Fast Algorithms for Relational Marginal Polytopes

Yuanhong Wang[1] , **Timothy van Bremen**[2] , **Juhua Pu**[1,3*] , **Yuyi Wang**[4] and **Ondřej Kuželka**[5*]
[1]State Key Laboratory of Software Development Environment & Engineering Research Center of ACAT, Ministry of Education. Beihang University, China
[2]KU Leuven, Belgium
[3]Research Institute of Beihang University in Shenzhen, China
[4]CRRC Zhuzhou Institute, China & ETH Zurich, Switzerland
[5]Czech Technical University in Prague, Czech Republic
{lucienwang, pujh}@buaa.edu.cn, timothy.vanbremen@cs.kuleuven.be, yuyiwang920@gmail.com, ondrej.kuzelka@fel.cvut.cz

Abstract

We study the problem of constructing the relational marginal polytope (RMP) of a given set of first-order formulas. Past work has shown that the RMP construction problem can be reduced to weighted first-order model counting (WFOMC). However, existing reductions in the literature are intractable in practice, since they typically require an infeasibly large number of calls to a WFOMC oracle. In this paper, we propose an algorithm to construct RMPs using fewer oracle calls. As an application, we also show how to apply this new algorithm to improve an existing approximation scheme for WFOMC. We demonstrate the efficiency of the proposed approaches experimentally, and find that our method provides speedups over the baseline for RMP construction of a full order of magnitude.

1 Introduction

We study the construction of *relational marginal polytopes* [Kuzelka *et al.*, 2018], which are objects that naturally arise in the study of Markov logic networks [Richardson and Domingos, 2006] and other statistical relational learning models. Informally, given first-order logic formulas $\alpha_1, \alpha_2, \ldots, \alpha_k$, which may contain free variables, and a set of constants $\{c_1, c_2, \ldots, c_n\}$, called a *domain*, the respective *relational marginal polytope* is the convex hull of the set of points obtained by taking a possible world ω on the given domain, counting the groundings of the formulas α_1, α_2, $\ldots$, α_k that are true in ω, and repeating this for all possible ω's. For instance, for the formulas $\alpha_1 = sm(x)$ and $\alpha_2 = sm(x) \wedge friends(x, y) \wedge sm(y)$, the points defining the polytope would be given by the numbers of smokers in the population (the formula α_1) and the number of pairs of people who are friends and, at the same time, both smoke (the formula α_2). Relational marginal polytopes can also be seen as "lifted" counterparts of standard marginal polytopes studied in the probabilistic graphical models literature (see, e.g., [Roughgarden and Kearns, 2013; Sontag and Jaakkola, 2008]). Relational marginal polytopes have already found applications, among others, in polynomial-time algorithms for maximum-likelihood learning [Kuzelka and Kungurtsev, 2019; Kuzelka *et al.*, 2020].

Kuzelka and Wang [2020] recently showed that, roughly speaking, if computing the partition function of a Markov logic network given by formulas $\alpha_1, \ldots, \alpha_k$ can be done in time polynomial in the domain size, then the relational marginal polytope for the same formulas can be constructed in polynomial time. On the one hand, this is an important result, because it allows one to exploit existing lifted inference algorithms for weighted first-order model counting (WFOMC) (such as that of [Van den Broeck *et al.*, 2011]) for the construction of relational marginal polytopes. On the other hand, despite being polynomial in the domain size, the algorithm proposed in [Kuzelka and Wang, 2020] is not practical. In this paper we propose an approach that uses significantly fewer calls to a WFOMC oracle than the aforementioned algorithm. This new algorithm also outperforms a subsequent algorithm based on discrete Fourier transforms proposed in [Kuzelka, 2020]. Moreover, as a secondary contribution, we show how an efficient algorithm for relational marginal polytope construction such as the one proposed here can be applied to speed up **ApproxWFOMC** [van Bremen and Kuzelka, 2020], a recent approach for approximate weighted first-order model counting.

2 Preliminaries

This section briefly reviews the syntax and semantics of Markov logic networks and weighted first-order model counting.

2.1 Markov Logic Networks

We consider a function-free first-order logic defined by a set of constants Δ, called a *domain*, a set of variables $\mathcal{V}$ and a set $\mathcal{R}_k$ of k-ary predicates for each $k \in \mathbb{N}$. An atom takes the form $r(a_1, \ldots, a_k)$ with $a_1, \ldots, a_k \in \Delta \cup \mathcal{V}$ and $r \in \mathcal{R}_k$. A literal is an atom or its negation. 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 formula in which

*Corresponding author

none of its literals contains any variables is called ground. The set of grounding substitutions $\Theta(\alpha, \Delta) = \{\theta_1, \ldots, \theta_m\}$ of a formula α w.r.t. a domain Δ is the set of substitutions on all free variables occurring in α using constants from Δ. A possible world ω is represented as a set of ground atoms that are true in ω. The satisfaction relation $\models$ is defined in the usual way: $\omega \models \alpha$ means that the formula α is true in ω, as per the standard semantics of first-order logic.

A *Markov logic network* (MLN) [Richardson and Domingos, 2006] is a set of weighted first-order logic formulas (α, w), where $w \in \mathbb{R}$ and α is a first-order formula. An MLN Φ induces a probability distribution over possible worlds $\omega \in \Omega : p_\Phi(\omega) = \frac{1}{Z} \exp\left(\sum_{(\alpha,w)\in\Phi} w \cdot N(\alpha, \omega)\right)$, where $N(\alpha, \omega) := \sum_{\theta\in\Theta(\alpha,\Delta)} \mathbb{1}(\omega \models \alpha\theta)$ is the number of groundings of α satisfied in ω, and Z, called the *partition function*, is a normalization constant to ensure that p_Φ is a probability distribution. We also allow infinite weights. A formula α with infinite weight ∞ is understood as a hard constraint imposing that all worlds ω in which $N(\alpha, \omega)$ is not maximal have zero probability. A literal with infinite weight is called an *evidence literal*. We denote by $vars(\alpha)$ the number of free variables in the formula α.

2.2 Inference Using WFOMC

The marginal inference task in MLNs can be reduced to *weighted first-order model counting* (WFOMC).

Definition 1 (WFOMC, [Van den Broeck *et al.*, 2011])**.** *Let $w(r)$ and $\bar{w}(r)$ be functions mapping predicates to complex numbers, Γ a sentence, and Δ a set of constants. Then*

$$\mathtt{WFOMC}(\Gamma, w, \bar{w}, \Delta) := \sum_{\omega\in\Omega_\Delta:\omega\models\Gamma} \prod_{a\in P(\omega)} w(Pred(a)) \cdot \prod_{a\in N(\omega)} \bar{w}(Pred(a)),$$

where Ω_Δ is the set of all possible worlds on the domain Δ (using the predicates in Γ), $P(\omega)$ and $N(\omega)$ denote the positive literals that are true and false in ω, respectively, and $Pred(a)$ denotes the predicate of a (e.g., Pred(friends(Alice, Bob)) = friends).

We may proceed as in [Van den Broeck *et al.*, 2011] to compute the partition function Z of a given MLN using WFOMC. Given an MLN Φ, for every weighted formula $(\alpha_i, w_i) \in \Phi$, where the free variables in α_i are exactly $\mathbf{x} = \{x_1, \ldots, x_k\}$ and $w \neq \infty$, we create a new formula $\forall \mathbf{x} : \xi_i(\mathbf{x}) \leftrightarrow \alpha_i(\mathbf{x})$ where ξ_i is a new predicate. When α_i has $w = \infty$, we instead create a new formula $\forall \mathbf{x} : \alpha_i(\mathbf{x})$. We denote the conjunction of the resulting set of sentences by Γ and set the weight function to be $w(\xi_i) = \exp(w_i)$ and $\bar{w}(\xi_i) = 1$, and for all other predicates we set both w and $\bar{w}$ to be 1. It is easy to check that $\mathtt{WFOMC}(\Gamma, w, \bar{w}, \Delta) = Z$.

Importantly, there are classes of first-order logic theories for which weighted first-order model counting can be performed in time polynomial in the domain size $|\Delta|$. In particular, as shown in [Van den Broeck *et al.*, 2014], this is the case when the theory in question consists only of first-order sentences each containing at most two logical variables. In statistical relational learning, the term used for classes of problems that allow such a polynomial-time algorithm is *domain liftability*.

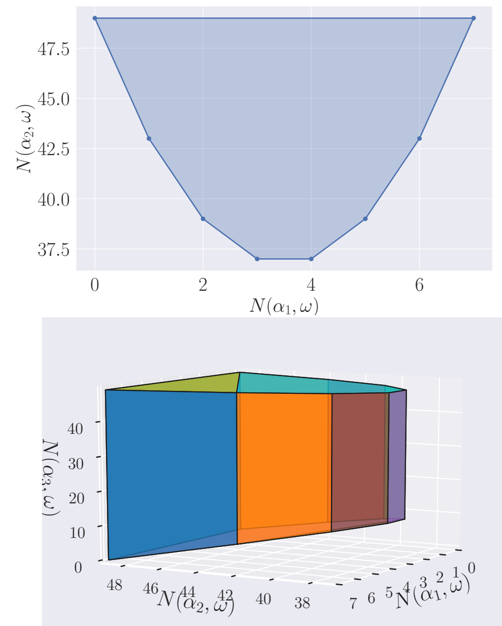


Figure 1: Examples of RMP of "friends-smokers" formulas Ψ_1 (left) and Ψ_2 (right) for domain size 7 (best viewed in color).

3 Relational Marginal Polytopes

Here we define relational marginal polytopes (RMP), which are the main focus of this paper. Intuitively, an RMP represents the possible expected values for the vectors of grounding counts of a given set of formulas.[1]

Definition 2 (Relational Marginal Polytope, [Kuzelka *et al.*, 2018])**.** *Let Δ be a set of constants and $\Psi = (\alpha_1, \ldots, \alpha_m)$ be a set of formulas. The relational marginal polytope of Ψ on Δ is defined as $\mathtt{RMP}(\Psi, \Delta) := \{(x_1, \ldots, x_m) \in \mathbb{R}^m \mid \exists dist.\ on\ \Omega_\Delta\ s.t.\ \mathbb{E}[N(\alpha_1, \omega)] = x_1 \wedge \cdots \wedge \mathbb{E}[N(\alpha_m, \omega)] = x_m\}$, where Ω_Δ is the set of all possible worlds on the domain Δ using the predicates in Ψ.*

3.1 RMPs as Convex Polytopes

The definition of $\mathtt{RMP}(\Psi, \Delta)$ is equivalent to the *convex hull* of the set of integral points $\{(N(\alpha_1, \omega), \ldots, N(\alpha_m, \omega)) \mid \omega \in \Omega_\Delta\}$. For simplicity, we denote $\mathbf{N}(\Psi, \omega) = (N(\alpha_1, \omega), \ldots, N(\alpha_m, \omega))$.

Example 1. *Consider "friends-smokers" formulas $\alpha_1 := sm(x)$, $\alpha_2 := fr(x, y) \wedge sm(x) \Rightarrow sm(y)$ and $\alpha_3 := friends(x, y)$ and denote $\Psi_1 = \{\alpha_1, \alpha_2\}$ and $\Psi_2 = \{\alpha_1, \alpha_2, \alpha_3\}$. Let Δ be a domain of size 7. The two relational marginal polytopes $\mathtt{RMP}(\Psi_1, \Delta)$ and $\mathtt{RMP}(\Psi_2, \Delta)$ are shown in Figure 1.*

A convex polytope $\mathcal{P}$ can be specified by a set of non-redundant *bounding half-spaces* $\{\langle \mathbf{a}_1, \mathbf{x}\rangle \leqslant b_1, \ldots, \langle \mathbf{a}_M, \mathbf{x}\rangle \leqslant b_M\}$, called the H-representation [Fukuda,

[1] Kuzelka *et al.* [2018] define a rescaled version of the polytope presented here.

2004]. Here we review some important terminologies of polytopes used in this paper.

Definition 3 (Bounding half-space). *A half-space $\langle \mathbf{a}, \mathbf{x} \rangle \leqslant b$ is called valid for $\mathcal{P}$ if $\langle \mathbf{a}, \mathbf{x} \rangle \leqslant b$ holds for all $\mathbf{x} \in \mathcal{P}$. A half-space $\langle \mathbf{a}, \mathbf{x} \rangle \leqslant b$ is a bounding half-space of the convex polytope $\mathcal{P}$, if it is valid for $\mathcal{P}$ and the intersection of its hyperplane $\langle \mathbf{a}, \mathbf{x} \rangle = b$ with $\mathcal{P}$ is non-empty.*

Definition 4 (Supporting hyperplane). *A supporting hyperplane of a polytope is the hyperplane of any of its bounding half-spaces.*

Definition 5 (Vertex). *A vertex of a polytope $\mathcal{P}$ is a 0-dimensional intersection of $\mathcal{P}$ with any of its supporting hyperplanes.*

3.2 A Lifted Reduction to WFOMC

Though constructing relational marginal polytopes in practice can be quite difficult, Kuzelka and Wang [2020] provide a lifted[2] reduction from this problem to WFOMC. They first enumerate the normal vectors of all possible bounding half-spaces of $\mathbf{RMP}(\Psi, \Delta)$. Since $\mathbf{N}(\Psi, \omega)$ can only take values in $\times_{\alpha \in \Psi} \{1, 2, \ldots, vars(\alpha)\}$, the number of normal vectors to be checked is at most $|\Delta|^{m \sum_{\alpha \in \Psi} vars(\alpha)}$. For each candidate normal vector $\mathbf{a}$, they use a version[3] of Lemma 1 below to find the integer b for the bounding half-space $\langle \mathbf{a}, \mathbf{x} \rangle \leqslant b$ by computing the partition function of an MLN.

Lemma 1 (Based on Theorem 3 in [Kuzelka and Wang, 2020]). *Given a normal vector $\mathbf{a} \in \mathbb{R}^m$ and a set of formulas $\Psi = \{\alpha_1, \ldots, \alpha_m\}$ over a domain Δ, let Z be the partition function of the MLN $\{(\alpha_i, 2a_i \ln |\Omega_\Delta|) \mid 1 \leqslant i \leqslant m\}$. Then the half-space $\langle \mathbf{a}, \mathbf{x} \rangle \leqslant \lceil \frac{\ln Z / \ln |\Omega_\Delta| - 1}{2} \rceil$ is a bounding half-space of $\mathbf{RMP}(\Psi, \Delta)$.*

When all bounding half-spaces of $\mathbf{RMP}(\Psi, \Delta)$ are available, constructing its H-representation is not a difficult problem. The problem is equivalent to removing redundant inequalities from a system of linear inequalities. A simple solution, which is employed by Kuzelka and Wang [2020], is to iteratively check whether a bounding half-space is redundant using linear programming (LP) (*c.f.* Lemma 2 therein). Since the number of enumerated bounding half-spaces is polynomial w.r.t. the domain size, they claim that if the MLNs used in Lemma 1 are domain-liftable, computing $\mathbf{RMP}(\Psi, \Delta)$ can be done in time polynomial in $|\Delta|$.

However, as we discussed above, the method that they proposed requires $|\Delta|^{m \sum_{\alpha \in \Psi} vars(\alpha)}$ WFOMC oracle calls, which is usually infeasible in practice. Although the complexity was later improved to $(|\Delta| + 1)^{\sum_{\alpha \in \Psi} vars(\alpha)} + 1$ calls using a technique based on computing the discrete Fourier transform (DFT) of the count distribution of an MLN [Kuzelka, 2020], invoking such a large number of WFOMC oracle calls still quickly becomes intractable as the domain size increases.

[2] The term "lifted" here means that the reduction can be performed in time polynomial in the domain size.

[3] Since Kuzelka and Wang use slightly different definitions and notation, we slightly adapted their Theorem 3 to the setting used in this paper.

4 A Faster Algorithm

In this section, we propose **Fast-RMP**, a faster algorithm for the RMP construction problem.

4.1 Intuition

Instead of traversing all candidate bounding half-spaces of $\mathbf{RMP}(\Psi, \Delta)$, we consider only a portion forming a convex polytope candidate $\mathcal{P}$, and iteratively check whether the vertices of $\mathcal{P}$ are vertices of $\mathbf{RMP}(\Psi, \Delta)$. It is clear that $\mathbf{RMP}(\Psi, \Delta) \subseteq \mathcal{P}$, and if we reach a point where all vertices of $\mathcal{P}$ are vertices of $\mathbf{RMP}(\Psi, \Delta)$, we know that $\mathcal{P}$ must be the desired polytope.

For each vertex $\mathbf{v}$ of $\mathcal{P}$, we know it must be the intersection of a set of m linearly independent bounding hyperplanes of $\mathbf{RMP}(\Psi, \Delta)$. Let $\{\langle \mathbf{a}_1, \mathbf{x} \rangle \leqslant b_1, \ldots, \langle \mathbf{a}_m, \mathbf{x} \rangle \leqslant b_m\}$ be the corresponding bounding half-spaces. To check if $\mathbf{v}$ is a true vertex of $\mathbf{RMP}(\Psi, \Delta)$, we construct a new normal vector $\mathbf{a}' = \sum_{i=1}^m \mathbf{a_i}$ and call the WFOMC oracle to obtain a new bounding half-space $\langle \mathbf{a}', \mathbf{v} \rangle \leqslant b'$ of $\mathbf{RMP}(\Psi, \Delta)$. By Lemma 2, if $b' = \sum_{i=1}^m b_i$, the candidate $\mathbf{v}$ is a true vertex of $\mathbf{RMP}(\Psi, \Delta)$.

Lemma 2. *Let $\mathcal{P}$ be a convex polytope in k-dimensional space, and let $\mathbf{H} := \{\langle \mathbf{a}_1, \mathbf{x} \rangle \leqslant b_1, \ldots, \langle \mathbf{a}_k, \mathbf{x} \rangle \leqslant b_k\}$ be k non-redundant half-spaces that are valid for $\mathcal{P}$. Denote $\mathcal{H}$ the corresponding hyperplanes of $\mathbf{H}$. Then the intersection of the hyperplanes in $\mathcal{H}$ is a vertex of $\mathcal{P}$, if and only if the half-space $H := \langle \sum_{i=1}^k \mathbf{a}_i, \mathbf{x} \rangle \leqslant \sum_{i=1}^k b_i$ is a bounding half-space of $\mathcal{P}$.*

Proof. Since $\mathbf{H}$ is non-redundant, it is clear that $\mathcal{H}$ is linearly independent and its intersection is a singleton containing a point. Let $\mathbf{v}$ be this point and h be the hyperplane of H. According to the definition of a vertex, it is sufficient to prove that $h \cap \mathcal{P} = \{\mathbf{v}\}$. Since H is a bounding half-space of $\mathcal{P}$, we have $h \cap \mathcal{P} \neq \emptyset$. Assume there exists a point $\mathbf{v}' \in h \cap \mathcal{P}$ and $\mathbf{v}' \neq \mathbf{v}$. By the condition $\mathbf{v}' \in h$, we have

$$\langle \sum_{i=1}^k \mathbf{a}_i, \mathbf{v}' \rangle = \sum_{i=1}^k b_i. \tag{1}$$

As $\mathbf{v}' \in \mathcal{P}$ and every half-space in $\mathbf{H}$ is valid for $\mathcal{P}$, we also have

$$\langle \mathbf{a}_i, \mathbf{v}' \rangle \leqslant b_i, i = 1, \ldots, k. \tag{2}$$

Since all hyperplanes in $\mathcal{H}$ are linearly independent, their intersection $\mathbf{v}$ is the unique solution of the system of linear equations $\langle \mathbf{a}_i, \mathbf{x} \rangle = b_i, i = 1, \ldots, k$. For the point $\mathbf{v}'$, there must exist $j \in \{1, \ldots, k\}$ such that $\langle \mathbf{a}_j, \mathbf{v}' \rangle \neq b_j$, which combined with (2) leads to a contradiction on (1). Thus the point $\mathbf{v}$ is the unique point in $h \cap \mathcal{P}$. The reverse direction is straightforward according to the definition of bounding half-spaces. □

4.2 Fast-RMP

The proposed approach, named **Fast-RMP**, is described in Algorithm 1. We denote by `boundingHalfSpace` the function that takes a set of formulas Ψ, a set of possible worlds Ω and a normal vector $\mathbf{a}$ as inputs, and calculates the intercept of the bounding half-space of $\mathbf{RMP}(\Psi, \Delta)$ with the norm $\mathbf{a}$ according to Lemma 1. `convexHull` calls an oracle that produces a convex hull from a set of points. `notVisited`$(\mathcal{P}, visited)$ returns vertices of $\mathcal{P}$ that are not contained in the set $visited$.

Algorithm 1 Fast-RMP

Input: A list of formulas $\Psi = \{\alpha_1, \dots, \alpha_m\}$ and a set of possible worlds Ω
Output: Relational marginal polytope $\texttt{RMP}(\Psi, \Delta)$
for $i \leftarrow \{1, \dots, m\}$ **do**
  // $\mathbf{e}_i$ is the ith basis vector
  $b_i \leftarrow \texttt{boundingHalfSpace}(\Psi, \Omega, \mathbf{e}_i)$
$I \leftarrow \{\mathbf{x} = (x_1, \dots, x_m) \mid \mathbf{x} \in \mathbb{Z}^m, 0 \leqslant x_1 \leqslant b_1, \dots, 0 \leqslant x_m \leqslant b_m\}$
$\mathcal{P} \leftarrow \texttt{convexHull}(I)$
$visited \leftarrow ()$
while $\texttt{notVisited}(\mathcal{P}, visited)$ is not empty **do**
  $\mathbf{v} \leftarrow \texttt{notVisited}(\mathcal{P}, visited)[0]$
  $\{\langle \mathbf{a}_1, \mathbf{x}\rangle \leqslant b_1, \dots, \langle \mathbf{a}_m, \mathbf{x}\rangle \leqslant b_m\} \leftarrow \texttt{HRepresentation}(\mathcal{P}, \mathbf{v})$
  $visited.add(\mathbf{v})$
  $\mathbf{a}' \leftarrow \sum_{i=1}^m \mathbf{a}_m$
  $b' \leftarrow \texttt{boundingHalfSpace}(\Psi, \Omega, \mathbf{a}')$
  if $b' = \sum_{i=1}^m b_i$ **then**
    // Found a vertex of $\texttt{RMP}(\Psi, \Delta)$
    continue
  $I \leftarrow \{\mathbf{x} \mid \mathbf{x} \in I, \langle \mathbf{a}', \mathbf{x}\rangle \leqslant b'\}$
  $\mathcal{P} \leftarrow \texttt{convexHull}(I)$
return $\mathcal{P}$

We use $\texttt{HRepresentation}(\mathcal{P}, \mathbf{v})$ to obtain a set of m non-redundant (linearly independent) bounding half-spaces of $\mathcal{P}$ related to the vertex $\mathbf{v}$.

We initialize the convex polytope $\mathcal{P}$ as the bounding box specified by bounding half-spaces of $\texttt{RMP}(\Psi, \Delta)$ with normal vectors parallel to axes. We also keep a set of integral points I contained in $\mathcal{P}$. At each step, to inspect the vertex of $\mathcal{P}$, we first construct the normal vector according to Lemma 2, and then find the bounding half-space with this normal vector using `boundingHalfSpace` based on Lemma 1. If the bounding half-space satisfies the condition in Lemma 2, we confirm that the vertex candidate is a true vertex of $\texttt{RMP}(\Psi, \Delta)$. If the vertex candidate fails to pass the check, we purge it along with points in I not in the new half-space H, and then compute a new convex polytope from the remaining points in I using an oracle for convex hulls. It is easy to check that all bounding half-spaces of the new polytope are valid for $\texttt{RMP}(\Psi, \Delta)$. The set of points I serves as a filter to make sure all vertices of the processed convex polytope are integral, since only an integral point can be a vertex of $\texttt{RMP}(\Psi, \Delta)$. When all vertices of $\mathcal{P}$ are verified, the current convex polytope is exactly $\texttt{RMP}(\Psi, \Delta)$.

In the worst case, we would need to check all integral points in the original bounding box. Since in this case we need only one WFOMC call per integral point in the initial bounding box, the number of WFOMC calls in Algorithm 1 is never more than the $(|\Delta| + 1)^{\sum_{\alpha \in \Psi} |vars(\alpha)|} + 1$ calls needed in [Kuzelka, 2020].

Remark 1. *We choose QuickHull [Barber* et al.*, 1996] as the oracle for* `convexHull`*, which is polynomial-time w.r.t. the number of integral points $|I|$. It follows that the guarantee that if MLNs given by Ψ are domain-liftable, constructing its RMP is also domain-liftable still holds for Alg. 1.*

5 An Application of Fast-RMP

In this section, we present an application of **Fast-RMP** to improve an existing approximation algorithm for WFOMC [van Bremen and Kuzelka, 2020]. To simplify the exposition, we describe the method only for WFOMC problems obtained from MLNs. Still, the proposed method can also be applied to general WFOMC problems (with a generalized **Fast-RMP**). Given an MLN Φ, denote by Ψ^∞ the set of formulas with infinite weight in Φ. Let $\Phi^\natural$ be the tuples with finite weight in Φ and $\Omega_{\Delta,\Psi}$ be the set of all possible worlds on the domain Δ that satisfy all formulas in some set Ψ. Using the reduction in Section 2.2, we can overload the definition of WFOMC to allow MLNs as input, yielding the partition function of Φ on a domain Δ:

$$\texttt{WFOMC}(\Phi, \Delta) := \sum_{\omega \in \Omega_{\Delta,\Psi^\infty}} \prod_{(\alpha,w) \in \Phi^\natural} \prod_{\theta \in \Theta(\alpha,\Delta): \omega \models \alpha\theta} \exp(w).$$

5.1 ApproxWFOMC

In general, computing $\texttt{WFOMC}(\Phi, \Delta)$ exactly is a #P_1-hard problem if Φ contains a formula with more than two logical variables [Beame *et al.*, 2015]. Therefore, to deal effectively with non-liftable formulas, van Bremen and Kuzelka [2020] proposed an approximation algorithm for WFOMC. Given an MLN Φ and a domain Δ, they define the first-order model counting function as

$$\texttt{MC}_{\Phi,\Delta}(\mathcal{Z}) = |\{\omega \in \Omega_{\Delta,\Psi^\infty} \mid \mathbf{N}(\Psi^\natural, \omega) \in \mathcal{Z}\}|$$

where $\mathcal{Z}$ is a set of integer vectors and $\Psi^\natural$ is the list of formulas in $\Phi^\natural$. The WFOMC $\texttt{WFOMC}(\Phi, \Delta)$ then can be decomposed into:

$$\texttt{WFOMC}(\Phi, \Delta) = \sum_{\mathbf{k} \in \mathcal{K}} \texttt{MC}_{\Phi,\Delta}(\{\mathbf{k}\}) \cdot \prod_{i=1}^m \exp(k_i w_i) \quad (3)$$

where $\mathbf{k} = (k_1, \dots, k_m)$, $\mathcal{K} = \times_{i=1}^m \{0, 1 \dots, |\Delta|^{vars(\alpha_i)}\}$. The first-order model counts are estimated using a hashing-based approximation algorithm [Chakraborty *et al.*, 2016; Soos and Meel, 2019], which allows for PAC-guarantees on the WFOMC returned; we refer the reader to [van Bremen and Kuzelka, 2020] for more details.

In their algorithm **ApproxWFOMC**, they approximate (3) using a divide and conquer strategy that splits the space specified by $\mathcal{K}$ into disjoint small rectangular boxes. For each box and a set of integer points in it $\mathcal{C} = \{\mathbf{k}^{(1)}, \dots, \mathbf{k}^{(t)}\}$, they compute $\texttt{MC}_{\Phi,\Delta}(\mathcal{C})$ and the minimum and maximum of the product of weights $lo = \min_{j=1}^t \prod_{i=1}^m \exp(\mathbf{k}_i^{(j)} w_i)$ and $hi = \max_{j=1}^t \prod_{i=1}^m \exp(\mathbf{k}_i^{(j)} w_i)$ respectively. The value of the weighted first-order model count constrained by this box can be bounded by $\texttt{MC}_{\Phi,\Delta}(\mathcal{C}) \cdot lo$ and $\texttt{MC}_{\Phi,\Delta}(\mathcal{C}) \cdot hi$.

5.2 Improvement on ApproxWFOMC

We present an improved **ApproxWFOMC**, named **Fast-ApproxWFOMC**, in Algorithm 2. The main idea is that if $\texttt{RMP}(\Psi, \Delta)$ is already known, for the decomposition given in (3),

1. we can consider only the summation on $\mathbf{k}$'s contained in $\texttt{RMP}(\Psi, \Delta)$ rather than in all of $\mathcal{K}$; and,

Algorithm 2 Fast-ApproxWFOMC

Input: an MLN Φ of size m with a surrogate RMP $\mathcal{P}$, a domain Δ and tolerance τ
Output: (b_1, b_2) s.t. $b_1 \leqslant \texttt{WFOMC}(\Phi, \Delta) \leqslant b_2$, and $\frac{b_2}{b_1} < 1 + \tau$

$queue \leftarrow$ new priority queue
for $i \in \{1, \ldots, m\}$ **do**
  // Improvement 1
  $constraints[i] \leftarrow (\min_{\mathbf{x}_i} \mathbf{x} \in \mathcal{P}, \max_{\mathbf{x}_i} \mathbf{x} \in \mathcal{P})$
$\mathcal{C} \leftarrow \texttt{integerPoints}(constraints)$
$mc \leftarrow \texttt{MC}_{\Phi,\Delta}(\mathcal{C})$
// Improvement 2
$lb, ub \leftarrow \texttt{getBounds}(\mathcal{P}, constraints, \Phi)$
$LB, UB \leftarrow mc \cdot lb, mc \cdot ub$
Store $(constraints, LB, UB)$ in $queue$
while $\frac{UB}{LB} \geqslant 1 + \tau$ and $queue$ is not empty **do**
  Pop $(constraints, lb, ub)$ from $queue$
  if $constraints$ cannot be split further **then**
    continue
  $i \leftarrow$ the index of optimal formula by heuristic
  $l, u \leftarrow constraints[i]$
  $leftConstr \leftarrow constraints$
  $rightConstr \leftarrow constrains$
  $leftConstr[i] \leftarrow (l, \lfloor \frac{l+u}{2} \rfloor)$
  $rightConstr[i] \leftarrow (\lfloor \frac{1+u}{2} \rfloor + 1, u)$
  $LB \leftarrow LB - lb$
  $UB \leftarrow UB - ub$
  for $refinedConstr$ in $\{leftConstr, rightConstr\}$ **do**
    // Improvement 1
    if $refinedConstr \cap \mathcal{P}$ is $\varnothing$ **then**
      continue
    // Improvement 2
    $lo, hi \leftarrow \texttt{getBounds}(\mathcal{P}, refinedConstr, \Phi)$
    $\mathcal{C} \leftarrow \texttt{integerPoints}(refinedConstr)$
    $mc \leftarrow \texttt{MC}_{\Phi,\Delta}(\mathcal{C})$
    $LB \leftarrow LB + lo \cdot mc$
    $UB \leftarrow UB + hi \cdot mc$
    Push $(refinedConstr, lo \cdot mc, hi \cdot mc)$ to $queue$
return (LB, UB)

function $\texttt{getBounds}(\mathcal{P}, constraints, \Phi)$
  Get H-representation $\{\langle \mathbf{a}_1, \mathbf{x} \rangle \leqslant b_1, \ldots, \langle \mathbf{a}_M, \mathbf{x} \rangle \leqslant b_M\}$ of $\mathcal{P}$
  $\mathbf{w} \leftarrow (w_1, \ldots, w_m)$
  Solve the following LPs and obtain the optimal objectives o_l and o_u respectively

$$\begin{aligned} &\min \langle \mathbf{w}, \mathbf{x} \rangle \quad \text{or} \quad \min -\langle \mathbf{w}, x \rangle \\ &s.t.\ \langle \mathbf{a}_i, \mathbf{x} \rangle \leqslant b_i, i = 1, \ldots, M \\ &\quad \mathbf{x}_j \geqslant constraints[j][0], j = 1, \ldots, m \\ &\quad \mathbf{x}_j \leqslant constraints[j][1], j = 1, \ldots, m \end{aligned}$$

  return $(\exp(o_l), \exp(-o_u))$

2. in each split box generated by **ApproxWFOMC**, we can further constrain $\mathbf{k}$ to lie in $\texttt{RMP}(\Psi, \Delta)$ to obtain better lower and upper bounds of the product of weights.

In improvement 2, to avoid intractable integer programming, we constrain $\mathbf{k}$ to lie in the intersection of $\texttt{RMP}(\Psi, \Delta)$ with the continuous area specified by the box.

Example 2. *Consider the convex polytope of the MLN* $\Phi = \{(\alpha_1 : 1), (\alpha_2 : 2)\}$ *for domain size 3, where* α_1 *and* α_2 *are given in Example 1. Figure 2 shows a possible split produced by* ***ApproxWFOMC****. In this split, the original lower and upper bounds for each box are* $(e^{10}, e^{19}), (e^{12}, e^{21}), (e^0, e^9), (e^2, e^{11})$ *respectively, while in our algorithm, boxes* 3 *and* 4 *would be ignored since they do not intersect with the polytope, and the bounds for boxes* 1 *and* 2 *are improved to* $(e^{15}, e^{19}), (e^{16}, e^{21})$.

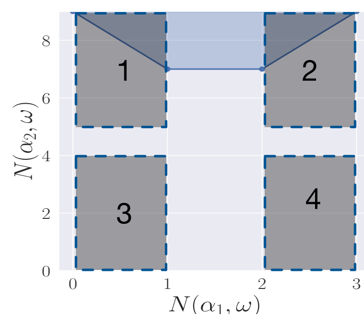


Figure 2: A possible split made by **ApproxWFOMC** to approximate the partition function of the example MLN. The blue area represents the RMP of this MLN and the four gray rectangles are the split boxes. The boundary of each box is inclusive.

Note our enhancements do not require an *exact* $\texttt{RMP}(\Psi, \Delta)$. If only the RMP of a subset of formulas with indices $\{i_1, \ldots, i_t\}$ is available, e.g., the RMP problem for the whole formulas is not domain-liftable, we can construct a *surrogate* convex polytope $\texttt{RMP}'(\Psi, \Delta) := \texttt{RMP}(\{\alpha_{i_1}, \ldots, \alpha_{i_t}\}, \Delta) \cap \mathcal{J}$, where $\mathcal{J} = \{\mathbf{x} \mid 0 \leqslant \mathbf{x}_i \leqslant |\Delta|^{vars(\alpha_i)}, i = 1, \ldots, m\}$ is the continuous area of $\mathcal{K}$. Since $\texttt{RMP}'(\Psi, \Delta) \subseteq \mathcal{J}$, the algorithm with $\texttt{RMP}'(\Psi, \Delta)$ can still benefit from the improvements proposed above.

5.3 Algorithm Details

In Algorithm 2, we use *constraints* to represent a rectangular box with its ith component specifying the boundary of the box along the ith dimension. The function $\texttt{integerPoints}(constraints)$ returns a set of the integer points within the box specified by *constraints*: $\{\mathbf{k} \in \mathbb{Z}^m \mid constraints[i][0] \leqslant \mathbf{k}_i \leqslant constraints[i][1], i = 1, \ldots, m\}$.

Using the divide and conquer strategy, we progressively split $\mathcal{K}$ into several disjoint boxes and compute the lower and upper bounds of WFOMC constrained by these boxes. The whole WFOMC then can be estimated by adding up the lower and upper bounds over all these boxes as shown in (3). We store these boxes as well as their bounds in a queue that is sorted according to a heuristic function on these bounds. At each step of the while loop, we process one of the boxes, evenly partitioning it into two boxes along a heuristic-optimal dimension (lines 14-19) resulting in a new split of $\mathcal{K}$, and refine the lower and upper bounds of the WFOMC on the new split (lines 20-30). We refer to Algorithm 1 in [van Bremen and Kuzelka, 2020] for more details.

Our improvements on the original **ApproxWFOMC** are highlighted by comments. In particular, the function getBounds$(\mathcal{P}, constraints, \Phi)$ realizes improvement 2, returning refined lower and upper bounds of the WFOMC within the intersection of $\mathcal{P}$ with the continuous area of $constraints$.

6 Experiments

We conduct several experiments on different benchmark problems to show the efficiency of our proposed algorithms **Fast-RMP** and **Fast-ApproxWFOMC**.

6.1 Implementation and Setup

We use Forclift[4] as the WFOMC oracle, QuickHull[5] as the backend for the convex hull problem, and an interior point method [Boyd and Vandenberghe, 2004] to solve LPs. We follow van Bremen and Kuzelka [2020] and utilize ApproxMC4 [Soos and Meel, 2019] as an approximate MC oracle. The hyperparameters in our WFOMC approximation experiments are set to $\epsilon = 0.8$, $\delta = 0.2$ and $\tau = 0.5$, just as in the original paper.

For **Fast-RMP**, we further adopt a pre-compilation technique [Van Haaren *et al.*, 2016] that only compiles Ψ into a first-order d-DNNF circuit in the first oracle call, and evaluates the circuit on the given set of weights to compute the partition function in later WFOMC calls. This further speeds up the RMP computation.

All experiments are performed on a computer with an eight-core Intel i7 3.60GHz processor and 32 GB of RAM.

6.2 RMP Construction

We use the DFT-based algorithm proposed by Kuzelka [2020] as a baseline method, since it is, to the best of our knowledge, the state-of-art for the RMP construction problem. We test **Fast-RMP** on four problems:

- Friends and smokers: two variants corresponding to Ψ_1 and Ψ_2 from Example 1.
- Friends, smokers and drinkers: $\Psi_3 = \{str(x) \Rightarrow sm(x), sm(x) \wedge fr(x,y) \Rightarrow sm(y), str(x) \Rightarrow dr(x), dr(x) \wedge fr(x,y) \Rightarrow dr(y)\}$
- Infection: $\Psi_4 = \{disease(x) \Rightarrow cough(x), disease(x) \wedge contact(x,y) \Rightarrow disease(y), disease(x) \Rightarrow \neg contact(x,y)\}$ in which the third formula simulates a quarantine on infected people.

In Figure 4a, we show the performance gain of **Fast-RMP** over the baseline, by quantifying the ratio between our algorithm and the baseline for both runtime and the number of WFOMC oracle calls. The baseline already times out (10 hours) for Ψ_3 with a domain of size 10, so the domain size we test for Ψ_3 is up to 10. **Fast-RMP** is more than a hundred

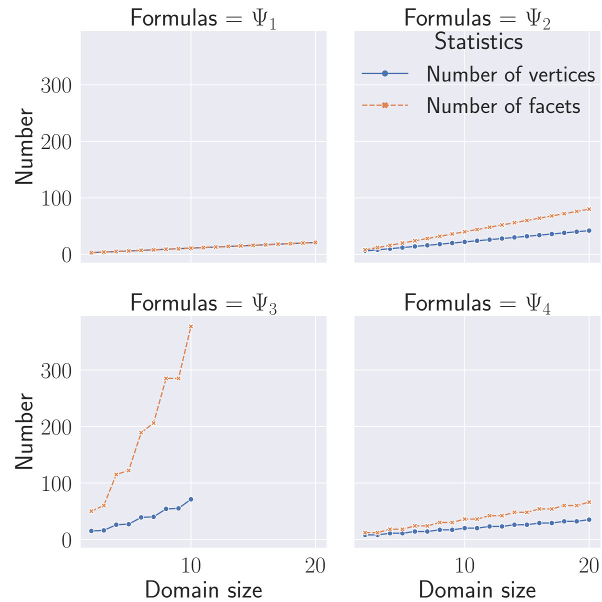


Figure 3: Number of vertices and facets of convex polytopes of Ψ_1, Ψ_2, Ψ_3 and Ψ_4 on different domain sizes.

times faster. On all tested problems, the ratio of WFOMC calls steadily decrease with domain size. Overall, **Fast-RMP** is approximately 10–100 times faster than the baseline.

We also analyze the statistics of convex polytopes of the four examples on different domain sizes as shown in Figure 3. The performance of our method obviously depends on the number of vertices of the convex polytope, but it seems that the complexity of convex polytopes does not hurt the efficiency of our approach. For example, the polytope of Ψ_3 is much more complicated than others, while the improvement of **Fast-RMP** on it is more significant.

6.3 WFOMC Approximation

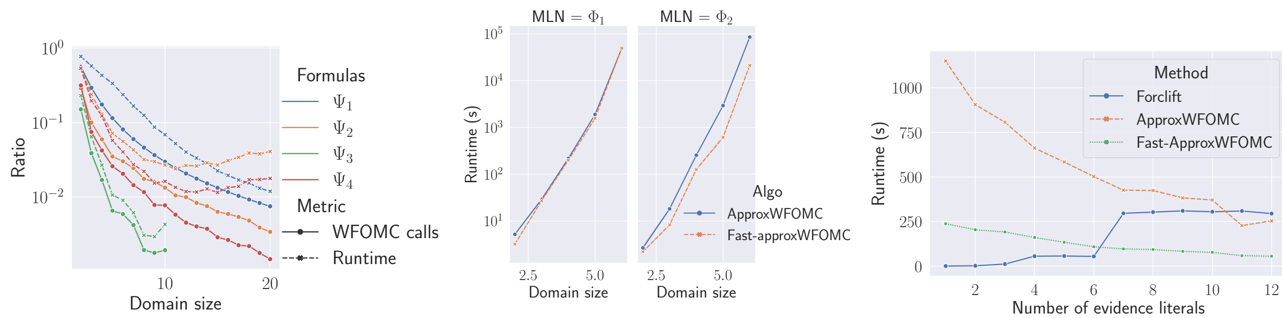
To evaluate our new approximate WFOMC algorithm **Fast-ApproxWFOMC**, we conduct inference experiments in two different WFOMC settings: (i) with MLNs that are not domain-liftable and (ii) with MLNs that are domain-liftable, but have been augmented with binary evidence literals. We then compare our algorithm with the original one **ApproxWFOMC** and, for the experiments with domain-liftable MLNs, also with Forclift [Van den Broeck *et al.*, 2011]. Forclift computes the exact WFOMC, but works only on domain-liftable problems. **ApproxWFOMC** and **Fast-ApproxWFOMC** provide PAC-guarantees on the approximation [van Bremen and Kuzelka, 2020]. We do not compare with algorithms that do not give any guarantees (such as Gibbs sampling); even though they can scale to large domains, estimates obtained using such methods can be arbitrarily bad. Such a comparison would therefore not make sense here.

Inference in Non-liftable MLNs

We perform experiments with the following two MLNs:

- Transitive smokers and drinkers: $\Phi_1 = \{(str(x) \Rightarrow sm(x), 1.22), (sm(x) \wedge fr(x,y) \Rightarrow sm(y), 2.08),$

[4]https://github.com/UCLA-StarAI/Forclift
[5]http://www.qhull.org/



(a) (b) (c)

Figure 4: (a) Ratio of WFOMC calls and runtime made by **Fast-RMP** to the baseline. (b) Runtime of **Fast-ApproxWFOMC** compared to **ApproxWFOMC** on the "transitive smokers and drinkers" and "transitive infection" examples with various domain sizes. (c) Runtime of Forclift, **ApproxWFOMC** and **Fast-ApproxWFOMC** on the "smokers and drinkers with evidence" MLN with a varying number of evidence literals.

$(fr(x,y) \wedge fr(y,z) \Rightarrow fr(x,z), 0.69)$, $(str(x) \Rightarrow dr(x), 2)$, $(dr(x) \wedge fr(x,y) \Rightarrow dr(y), 1.5)\}$, with the same weights as in [van Bremen and Kuzelka, 2020].

- Transitive infection: $\Phi_2 = \{(disease(x) \Rightarrow cough(x), -0.69), (disease(x) \wedge contact(x,y) \Rightarrow disease(y), 0.69), (contact(x,y) \wedge contact(y,z) \Rightarrow contact(x,z), 0.83), (disease(x) \Rightarrow \neg contact(x,y), -0.11)\}$

Note that the third formula in the MLNs is a transitive rule, meaning neither problem is domain-liftable. To compute their polytopes, we consider the liftable part of each of these two MLNs and feed their convex polytopes as surrogate polytopes to our algorithm as described in Section 5.2.

Figure 4b presents the runtime of our algorithm compared to the baseline. On the "transitive smokers and drinkers" problem, **Fast-ApproxWFOMC** performs on par with the baseline, but does better on the transitive infection problem.

To better understand why **Fast-ApproxWFOMC** does not improve over **ApproxWFOMC** on the "transitive smokers and drinkers" MLN, we further visualized the convergence rate of the bounds in Figure 5. As can be seen, though our new algorithm requires a similar number of oracle calls as the baseline to achieve an approximation within the prescribed accuracy, it converges much faster at early iterations. This suggests that, in some cases where **Fast-ApproxWFOMC** is not faster, overall it may still be more efficient for computing rough bounds, i.e., when the value of the tolerance τ is large.

Inference in Liftable MLNs with Binary Evidence

A "smokers and drinkers with evidence" MLN is used here: $\Phi_3 = \{(sm(x), 1.22), (sm(x) \wedge fr(x,y) \Rightarrow sm(y), 2.08), (fr(x,y), 0.69), (dr(x) \wedge fr(x,y) \Rightarrow dr(y), 1.5), (l_1, +\infty), \ldots, (l_n, +\infty)\}$, where $l_1, \ldots, l_n$ are randomly generated literals using the binary predicate $fr(\cdot,\cdot)$. The domain size is fixed to 5. Note that approximating the WFOMC of this MLN makes sense, as the exact WFOMC problem is usually hard for MLNs with binary evidence literals, even if the MLN is otherwise domain-liftable [Van Den Broeck and Davis, 2012]. However, computing the RMP used in **Fast-ApproxWFOMC** is tractable since it does not involve any evidence.

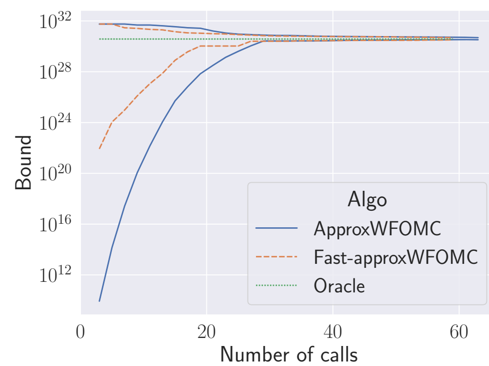


Figure 5: Bounds obtained by **ApproxWFOMC** versus **Fast-ApproxWFOMC** for a varying number of ApproxMC4 calls on the "transitive smokers and drinkers" example, for a representative domain size 3. The green dashed line represents the true WFOMC.

Figure 4c compares the runtime of Forclift, **ApproxWFOMC** and **Fast-ApproxWFOMC** for computing the partition function of the MLN above with a varying number of evidence literals. Forclift performs better than the approximation algorithms with a small number of evidence literals, while **ApproxWFOMC** and **Fast-ApproxWFOMC** surpass Forclift after 7 and 11 evidence literals respectively. The runtime of approximation algorithms decreases gradually as the size of evidence increases. The evidence literal sufficiently simplifies the theory fed to the model counting oracle, thus reducing the time needed for the propositional model counter calls. Importantly, for all evidence sizes, **Fast-ApproxWFOMC** is consistently faster than **ApproxWFOMC**.

7 Conclusions

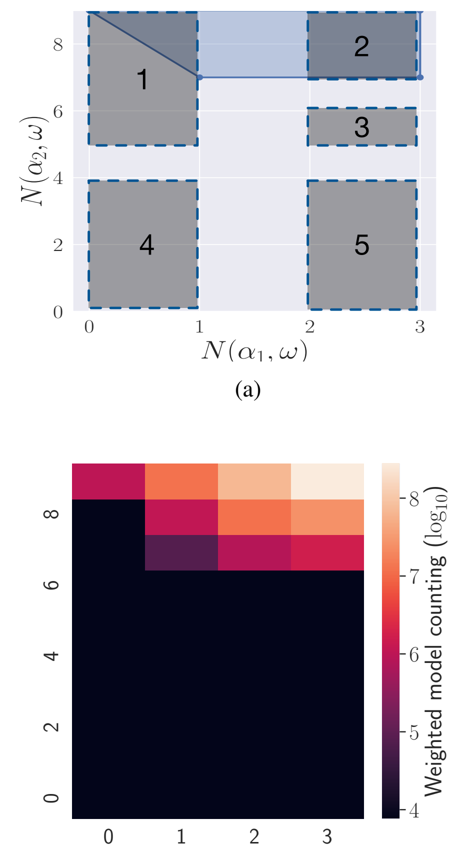
We considered the problem of constructing relational marginal polytopes, and proposed an algorithm that is successful in reducing both the number of WFOMC oracle calls as well as overall runtime by orders of magnitude as compared to the current state of the art. We further applied our approach to an improved algorithm for approximate weighted first-order model counting. In the future, we would like to further investigate applications of RMPs in other statistical-relational models beyond MLNs.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments. Yuanhong Wang and Juhua Pu are supported in part by the National Key R&D Program of China (2018YFB2100800), Science Technology and Innovation Commission of Shenzhen Municipality (JCYJ20180307123659504) , and the State Key Laboratory of Software Development Environment in Beihang University (SKLSDE-2020ZX-22). TvB is supported by the Research Foundation – Flanders [G095917N]. OK is supported by the Czech Science Foundation project "Generative Relational Models" (20-19104Y) and partially also by the OP VVV project *CZ.02.1.01/0.0/0.0/16_019/0000765* "Research Center for Informatics".

A Further Analysis of the Performance of Fast-ApproxWFOMC

In this section, we try to explain the actual reason why **Fast-ApproxWFOMC** does not help much on the "transitive smokers and drinkers" example. One possible reason is that the "mass" of the weighted first-order model count of this example is concentrated in some rectangular boxes in the convex polytope. For instance, a possible split made by **Fast-ApproxWFOMC** for the subset of "transitive smokers and drinkers" MLN: $\{(str(x) \Rightarrow sm(x), 1.22), (sm(x) \wedge fr(x, y) \Rightarrow sm(y), 2.08)\}$ for domain size 3 is presented in Figure 6a. The distribution of the value of $\mathtt{MC}_{\Psi,\Delta}(\mathbf{k}) \cdot \prod_{i=1}^{m} \exp(k_i w_i)$ in (3) of this example is shown in Figure 6b. It is easy to see that the large part of WFOMC of this example is mainly concentrated at the top right corner of the polytope, which is approximated by box 2. However, box 2 is contained in the polytope, and thus for this box **Fast-ApproxWFOMC** cannot provide better lower and upper bounds than **ApproxWFOMC**. The consequence is that the overall bounds cannot be improved much by **Fast-ApproxWFOMC** in this specific case.



(a)

(b)

Figure 6: (a): A possible split made by **Fast-ApproxWFOMC** to approximate WFOMC of an example MLN: $\{(str(x) \Rightarrow sm(x), 1.22), (sm(x) \wedge fr(x, y) \Rightarrow sm(y), 2.08)\}$ for domain size 3. The blue area represents the convex polytope of this MLN and four gray rectangles are the split boxes. (b): The value of $\mathtt{MC}_{\Psi,\Delta}(\mathbf{k}) \cdot \prod_{i=1}^{m} \exp(k_i w_i)$ of the example for all feasible $\mathbf{k}$.

References

[Barber *et al.*, 1996] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996.

[Beame *et al.*, 2015] Paul Beame, Guy Van den Broeck, Eric Gribkoff, and Dan Suciu. Symmetric weighted first-order model counting. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 313–328, 2015.

[Boyd and Vandenberghe, 2004] Stephen P. Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[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.

[Fukuda, 2004] Komei Fukuda. Frequently Asked Questions in Polyhedral Computation. https://people.inf.ethz.ch/fukudak/polyfaq/polyfaq.html, 2004. Accessed: 2021-05-20.

[Kuzelka and Kungurtsev, 2019] Ondrej Kuzelka and Vyacheslav Kungurtsev. Lifted weight learning of markov logic networks revisited. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019*, pages 1753–1761, 2019.

[Kuzelka and Wang, 2020] Ondrej Kuzelka and Yuyi Wang. Domain-liftability of relational marginal polytopes. In *The*

23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, pages 2284–2292, 2020.

[Kuzelka *et al.*, 2018] Ondrej Kuzelka, Yuyi Wang, Jesse Davis, and Steven Schockaert. Relational marginal problems: Theory and estimation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, pages 6384–6391. AAAI Press, 2018.

[Kuzelka *et al.*, 2020] Ondrej Kuzelka, Vyacheslav Kungurtsev, and Yuyi Wang. Lifted weight learning of markov logic networks (revisited one more time). In *The 10th International Conference on Probabilistic Graphical Models, PGM 2020*, 2020.

[Kuzelka, 2020] Ondrej Kuzelka. Complex markov logic networks: Expressivity and liftability. In *Conference on Uncertainty in Artificial Intelligence*, pages 729–738. PMLR, 2020.

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

[Roughgarden and Kearns, 2013] Tim Roughgarden and Michael Kearns. Marginals-to-models reducibility. In *Advances in Neural Information Processing Systems*, pages 1043–1051, 2013.

[Sontag and Jaakkola, 2008] David Sontag and Tommi S Jaakkola. New outer bounds on the marginal polytope. In *Advances in Neural Information Processing Systems*, pages 1393–1400, 2008.

[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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1592–1599, 2019.

[van Bremen and Kuzelka, 2020] Timothy van Bremen and Ondrej Kuzelka. Approximate weighted first-order model counting: Exploiting fast approximate model counters and symmetry. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4252–4258. International Joint Conferences on Artificial Intelligence Organization, 7 2020.

[Van Den Broeck and Davis, 2012] Guy Van Den Broeck and Jesse Davis. Conditioning in first-order knowledge compilation and lifted probabilistic inference. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, page 1961–1967. AAAI Press, 2012.

[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 *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, pages 2178–2185. AAAI Press/International Joint Conferences on Artificial Intelligence, 2011.

[Van den Broeck *et al.*, 2014] Guy Van den Broeck, Wannes Meert, and Adnan Darwiche. Skolemization for weighted first-order model counting. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 1–10, 2014.

[Van Haaren *et al.*, 2016] Jan Van Haaren, Guy Van den Broeck, Wannes Meert, and Jesse Davis. Lifted generative learning of markov logic networks. *Machine Learning*, 103(1):27–55, 2016.