained top- k set.

The partition of variables into relevant variables W_r and irrelevant variables W_i is analogous to the distinction used in projected model enumeration: W_r plays the role of the projected variable set, while W_i contains variables that are existentially completed. Here, however, the projection is used only for weight reasoning. Variables in W_i may still be necessary to satisfy the formula, but they do not affect the ranking of assignments once an extension exists.

4. Chronological vs. Non-Chronological Backtracking

Integrating weight reasoning into CDCL-based enumeration gives rise to two distinct solver designs, depending on how conflicts (both Boolean and weight-based) are resolved. The first adopts the classical non-chronological clause-learning paradigm of modern SAT solvers [11, 12]. The second combines CDCL with *chronological backtracking* as used in recent AllSAT engines [11, 25]. These two strategies have been explored in the non-weighted setting [14, 25], but weight-based pruning could accentuate their trade-offs in the weighted case. We emphasize that the distinction between chronological and non-chronological backtracking is well established in prior work [11, 43, 14]. Rather than redefining these notions, we focus on the different implementation requirements that arise when they are instantiated in AllSAT-style enumeration, in particular with respect to model blocking and exhaustive traversal.

4.1. Chronological Backtracking Framework

In the chronological framework, conflicts trigger chronological backtracking, but the solvers still incorporate CDCL-style conflict analysis [43, 14]. To maintain chronological traversal, analysis is carried out up to the last unique implication point (UIP), instead of performing the usual first-UIP backjumping [14]. Blocking previously explored models is *implicit*: once a branch is closed, the chronological traversal ensures it will not be revisited [43].

The main advantage of this approach is its low memory footprint and fast propagation. Nevertheless, the performance of this design depends heavily on decision order. Early poor branching choices can delay the discovery of high-weight models, reducing the effectiveness of pruning. Restarts would break the implicit blocking guarantee, so they must be disabled [14], which further increases sensitivity to branching heuristics. In practice, this variant performs best when the solution space is dense, such that implicit blocking remains efficient.

4.2. Non-Chronological Backtracking Framework

In the non-chronological framework, the solver backjumps to the first UIP rather than the last, thereby returning shorter clauses and enabling more effective backjumping [44]. Model blocking is also *explicit*: whenever a complete model is found, a clause consisting of the negation of all its decision literals is added to exclude it in subsequent search. These blocking clauses persist across restarts, enabling the solver to reset the trail while retaining information about previously explored or infeasible regions.

This approach is more robust to decision order, since restarts allow the solver to dynamically reorganize the search. Even if early decisions move the search away from promising regions, accumulated blocking and weight-conflict clauses guide the search toward high-weight models over time. The trade-off is that propagation becomes increasingly expensive as clauses accumulate, and memory usage may quickly increase.

5. Experimental Evaluation

We empirically evaluate the proposed WME framework on both synthetic and real-world benchmarks. Two tasks are considered: *top-k* and *threshold-based* enumeration. All experiments were conducted on a high-performance computing cluster. Each run was bound to a single core of an Intel Xeon

E5-2650 v2 @ 2.60GHz with 25 GB RAM. We implemented two CDCL-based WME variants, available at <https://github.com/giuspek/WME>, atop the same CaDiCaL [45] codebase⁴:

- WME-NCB: Weighted model enumeration with *non-chronological backtracking*;
- WME-CB: Weighted model enumeration with *chronological backtracking*.

For the top- k experiments, we also compare against MaxSAT-based enumeration baselines, namely the top- k versions of RC2 and MaxHS discussed in the 2020 competition. We use the latest available versions of both tools, with the configurations described for the MaxSAT top- k setting. For RC2, we use its public `enumerate()` interface. For MaxHS, the top- k variant was released for the 2020 MaxSAT Evaluation. However, the currently available distribution does not provide a documented way to invoke this mode. In particular, the accompanying helper script does not expose usable invocation information through its help interface, preventing a reliable reproduction of the competition-specific wrapper. We therefore reconstructed the procedure according to the competition description: after each optimum is found, we add a blocking clause for the returned solution and re-optimize under the accumulated blocking clauses.

5.1. Benchmarks

We evaluate weighted model enumeration on five benchmark families covering random, weighted-counting, and MaxSAT-derived instances. All instances are provided in the WME codebase.

The first family, `rnd3sat-1.5`, consists of 525 random 3-CNF formulae with 25 to 45 variables and clause-to-variable ratio 1.5. This low-density regime is commonly used in AllSAT evaluations because it yields many satisfying assignments while remaining non-trivial for exhaustive enumeration [46, 25]. These instances typically admit many satisfying assignments, making them suitable for evaluating threshold-based enumeration and for isolating the effect of weight-based pruning. The second family, `uf200-860`, contains 100 SATLIB random 3-CNF instances with 200 variables and ratio 4.28, a standard hard SAT regime used here as a stress test for weighted pruning. Literal weights for both random families are sampled from $(0, 1)$.

The third family, `bayes`, consists of 389 literal-weighted CNF instances from [39], derived from Bayesian networks in [47]. These benchmarks are commonly used for weighted model counting and MPE inference, and therefore naturally induce a ranking of satisfying assignments by weight. We also include instances from the 2020 Weighted Model Counting Competition, denoted `wmc-comp`. We use the 2020 set to match the year of the MaxSAT top- k competition and to avoid later benchmark families with more articulated backbone structure, which could disproportionately affect the MaxSAT-based baselines.

Finally, `maxsat-comp` contains a subset of instances from the MaxSAT 2020 top- k competition. Since WME assigns weights to literals, while MaxSAT instances may contain arbitrary soft clauses, not all MaxSAT benchmarks are directly convertible to our setting. We therefore retain only instances whose soft clauses are all unit clauses. Among the 70 available top- k instances, 39 satisfy this condition and are converted into literal-weighted WME instances.

5.2. Top- k Enumeration

We first evaluate the *top-50* task, where the goal is to enumerate the 50 highest-weight models within the time limit (top-50 was one of the settings considered in the MaxSAT top- k competition).

Figure 1 compares WME-NCB against WME-CB, MaxHS, and RC2. Overall, WME-NCB is the most robust WME configuration. Compared with WME-CB, WME-NCB benefits from non-chronological backtracking, explicit blocking clauses, and restarts, which allow the solver to move more aggressively across the search space after promising regions have been explored.

The comparison with MaxHS and RC2 is more nuanced. On several benchmark families, especially `wmc-comp`, `bayes`, and `rnd3sat-1.5`, WME-NCB is competitive and often faster. This supports the main

⁴Weights are represented using arbitrary-precision floating-point numbers in the implementation.

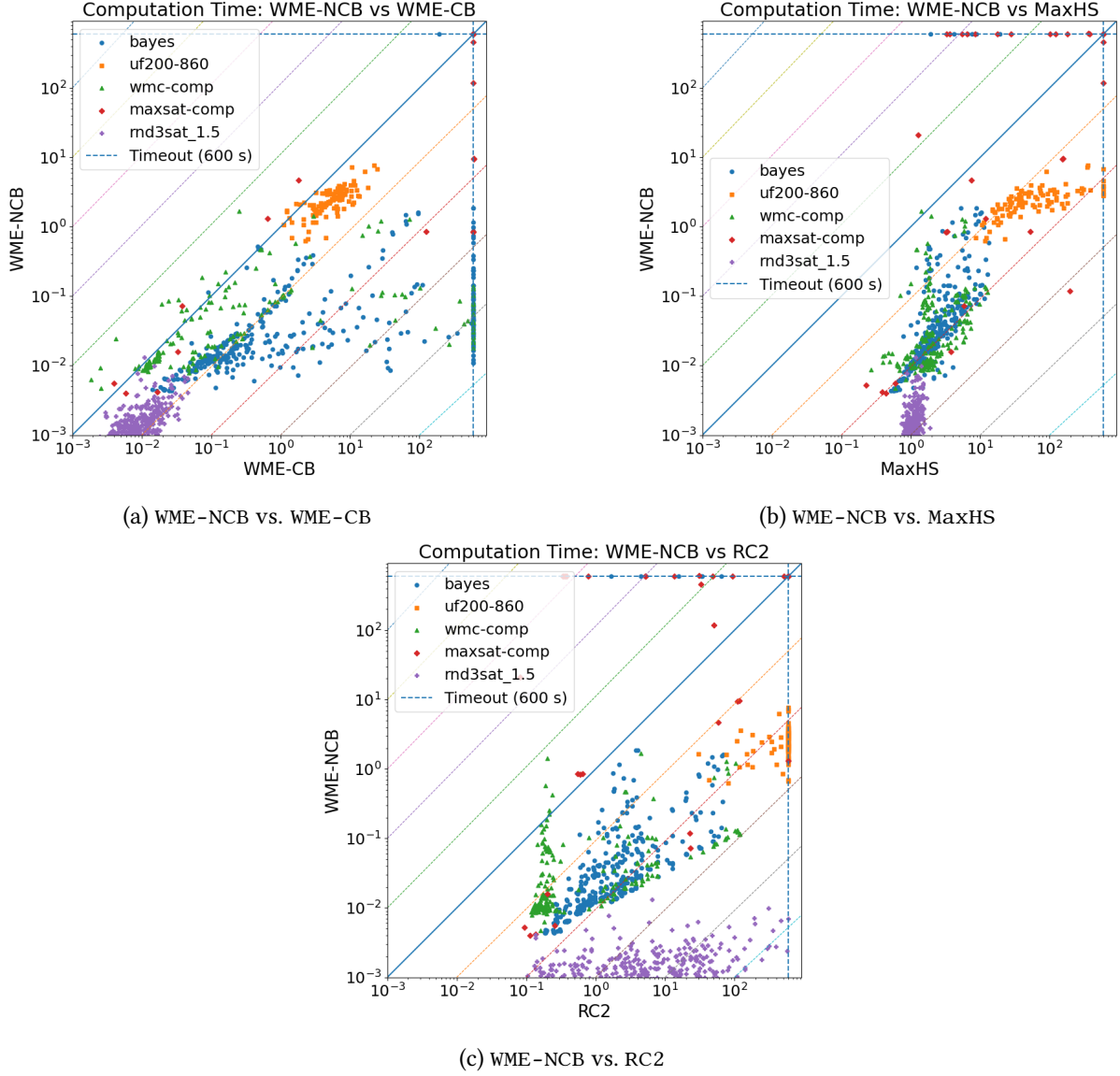


Figure 1: Top-50 enumeration experiments.

motivation of our approach: once multiple high-weight solutions must be produced, integrating weight reasoning directly inside the enumeration procedure can avoid the repeated optimization-and-blocking pattern used by MaxSAT-based approaches.

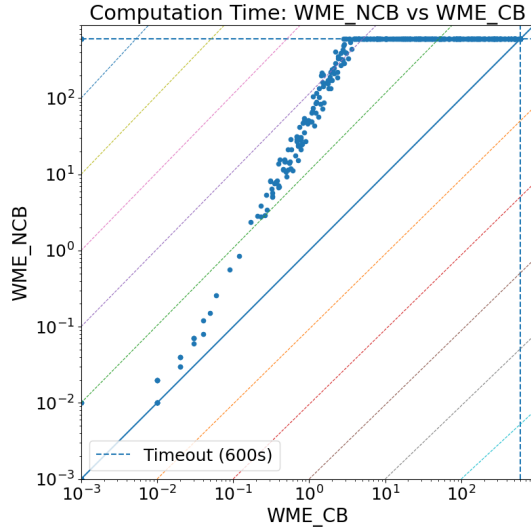
On the maxsat-comp family, however, the picture is mixed. Both MaxHS and RC2 perform very well on many of these instances, whereas WME-NCB exhibits several timeouts. We attribute this behavior to the structure of these MaxSAT-derived benchmarks. In particular, many of the best solutions have very similar, and in some cases identical, weights. In this regime, the top-50 cutoff does not become sufficiently selective: once the current bound reaches a value shared by many assignments, weight-based pruning cannot discard large parts of the remaining search space without risking the loss of tied top- k solutions. By contrast, specialized MaxSAT solvers can exploit their highly optimized core-guided and branch-and-bound machinery on these instances.

Table 3 reports, for each benchmark family, the number of solved instances and the number of instances on which each solver is the fastest. The results further suggest that the effectiveness of top- k weighted enumeration depends not only on the number of requested models, but also on the distribution of model weights: when the weight spectrum is well-separated, the cutoff quickly becomes informative; when many top solutions are tied or nearly tied, pruning is weaker.

Solver	maxsat-comp (of 39)		wmc-comp (of 200)		bayes (of 389)		uf200-860 (of 100)		rnd3sat-1.5 (of 525)	
	Solved	Faster	Solved	Faster	Solved	Faster	Solved	Faster	Solved	Faster
MaxHS	31	4	200	0	389	5	92	0	525	0
RC2	30	10	200	4	389	1	23	0	525	0
WME-CB	19	14	181	19	310	3	100	6	525	1
WME-NCB	21	9	200	177	383	380	100	94	525	524

Table 3

Top-50 enumeration statistics across benchmark families. Notice that 2 problems from maxsat-comp were solved by no solver.



Solver	Solved (out of 525)	PAR2 (s)
WME-NCB	356	457.5
WME-CB	450	267.5

Table 4: Timeout statistics for thresholded WME framework variants on rnd3sat-1.5.

Figure 2: Threshold-based enumeration evaluation among the two WME frameworks on the rnd3sat-1.5.

5.3. Threshold-Based Enumeration

We next consider *threshold-based* enumeration. We focus on the rnd3sat-1.5 benchmarks, which feature a larger solution space and are standard for evaluating enumeration performance [46, 25]. For a formula with n variables, we set the threshold θ to 0.5^n , retaining a non-trivial fraction of solutions above it. Since this setting typically requires enumerating many models, we use a timeout of 600 seconds.

Figure 2 reports the runtime comparison. Unlike in the top- k case, WME-CB outperforms WME-NCB, especially on larger instances with many feasible models. This is expected, as non-chronological search accumulates blocking clauses that increasingly hinder propagation. These results are consistent with prior observations in unweighted enumeration [11, 25].

5.4. Ablation Studies

Since WME integrates weight reasoning directly into the CDCL loop, it is natural to ask whether this integration is essential, or whether similar performance can be obtained by a pure AllSAT-style traversal followed by post hoc weight filtering. To this end, we compare our best-performing configuration with variants in which weight-based pruning is disabled.

For top-50 enumeration, we evaluate WME-NCB on the challenging bayes benchmarks. For threshold-based enumeration, we consider WME-CB on the rnd3sat-1.5 instances. In both cases (Figure 3), enabling weight-based conflict analysis achieves a substantial speedup, confirming that early weight-driven pruning is essential under weight constraints.

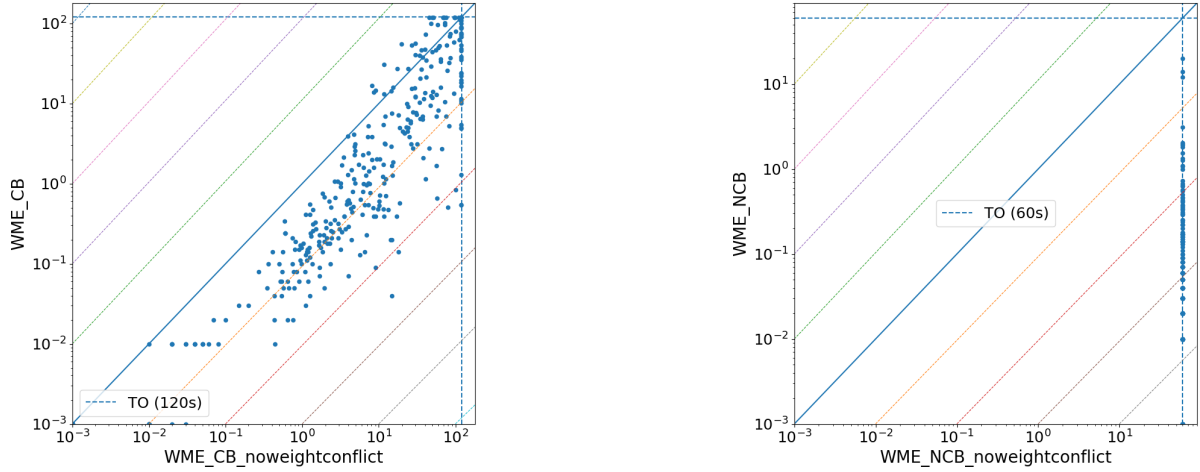


Figure 3: Ablation studies. **Left:** threshold-based enumeration. **Right:** top-50 enumeration.

6. Conclusion

We introduced WME, a CDCL-based framework for enumerating models of a formula F under weight-based criteria. The framework combines weight-conflict analysis for early pruning with residual-aware backtracking, allowing the search to resume only from states that may still lead to improved solutions.

Our empirical results show that the choice between chronological and non-chronological backtracking is regime-dependent. In top- k settings, where the bound typically tightens quickly, non-chronological backtracking with blocking clauses often reaches high-weight solutions faster. In threshold-based enumeration, where many feasible models may need to be generated, chronological backtracking avoids the additional propagation burden of blocking clauses and is often preferable. Overall, the most effective design depends on the interaction between bound tightening, feasible-region density, and propagation overhead.

One natural step is to make the solver adaptive, selecting between chronological and non-chronological reasoning based on structural properties of the instance and of the associated weight distribution. A second direction is to extend WME with projection and SMT reasoning, enabling weighted enumeration over variables of interest and under richer background theories. More broadly, these ideas may also be applied in future CDCL-based Weighted Model Integration algorithms to reduce the enumeration time [48, 49].

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly for grammar and spelling checks. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication’s content.

References

- [1] W. Li, P. Poupart, P. van Beek, Exploiting structure in weighted model counting approaches to probabilistic inference, *Journal of Artificial Intelligence Research* 40 (2011) 729–765. doi:10.1613/jair.3232.
- [2] Q. Liu, A. Ihler, Belief propagation for structured decision making, in: *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, UAI’12*, AUAI Press, Arlington, Virginia, USA, 2012, p. 523–532. <https://dl.acm.org/doi/10.5555/3020652.3020708>.

- [3] B. Cai, L. Huang, M. Xie, Bayesian networks in fault diagnosis, *IEEE Transactions on industrial informatics* 13 (2017) 2227–2240. doi:10.1109/TII.2017.2695583.
- [4] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, C. Zhong, Interpretable machine learning: Fundamental principles and 10 grand challenges, *Statistic Surveys* 16 (2022) 1–85. doi:10.1214/21-ss133.
- [5] D. Tolpin, F. Wood, Maximum a posteriori estimation by search in probabilistic programs, in: *Proceedings of the International Symposium on Combinatorial Search*, volume 6, 2015, pp. 201–205. doi:10.1609/socs.v6i1.18369.
- [6] T. Sang, P. Beame, H. A. Kautz, A Dynamic Approach for MPE and Weighted MAX-SAT., in: *IJCAI*, 2007, pp. 173–179. URL: <https://dl.acm.org/doi/10.5555/1625275.1625302>.
- [7] H. Chan, A. Darwiche, On the robustness of most probable explanations, in: *UAI '06, Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence*, Cambridge, MA, USA, July 13–16, 2006, AUAI Press, 2006. URL: <https://dl.acm.org/doi/10.5555/3020419.3020428>.
- [8] E. L. Lawler, A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem, *Management science* 18 (1972) 401–405. doi:10.1287/mnsc.18.7.401.
- [9] R. Fagin, A. Lotem, M. Naor, Optimal aggregation algorithms for middleware, in: *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2001, pp. 102–113. URL: <https://dl.acm.org/doi/10.1145/375551.375567>.
- [10] C. Yanover, Y. Weiss, Finding the M most probable configurations using loopy belief propagation, *Advances in neural information processing systems* 16 (2003). URL: <https://dl.acm.org/doi/10.5555/2981345.2981382>.
- [11] T. Toda, T. Soh, Implementing efficient all solutions SAT solvers, *Journal of Experimental Algorithmics (JEA)* 21 (2016) 1–44. doi:doi.org/10.1145/2975585.
- [12] J. Liang, F. Ma, J. Zhou, M. Yin, AllSATCC: Boosting AllSAT Solving with Efficient Component Analysis., in: *IJCAI*, 2022, pp. 1866–1872. doi:10.24963/ijcai.2022/259.
- [13] M. Gebser, B. Kaufmann, A. Neumann, T. Schaub, clasp: A conflict-driven answer set solver, in: *International Conference on Logic Programming and Nonmonotonic Reasoning*, Springer, 2007, pp. 260–265. doi:10.1007/978-3-540-72200-7_23.
- [14] G. Spallitta, R. Sebastiani, A. Biere, Disjoint partial enumeration without blocking clauses, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 2024, pp. 8126–8135. doi:10.1609/aaai.v38i8.28652.
- [15] T. Sang, F. Bacchus, P. Beame, H. A. Kautz, T. Pitassi, Combining component caching and clause learning for effective model counting., *SAT 4* (2004) 7th. URL: <https://www.satisfiability.org/SAT04/accepted/65.html>.
- [16] U. Oztok, A. Darwiche, A top-down compiler for sentential decision diagrams., in: *IJCAI*, volume 15, 2015, pp. 3141–3148. URL: <https://dl.acm.org/doi/10.5555/2832581.2832687>.
- [17] J. Lagniez, P. Marquis, An improved decision-dnnf compiler., in: *IJCAI*, volume 17, 2017, pp. 667–673. doi:10.24963/ijcai.2017/93.
- [18] J. M. Dudek, V. H. Phan, M. Y. Vardi, DPMC: weighted model counting by dynamic programming on project-join trees, in: *International Conference on Principles and Practice of Constraint Programming*, Springer, 2020, pp. 211–230. doi:10.1007/978-3-030-58475-7_13.
- [19] P. Dilkas, V. Belle, Weighted model counting with conditional weights for bayesian networks, in: *Uncertainty in Artificial Intelligence*, PMLR, 2021, pp. 386–396. URL: <https://proceedings.mlr.press/v161/dilkas21a.html>.
- [20] J. Davies, F. Bacchus, Solving MAXSAT by solving a sequence of simpler SAT instances, in: *International conference on principles and practice of constraint programming*, Springer, 2011, pp. 225–239. doi:10.1007/978-3-642-23786-7_19.
- [21] A. Nadel, TT-Open-WBO-Inc: An Efficient Anytime MaxSAT Solver, *Journal on Satisfiability, Boolean Modeling and Computation* 15 (2024) 1–7. doi:10.3233/sat-231504.
- [22] S. Joshi, P. Kumar, S. Rao, R. Martins, Open-WBO-Inc: Approximation Strategies for Incomplete Weighted MaxSAT, *Journal on Satisfiability, Boolean Modeling and Computation* 11 (2019) 73–97.

doi:10.3233/sat190118.

- [23] F. Bacchus, M. Järvisalo, J. Berg, R. Martins, A. Niskanen, Maxsat evaluation 2020, 2020.
- [24] J. P. Marques-Silva, K. A. Sakallah, GRASP: A Search Algorithm for Propositional Satisfiability, *IEEE Trans. Comput.* 48 (1999) 506–521. doi:10.1109/12.769433.
- [25] G. Spallitta, R. Sebastiani, A. Biere, Disjoint projected enumeration for SAT and SMT without blocking clauses, *Artificial Intelligence* (2025) 104346. doi:10.1016/j.artint.2025.104346.
- [26] S. Sharma, S. Roy, M. Soos, K. S. Meel, Ganak: A scalable probabilistic exact model counter, in: *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2019. doi:10.24963/ijcai.2019/163.
- [27] M. Soos, K. S. Meel, Engineering an efficient probabilistic exact model counter, in: *Proceedings of the International Conference on Computer Aided Verification (CAV)*, 2025. doi:10.1007/978-3-031-98682-6_5.
- [28] J. M. Dudek, V. H. N. Phan, M. Y. Vardi, ADDMC: Weighted Model Counting with Algebraic Decision Diagrams, *Proceedings of the AAAI Conference on Artificial Intelligence* 34 (2020) 8730–8737. doi:10.1609/aaai.v34i02.5505.
- [29] T. Korhonen, M. Järvisalo, SharpSAT-TD in model counting competitions 2021-2023, *arXiv preprint arXiv:2308.15819* (2023). doi:10.48550/arXiv.2308.15819.
- [30] R. Martins, V. Manquinho, I. Lynce, Open-WBO: A Modular MaxSAT Solver, in: C. Sinz, U. Egly (Eds.), *Theory and Applications of Satisfiability Testing – SAT 2014*, 2014, pp. 438–445. doi:10.1007/978-3-319-09284-3_33.
- [31] C. Li, Z. Xu, J. Coll, F. Manyà, D. Habet, K. He, Combining Clause Learning and Branch and Bound for MaxSAT, in: L. D. Michel (Ed.), *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*, volume 210 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2021, pp. 38:1–38:18. URL: <https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CP.2021.38>. doi:10.4230/LIPIcs.CP.2021.38.
- [32] A. Nadel, Anytime weighted maxsat with improved polarity selection and bit-vector optimization, in: *2019 Formal Methods in Computer Aided Design (FMCAD)*, 2019, pp. 193–202. doi:10.23919/FMCAD.2019.8894273.
- [33] A. Ignatiev, A. Morgado, J. Marques-Silva, RC2: an efficient maxsat solver, *J. Satisf. Boolean Model. Comput.* 11 (2019) 53–64. doi:10.3233/SAT190116.
- [34] A. Darwiche, P. Marquis, A knowledge compilation map, *Journal of Artificial Intelligence Research* 17 (2002) 229–264. doi:10.1613/jair.989.
- [35] A. Amarilli, P. Bourhis, F. Capelli, M. Monet, Ranked Enumeration for MSO on Trees via Knowledge Compilation, in: G. Cormode, M. Shekelyan (Eds.), *27th International Conference on Database Theory, ICDT 2024, Paestum, Italy, March 25-28, 2024*, volume 290 of *LIPIcs*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, pp. 25:1–25:18. doi:10.4230/LIPIcs.ICDT.2024.25.
- [36] S. Minato, J. Kawahara, M. Banbara, T. Horiyama, I. Takigawa, Y. Yamaguchi, Fast enumeration of all cost-bounded solutions for combinatorial problems using zdds, *Discrete Applied Mathematics* 360 (2025) 467–486. doi:10.1016/j.dam.2024.10.003.
- [37] M. Michelutti, G. Masina, G. Spallitta, R. Sebastiani, Canonical Decision Diagrams Modulo Theories, in: *Proceedings of the European Conference on Artificial Intelligence - ECAI24*, 2024. doi:10.3233/faia241007.
- [38] R. Nieuwenhuis, A. Oliveras, C. Tinelli, Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL (T), *Journal of the ACM (JACM)* 53 (2006) 937–977. doi:10.1145/1217856.1217859.
- [39] V. H. N. Phan, M. Y. Vardi, DPO: Dynamic-Programming Optimization on Hybrid Constraints, *arXiv preprint arXiv:2205.08632* (2022). doi:10.48550/arXiv.2205.08632.
- [40] C. Barrett, R. Sebastiani, A. A. Seshia, C. Tinelli, Satisfiability Modulo Theories, in: *Handbook of Satisfiability*, volume 336 of *FAIA*, 2 ed., IOS Press, 2021, pp. 1267–1329. doi:10.3233/FAIA201017.
- [41] G. S. Tseitin, On the complexity of derivation in propositional calculus, in: *Automation of Reasoning 2: Classical Papers on Computational Logic 1967-70*, Springer, 1983, pp. 466–483.

doi:10.1007/978-3-642-81955-1_28.

- [42] D. A. Plaisted, S. Greenbaum, A structure-preserving clause form translation, *Journal of Symbolic Computation* 2 (1986) 293–304. doi:10.1016/s0747-7171(86)80028-1.
- [43] S. Möhle, A. Biere, Combining Conflict-Driven Clause Learning and Chronological Backtracking for Propositional Model Counting., in: *GCAI*, 2019, pp. 113–126. doi:doi.org/10.29007/vgg4.
- [44] A. Biere, M. Heule, H. van Maaren, T. Walsh, *Handbook of satisfiability*, volume 336, IOS press, 2021. doi:10.3233/faia336.
- [45] A. Biere, T. Faller, K. Fazekas, M. Fleury, N. Froleys, F. Pollitt, CaDiCaL 2.0, in: *International Conference on Computer Aided Verification*, Springer, 2024, pp. 133–152. doi:10.1007/978-3-031-65627-9_7.
- [46] R. J. Bayardo Jr, R. Schrag, Using CSP look-back techniques to solve real-world SAT instances, in: *Aaai/iaai*, Providence, RI, 1997, pp. 203–208. doi:10.1007/3-540-61551-2_65.
- [47] T. Sang, P. Beame, H. A. Kautz, Performing Bayesian inference by weighted model counting, in: *AAAI*, volume 5, 2005, pp. 475–481. URL: <https://dl.acm.org/doi/10.5555/1619332.1619409>.
- [48] G. Spallitta, G. Masina, P. Morettin, A. Passerini, R. Sebastiani, SMT-based weighted model integration with structure awareness, in: *Uncertainty in Artificial Intelligence*, PMLR, 2022, pp. 1876–1885. doi:10.48550/arXiv.2206.13856.
- [49] G. Spallitta, G. Masina, P. Morettin, A. Passerini, R. Sebastiani, Enhancing SMT-based Weighted Model Integration by structure awareness, *Artificial Intelligence* 328 (2024) 104067. doi:10.1016/j.artint.2024.104067.