Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)
Special Session on KR and Machine Learning

On the Approximability of Weighted Model Integration on DNF Structures

Ralph Abboud , **İsmail İlkan Ceylan** and **Radoslav Dimitrov**
Department of Computer Science, University of Oxford
{firstname.lastname}@cs.ox.ac.uk

Abstract

Weighted model counting (WMC) consists of computing the weighted sum of all satisfying assignments of a propositional formula. WMC is well-known to be #P-hard for exact solving, but admits a fully polynomial randomized approximation scheme (FPRAS) when restricted to DNF structures. In this work, we study *weighted model integration*, a generalization of weighted model counting which involves real variables in addition to propositional variables, and pose the following question: Does weighted model integration on DNF structures admit an FPRAS? Building on classical results from approximate volume computation and approximate weighted model counting, we show that weighted model integration on DNF structures can indeed be approximated for a class of weight functions. Our approximation algorithm is based on three subroutines, each of which can be a *weak* (i.e., approximate), or a *strong* (i.e., exact) oracle, and in all cases, comes along with accuracy guarantees. We experimentally verify our approach over randomly generated DNF instances of varying sizes, and show that our algorithm scales to large problem instances, involving up to 1K variables, which are currently out of reach for existing, general-purpose weighted model integration solvers.

1 Introduction

Weighted model counting (WMC) has been introduced as a unifying approach for encoding probabilistic inference problems that arise in various formalisms. Informally, given a propositional formula, and a weight function that assigns every truth assignment a weight, WMC amounts to computing the weighted sum of all the satisfying assignments (Gomes, Sabharwal, and Selman 2009). Many probabilistic inference problems in *probabilistic graphical models* (Koller and Friedman 2009), *probabilistic planning* (Domshlak and Hoffmann 2007), *probabilistic logic programming* (De Raedt, Kimmig, and Toivonen 2007), *probabilistic databases* (Suciu et al. 2011), and *probabilistic knowledge bases* (Borgwardt, Ceylan, and Lukasiewicz 2017) can be reduced to a form of WMC.

Despite its wide applicability, WMC is limited to discrete domains and thus cannot be applied to domains involving real variables, and this motivated the study of *weighted model integration* (WMI) (Belle, Passerini, and Van Den Broeck 2015), as a generalization of WMC.

Building on the foundations of *satisfiability modulo theories (SMT)* (Barrett et al. 2009), WMI can capture *hybrid domains* with mixtures of Boolean and continuous variables. Briefly, the input to WMI is a *hybrid* propositional formula that additionally involves arithmetic constraints (e.g., linear constraints over real, or integer variables), and a weight function that defines a *density* for every truth assignment of the formula. WMI is then the task of computing the sum of *integrals* over the densities of all the satisfying assignments of the given hybrid propositional formula (Belle, Passerini, and Van Den Broeck 2015; Morettin, Passerini, and Sebastiani 2019).

The standard formulation of WMI assumes a formula in *conjunctive normal form* (CNF) as an input, and to date, there is no study of WMI which is specifically tailored to formulas in *disjunctive normal form* (DNF). This is surprising, as both variants are widely investigated for WMC. We write WMI(CNF) and WMI(DNF) in the sequel to distinguish between these cases. These problems are clearly #P-hard for *exact solving*, as are their respective special cases WMC(CNF) and WMC(DNF) (Valiant 1979). For *approximate solving*, however, there is a strong contrast in computational complexity between variants of weighted model counting problems: WMC(DNF) has a *fully polynomial randomized algorithm scheme (FPRAS)* (Karp, Luby, and Madras 1989), producing polynomial-time approximations with guarantees, whereas WMC(CNF) is NP-hard to approximate (Roth 1996). The latter polynomial-time inapproximability result immediately propagates to WMI(CNF), while the approximability status of WMI(DNF) remains *open*. In this paper, we pose the following question: Does WMI(DNF) admit an FPRAS?

We answer this question in the affirmative, and provide a polynomial-time algorithm for WMI(DNF) with probabilistic accuracy guarantees. The intuition behind our result is based on two observations. First, the special case of WMI(DNF) without any arithmetic constraints corresponds to WMC(DNF) which has an FPRAS (Karp, Luby, and Madras 1989). Second, the special case of WMI(DNF) with constant weight functions, and without any Booleans, corresponds to computing the volume of *unions of convex bodies*, which also has an FPRAS (Bringmann and Friedrich 2010). Our result builds on these results, and extends them, by allowing extra constructs essential for WMI, while preserving

the approximation guarantees. Our main contributions can be summarized as follows:

- We propose an efficient approximation algorithm for WMI(DNF), called APPROXWMI, extending the algorithm given in (Bringmann and Friedrich 2010).
- We prove that APPROXWMI is an FPRAS provided that the weight functions are *concave*, and can be factorized into products of weights of literals. We provide asymptotic bounds for the running time of the algorithm.
- We extend APPROXWMI to the case where the products of weights assumption is relaxed, and provide asymptotic bounds for the running time of the algorithm.
- We experimentally verify our approach, using a strong oracle for computing the volume of a body. Our experiments suggest that APPROXWMI solves large problem instances, including up to 1K variables, which are out of reach for any existing, general-purpose WMI solver.

The full proofs of our results can be found in the long version of this paper, available at: https://arxiv.org/abs/2002.06726.

2 Preliminaries

We briefly recall *propositional logic*, *linear real arithmetic* and *weighted model integration*, where we also settle the notation and assumptions used throughout the paper.

2.1 Logic and Linear Real Arithmetic

Let us denote by $\mathbb{R}$ the real domain, and by $\mathbb{B}$ the Boolean domain $\{0, 1\}$. Let $\mathbf{X}$ be a set of n real variables, and $\mathbf{V}$ be a set of m Boolean variables (or atoms). An *LRA atom* is of the form

$$\sum_i c_i x_i \bowtie c,$$

where c and c_i are rational values/constants, $x_i \in \mathbf{X}$, and $\bowtie \in \{<, \leq, >, \geq, =, \neq\}$ with their usual semantics. We write $\mathsf{atoms}(\mathbf{X}, \mathbf{V})$ to denote the set of atoms over $\mathbf{X} \cup \mathbf{V}$. A *literal* is either an atom or its negation. We sometimes write *LRA literal*, or a *Boolean literal*, to distinguish the literals depending on the domain of their corresponding variables.

A *propositional formula* ϕ over $\mathsf{atoms}(\mathbf{X}, \mathbf{V})$ is defined as a Boolean combination of literals via the logical connectives $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$. If $\mathbf{V} = \varnothing$, we say ϕ is an *LRA formula*, and if $\mathbf{X} = \varnothing$ then ϕ corresponds to a standard propositional formula defined only over Boolean variables.

Example 1. *Let us consider the formula ϕ_{ex} given as*

$$((0 \leq x_1 \leq 5) \vee \neg p_1) \wedge (p_2 \vee \neg(10 \leq x_1 + x_2 \leq 15)),$$

which contains 2 Boolean and 2 LRA literals. ∎

Given a propositional formula ϕ over $\mathsf{atoms}(\mathbf{X}, \mathbf{V})$, a *truth assignment* $\tau : \mathsf{atoms}(\mathbf{X}, \mathbf{V}) \mapsto \mathbb{B}$, maps every atom to either 0 (false), or 1 (true). A truth assignment τ *satisfies* a propositional formula ϕ, denoted $\tau \vDash \phi$, in the usual sense, where $\vDash$ is the propositional entailment relation. We sometimes say τ *propositionally satisfies* ϕ to make the underlying entailment relation explicit.

Observe that a propositional formula over $\mathsf{atoms}(\mathbf{X}, \mathbf{V})$ may have a propositionally satisfying truth assignment but not admit a solution to the LRA constraints, i.e., the relevant LRA constraints define an empty polytope. An assignment τ is *LRA-satisfiable* if the solution space to the set of linear inequalities induced by the mapping τ is non-empty. The classical SMT problem over LRA constraints is, for a given a propositional formula ϕ over $\mathsf{atoms}(\mathbf{X}, \mathbf{V})$, to decide whether there exists an assignment τ such that $\tau \vDash \phi$ (propositionally satisfiable), and τ is LRA-satisfiable.

Example 2. *Consider again the formula ϕ_{ex}, and an assignment τ with $\tau(p_1) = 1$, $\tau(p_2) = 0$, $\tau(0 \leq x_1 \leq 5) = 1$, and $\tau(10 \leq x_1 + x_2 \leq 15) = 0$. Clearly, ϕ_{ex} is propositionally satisfiable as witnessed by τ; it is also LRA satisfiable, e.g., for values $x_1 = 2$, and $x_2 = 30$, the assignment τ is LRA satisfiable. By contrast, the formula*

$$\phi_{ex'} = (1 \leq x_1 + x_2 \leq 4) \wedge \neg p_1 \wedge (x_1 \leq -2) \wedge (x_2 \leq 2)$$

is trivially propositionally satisfiable, but is not LRA satisfiable, since $(x_1 \leq -2) \wedge (x_2 \leq 2)$ and $(1 \leq x_1 + x_2 \leq 4)$ cannot be satisfied simultaneously. ∎

We recall the fragments of propositional logic. A *conjunctive clause* is a conjunction of literals, and a *disjunctive clause* is a disjunction of literals. A propositional formula ϕ is in *conjunctive normal form* (CNF) if it is a conjunction of disjunctive clauses, and it is in *disjunctive normal form* (DNF) if it is a disjunction of conjunctive clauses. A clause has *width* k if it has exactly k literals. We say that a DNF (resp., CNF) has width k if it contains clauses of width at most k.

2.2 Weighted Model Integration

Let $\mathbf{X}$ be a set of n real variables, and $\mathbf{V}$ a set of m Boolean variables. We consider a weight function $w : (\mathbb{R}^n \times \mathbb{B}^m) \mapsto \mathbb{R}^+$, and propositional formulas $\phi(\mathbf{X}, \mathbf{V})$ such that $\phi : (\mathbb{R}^n \times \mathbb{B}^m) \mapsto \mathbb{B}$.

The *weighted model integral* is defined as:

$$\mathrm{WMI}(\phi, w \mid \mathbf{X}, \mathbf{V}) = \sum_{\mathbf{v}} \int_{\mathbf{x}_\phi} w(\mathbf{x}, \mathbf{v}) d\mathbf{x} \tag{1}$$

where $\mathbf{v}$ is an assignment to Boolean variables $\mathbf{V}$, and $\mathbf{x}_\phi$ denotes the real valuations of $\mathbf{X}$ satisfying $\phi(\mathbf{x}, \mathbf{v})$.

Note that the weight function depends both on Boolean and real variables. It is common to employ a simplifying assumption to $w(\mathbf{x}, \mathbf{v})$ (e.g., Section 2.2 of (Martires, Dries, and De Raedt 2019)):

$$w(\mathbf{x}, \mathbf{v}) = w_x(\mathbf{x}) \prod_{i=1}^{m} w_b(p_i), \tag{2}$$

where $w_x : \mathbb{R}^n \rightarrow \mathbb{R}^+$ and $w_b : \mathbb{B} \rightarrow]0, 1[$ are functions, and $w_b(p_i)$ returns the probability of Boolean literal p_i. This implies that the weight function from (1) can be factorized into a product of a real variable weight function w_x and a product of individual Boolean literal weights as in (2), and we refer to this as the *factorization assumption*.

Example 3. *Consider the formula* ϕ_{DNF} :

$$(p_1 \wedge (0 \leq x_1 \leq 5) \wedge \neg p_2) \vee (p_2 \wedge \neg(2 \leq x_1 \leq 4)).$$

Let $0 \leq x_1 \leq 10$*, and* $w(\mathbf{x}, \mathbf{v}) = x_1 \cdot w_b(p_1) \cdot w_b(p_2)$*, with* $w_b(p_1) = 0.6$*, and* $w_b(p_2) = 0.1$*. Hence, we obtain:*

$$\text{WMI}(\phi_{\text{DNF}}) = 0.6 \cdot (1 - 0.1) \cdot \int_0^5 x_1 \, dx_1 \; + \; 0.1 \cdot \Big(\int_0^2 x_1 \, dx_1 + \int_4^{10} x_1 \, dx_1 \Big) = 11.15$$

■

Weighted model integration is defined on fragments of propositional logic in the obvious way, i.e., WMI over DNF formulas (resp. CNF formulas) is the weighted model integration problem where the class of input formulas is restricted to formulas in DNF (resp., CNF). We write WMI(DNF) (resp., WMI(CNF)) to denote the specific problem.

Finally, weighted model counting can be viewed as a special case of WMI where ϕ is restricted to Boolean variables. Formally, the *weighted model count* (WMC) of ϕ is given by $\sum_{\tau \models \phi} w(\tau)$, where $w : \mathbb{B}^m \mapsto \mathbb{R}^+$ is a *weight function*.

2.3 Approximations with Guarantees

Model counting problems are #P-hard to solve exactly, and thus are intractable for exact computation. As a result, techniques for efficient *approximations* to model counting have been devised, a special class of which being *fully polynomial randomized approximation schemes (FPRAS)*. Given a target error $0 < \epsilon < 1$ and confidence $0 < \delta < 1$, an FPRAS computes an approximation $\hat{\mu}$ of the actual solution μ, in polynomial time w.r.t the input, $\frac{1}{\epsilon}$, and $\frac{1}{\delta}$, such that

$$\Pr\big(\mu(1-\epsilon) \leq \hat{\mu} \leq \mu(1+\epsilon)\big) \geq 1 - \delta.$$

For WMC on DNF structures, the Karp, Luby, and Madras (1989) algorithm (KLM), a special case of the Linear-Time Coverage (LTC) (Luby 1983) algorithm, is an FPRAS. For a DNF ϕ with n variables and m clauses, KLM runs $T = 8(1+\epsilon)m \log(\frac{2}{\delta})\frac{1}{\epsilon^2}$ trials to compute a successful trial count N. At every trial, KLM performs the following:

1. If no current sample assignment τ exists, then a random clause c_i is selected with probability $\Pr(c_i)/\sum_{j=1}^m \Pr(c_j)$, where $\Pr(c_i) = \prod_{v \in c_i} w_b(v)$. Afterwards, τ is sampled uniformly from the set of satisfying assignments for c_i.
2. Another clause c_k (which could be identical to c_i) is *uniformly* randomly sampled, and τ is checked against c_k. If $\tau \models c_k$, N is incremented and τ is re-sampled. Otherwise, τ is re-used in the next trial.

KLM returns $T \sum_{j=1}^m p(c_j)/mN$ as an estimate for WMC(DNF). Since assignment checking runs in $O(n)$, KLM thus runs in time $O\big(nm\epsilon^{-2}\log(\frac{1}{\delta})\big)$.

For volume computation of convex bodies, Lovàsz and Vempala (2006) provide an FPRAS based on Multi-Phase Monte Carlo. In $n-$dimensional space, it uses $O^*(n)$ phases, where the asterisk denotes suppressed logarithmic factors, such that at every phase, random walk algorithms such as *hit-and-run* (Chen and Schmeiser 1996) are called over convex bodies at consecutive phases to approximate the ratios between their volumes. Since hit-and-run runs in $O^*(n^3)$, the overall volume computation runs in time $O^*(n^4)$.

3 Weighted Model Integration over DNFs

In this section, we propose APPROXWMI, an algorithm for WMI(DNF), and prove that it is an FPRAS. APPROXWMI builds on work for approximately computing the volume of unions of convex bodies (Bringmann and Friedrich 2010), which we introduce next.

3.1 The Volume of Union of Convex Bodies

APPROXUNION is an FPRAS for computing the volume of the union of convex bodies (Bringmann and Friedrich 2010). More formally, given k convex bodies $B_1, ..., B_k$, APPROXUNION returns an approximation of the volume of $\bigcup_{i=1}^k B_i$, denoted $\mathsf{Vol}(\bigcup_{i=1}^k B_i)$. This algorithm is based on the LTC algorithm, and extends it with approximate ("weak") oracles in order to tackle the underlying volume computations.

APPROXUNION first computes the volume of every convex body approximately, with multiplicative error ϵ_v, using an oracle VOLUMEQUERY. Following this, it repeats the following procedure for T trials to compute a successful trial count N.

1. If no sample point p exists, APPROXUNION samples a body B_i with probability $\mathsf{Vol}(B_i)/\sum_{j=1}^k \mathsf{Vol}(B_j)$, and then approximately uniformly samples p from this body using an oracle SAMPLEQUERY with error ϵ_S.
2. It then checks whether p belongs to another uniformly chosen body B' using another approximate oracle, POINTQUERY, with error ϵ_p. If $p \in B'$, a new B_i and p are sampled, and N is incremented. Otherwise, p is re-used in the next trial.

Following T calls to POINTQUERY, the algorithm returns $T \sum_{j=1}^k \mathsf{Vol}(B_j)/kN$ as an estimate of $\mathsf{Vol}(\bigcup_{i=1}^k B_i)$.

APPROXUNION is an FPRAS for the volume computation of a union of convex bodies under certain conditions, as stated in Theorem 2 of (Bringmann and Friedrich 2010), and these conditions are satisfied with closed-form bounds from Lemma 3 of that work. All in all, the following result holds:

Theorem 1 (Theorem 2 and Lemma 3, (Bringmann and Friedrich 2010))**.** APPROXUNION *relative to oracles* VOLUMEQUERY, SAMPLEQUERY, POINTQUERY *with errors* ϵ_V, ϵ_S, *and* ϵ_P *respectively, is an FPRAS for* $\mathsf{Vol}(\bigcup_{i=1}^k B_i)$ *with error* ϵ *and confidence* $\frac{1}{4}$ *using* $T = \frac{24\ln(2)(1+\tilde{\epsilon})k}{\tilde{\epsilon}^2 - 8(\tilde{C}-1)k}$ *iterations, with* $\tilde{\epsilon} = \frac{\epsilon - \epsilon_V}{1+\epsilon_V}$ *and* $\tilde{C} = \frac{(1+\epsilon_S)(1+\epsilon_V)(1+k\epsilon_P)}{(1-\epsilon_V)(1-\epsilon_P)}$*, for* $\epsilon_V, \epsilon_S \leq \frac{\epsilon^2}{47k}$*, and* $\epsilon_P \leq \frac{\epsilon^2}{47k^2}$.

Furthermore, when this theorem holds, T is $O(\frac{k}{\epsilon^2})$. APPROXUNION generalizes LTC to allow errors within sampling, membership checking, and volume computation, so long as these can be made arbitrarily small, and computes unions of continuous sets, as opposed to only discrete sets.

Algorithm 1 APPROXWMI for WMI(DNF).

Input: $\mathbf{X}$: a set of n real variables; $\mathbf{V}$: a set of m Boolean variables; ϕ: a DNF consisting of k clauses c_i; w: a concave factorized weight function.
Parameters: ϵ: error, δ: confidence.
Output: $\text{WMI}(\phi, w \mid \mathbf{X}, \mathbf{V})$.

1: $T \leftarrow \frac{8\ln(\frac{8}{\delta})(1+\tilde{\epsilon})k}{\tilde{\epsilon}^2 - 8(\tilde{C}-1)k}$ ▷ T is $O(\frac{k}{\epsilon^2}\ln(\frac{1}{\delta}))$
2: $\delta_V \leftarrow \frac{\delta}{4k}$, $\delta_S \leftarrow \frac{\delta}{2276\ln(\frac{8}{\delta})\frac{k}{\epsilon^2}}$ ▷ Confidence parameters
3: **for** $a \leftarrow 1$ to k **do**
4: $\quad U_i \leftarrow \text{CLAUSEWEIGHT}(c_i, w, \mathbf{X}, \mathbf{V})[\frac{\epsilon^2}{47k},\ \delta_V]$
5: $U \leftarrow \sum_{i=1}^{m} U_i$ ▷ Sampling trials
6: time $\leftarrow 0$, $N_T \leftarrow 0$
7: **while** time $< T$ **do**
8: $\quad$ Randomly select c_i, $i \in 1, ..., k$ with probability $\frac{U_i}{U}$
9: $\quad p_{\text{Bool}}, p_{\text{Real}} \leftarrow \text{SAMPLE}(c_i, w, \mathbf{X}, \mathbf{V})[\frac{\epsilon^2}{47k},\ \delta_S]$
10: $\quad c_{\text{sat}} \leftarrow \textit{false}$
11: $\quad$ **while** $\neg c_{\text{sat}}$ **do**
12: $\quad\quad$ Uniformly select c_j, where $j \in 1, ..., k$
13: $\quad\quad$ time $\leftarrow$ time + 1
14: $\quad\quad$ **if** time $\geq T$ **then** ▷ If T reached during trial
15: $\quad\quad\quad$ **return** TU/kN_T
16: $\quad\quad$ **if** $\text{EVALUATE}(c_j, p_{\text{Bool}}, p_{\text{Real}})$ **then**
17: $\quad\quad\quad$ $c_{\text{sat}} \leftarrow \textit{true}$, $N_T \leftarrow N_T + 1$
18: **return** TU/kN_T

APPROXUNION does *not* allow for confidence parameters in the oracles, but it can be extended to allow for FPRAS oracles having confidence δ using standard tools of probability.

Lemma 2. APPROXUNION *relative to FPRAS oracles* VOLUMEQUERY, SAMPLEQUERY, *and* POINTQUERY *with errors* ϵ_V, ϵ_S, ϵ_P *and confidence values* δ_V, δ_S, δ_P, *respectively, is an FPRAS with error* ϵ *and confidence* δ *for* $\mathsf{Vol}(\bigcup_{i=1}^{k} B_i)$ *using* $T = \frac{8\ln(\frac{8}{\delta})(1+\tilde{\epsilon})k}{\tilde{\epsilon}^2 - 8(\tilde{C}-1)k}$ *iterations, for*

1. $\epsilon_V, \epsilon_S \leq \frac{\epsilon^2}{47k}$, $\epsilon_P \leq \frac{\epsilon^2}{47k^2}$, *and*
2. $\delta_V \leq \frac{\delta}{4k}$, $\delta_S + \delta_P \leq \frac{\delta}{2276\ln(\frac{8}{\delta})\frac{k}{\epsilon^2}}$.

3.2 Approximating the WMI over DNFs

We can now introduce the algorithm APPROXWMI (Algorithm 1) for WMI(DNF), assuming w is concave, i.e., $\forall x, y \in \mathbb{R}^n$, and $\lambda \in [0, 1]$, it holds that

$$\lambda w(x) + (1-\lambda)w(y) \leq w(\lambda x + (1-\lambda)y).$$

This assumption ensures that the bodies resulting from the application of weight functions on convex polytopes are also convex, which in turn enables the use of volume computation FPRAS algorithms (Lovász and Vempala 2006; Kannan, Lovász, and Simonovits 1997) over these bodies. We also assume that w uses the factorization assumption.

Overview of APPROXWMI. Given a DNF ϕ over a hybrid domain $\mathbf{X} \cup \mathbf{V}$, and a concave weight function w that factorizes, APPROXWMI computes an ϵ, δ approximation of $\text{WMI}(\phi, w \mid \mathbf{X}, \mathbf{V})$. More specifically, APPROXWMI extends the oracle functions of APPROXUNION in order to allow unreliable oracles (with confidence parameter δ), hybrid domains, and arbitrary factorized concave weight functions over convex bodies.

Algorithm 2 Subroutines of APPROXWMI as functions CLAUSEWEIGHT, SAMPLE, and EVALUATE.

function $\text{CLAUSEWEIGHT}(c, w, \mathbf{X}, \mathbf{V})[\epsilon, \delta]$
$\quad w_{\text{Bool}} \leftarrow \prod_{p \in c_i} w_b(p)$ ▷ (c_1)
$\quad$ Introduce a new variable d in $\mathbf{X}$ ▷ (c_2)
$\quad \mathbf{x}'_c \leftarrow \mathbf{x}_c \wedge (0 \leq d \leq w_x(x))$
$\quad w_{\text{Real}} \leftarrow \text{VOLUME}(\mathbf{x}'_c)[\epsilon, \delta]$ ▷ (c_3)
$\quad$ **return** $w_{\text{Bool}} \cdot w_{\text{Real}}$ ▷ (c_4)

function $\text{SAMPLE}(c, w, \mathbf{X}, \mathbf{V})[\epsilon, \delta]$
$\quad p_{\text{Bool}} \leftarrow b_c$, $\mathbf{V} \setminus b_c$ sampled according to w_b ▷ (s_1)
$\quad$ Introduce a new variable d in $\mathbf{X}$ ▷ (s_2)
$\quad \mathbf{x}'_c \leftarrow \mathbf{x}_c \wedge (0 \leq d \leq w_x(x))$
$\quad p_{\text{Real}} \leftarrow \text{CONVEXBODYSAMPLER}(\mathbf{x}'_c)[\epsilon, \delta]$ ▷ (s_3)
$\quad$ **return** $p_{\text{Bool}}, p_{\text{Real}}$ ▷ (s_4)

function $\text{EVALUATE}(c, p_{\text{Bool}}, p_{\text{Real}})$
$\quad$ Compute convex polytope $\mathbf{x}_c$ from c
$\quad$ **if** $p_{\text{Real}} \notin \mathbf{x}_c$ **then return** *false* ▷ (e_1)
$\quad$ **if** $p_{\text{Bool}} \notin c$ **then return** *false* ▷ (e_2)
$\quad$ **return** *true*

The main steps of APPROXWMI are as follows: After initializing the parameters (1-2), the first step is to compute the weighted model integral of the individual clauses in ϕ, using the function CLAUSEWEIGHT (3-4). Then, the algorithm runs T sampling trials to compute a successful trial count N_T (7-18), similarly to LTC. In a sampling trial, a random clause c is selected with probability proportional to its weight U_i, and then a point p is sampled from c according to w using the function SAMPLE (8-9). Afterwards, a clause c' (possibly c), is uniformly chosen from ϕ (12), and a check is made via the function EVALUATE, to verify the membership of p to c', and the estimator N_T is incremented accordingly (16-17).

We now explain the subroutines of CLAUSEWEIGHT, SAMPLE, and EVALUATE, given in Algorithm 2, in detail.

CLAUSEWEIGHT. This function returns an ϵ, δ approximation of $\text{WMI}(c, w \mid \mathbf{X}, \mathbf{V})$ from a given conjunct c and weight function w over the domains $\mathbf{X}, \mathbf{V}$. CLAUSEWEIGHT computes the product of probabilities for all Boolean literals, denoted by p, appearing in c (c_1). It then transforms the polytope $\mathbf{x}_c$ defined by the LRA constraints in c, into $\mathbf{x}'_c \in \mathbb{R}^{n+1}$, by adding another constraint encoding the weight function, with this operation denoted by $\wedge$ (c_2). Thus, $\mathbf{x}'_c$ is identical to $\mathbf{x}_c$ across the first n dimensions, with an added dimension d verifying $0 \leq d \leq w_x(x), x \in \mathbf{x}_c$. CLAUSEWEIGHT approximates the volume of $\mathbf{x}'_c$, as a proxy for computing the integral of w over $\mathbf{x}_c$, using a convex body volume computation algorithm (Lovász and Vempala 2006; Kannan, Lovász, and Simonovits 1997), denoted by VOLUME (c_3). Since w is concave, the added dimen-

sion in $\mathbf{x}'_c$ maintains the convexity of $\mathbf{x}_c$, and $\mathbf{x}'_c$ is therefore convex. Finally, CLAUSEWEIGHT returns the product of the steps (c_1) and (c_3) outputs, as its estimate for $\text{WMI}(c, w \mid \mathbf{X}, \mathbf{V})$ (c_4).

SAMPLE. This function samples a point p over the domain $\mathbf{X} \cup \mathbf{V}$ from a given conjunct c and a weight function w, as follows: It first samples Boolean assignments p_{Bool} satisfying c by setting all Boolean literals appearing in c to their required values and randomly sampling all remaining variables according to w_b (s_1). Then, it computes an analogous transformation from $\mathbf{x}_c$ to $\mathbf{x}'_c$ as in CLAUSEWEIGHT (s_2). Afterwards, SAMPLE samples a point p_{Real} approximately uniformly from $\mathbf{x}'_c$, using standard sampling approaches for convex bodies such as hit-and-run (Chen and Schmeiser 1996), denoted by CONVEXBODYSAMPLER (s_3). It then discards the $n+1^{th}$ dimension to yield approximate samples p_{Real} from $\mathbf{x}_c$ weighted according to w. Finally, SAMPLE (s_4) returns the concatenation of the outputs of (s_1) and (s_3) as a sample from c. Note that, since the weight factorization makes Boolean variable and real variable weights independent, p_{Bool} and p_{Real} can be sampled separately, as we outline here.

EVALUATE. This function determines the membership of a point $p \in \mathbb{R}^n \times \mathbb{B}^m$ to the body defined by a conjunct c. Specifically, it checks the membership of a point p to the polytope defined by c in two steps: EVALUATE first verifies the Boolean component of p (e_1), and then verifies the real component of p (i.e., that all the LRA constraints of c are satisfied) (e_2). If both conditions are met, then p satisfies c. Unlike the earlier two functions, EVALUATE is deterministic, and its outputs have no attached uncertainty.

3.3 APPROXWMI is an FPRAS

We show the correctness of APPROXWMI, and prove that, under the error and confidence settings presented in Algorithm 1, it is an FPRAS for WMI (DNF) with concave weight functions w, respecting the factorization assumption.

Theorem 3. APPROXWMI *relative to FPRAS oracles* CLAUSEWEIGHT, SAMPLE, *and* EVALUATE *having error* ϵ_V, ϵ_S, ϵ_P *and confidence* δ_V, δ_S, δ_P, *respectively, is an FPRAS for* WMI(DNF) *over a concave and factorized weight function* w, *with error* ϵ *and confidence* δ *and using* $T = \frac{8\ln(\frac{8}{\delta})(1+\tilde{\epsilon})k}{\tilde{\epsilon}^2 - 8(\tilde{C}-1)k}$ *iterations, for*

1. $\epsilon_V, \epsilon_S \leq \frac{\epsilon^2}{47k}$, $\epsilon_P \leq \frac{\epsilon^2}{47k^2}$, *and*
2. $\delta_V \leq \frac{\delta}{4k}$, $\delta_S + \delta_P \leq \frac{\delta}{2276\ln(\frac{8}{\delta})\frac{k}{\epsilon^2}}$.

Proof sketch. APPROXWMI is a variant of APPROXUNION, where the oracles are replaced with the specific oracles for WMI. Thus, it suffices to show that all oracles in APPROXWMI satisfy the conditions of Lemma 2. EVALUATE is deterministic, so trivially satisfies the conditions. As for CLAUSEWEIGHT, we first verify that it correctly computes $\text{WMI}(c, w \mid \mathbf{X}, \mathbf{V})$. We then show that CLAUSEWEIGHT meets the conditions of Lemma 2: as multiplication of Boolean weights is error-free, it is sufficient for VOLUME to have error $\epsilon \leq \epsilon_V$ and confidence $\delta \leq \delta_V$. As for SAMPLE, we first show that sampling from the transformation result $\mathbf{x}'_c$ is equivalent to sampling from $\mathbf{x}_c$ according to w. Then, it suffices to run CONVEXBODYSAMPLER with parameters ϵ_S and δ_S to ensure SAMPLE meets the requirements. □

The correctness of APPROXWMI is clearly independent from the specific choice of oracles, and both for weak and strong oracles, the accuracy guarantees are preserved. Assuming additionally that the oracles have an FPRAS, APPROXWMI runs in polynomial time. More specifically, for CLAUSEWEIGHT runtime r_c, SAMPLE runtime r_s, and EVALUATE runtime r_e, APPROXWMI runs in $O(k \cdot r_c + T(r_s + r_e))$.

To illustrate, for the most general case of arbitrary concave weight functions and convex bodies, we can use the algorithm of Lovász and Vempala (2006), as the VOLUME oracle in CLAUSEWEIGHT, and hit-and-run for CONVEXBODYSAMPLER (Chen and Schmeiser 1996). These choices make CLAUSEWEIGHT run in time $O^*(m + n^4(\frac{1}{\epsilon_V})^2)$, where m is the number of Boolean variables and n is the number of real variables, and SAMPLE run in time $O^*(m + n^3(\frac{1}{\epsilon_S})^2)$. Since EVALUATE runs in deterministic polynomial time, namely $O(m + Wn)$, where W is the width of a conjunction c, APPROXWMI therefore runs in time:

$$
\begin{aligned}
& O^*\Big(km + kn^4(\frac{1}{\epsilon_V})^2 + Tm + Tn^3(\frac{1}{\epsilon_S})^2 + T(m + Wn)\Big) \\
= \; & O^*\Big(km + \frac{k^3}{\epsilon^4}n^4 + \frac{k}{\epsilon^2}m + \frac{k^3}{\epsilon^6}n^3 + \frac{k(Wn+m)}{\epsilon^2}\Big) \\
= \; & O^*\Big(\frac{k^3}{\epsilon^4}n^4 + \frac{k^3}{\epsilon^6}n^3 + \frac{k}{\epsilon^2}(m + Wn)\Big).
\end{aligned}
$$

Note that the time complexity of SAMPLE can be reduced by restricting the class of bodies and weight functions used. Indeed, if w is restricted to be linear, e.g., $3x_1 + 2x_2 - x_4$, then all bodies can be sampled approximately using optimized polytope sampling methods such as geodesic walks (Lee and Vempala 2017), which can run significantly faster for bodies defined with a small number of LRA constraints. Further to this, box-shaped bodies, defined by using no more than one variable per constraint, paired with a constant w, i.e., a uniform distribution over the problem domain, are an instance of unweighted model integration, and can be trivially sampled using uniform sampling. Nonetheless, we assume the general case of convex bodies in this work, and build our algorithm accordingly.

3.4 Extending APPROXWMI: APPROXWMI$_\mathbf{D}$

In Section 3.2, we presented APPROXWMI, an FPRAS for WMI over DNF formulas with factorized and concave weight function w. The factorization of w simplifies WMI, in that (i) it makes Boolean variable weights independent from real variables, and vice-versa, and (ii) simplifies the joint distribution over Booleans to a product of weights. This standard factorization prevents any changes in the Boolean domain from affecting w_x, but can be used to capture weight functions with this behaviour by defining mutually exclusive Boolean partitions of the problem domain,

in which no dependencies between Boolean and real variables exist, and then applying the factorization separately over each of these partitions (Martires, Dries, and De Raedt 2019). However, the number of such partitions can be exponential in the worst-case, which makes this approach intractable in practice.

In this section, we introduce a more general factorization for w, such that

$$w(\mathbf{x}, \mathbf{v}) = w_x(\mathbf{x}, \mathbf{v}) \prod_{i=1}^{m} w_b(p_i), \tag{3}$$

where w_x also depends on Boolean variables. We then propose an FPRAS APPROXWMI$_\mathrm{D}$ for this factorization. APPROXWMI$_\mathrm{D}$ extends APPROXWMI, in particular its oracle CLAUSEWEIGHT, to function under this more general factorization.

We now describe how the more general weight function w is defined. Let $\mathbf{x}_c \subseteq \mathbb{R}^n$ be the polytope defined by the LRA constraints of a conjunct c, and $v \in \mathbb{B}^m$ be a Boolean assignment. Let $a, b : P(\mathbb{R}^n) \mapsto \mathbb{R}^+$, where P denotes the power set operation, be functions such that

$$a(\mathbf{x}_c) = \min_{\mathbf{v}} \int_{\mathbf{x}_c} w_x(\mathbf{y}, \mathbf{v})\, d\mathbf{y}, \text{ and}$$

$$b(\mathbf{x}_c) = \max_{\mathbf{v}} \int_{\mathbf{x}_c} w_x(\mathbf{y}, \mathbf{v})\, d\mathbf{y}.$$

These functions are guaranteed to exist and are finite as the Boolean domain $\mathbb{B}^m$ is finite and all integral computations in the scope of WMI are finite-valued. Hence, we note that $\forall \mathbf{x}_c \in P(\mathbb{R}^n), \forall \mathbf{v} \in \mathbb{B}^m, \exists a, b : P(\mathbb{R}^n) \mapsto \mathbb{R}^+$, such that

$$a(\mathbf{x}_c) \leq \int_{\mathbf{x}_c} w_x(\mathbf{y}, \mathbf{v})\, d\mathbf{y} \;\leq\; b(\mathbf{x}_c). \tag{4}$$

Let us define $\rho = \max_{\mathbf{x}_c} \frac{b(\mathbf{x}_c)}{a(\mathbf{x}_c)}$. In what follows, we restrict the choice of w such that, for all $\mathbf{x}_c$, ρ is upper-bounded by a polynomial in $\frac{1}{\epsilon_{\text{Samp}}}$, $\frac{1}{\delta_{\text{Samp}}}$, n, m, and k. In this case, we say that a weight function w is *ρ-restricted*.

Intuitively, *ρ-restriction* ensures that integrals over the same body, computed with different Boolean instantiations of the weight function, yield results that are within a tractable ratio from one another, which in turn allows the efficient use of sampling techniques.

While this restriction is similar in nature to restrictions on *tilt* θ in existing works on weighted model counting (see e.g. (Chakraborty et al. 2014)), it is not identical to them. More specifically, tilt is the ratio between the maximum weight of a satisfying assignment and the minimum weight of a satisfying assignment, whereas ρ is the ratio between the maximum integral and minimum integral of a weight function over any given real body. In fact, restrictions on ρ are looser than restrictions on θ. Indeed, when θ is bounded by a value H, it is simple to show that ρ is also bounded by H, whereas the same cannot be said in the opposite direction.

Consider a simple example, with one Boolean variable v and one real variable x, such that $0 \leq x \leq 10$, and let

$$w_x(x, v) = \begin{cases} \frac{e^{10}}{1+e^{-x}} - 0.5e^{10} + 1 & \text{if } v \\ \frac{2e^{10}}{1+e^{-x}} - e^{10} + 2 & \text{otherwise,} \end{cases}$$

Algorithm 3 Subroutines of APPROXWMI$_\mathrm{D}$ as functions CLAUSEWEIGHT$_\mathrm{D}$ and SAMPLE$_\mathrm{D}$.

function CLAUSEWEIGHT$_\mathrm{D}(c, w, \mathbf{X}, \mathbf{V})[\epsilon, \delta]$
- $\epsilon_{\text{Samp}}, \epsilon_{\text{Comp}} \leftarrow \frac{\epsilon}{1+\sqrt{2}}$, $\delta_{\text{Samp}} \leftarrow \frac{\delta}{2}$
- $s \leftarrow \ln(\frac{2}{\delta_{\text{Samp}}}) \frac{1}{\epsilon^2_{\text{Samp}}} \rho^2$ ▷ # of sampling trials
- $\delta_{\text{Comp}} \leftarrow \frac{\delta}{2s}$
- **for** $i \leftarrow 1$ to s **do**
  - $\tau_i \leftarrow b_c, \mathbf{V} \setminus b_c$ sampled according to w_b ▷ (c_1)
  - Introduce a new variable d in $\mathbf{X}$ ▷ (c_2)
  - $\mathbf{x}'_c \leftarrow \mathbf{x}_c \wedge (0 \leq d \leq w_x(\mathbf{x}, \tau_i))$
  - $X_i \leftarrow \text{VOLUME}_{\epsilon_{\text{Comp}}, \delta_{\text{Comp}}}(\mathbf{x}'_c)$ ▷ (c_3)
- BoolWeight $\leftarrow \prod_{p \in c_i} w_b(p)$ ▷ (c_4)
- **return** BoolWeight $\cdot \frac{1}{s} \sum_{i=1}^{s} X_i$ ▷ (c_5)

function SAMPLE$_\mathrm{D}(c, w, \mathbf{X}, \mathbf{V})[\epsilon, \delta]$
- $p_{\text{Bool}} \leftarrow b_c, \mathbf{V} \setminus b_c$ sampled according to w_b ▷ (s_1)
- Introduce a new variable d in $\mathbf{X}$ ▷ (s_2)
- $\mathbf{x}'_c \leftarrow \mathbf{x}_c \wedge (0 \leq d \leq w_x(\mathbf{x}, p_{\text{Bool}}))$
- $p_{\text{Real}} \leftarrow \text{CONVEXBODYSAMPLER}(\mathbf{x}'_c)[\epsilon, \delta]$ ▷ (s_3)
- **return** $p_{\text{Bool}}, p_{\text{Real}}$ ▷ (s_4)

In this example, ρ is clearly upper-bounded by 2, but θ is upper-bounded by $2 + e^{10}$. To upper-bound ρ easily in practice, it is sufficient to upper-bound, for all real assignments x, the ratio between the maximum $w(\mathbf{x}, \mathbf{v})$ and the minimum $w(\mathbf{x}, \mathbf{v})$ over all Boolean assignments, or more formally, $w_{\max}(\mathbf{x})/w_{\min}(\mathbf{x})$, where $w_{\min}(\mathbf{x}) = \min_{\mathbf{v}} w(\mathbf{x}, \mathbf{v})$ and $w_{\max}(\mathbf{x}) = \max_{\mathbf{v}} w(\mathbf{x}, \mathbf{v})$.

APPROXWMI$_\mathrm{D}$ extends APPROXWMI to also handle weight functions w, where w_x may depend on Boolean variables. To achieve this, APPROXWMI$_\mathrm{D}$ replaces the oracles CLAUSEWEIGHT and SAMPLE used in Algorithm 1, with CLAUSEWEIGHT$_\mathrm{D}$ and SAMPLE$_\mathrm{D}$, respectively (while other details remain unaffected). Hence, we present these extended oracles CLAUSEWEIGHT$_\mathrm{D}$ and SAMPLE$_\mathrm{D}$, presented in Algorithm 3, in more detail.

CLAUSEWEIGHT$_\mathrm{D}$. CLAUSEWEIGHT$_\mathrm{D}$ performs sampling over the set of Boolean assignments to estimate the WMI of c, and this sampling occurs with error ϵ_{Samp} and confidence δ_{Samp}. The number s of sampling trials is a function of ϵ_{Samp} and δ_{Samp}, as well as ρ, the integral ratio defined earlier. Within every trial, VOLUME is called with error ϵ_{Comp} and confidence δ_{Comp}.

More specifically, given a conjunct c and a weight function w, CLAUSEWEIGHT$_\mathrm{D}$ performs s *sampling rounds* to estimate WMI$(c, w \mid \mathbf{X}, \mathbf{V})$, where each sampling round consists of randomly sampling a Boolean assignment τ satisfying c according to w_b (c_1), computing the induced weight function $w_x(x, \tau)$ and using it to obtain the transformed convex body $\mathbf{x}'_c$ (c_2), and computing the weighted integral over c using VOLUME (c_3). At the end of the s sampling steps, the product of all b_c literal weights is computed (c_4) , and the function returns this product, multiplied by the average of all sampling results, as an approximation of WMI$(c, w \mid \mathbf{X}, \mathbf{V})$ (c_5).

SAMPLE$_\mathbf{D}$. This function is defined almost identically to SAMPLE in APPROXWMI, with the only minor difference being that $w_x(\mathbf{x}, p_{\text{Bool}})$ must be induced from p_{Bool} in SAMPLE$_\text{D}$ (s_2) prior to applying the transformation. This is because w_x also depends on Boolean variables in this setting. Unlike SAMPLE, where Boolean and real variable sampling can be done in any order, it is necessary for Boolean sampling to run first in SAMPLE$_\text{D}$, so as to condition w_x on the Boolean sample output and subsequently sample from $\mathbf{x}_c$ according to the induced weight function $w_x(\mathbf{x}, p_{\text{Bool}})$.

3.5 APPROXWMI$_\text{D}$ is an FPRAS

We show that APPROXWMI$_\text{D}$ is an FPRAS for WMI(DNF), by lifting the result given in Theorem 3. To do so, we first need to show the correctness of CLAUSEWEIGHT$_\text{D}$, and prove that it is an FPRAS for WMI$(c, w \mid \mathbf{X}, \mathbf{V})$ over concave, ρ-restricted w factorized using this more general factorization.

Lemma 4. CLAUSEWEIGHT$_\text{D}$ *relative to Monte-Carlo sampling error ϵ_{Samp} and confidence δ_{Samp}, and an FPRAS* VOLUME *with error ϵ_{Comp} and confidence δ_{Comp}, is an FPRAS for* WMI$(c, w|\mathbf{X}, \mathbf{V})$*, where c is a clause and w is concave, ρ-restricted, and factorized according to Equation 3, with error ϵ, confidence δ, and using $s = \ln(\frac{2}{\delta_{Samp}})\frac{1}{\epsilon^2_{Samp}}\rho^2$ iterations, for*

1. $\epsilon_{Samp}, \epsilon_{Comp} \leq \frac{\epsilon}{1+\sqrt{2}}$,
2. $\delta_{Samp} \leq \frac{\delta}{2}$, *and*
3. $\delta_{Comp} \leq \frac{\delta}{2s}$.

Proof sketch. Under this more general factorization, WMI for a conjunct c can be computed as a sum of integrals over the real polytope $\mathbf{x}_c$ given all (possibly exponential) possible weight functions induced by Boolean assignments. Monte-Carlo sampling approximates WMI$(c, w \mid \mathbf{X}, \mathbf{V})$, avoids the exponential blow-up, and can provide guarantees, since w is ρ-restricted. Clearly, the results of every sampling step are s independent and identically distributed (i.i.d) random variables which, from Equation 4, are bounded by $a(\mathbf{x}_c)$ and $b(\mathbf{x}_c)$. Applying the Hoeffding bound with the target additive difference a ϵ_{Samp} multiple of the expected WMI yields the lower bound for s shown in Algorithm 3. CLAUSEWEIGHT$_\text{D}$ therefore also runs in polynomial time with respect to $\frac{1}{\epsilon_{\text{Samp}}}$, $\frac{1}{\delta_{\text{Samp}}}$, n, m, and k , since a sampling iteration runs in polynomial time (VOLUME is an FPRAS), and s is polynomial given that w is ρ-restricted.

It now remains to show that, under the conditions of Lemma 4, CLAUSEWEIGHT$_\text{D}$ produces an ϵ, δ approximation of WMI$(c, w|\mathbf{X}, \mathbf{V})$. First, we prove that, when no failure occurs within CLAUSEWEIGHT$_\text{D}$ (i.e., all oracles and the sampling procedure respect their error bounds), the provided error bounds in Lemma 4 produce an estimate within the overall ϵ error requirement, and the multiplication of the two $\frac{\epsilon}{1+\sqrt{2}}$ bounds for sampling and volume computation, combined with $\epsilon \leq 1$, yields the desired result. Second, we show that, with the asserted confidence bounds, the probability of any failure in CLAUSEWEIGHT$_\text{D}$, is upper-bounded by δ, using the union bound. □

We can now combine Theorem 3 and Lemma 4 to show that APPROXWMI$_\text{D}$ is an FPRAS for WMI (DNF) for this more general factorization of w, and given a concave and ρ-restricted w.

Theorem 5. APPROXWMI$_\text{D}$*, relative to FPRAS oracles* CLAUSEWEIGHT$_\text{D}$*,* SAMPLE$_\text{D}$*, and* EVALUATE *having error ϵ_V, ϵ_S, ϵ_P and confidence δ_V, δ_S, δ_P, respectively, is an FPRAS for* WMI(DNF) *over a w that is concave, ρ-restricted, and factorized according to Equation 3, with error ϵ and confidence δ and using $T = \frac{8\ln(\frac{8}{\delta})(1+\bar{\epsilon})k}{\bar{\epsilon}^2-8(\bar{C}-1)k}$ iterations, for*

1. $\epsilon_V, \epsilon_S \leq \frac{\epsilon^2}{47k}$, $\epsilon_P \leq \frac{\epsilon^2}{47k^2}$, *and*
2. $\delta_V \leq \frac{\delta}{4k}$, $\delta_S + \delta_P \leq \frac{\delta}{2276\ln(\frac{8}{\delta})\frac{k}{\epsilon^2}}$.

Finally, the running time of APPROXWMI$_\text{D}$, as a function of CLAUSEWEIGHT$_\text{D}$ runtime r'_c, SAMPLE$_\text{D}$ runtime r'_s, and EVALUATE runtime r_e, is $O(k \cdot r'_c + T(r'_s + r_e))$, analogously to APPROXWMI. Furthermore, for the same choice of CONVEXBODYSAMPLER, it is easy to see that $r_s = r'_s$. However, the main difference in running time between APPROXWMI and APPROXWMI$_\text{D}$ comes from the difference between r'_c and r_c. Indeed, given the same choice of VOLUME oracle, with running time r_v, $r_c = O(m + r_v)$, whereas $r'_c = O(s(m+r_v)) = O(\frac{1}{\epsilon^4}\ln(\frac{1}{\delta})(m+r_v))$. Hence, the wider applicability of APPROXWMI$_\text{D}$ comes at the expense of a larger runtime complexity, owing to the larger power of $\frac{1}{\epsilon}$ required for CLAUSEWEIGHT$_\text{D}$.

4 Experimental Evaluation

To evaluate the performance of APPROXWMI, we generate random DNF formulas and measure the time APPROXWMI requires to solve them. We explain data generation and experimental setup in the following subsections.

4.1 Generating Evaluation Data

To evaluate APPROXWMI, we generate DNF formulas with total number of variables between 100 and $1K$ inclusive, in increments of 100. Variables are split equally between real and Boolean (i.e., $m = n$). Clause width W is fixed between 3, 5, 8, 13, and the number of clauses is $k = \lfloor \frac{m+n+20}{W} \rfloor$. For every configuration (m, n, k, W), we generate 4 formulas, resulting in a total of 176 formulas used for evaluation.

Given a configuration, we generate a propositional DNF with $m + n$ variables. This DNF has $k \cdot W$ "slots", corresponding to the vacant literal positions to be filled in its clauses, and these slots are allocated to variables such that every variable appears at least once in the DNF.

With probability 0.5, this allocation occurs uniformly. However, when this isn't the case, a "privileged mechanism" is used, such that a randomly selected small subset of "privileged" variables is allocated significantly more slots than the remaining variables (to encourage more dependencies across clauses), thereby giving these variables a larger impact on the formula WMI. Once all variables are allocated, their literal sign is chosen uniformly at random. This data generation is also used and presented in more detail earlier (Abboud, Ceylan, and Lukasiewicz 2020).

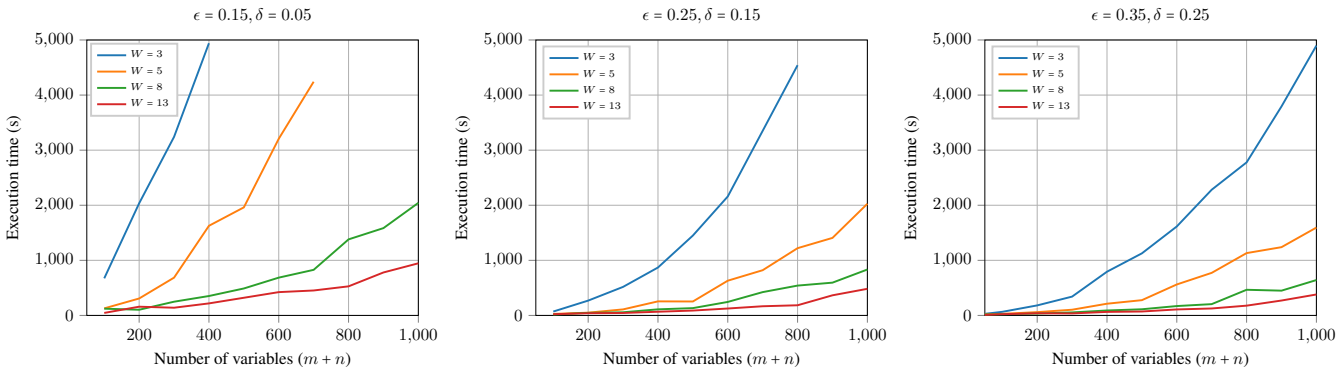

Figure 1: Runtime results for APPROXWMI relative to different ϵ, δ. For $\epsilon = 0.35, \delta = 0.25$, all instances, including those with 1K variables, terminate within 5000 seconds, and higher-width instances ($W = 8, 13$) all terminate within 2000 seconds for all ϵ, δ.

Once a propositional DNF formula is generated, LRA constraints are then incorporated as follows: First, the slots for n variables, corresponding to variable indices $m + 1$ to $m + n$, are all replaced with LRA constraints, which in turn are generated at the clause level. For a conjunctive clause c with q LRA constraints to be generated:

1. A random point p in the real domain is uniformly sampled such that, by construction, $p \models c$, so that the real polytope defined by LRA constraints in c is non-empty.
2. Generate q constraints by (i) randomly selecting a subset S of $Geom(1/L)$ real variables, where $Geom$ denotes the geometric distribution and $L = 2$, (ii) randomly sampling $w_s \in \mathbb{R}^{|S|}$ weights for these variables and (iii) generating a random value v and setting the linear constraint $w.S \leq v$.
3. If p satisfies $w.S \leq v$, then $w.S \leq v$ is added to c, otherwise, $w.S \geq v$, which p must then satisfy, is added.

4.2 Experimental Setup

In our experiments, we bound all real variables such that $\forall x \in \mathbf{X}, 0 \leq x \leq 10$, to ensure finite integrals. We then evaluate APPROXWMI with *polynomial* weight functions: w is a sum of up to 4 polynomial terms, each with a random constant weight, and degree randomly chosen using $Geom(0.6)$ and upper-bounded by 5. For terms with degree of 2 or higher, the constant weight is constrained to be negative to maintain concavity, e.g. $w = 20 - 3x_1^2 x_4 - 2.2x_3^4 + x_2$.

We run APPROXWMI using LattE (Baldoni et al. 2014) for VOLUME within CLAUSEWEIGHT as, despite being an exact solver, it supports polynomial weight functions while performing reliably in practice for smaller-scale formulas compared with approximate techniques (Emiris and Fisikopoulos 2018). We optimize our use of LattE by separately (trivially) integrating over variables not appearing in a clause or a term of w, and only running LattE over the appearing variables. This optimization is effective as (i) the number of real variables appearing in a clause is small in expectation (at worst LW) and (ii) the DNF formula structure breaks down the n-dimensional integration task into k smaller parts which can be solved more efficiently (unlike for CNF). For sampling, we use hit-and-run (Chen and Schmeiser 1996) for CONVEXBODYSAMPLER within SAMPLE, with a constant factor 10^4 used to compute the number of walk iterations, as is standard in practice. It is *unknown* whether this constant preserves theoretical guarantees, but it is widely used given that the best-known theoretical constant factors for hit-and-run are loose and prohibitively large (i.e., 10^{30}) (Lovász and Vempala 2003).

Finally, results were averaged over 5 runs with 3 (ϵ, δ) settings: (0.15, 0.05), (0.25, 0.15), and (0.35, 0.25). Experiments ran with a timeout of 5000 seconds on a server with a Haswell 5-2640v3, 2.60GHz CPU and 12 GB of RAM.

4.3 Experimental Results

All APPROXWMI running times w.r.t. $m + n$, W, ϵ, and δ, are presented in Figure 1: APPROXWMI performs very encouragingly, and solves DNF instances with up to $1K$ variables within 5000 seconds over all clause widths W for $\epsilon = 0.35$ and $\delta = 0.25$. In fact, instances with $1K$ variables and $W = 5, 8, 13$ all run within 1600 seconds. For tighter ϵ and δ, the system maintains high performance, despite the high power of ϵ^{-1} in the running time of APPROXWMI due to SAMPLE (cf. Section 3.3). Indeed, even with $\epsilon = 0.15$ and $\delta = 0.05$, all instances with widths 8 and 13 finish within ~2000 seconds, whereas large instances of width 3 and $m + n \geq 500$, and width 5, $m + n \geq 800$, time out.

Somewhat unintuitively, system performance worsens as W *decreases*. For DNF instances with $W = 3$, APPROXWMI requires almost triple the time compared to instances with higher W. Though this behavior is surprising, it can be attributed to an increased number of sampling replacements within APPROXWMI. Indeed, for smaller widths, it is likelier that a call to EVALUATE will yield "True" since there are less constraints to satisfy. Hence, further calls to SAMPLE, which runs in $O^*(m+n^3)$, will be required, imposing a significant computational overhead on APPROXWMI. This behavior justifies improved performance with increased width, as the expected number of calls to SAMPLE decreases.

These results confirm our intuitions about APPROXWMI. First, they highlight that APPROXWMI can indeed scale to large instances having up to 1K total variables, even with tight ϵ and δ. Second, they show that CLAUSEWEIGHT calls are not a bottleneck, as they all run in less than 450 seconds

in any instance, due to the bounded width W used in our experiments. Third, our results show that the main performance bottleneck for APPROXWMI is the number of calls to SAMPLE. This is particularly evident from the high dependence of running time on width W, and shows that, even with a reduced constant factor, convex body sampling remains a highly costly operation.

The use of an exact CLAUSEWEIGHT oracle allows significant gains, owing to a reduced error requirement for SAMPLE. Indeed, given exact CLAUSEWEIGHT and EVALUATE, i.e., $\epsilon_V, \epsilon_P, \delta_V, \delta_P = 0$, Theorem 1 and the union bound yield $\epsilon_S < \frac{\epsilon^2}{8k}$ and $\delta_S \leq \frac{\delta}{1518 \ln(\frac{8}{\delta})\frac{k}{\epsilon^2}}$ respectively, and these looser bounds for sampling reduce the running time of APPROXWMI significantly. Overall, APPROXWMI scales to DNF instances with up to 1K variables using standard oracle implementations. This is particularly true for larger widths, as the number of SAMPLE calls decreases.

5 Related Work

WMC is a unifying tool for probabilistic inference. Inference in probabilistic graphical models (Koller and Friedman 2009) reduces to WMC(CNF) (Sang, Bearne, and Kautz 2005; Chavira and Darwiche 2008). Similar reductions exist for Markov Logic Networks (MLNs) (Richardson and Domingos 2006), probabilistic logic programming (Fierens et al. 2015), and more generally, for relational models (Gogate and Domingos 2011). Still, WMC cannot capture hybrid probabilistic models that are extensively studied; see e.g. hybrid MLNs (Wang and Domingos 2008), and hybrid Bayesian networks (Gogate and Dechter 2005; Sanner and Abbasnejad 2012). WMI is proposed as a unifying inference tool in these hybrid models (Belle, Passerini, and Van Den Broeck 2015).

Weighted model integration/counting is #P-hard (Valiant 1979), so is highly intractable. Nonetheless, many general-purpose exact solvers, based on several optimizations, have been developed. For instance, Morettin, Passerini, and Sebastiani (2019) propose a tool which uses SMT predicate abstraction techniques to reduce the number of models, for which an integration tool is called. Symbo (Martires, Dries, and De Raedt 2019) uses knowledge compilation to push computational overhead to an offline phase, and subsequently allow efficient online WMI computation for factorized w. Finally, a technique is proposed to compute exact lower and upper bounds for WMI based on hyper-rectangular decomposition and orthogonal transformations (Merrell, Albarghouthi, and D'Antoni 2017).

Besides exact solvers, many approximate solvers have been developed for WMI. For example, a hashing-based approach for WMI is proposed (Belle, den Broeck, and Passerini 2015), which extends existing hashing methods for WMC, and uses propositional abstraction and requires a polynomial number of NP-oracle calls. Sampo (Martires, Dries, and De Raedt 2019) extends Symbo with Monte Carlo sampling for integral computation, and thus leverages knowledge compilation to quickly evaluate sampled densities. Furthermore, Markov Chain Monte Carlo methods have been applied to WMI, but such approaches do not provide any guarantees. The only exception is the general tool of Chistikov, Dimitrova, and Majumdar (2017) for #SMT, which is the unweighted case of WMI. This tool approximates the solution of a #SMT instance by calling a SAT solver. This comes with approximation guarantees as in WMC(CNF), using a randomized algorithm that makes polynomially many calls to the SAT solver (Jerrum, Valiant, and Vazirani 1986).

To the best of our knowledge, there is no dedicated study for WMI(DNF), despite WMC(DNF) being extensively studied, partly motivated from the rich literature on probabilistic data management (Suciu et al. 2011): query answering in probabilistic databases reduces to WMC(DNF) as every conjunctive query is equivalent to a DNF via its lineage representation. The study of WMC(DNF) has led to a number of tools, or algorithms from KLM (Karp, Luby, and Madras 1989) to hashing-based techniques (Meel, Shrotri, and Vardi 2017), and recently to neural model counting approaches (Abboud, Ceylan, and Lukasiewicz 2020).

Existing tools for WMC(DNF) cannot handle extended data models which also include continuous distributions. Such data models are quite common, see e.g. Monte Carlo Databases (MCDBs) (Jampani et al. 2008), and the system PIP (Kennedy and Koch 2010), which extends MCDBs to efficiently query probabilistic data defined over both discrete and continuous distributions. Abstracting away from subtle differences, these works also introduce approximate inference algorithms, but unlike our approach, these do not provide guarantees. The study of WMI(DNF) hence serves as a unifying perspective for a class of data models.

6 Summary and Outlook

In this work, we studied weighted model integration on DNF structures and presented APPROXWMI, which is an FPRAS given a concave and factorized weight function w. We also presented $\text{APPROXWMI}_\text{D}$, an extension to APPROXWMI allowing more relaxed factorizations of w. Our FPRAS results for WMI(DNF) complement the result of WMC(DNF), and help draw a more complete picture of approximability for these problems. Our experimental analysis further shows the potential of APPROXWMI.

Looking forward, we aim to investigate alternative approaches for approximate WMI with guarantees, e.g., hashing-based approaches, to minimize the impact of sampling. We hope this work leads to further investigation of WMI techniques, leading to more robust WMI systems.

Acknowledgements

This work was supported by the Alan Turing Institute under the UK EPSRC grant EP/N510129/1, the AXA Research Fund, and by the EPSRC grants EP/R013667/1, EP/L012138/1, and EP/M025268/1. Ralph Abboud is funded by the Oxford-DeepMind Graduate Scholarship and the Alun Hughes Graduate Scholarship. Experiments for this work were conducted on servers provided by the Advanced Research Computing (ARC) cluster administered by the University of Oxford.

References

Abboud, R.; Ceylan, İ. İ.; and Lukasiewicz, T. 2020. Learning to reason: Leveraging neural networks for approximate DNF counting. In *Proc. of AAAI*, 3097–3104.

Baldoni, V.; Berline, N.; De Loera, J. A.; Dutra, B.; Köppe, M.; Moreinis, S.; Pinto, G.; Vergne, M.; and Wu, J. 2014. A user's guide for LattE integrale v1.7.2. *Optimization* 22(2).

Barrett, C.; Sebastiani, R.; Seshia, S.; and Tinelli, C. 2009. *Satisfiability modulo theories*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. 825–885.

Belle, V.; den Broeck, G. V.; and Passerini, A. 2015. Hashing-based approximate probabilistic inference in hybrid domains. In *Proc. of UAI*, 141–150.

Belle, V.; Passerini, A.; and Van Den Broeck, G. 2015. Probabilistic inference in hybrid domains by weighted model integration. In *Proc. of IJCAI*, 2770–2776.

Borgwardt, S.; Ceylan, İ. İ.; and Lukasiewicz, T. 2017. Ontology-mediated queries for probabilistic databases. In *Proc. of AAAI*, 1063–1069.

Bringmann, K., and Friedrich, T. 2010. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Comput. Geom.* 43(6-7):601–610.

Chakraborty, S.; Fremont, D. J.; Meel, K. S.; Seshia, S. A.; and Vardi, M. Y. 2014. Distribution-aware sampling and weighted model counting for SAT. In *Proc. of AAAI*, 1722–1730.

Chavira, M., and Darwiche, A. 2008. On probabilistic inference by weighted model counting. *Artificial Intelligence* 172(6):772 – 799.

Chen, M., and Schmeiser, B. W. 1996. General hit-and-run Monte Carlo sampling for evaluating multidimensional integrals. *Oper. Res. Lett.* 19(4):161–169.

Chistikov, D.; Dimitrova, R.; and Majumdar, R. 2017. Approximate counting in smt and value estimation for probabilistic programs. *Acta Informatica* 54(8):729–764.

De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. ProbLog: A probabilistic prolog and its application in link discovery. In *Proc. of IJCAI*, 2462–2467.

Domshlak, C., and Hoffmann, J. 2007. Probabilistic planning via heuristic forward search and weighted model counting. *JAIR* 30(1):565–620.

Emiris, I. Z., and Fisikopoulos, V. 2018. Practical polytope volume approximation. *ACM TOMS* 44(4):38:1–38:21.

Fierens, D.; Van Den Broeck, G.; Renkens, J.; Shterionov, D.; Gutmann, B.; Thon, I.; Janssens, G.; and De Raedt, L. 2015. Inference and learning in probabilistic logic programs using weighted boolean formulas. *TPLP* 15(3):358–401.

Gogate, V., and Dechter, R. 2005. Approximate inference algorithms for hybrid bayesian networks with discrete constraints. In *Proc. of UAI*, 209–216.

Gogate, V., and Domingos, P. 2011. Probabilistic theorem proving. In *Proc. of UAI*, 256–265.

Gomes, C. P.; Sabharwal, A.; and Selman, B. 2009. Model counting. In *Handbook of Satisfiability*. IOS Press. 633–654.

Jampani, R.; Xu, F.; Wu, M.; Perez, L. L.; Jermaine, C.; and Haas, P. J. 2008. Mcdb: A Monte Carlo approach to managing uncertain data. In *Proc. of SIGMOD*, 687–700.

Jerrum, M. R.; Valiant, L. G.; and Vazirani, V. V. 1986. Random generation of combinatorial structures from a uniform. *TCS* 43(2-3):169–188.

Kannan, R.; Lovász, L.; and Simonovits, M. 1997. Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. *Random Struct. Algorithms* 11(1):1–50.

Karp, R. M.; Luby, M.; and Madras, N. 1989. Monte-Carlo approximation algorithms for enumeration problems. *J. Algorithms* 10(3):429–448.

Kennedy, O., and Koch, C. 2010. PIP: A database system for great and small expectations. In *Proc. of ICDE*, 157–168.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Lee, Y. T., and Vempala, S. S. 2017. Geodesic walks in polytopes. In *Proc. of STOC*, 927–940.

Lovász, L., and Vempala, S. 2003. Where to start a geometric random walk. Technical report, Microsoft Research.

Lovász, L., and Vempala, S. S. 2006. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. *JCSS* 72(2):392–417.

Luby, M. G. 1983. *Monte-Carlo Methods for Estimating System Reliability*. Ph.D. Dissertation, UC Berkeley.

Martires, P.; Dries, A.; and De Raedt, L. 2019. Exact and approximate weighted model integration with probability density functions using knowledge compilation. In *Proc. of AAAI*, 7825–7833.

Meel, K. S.; Shrotri, A. A.; and Vardi, M. Y. 2017. On hashing-based approaches to approximate DNF-counting. In *Proc. of FSTTCS*, 41:1–41:14.

Merrell, D.; Albarghouthi, A.; and D'Antoni, L. 2017. Weighted model integration with orthogonal transformations. In *Proc. of IJCAI*, 4610–4616.

Morettin, P.; Passerini, A.; and Sebastiani, R. 2019. Advanced SMT techniques for weighted model integration. *AIJ* 275:1 – 27.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62(1):107–136.

Roth, D. 1996. On the Hardness of Approximate Reasoning. *AIJ* 82(1-2):273–302.

Sang, T.; Bearne, P.; and Kautz, H. 2005. Performing bayesian inference by weighted model counting. In *Proc. of AAAI*, 475–482.

Sanner, S., and Abbasnejad, E. 2012. Symbolic variable elimination for discrete and continuous graphical models. In *Proc. of AAAI*, 1954–1960.

Suciu, D.; Olteanu, D.; Ré, C.; and Koch, C. 2011. *Probabilistic Databases*. Morgan & Claypool.

Valiant, L. G. 1979. The complexity of computing the permanent. *TCS* 8(2):189–201.

Wang, J., and Domingos, P. 2008. Hybrid Markov logic networks. In *Proc. of AAAI*, 1106–1111.