Approximate Weighted First-Order Model Counting: Exploiting Fast Approximate Model Counters and Symmetry

Timothy van Bremen[1] and **Ondřej Kuželka**[2]
[1]KU Leuven, Belgium
[2]Czech Technical University in Prague, Czech Republic
timothy.vanbremen@cs.kuleuven.be, ondrej.kuzelka@fel.cvut.cz

Abstract

We study the symmetric weighted first-order model counting task and present ApproxWFOMC, a novel anytime method for efficiently bounding the weighted first-order model count of a sentence given an unweighted first-order model counting oracle. The algorithm has applications to inference in a variety of first-order probabilistic representations, such as Markov logic networks and probabilistic logic programs. Crucially for many applications, no assumptions are made on the form of the input sentence. Instead, the algorithm makes use of the symmetry inherent in the problem by imposing cardinality constraints on the number of possible true groundings of a sentence's literals. Realising the first-order model counting oracle in practice using the approximate hashing-based model counter ApproxMC3, we show how our algorithm is competitive with existing approximate and exact techniques for inference in first-order probabilistic models. We additionally provide PAC guarantees on the accuracy of the bounds generated.

1 Introduction

Given a propositional formula ϕ, the *model counting* problem asks for the number of models (satisfying assignments) of ϕ. Model counting is the prototypical $\#\mathsf{P}$-complete problem. The *weighted model counting* (WMC) problem generalizes this task by associating each assignment with a real-valued *weight*, and asks for the weighted sum of the formula's models. In the past several years, the WMC task has attracted great interest as an "assembly language" for probabilistic inference, as inference in various formalisms such as graphical models [Chavira and Darwiche, 2008] and probabilistic logic programming languages [Fierens *et al.*, 2015] can be reduced to WMC. Many practical implementations of (weighted) model counters have also been introduced, such as DSHARP [Muise *et al.*, 2012], MINIC2D [Oztok and Darwiche, 2015], and d4 [Lagniez and Marquis, 2017]. In addition to exact weighted model counters, another line of research has unfolded among approximate model counters [Chakraborty *et al.*, 2013; Chakraborty *et al.*, 2014], which are often capable of scaling to much larger problem sizes than exact methods.

In practice, however, logical representations of real-world domains are often first-order, and are typically grounded into propositional logic before a weighted model counter can be used to infer probabilities. In general, a first-order probabilistic inference task can be reduced to an instance of the *weighted first-order model counting* (WFOMC) problem, in which weights are assigned to interpretations of a first-order formula. In this paper, we consider the *symmetric* WFOMC problem, where weights are associated with each predicate, as opposed to the asymmetric case where each possible grounding of a predicate may have a distinct weight.

WFOMC remains a difficult task. From a complexity point of view, Beame *et al.* [2015] showed that the data complexity of symmetric WFOMC for FO^k ($k \geq 3$) is $\#\mathsf{P}_1$-hard, suggesting that in general sentences with at least three distinct logical variables are not domain-liftable.[1] Nevertheless, the search for practical methods for performing WFOMC remains an active area of research. Some algorithms, such as FORCLIFT [Van den Broeck *et al.*, 2011] and ALCHEMY2 [Gogate and Domingos, 2011], operate directly on the first-order representation in order to circumvent the grounding step. The alternative approach requires first grounding out the problem and then passing it to a propositional weighted model counter.

One question that has received relatively little attention is how one can efficiently exploit propositional model counters in practice for first-order problems: in other words, can we leverage off-the-shelf propositional model counters for WFOMC, in a more efficient manner than the naïve ground-and-solve approach? In this paper, we answer this question in the affirmative, and show how such a strategy can be efficiently implemented using hashing-based approximate model counters. As we shall show later, existing hashing-based approximate model counting algorithms capable of dealing with weighted instances need an exponential number of SAT queries in the size of the domain when dealing with grounded first-order formulas. In order to overcome this, we first propose a decomposition of the weighted first-order model count (which, like its corresponding problem, we abbreviate as WFOMC) into the weighted sum of a number of (unweighted) first-order model counts of the input formula conjoined with cardinality con-

[1]In the artificial intelligence literature, a problem is said to be *domain-liftable* if inference can be performed in polynomial time in the size of the domain [Van den Broeck, 2011].

straints. These cardinality constraints serve to limit the number of true instances of the formula's atoms. We then extend our approach to an anytime iterative algorithm with guarantees that uses an intuitive search procedure to find dense regions in the space of weighted models. We evaluate our approach by computing the partition functions of challenging Markov logic network instances that are unlikely to be domain-liftable [Beame *et al.*, 2015], and show that we are competitive with existing methods, especially when seeking approximations.

2 Background

In this section, we first briefly review the syntax of first-order logic, and formally define the WFOMC problem. We then explain the principles of hashing-based approximate model counting techniques, and show their pitfalls when applied to first-order problems.

2.1 First-order Logic

We deal with the function-free, finite domain fragment of first-order logic. An *atom* of arity n takes the form $P(t_1, \ldots, t_n)$, where P/n comes from a vocabulary of *predicates*, and each argument t_i is either a constant from a finite domain $\mathbf{D}$, or a logical variable from a vocabulary of variables. A *literal* is an atom or its negation. A *formula* is formed by connecting one or more literals together using conjunction or disjunction. A formula may optionally be surrounded by one or more quantifiers of the form $\exists x$ or $\forall x$, where x is a logical variable. A logical variable in a formula is said to be *free* if it is not bound by any quantifier. A formula with no free variables is called a *sentence*. A *clause* is a sentence consisting of a disjunction of literals. A formula is in *conjunctive normal form* (CNF) if it is the conjunction of one or more clauses containing only universal quantification. We follow the usual semantics of first-order logic.

2.2 Weighted First-order Model Counting

We review the definition of the (unweighted) first-order model count of a sentence. Throughout this section, we fix a sentence ϕ containing predicates $P_1/r_1, \ldots, P_k/r_k$.

Definition 1. *The first-order model count (FOMC) of ϕ over a domain of size d is defined as:* $\mathsf{FOMC}(\phi, d) = |\mathsf{models}_d(\phi)|$ *where* $\mathsf{models}_d(\phi)$ *denotes the set of all models of ϕ under the domain* $\mathbf{D} = \{1, \ldots, d\}$.

In order to define the WFOMC of the formula, we must first define the notion of a weighting.

Definition 2. *Denote the set of predicates appearing in ϕ by P_ϕ. A weighting on ϕ is a pair of mappings $w : P_\phi \to \mathbb{R}$ and $\bar{w} : P_\phi \to \mathbb{R}$.*

Definition 3. *Let $(w, \bar{w})$ be a weighting on ϕ. The weighted first-order model count of ϕ over a domain of size d under $(w, \bar{w})$ is:* $\mathsf{WFOMC}(\phi, d, w, \bar{w}) = \sum_{\mu \in \mathsf{models}_d(\phi)} \prod_{L \in \mu_T} w(\mathsf{pred}(L)) \cdot \prod_{L \in \mu_F} \bar{w}(\mathsf{pred}(L))$ *where μ_T denotes the set of true ground atoms in the model μ, and μ_F the false ground atoms. The notation* $\mathsf{pred}(L)$ *maps an atom L to its corresponding predicate name.*

2.3 Hashing-based Approximate Model Counting

The most obvious way to solve a WFOMC problem instance is to simply ground it out and pass it to a (weighted) propositional model counter. One algorithm for model counting that has enjoyed great success in recent years involves exploiting universal hash functions to get approximate counts. We explain this idea here, along with the drawbacks of this approach when applied to first-order problems.

Chakraborty *et al.* [2013] proposed an algorithm, ApproxMC, which uses XOR-based hash functions in order to obtain an approximate model count with arbitrary tolerance and confidence guarantees. The basic working principle of this idea involves adding a XOR constraint on a random subset of the variables appearing in the formula, which cuts the number of models approximately in half. After repeating this procedure a sufficient number of times, we may compute exactly the number of models in the constrained formula, and repeat this procedure a number of times to get a good sample of the size of an average "cell". Multiplying the median cell size by the number of cells created from imposing the XOR constraints then gives an approximation of the overall model count.

This work later led to the development of even more efficient model counters using the same underlying principle. In particular, ApproxMC2 [Chakraborty *et al.*, 2016] was developed which reduced the number of calls needed to a SAT oracle to logarithmic in the number of variables of the input. Finally, the latest revision, ApproxMC3 [Soos and Meel, 2019], was released, which processes the constructed CNF-XOR formulas in a more efficient manner.

Crucial to all of these tools is that they give PAC guarantees on the resulting model count. We follow the notation of the papers above and denote by R_F the set of models of a propositional formula F, and by $R_{F\downarrow S}$ the projection of R_F onto a subset S of variables in the formula.

Theorem 1 ([Chakraborty *et al.*, 2016])**.** *Given a formula F, sampling set $S \subseteq \mathsf{Vars}(F)$, a tolerance $\varepsilon > 0$, and a confidence $1 - \delta \in (0, 1]$, ApproxMC3 returns a count c such that $P(|R_{F\downarrow S}|/(1+\varepsilon) \leq c \leq (1+\varepsilon)|R_{F\downarrow S}|) \geq 1 - \delta$. Moreover, the number of SAT oracle calls required is $k \in \mathcal{O}\left(\frac{\log(|S|)\log(\frac{1}{\delta})}{\varepsilon^2}\right)$.*

This approach was extended to WISH [Ermon *et al.*, 2013] and WeightMC [Chakraborty *et al.*, 2014], which each leverage related techniques to allow for weighted model counting. In the latter paper, the authors identify a parameter, *tilt*, which is the ratio of the maximum weight of all satisfying assignments to the minimum weight of all satisfying assignments, and show that their procedure runs in time polynomial in the tilt of the input formula when equipped with a SAT oracle. However, the tilt of the grounding of a first-order formula can grow exponentially in the size of the domain.

Example 1. *Let* $\Phi = \forall x.Heads(x) \vee Tails(x) \wedge \forall x.\neg Heads(x) \vee \neg Tails(x)$*, and let* $w(Heads) = 0.5$*,* $\bar{w}(Heads) = 1$*, and* $w(Tails) = 0.1$*,* $\bar{w}(Tails) = 1$*. Let* $\mathbf{D} = \{coin_1, \ldots, coin_n\}$*. Then* $tilt(\Phi) = (\frac{0.5}{0.1})^n = 5^n$.

Thus, using WeightMC on a first-order model may require an exponential number of SAT queries in the size of the domain. Although Chakraborty *et al.* [2014] also describe (in

Section 6 of their paper) a way to theoretically reduce the runtime by adding constraints that split the space of solutions into regions with small enough tilt, they mention that this approach would require a pseudo-boolean solver capable of efficiently handling XOR constraints, so there is no practical implementation of this theoretical extension of WeightMC.

In this paper, we show how to build on unweighted hashing-based approaches to solve first-order problems in such a way that the number of SAT queries can be reduced to a number polynomial in the domain size. This gives us a practical algorithm that enables us to scale to problem instances that are too large for exact approaches, and for which there currently exist no other practical methods with PAC guarantees.

3 Algorithm

In this section, we first show how the WFOMC of a sentence can be decomposed into a series of terms by making use of cardinality constraints. Although the utility of this decomposition is limited in practice, it forms the basis for the next section, where we show how we can further take advantage of cardinality constraints to design an anytime algorithm, ApproxWFOMC, that computes bounds for the WFOMC. We will initially assume the existence of an FOMC oracle, and later explain how this can be realised in practice using the hashing-based approximate model counter ApproxMC3.

3.1 An Exact Decomposition of the WFOMC

We start by decomposing a WFOMC problem into a sum of terms.

Theorem 2. *Consider a sentence ϕ with predicates $P_1, \ldots, P_k$. Then the WFOMC of ϕ can be decomposed into a weighted sum of first-order model counts as:*

$$\mathsf{WFOMC}(\phi, d, w, \bar{w}) = \sum_{(n_1,\ldots,n_k)\in\mathcal{K}} \prod_{i=1}^{k} [\, w(P_i)^{n_i} \cdot \bar{w}(P_i)^{r_i - n_i} \cdot \mathsf{FOMC}(\phi \wedge \phi^{CARD}_{(n_1,\ldots,n_k)}, d) \,]$$

where $r_i = \mathsf{arity}(P_i)^d$, $\mathcal{K} = \{(n_1, \ldots, n_k) \mid n_i \in \{0, \ldots, r_i\}\}$, and $\phi^{CARD}_{(n_1,\ldots,n_k)}$ denotes the first-order cardinality constraint fixing every model of ϕ to have exactly n_i true instances of P_i.

The intuition behind Theorem 2 can be reasoned as follows: consider the case of a sentence ϕ with a single predicate P and a domain of size d, and suppose we add a cardinality constraint to ϕ to fix P to have precisely n true groundings. Then every model of ϕ with the cardinality constraint will have the same weight of $w(P)^n \bar{w}(P)^{d-n}$. The formula above generalises this to multiple predicates. In practice, however, the decomposition in Theorem 2 is typically too large to compute exactly, even though the number of terms grows polynomially in the size of the domain.

Remark 1. *The number of terms (and thus, FOMC oracle calls) in Theorem 2 for a sentence ϕ with k predicates over a domain of size d is: $M(\phi, d) = \prod_{i=1}^{k} (d^{\mathsf{arity}(P_i)} + 1)$.*

3.2 Approximating the WFOMC Using an Exact FOMC Oracle

Our approach, ApproxWFOMC, to bounding the value of $\mathsf{WFOMC}(\phi, d, w, \bar{w})$ is described in Algorithm 1. One begins by obtaining the coarsest bounds possible for the WFOMC. This is done by computing the unweighted FOMC and multiplying by the weight obtained when all groundings of each predicate are true, or the case when all are false, depending on which is larger. It is not difficult to see that this indeed gives valid bounds on the true WFOMC.

Example 2. *Consider again the coin toss example from Example 1, and fix a domain of size $d = 6$. We have $\mathsf{FOMC}(\phi) = 2^6 = 64$. Moreover, we know that the positive weights for both predicates are lower than their respective negative weights. Thus, we may compute the lower bound: $LB = w(Heads)^d \cdot w(Tails)^d \cdot \mathsf{FOMC}(\phi) = 0.5^6 \cdot 0.1^6 \cdot 64 = 10^{-6}$ and upper bound: $UB = \bar{w}(Heads)^d \cdot \bar{w}(Tails)^d \cdot \mathsf{FOMC}(\phi) = 1^6 \cdot 1^6 \cdot 64 = 64$. We therefore get the global bounds $(LB, UB) = (10^{-6}, 64)$ for the coarsest constraints possible $\{Heads \to (0, 6), Tails \to (0, 6)\}$.*

We then split the space by considering two possible cases for each weighted predicate: one where at most half of all groundings of the predicate are true, and one where at least half are true. Given p weighted predicates, we thus have $2p$ possible halves. Using an appropriately defined heuristic whose details are explained later, we may select one of the p predicates which corresponds to its two halves. For each half of the split, we can compute the FOMC using cardinality constraints, and bounds on the maximum and minimum possible weights for these regions can also be computed accordingly. Then, the upper and lower bound for the WFOMC for each half can be stored in a queue that is sorted according to another heuristic function on these bounds. Most importantly, the upper bounds and the lower bounds of the two non-overlapping halves can be used to improve the upper and lower bounds *UB* and *LB* that we have for $\mathsf{WFOMC}(\phi, d, w, \bar{w})$. Specifically, denoting the old upper and lower bounds (before splitting) as *u* and *l* and the new pair of upper and lower bounds by (l_1, u_1), and (l_2, u_2), we can update the upper bounds as $UB := UB - u + (u_1 + u_2)$ and $LB := LB - l + (l_1 + l_2)$.

Example 3. *We now take the constraints from Example 2 and split it into 4 possible subconstraints: $c_1 = \{Heads \to (0, 3), Tails \to (0, 6)\}$, $c_2 = \{Heads \to (4, 6), Tails \to (0, 6)\}$, $c_3 = \{Heads \to (0, 6), Tails \to (0, 3)\}$ and $c_4 = \{Heads \to (0, 6), Tails \to (4, 6)\}$. Note that the constraint pair c_1, c_2 (resp. c_3, c_4) describes non-overlapping regions and corresponds to a split on Heads (resp. $Tails$). Suppose our heuristic functions selects the predicate $Heads$. Then imposing each cardinality constraint in turn gives us $\mathsf{FOMC}(\phi \wedge \phi^{CARD}_{c_1}) = 42$ and $\mathsf{FOMC}(\phi \wedge \phi^{CARD}_{c_2}) = 22$. We may now follow a similar process as that in the last example and compute upper and lower bounds for each of these non-overlapping regions, and push these bounds along with their respective constraints onto a queue. We now also update our global bounds (LB, UB) on the WFOMC: suppose we compute the bounds (l_i, u_i) for each constraint c_i. Then we can tighten our bounds from $(10^{-6}, 64)$ to $(l_1 + l_2, u_1 + u_2) =$*

Algorithm 1 ApproxWFOMC

Input FO CNF ϕ, weights $(w, \bar{w})$, domain size d, tolerance τ
Output (b_1, b_2) s.t. $b_1 \leq \mathsf{WFOMC}(\phi, d, w, \bar{w}) \leq b_2$ and $\frac{b_2}{b_1} < 1 + \tau$

1: **/* Initialization */**:
2: *queue* $\leftarrow$ new priority queue
3: *LB*, *UB* $\leftarrow \mathsf{FOMC}(\phi, d)$
4: **for** P in $\mathsf{WeightedPredicates}(\phi)$ **do**
5: $\quad \xi \leftarrow d^{\mathsf{arity}(P)}$
6: $\quad$ *LB* $\leftarrow$ *LB* $\cdot \min(w(P)^{\xi}, \bar{w}(P)^{\xi})$
7: $\quad$ *UB* $\leftarrow$ *UB* $\cdot \max(w(P)^{\xi}, \bar{w}(P)^{\xi})$
8: $\quad$ *constraints*$[P] \leftarrow (0, \xi)$
9: Store (*constraints*, *newLb*, *newUb*) in *queue*
10: **/* Main loop */**:
11: **while** $\frac{newUb}{newLb} \geq 1 + \tau$ and *queue* is non-empty **do**
12: $\quad$ Pop (*constraints*, *lb*, *ub*) from *queue*
13: $\quad$ */* Constructing refined constraints (splitting) */*
14: $\quad$ **if** *constraints* cannot be split further **then**
15: $\quad\quad$ **continue**
16: $\quad O \leftarrow$ select optimal predicate according to heuristic
17: $\quad leftConstr, rightConstr \leftarrow constraints$
18: $\quad leftConstr[O] \leftarrow (l, \lfloor \frac{l+u}{2} \rfloor)$
19: $\quad rightConstr[O] \leftarrow (\lfloor \frac{l+u}{2} \rfloor + 1, u)$
20: $\quad$ */* Recomputing LB and UB using split constraints */*
21: $\quad$ *LB* $\leftarrow$ *LB* $- lb$
22: $\quad$ *UB* $\leftarrow$ *UB* $- ub$
23: $\quad$ **for** *refinedConstr* in $\{leftConstr, rightConstr\}$ **do**
24: $\quad\quad lo,\ hi \leftarrow 1$
25: $\quad\quad$ **for** P in $\mathsf{WeightedPredicates}(\phi)$ **do**
26: $\quad\quad\quad \xi \leftarrow d^{\mathsf{arity}(P)}$
27: $\quad\quad\quad (l, u) \leftarrow refinedConstr[P]$
28: $\quad\quad\quad lo \leftarrow lo \cdot \min(w(P)^l \bar{w}(P)^{\xi - l}, w(P)^u \bar{w}(P)^{\xi - u})$
29: $\quad\quad\quad hi \leftarrow hi \cdot \max(w(P)^l \bar{w}(P)^{\xi - l}, w(P)^u \bar{w}(P)^{\xi - u})$
30: $\quad\quad mc \leftarrow \mathsf{FOMC}(\phi \wedge \phi^{CARD}_{refinedConstr}, d)$
31: $\quad\quad$ *LB* $\leftarrow$ *LB* $+ lo \cdot mc$
32: $\quad\quad$ *UB* $\leftarrow$ *UB* $+ hi \cdot mc$
33: $\quad\quad$ Push $(refinedConstr, lo \cdot mc, hi \cdot mc)$ to *queue*
return (*LB*, *UB*)

$(6.5625 \times 10^{-4} + 3.4375 \times 10^{-7}, 42 + 2.2 \times 10^{-3}) = (6.5659375 \times 10^{-4}, 42.0022)$.

The first element is then popped from the queue, and the procedure repeats until the bounds are sufficiently tight.

Details

The pseudocode shown in Algorithm 1 uses some notation that has not been described yet. We provide the details here. The function $\mathsf{WeightedPredicates}(\phi)$ returns the set of all non-neutral predicates in ϕ (i.e. all predicates having a positive or negative weight other than 1). The notation ϕ^{CARD}_a denotes the first-order formula imposing the cardinality constraints contained in the dictionary a. For example, if $a = \{P_1 \to (0, 1), P_2 \to (0, 2)\}$, then ϕ^{CARD}_a would impose the constraint that predicates P_1 and P_2 have at most one and two true groundings respectively.

The "optimal" predicate on line 16 is selected according to the following heuristic: first, refine the bounds for WFOMC on the currently processed region by updating the bounds using the binary split on that predicate (as shown above). Then select the predicate which minimises $\ln U_i - \ln L_i$.

The priority queue is sorted in decreasing order according to another heuristic function on the elements: given a tuple $(constraints, lb, ub)$, its heuristic is computed as $ub - lb$. It thus splits regions with the largest gap between upper and lower bounds first.

3.3 Approximating the WFOMC Using an Approximate FOMC Oracle

In practice we may only have access to an approximate FOMC oracle rather than an exact one: for example, in our implementation we ground the input sentence and use ApproxMC3 to provide such an oracle. In this case, in order to provide ε-δ style guarantees in ApproxWFOMC, we need to set the correct parameters to ApproxMC3.

Theorem 3. *Given a sentence ϕ, let $(LB, UB) =$ ApproxWFOMC$(\phi, w, \bar{w}, d, \tau)$. Suppose each FOMC oracle call is made by grounding the problem and calling ApproxMC3 with tolerance ε and confidence δ_i, where i is the number of oracle calls made so far on lines 3 and 30, and let $\delta = \sum_i \delta_i$. Then we have:*

$$Pr\left[\frac{LB}{1+\varepsilon} \leq \mathsf{WFOMC}(\phi, w, \bar{w}, d) \leq UB\,(1+\varepsilon)\right] \geq 1 - \delta.$$

Proof. Let M denote the number of calls to ApproxMC3 made by ApproxWFOMC and let c_i denote the output of the i-th call to ApproxMC3. Observe that at any point in ApproxWFOMC's run both *LB* and *UB* are weighted sums of the outputs of ApproxMC3: $UB = \sum_{i=1}^{M} \gamma_i \cdot c_i$ and $LB = \sum_{i=1}^{M} \gamma'_i \cdot c_i$ (the values of the coefficients γ_i and γ'_i are not important for the purposes of the proof). Next let c^*_i denote the true model count corresponding to the approximate value c_i returned by ApproxMC3 and let $UB^* = \sum_{i=1}^{M} \gamma_i \cdot c^*_i$ and $LB^* = \sum_{i=1}^{M} \gamma'_i \cdot c^*_i$ be the respective bounds returned by ApproxWFOMC. It follows from the guarantees on ApproxMC3 (Theorem 1) that the probability that $c_i \notin [\frac{c^*_i}{1+\varepsilon}, (1+\varepsilon) \cdot c^*_i]$ is no greater than δ_i. Then by the union bound, we have that the probability that at least one $c_i \notin [\frac{c^*_i}{1+\varepsilon}, (1+\varepsilon) \cdot c^*_i]$ is at most $\delta = \sum_{i=1}^{M} \delta_i$. Hence, with probability at least $1 - \delta$ it holds $LB \leq (1+\varepsilon) LB^*$ and $\frac{UB^*}{1+\varepsilon} \leq UB$ from which we then have $\frac{LB}{(1+\varepsilon)} \leq LB^*$ and $UB^* \leq (1+\varepsilon) UB$. Next it follows from a simple inspection of the pseudocode of ApproxWFOMC that $LB^* \leq \mathsf{WFOMC}(\phi, w, \bar{w}, d) \leq UB^*$. Using the probabilistic bounds just derived for *LB* and *UB*, we obtain that $\frac{LB}{1+\varepsilon} \leq \mathsf{WFOMC}(\phi, w, \bar{w}, d) \leq UB\,(1+\varepsilon)$ with probability at least $1 - \delta$. □

If we do not want to bother the user with setting τ and ε separately, we can also just ask for δ and τ and then set $\varepsilon' := \sqrt[3]{1+\tau} - 1$ and $\tau' := \sqrt[3]{1+\tau} - 1$, then call ApproxWFOMC with the parameters ε', τ' and δ. The algorithm will return numbers *LB* and *UB*: we set $LB' := LB/(1+\varepsilon')$ and $UB' := (1+\varepsilon') UB$ and return these two numbers to the user. It follows from Theorem 1 (and simple algebraic manipulations) that we are then guaranteed that, with probability at least $1 - \delta$, the

following holds: $LB' \leq \mathsf{WFOMC}(\phi, w, \bar{w}, d) \leq UB'$ and $UB'/LB' \leq 1 + \tau$.

One remaining question is how to set the values of the individual δ_i's. One obvious possibility is to set $\delta_i := \delta/M(\phi, d)$ where $M(\phi, d)$ is the theoretical maximum number of calls to ApproxMC3 defined as in Remark 1, and δ is the confidence parameter set by the user. However, we can do better, by exploiting the fact that in most cases the algorithm will need fewer calls to ApproxMC3 than the worst-case upper bound.

Proposition 1. *For the PAC bounds in Theorem 3 to hold for a given $\delta > 0$, it suffices to set $\delta_i := \delta/(i \cdot (\ln M(\phi, d) + 1))$.*

Proof. Since the sum of the first $M(\phi, d)$ elements of the harmonic series is at most $\ln(M(\phi, d)) + 1$, we will always have $\sum_{i=1}^{M} \delta_i \leq \delta$ for $M \leq M(\phi, d)$. □

Proposition 2. *Let M be the number of calls to ApproxMC3 made by a run of ApproxWFOMC with δ_i's as defined in Proposition 1. Then in total $\mathcal{O}\left(M \cdot \frac{\log(|S|)\cdot(\log(\frac{1}{\delta})+\log M+\log(\ln M(\phi,d)+1))}{\varepsilon^2}\right)$ calls to a SAT oracle will be made by ApproxWFOMC.*

Proof. Follows directly from Theorem 1 and Proposition 1. □

Cardinality constraints. One point that has not yet been explained is how the cardinality constraints in the algorithm can be implemented in practice. Cardinality constraints express bounds on the number of true instances of members of a set of propositions: for ApproxWFOMC, we are interested in expressing this constraint on the number of true groundings of a first-order predicate. As we use ApproxMC3 to provide an FOMC oracle by grounding out first-order formulas, we may express this constraint in propositional form. In our implementation, we employ an efficient encoding by Bailleux and Boufkhad [2003]. It adds $\mathcal{O}(n \log n)$ auxiliary variables and $\mathcal{O}(n^2)$ additional clauses of length at most 3, where n is the size of the constrained variable set.

4 Implementation and Experiments

We have implemented our algorithm and tested it on the encodings of two Markov logic network (MLN) instances [Richardson and Domingos, 2006]. Inference in the particular networks we use is known to be especially challenging, as the corresponding formulas are conjectured to not be liftable [Beame *et al.*, 2015]. We first review how computation of the partition function can be cast as a WFOMC task, and follow with an analysis of our experimental results. The FOMC oracle is implemented using ApproxMC3 as described earlier. Throughout this section, we fix $\epsilon = 0.8$, $\delta = 0.2$, and $\tau = 0.5$. All experiments were performed on a computer with a six-core Intel i7 2.2GHz processor and 16 GB of RAM.

4.1 Encoding an MLN

Recall that an MLN comprises a set of tuples (w, ϕ), where w is a real-valued weight and and ϕ is a first-order formula. For example, consider the MLN below, "`transitive-smokers-mln`":

$$1.22 \quad \texttt{stress}(X) \rightarrow \texttt{smokes}(X)$$
$$2.08 \quad \texttt{friend}(X, Y) \wedge \texttt{smokes}(X) \rightarrow \texttt{smokes}(Y)$$
$$0.69 \quad \texttt{friend}(X, Y) \wedge \texttt{friend}(X, Z) \rightarrow \texttt{friend}(X, Z)$$

The first rule states that people who are stressed are likely to smoke. The second states that friends of smokers tend to also be smokers. The last rule states that the `friends` relation is typically transitive. Note that it is the presence of this transitivity rule that is conjectured to make the problem not domain-liftable [Beame *et al.*, 2015]. That is, unlike in the standard "`smokers-mln`", inference in the "`transitive-smokers-mln`" is hard. In order to create a second even more challenging network, we also extended the MLN above with the following two rules to create "`transitive-smokers-drinkers-mln`":

$$2 \quad \texttt{stress}(X) \rightarrow \texttt{drinks}(X)$$
$$1.5 \quad \texttt{friend}(X, Y) \wedge \texttt{drinks}(X) \rightarrow \texttt{drinks}(Y)$$

Definition 4 ([Van den Broeck *et al.*, 2014])**.** *The WFOMC encoding $(\Delta, w, \bar{w})$ of an MLN is constructed as follows: for each tuple $(w_i, \phi_i(\boldsymbol{x}_i))$ in the MLN, where $\boldsymbol{x}_i$ denotes the free logical variables occurring in ϕ_i, we introduce an auxiliary predicate $P_i/|\boldsymbol{x}_i|$. Then for each formula in the MLN, Δ is formed by conjoining the sentences $\forall \boldsymbol{x}_i P_i \leftrightarrow \phi_i(\boldsymbol{x}_i)$. The weighting is defined by setting $w(P_i) = e^{w_i}$, $\bar{w}(P_i) = 1$, and $w(Q) = \bar{w}(Q) = 1$ for all other predicates Q.*

The encoding of the first rule of `transitive-smokers-mln` earlier is therefore:

$$\forall X P_1(X) \leftrightarrow (\texttt{stress}(X) \rightarrow \texttt{smokes}(X))$$

with $w(P_1) = e^{1.22}$, $\bar{w}(P_1) = 1$, and $w(\texttt{stress}) = w(\texttt{smokes}) = \bar{w}(\texttt{stress}) = \bar{w}(\texttt{smokes}) = 1$.

We can take advantage of some domain-specific knowledge of the MLN encoding in order to further optimize our algorithm when computing the partition function of an MLN. We first recall the definition of an (in)dependent support.

Definition 5 ([Ivrii *et al.*, 2016])**.** *Let F denote a propositional formula, and let X denote the set of variables appearing in F. Then $I \subseteq X$ is said to be an independent support of F if, for any two models $\sigma_1, \sigma_2 \in R_F$ that agree on I, we have $\sigma_1 = \sigma_2$. In other words, the truth values of I uniquely determine the truth value of every variable in $X \setminus I$. The remaining variables $X \setminus I$ are called a dependent support.*

Remark 2. *Let $(\Delta, w, \bar{w})$ denote the WFOMC encoding of an MLN. Then, after grounding Δ over some domain $\boldsymbol{D}$, the ground instances of all non-auxiliary predicates form an independent support for the grounding of Δ.*

Based on the observation in Lemma 2, we may pass the ground instances of all non-auxiliary predicates as a *sampling set* to every call of ApproxMC3 in ApproxWFOMC, and perform projected model counting. This has the effect of shortening the XOR constraints which must be processed by ApproxMC3, and provides a significant speed-up to the model counting times. Finally, observe that all non-auxiliary predicates have neutral weight, so constraints will only be imposed on the auxiliary predicates.

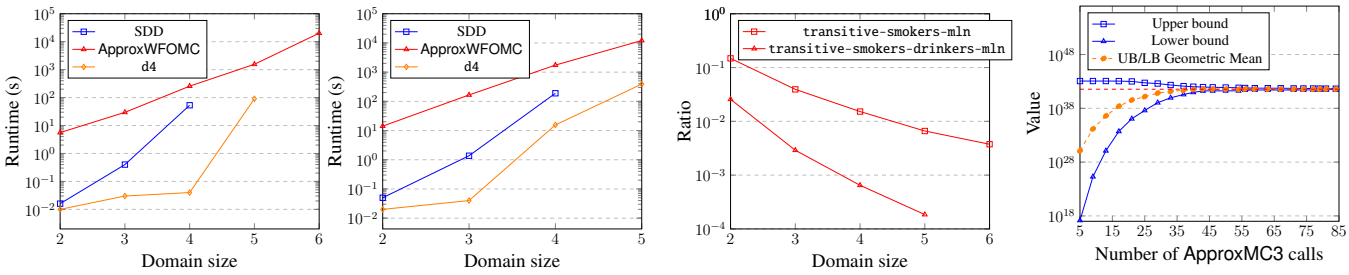

Figure 1: (a) Runtime of different WFOMC methods for the `transitive-smokers-mln` problem with various domain sizes. SDD compilation runs out of memory for domain sizes larger than 4. `d4` runs out of memory for domain sizes larger than 5. (b) Same as (a) for `transitive-smokers-drinkers-mln`. SDD compilation runs out of memory for domain sizes larger than 4. (c) Ratio of ApproxMC3 calls made by ApproxWFOMC (until convergence) to FOMC oracle calls in the decomposition in Theorem 2. (d) Bounds obtained by ApproxWFOMC versus true WFOMC for varying number of ApproxMC3 calls, for a representative domain size $d = 4$ on the `transitive-smokers-mln` problem. The true WFOMC is indicated by the red dashed line.

4.2 Experimental Results

We tested ApproxWFOMC on the encodings of the `transitive-smokers-mln` and `transitive-smokers-drinkers-mln` networks, and set out to answer the following questions:

1. How does the performance of ApproxWFOMC on first-order probabilistic models compare to solving the same problem using exact knowledge compilation?
2. How significant of an improvement does the search method proposed by ApproxWFOMC yield over the decomposition in Theorem 2, in terms of the number of FOMC oracle calls?
3. How quickly do the bounds converge as the number of FOMC oracle calls made by ApproxWFOMC increases?

We investigate each question individually.

Q1 In Figures 1a and 1b, we show how the domain size affects the runtime of ApproxWFOMC and compare it to the SDD library [Choi and Darwiche, 2013] and the Decision-DNNF compiler `d4` [Lagniez and Marquis, 2017]. Note that a small increase in domain size makes the resulting inference problem significantly harder, as the number of Boolean random variables in the joint distribution grows quadratically. Although SDDs outperform ApproxWFOMC at small domain sizes ($d = 2$, 3 and 4), ApproxWFOMC performs better with larger domains, with SDD compilation running out of memory already with a domain of size 5 on both networks. `d4` was more resilient, and is able to handle domains of size up to 5 (but no higher) in the `smokers` network, and all domain sizes we tested up to 5 in the `smokers-drinkers` network. We also tried to experiment with WeightMC but the *tilt* parameter (cf. Section 2.3) was so high that it rendered WeightMC unusable beyond a domain of size 2.

Q2 In Figure 1c, we show the efficiency gain of ApproxWFOMC over using a naïve decomposition of the form in Theorem 2, by quantifying the ratio between the number of FOMC oracle calls made by ApproxWFOMC to the number needed in the decomposition. In both networks the "efficiency ratio" improves significantly as domain size increases, showing that the heuristic-guided search becomes increasingly effective in finding dense regions in the space of weighted models.

Q3 In Figure 1d, we measure how quickly ApproxWFOMC converges to the true WFOMC for a representative domain size $d = 4$. The bounds rapidly approach a value near that of the true WFOMC (indicated by the red dashed line), meaning that most ApproxMC3 calls are made when the bounds are already relatively close to convergence. This suggests that ApproxWFOMC is well-suited to obtaining rough bounds very quickly (corresponding to large values of τ).

5 Conclusion

We introduced ApproxWFOMC, an anytime approximate WFOMC algorithm with PAC guarantees, and showed how it can be applied to inference in first-order probabilistic models. Results show that it is able to scale to domain sizes that are too large for existing exact methods. Other approximate methods such as MCMC either need exponential time to converge, or are used in practice in such a way that does not guarantee convergence to the stationary distribution (and can therefore give drastically incorrect results). Our method is somewhere in between these: it comes with guarantees, and scales beyond what exact methods can do. Even though a lot of future work is needed before methods like ours bring us efficient inference in large domains, we believe that this paper is an important first step in that direction. There are still several avenues for further research, including tighter integration between the cardinality constraints and ApproxMC3, and application of our method to inference in other statistical relational learning systems such as ProbLog [Fierens *et al.*, 2015].

Acknowledgements

TvB's work is supported by the Research Foundation – Flanders (grant G095917N). OK's work is supported by the OP VVV project *CZ.02.1.01/0.0/0.0/16_019/0000765* "Research Center for Informatics" and a donation from X-Order Lab. Part of this work was done while OK was already supported by the Czech Science Foundation project "Generative Relational Models" (20-19104Y).

References

[Bailleux and Boufkhad, 2003] Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of boolean cardinality constraints. In *CP*, volume 2833 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2003.

[Beame *et al.*, 2015] Paul Beame, Guy Van den Broeck, Eric Gribkoff, and Dan Suciu. Symmetric weighted first-order model counting. In *PODS*, pages 313–328. ACM, 2015.

[Chakraborty *et al.*, 2013] Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. A scalable approximate model counter. In *CP*, volume 8124 of *Lecture Notes in Computer Science*, pages 200–216. Springer, 2013.

[Chakraborty *et al.*, 2014] Supratik Chakraborty, Daniel J. Fremont, Kuldeep S. Meel, Sanjit A. Seshia, and Moshe Y. Vardi. Distribution-aware sampling and weighted model counting for SAT. In *AAAI*, pages 1722–1730. AAAI Press, 2014.

[Chakraborty *et al.*, 2016] Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. Algorithmic improvements in approximate counting for probabilistic inference: From linear to logarithmic SAT calls. In *IJCAI*, pages 3569–3576. IJCAI/AAAI Press, 2016.

[Chavira and Darwiche, 2008] Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artif. Intell.*, 172(6-7):772–799, 2008.

[Choi and Darwiche, 2013] Arthur Choi and Adnan Darwiche. Dynamic minimization of sentential decision diagrams. In *AAAI*. AAAI Press, 2013.

[Ermon *et al.*, 2013] Stefano Ermon, Carla P. Gomes, Ashish Sabharwal, and Bart Selman. Taming the curse of dimensionality: Discrete integration by hashing and optimization. In *ICML (2)*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 334–342. JMLR.org, 2013.

[Fierens *et al.*, 2015] Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. Inference and learning in probabilistic logic programs using weighted boolean formulas. *TPLP*, 15(3):358–401, 2015.

[Gogate and Domingos, 2011] Vibhav Gogate and Pedro M. Domingos. Probabilistic theorem proving. In *UAI*, pages 256–265. AUAI Press, 2011.

[Ivrii *et al.*, 2016] Alexander Ivrii, Sharad Malik, Kuldeep S. Meel, and Moshe Y. Vardi. On computing minimal independent support and its applications to sampling and counting. *Constraints*, 21(1):41–58, 2016.

[Lagniez and Marquis, 2017] Jean-Marie Lagniez and Pierre Marquis. An improved Decision-DNNF compiler. In *IJCAI*, pages 667–673. ijcai.org, 2017.

[Muise *et al.*, 2012] Christian J. Muise, Sheila A. McIlraith, J. Christopher Beck, and Eric I. Hsu. Dsharp: Fast d-DNNF compilation with sharpSAT. In *Canadian Conference on AI*, volume 7310 of *Lecture Notes in Computer Science*, pages 356–361. Springer, 2012.

[Oztok and Darwiche, 2015] Umut Oztok and Adnan Darwiche. A top-down compiler for sentential decision diagrams. In *IJCAI*, pages 3141–3148. AAAI Press, 2015.

[Richardson and Domingos, 2006] Matthew Richardson and Pedro M. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[Soos and Meel, 2019] Mate Soos and Kuldeep S. Meel. BIRD: engineering an efficient CNF-XOR SAT solver and its applications to approximate model counting. In *AAAI*, pages 1592–1599. AAAI Press, 2019.

[Van den Broeck *et al.*, 2011] Guy Van den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt. Lifted probabilistic inference by first-order knowledge compilation. In *IJCAI*, pages 2178–2185. IJCAI/AAAI, 2011.

[Van den Broeck *et al.*, 2014] Guy Van den Broeck, Wannes Meert, and Adnan Darwiche. Skolemization for weighted first-order model counting. In *KR*. AAAI Press, 2014.

[Van den Broeck, 2011] Guy Van den Broeck. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *NIPS*, pages 1386–1394, 2011.