Fast Algorithms for Relational Marginal Polytopes

Yuanhong Wang\textsuperscript{1}, Timothy van Bremen\textsuperscript{2}, Juhua Pu\textsuperscript{1,3*}, Yuyi Wang\textsuperscript{4} and Ondřej Kuželka\textsuperscript{5*}

\textsuperscript{1}State Key Laboratory of Software Development Environment \& Engineering Research Center of ACAT, Ministry of Education. Beihang University, China
\textsuperscript{2}KU Leuven, Belgium
\textsuperscript{3}Research Institute of Beihang University in Shenzhen, China
\textsuperscript{4}CRRC Zhuzhou Institute, China \& ETH Zurich, Switzerland
\textsuperscript{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 \cite{Kuzelka et al., 2018}, which are objects that naturally arise in the study of Markov logic networks \cite{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 $\omega$ on the given domain, counting the groundings of the formulas $\alpha_1, \alpha_2, \ldots, \alpha_k$ that are true in $\omega$, and repeating this for all possible $\omega$’s. For instance, for the formulas $\alpha_1 = \text{sm}(x)$ and $\alpha_2 = \text{sm}(x) \land \text{friends}(x,y) \land \text{sm}(y)$, the points defining the polytope would be given by the numbers of smokers in the population (the formula $\alpha_1$) and the number of pairs of people who are friends and, at the same time, both smoke (the formula $\alpha_2$). Relational marginal polytopes can also be seen as “lifted” counterparts of standard marginal polytopes studied in the probabilistic graphical models literature (see, e.g., \cite{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 \cite{Kuzelka and Kungurtsev, 2019; Kuzelka et al., 2020}.

Kuzelka and Wang \cite{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 \cite{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 \cite{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 \cite{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 $\Delta$, 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 \( \alpha \) w.r.t. a domain \( \Delta \) is the set of substitutions on all free variables occurring in \( \alpha \) using constants from \( \Delta \). A possible world \( \omega \) is represented as a set of ground atoms that are true in \( \omega \). The satisfaction relation \( \models \) is defined in the usual way: \( \omega \models \alpha \) means that the formula \( \alpha \) is true in \( \omega \), 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 \((\alpha, w)\), where \( w \in \mathbb{R} \) and \( \alpha \) is a first-order formula. An MLN \( \Phi \) induces a probability distribution over possible worlds \( \omega \in \Omega : \Phi(\omega) = \frac{1}{Z} \exp \left( \sum_{(\alpha, w) \in \Phi} w \cdot N(\alpha, \omega) \right) \), where \( N(\alpha, \omega) := \sum_{\theta \in \Theta(\alpha, \Delta)} \mathbb{I}(\omega = \alpha \theta) \) is the number of groundings of \( \alpha \) satisfied in \( \omega \), and \( Z \), called the partition function, is a normalization constant to ensure that \( \Phi(\omega) \) is a probability distribution. We also allow infinite weights. A formula \( \alpha \) with infinite weight \( \infty \) is understood as a hard constraint imposing that all worlds \( \omega \) in which \( N(\alpha, \omega) \) is not maximal have zero probability. A literal with infinite weight is called an evidence literal. We denote by \( \text{vars}(\alpha) \) the number of free variables in the formula \( \alpha \).

### 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, \( \Gamma \) a sentence, and \( \Delta \) a set of constants. Then

\[
\text{WFOMC}(\Gamma, w, \bar{w}, \Delta) := \sum_{\omega \in \Omega_\Delta} \prod_{a \in P(\omega)} w(Pred(a)) \cdot \prod_{\alpha \in N(\omega)} \bar{w}(\bar{Pred}(a)),
\]

where \( \Omega_\Delta \) is the set of all possible worlds on the domain \( \Delta \) (using the predicates in \( \Gamma \)), \( P(\omega) \) and \( N(\omega) \) denote the positive literals that are true and false in \( \omega \), respectively, and \( \bar{Pred}(a) \) denotes the predicate of \( a \) (e.g., \( \bar{Pred}(\text{friends}(\text{Alice}, \text{Bob})) = \text{friends} \)).

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 \( \Phi \), for every weighted formula \((\alpha_i, w_i) \in \Phi \), where the free variables in \( \alpha_i \) are exactly \( x = \{ x_1, \ldots, x_k \} \) and \( w \neq \infty \), we create a new formula \( \forall x : \xi_i(x) \iff \alpha_i(x) \) where \( \xi_i \) is a new predicate. When \( \alpha_i \) has \( w = \infty \), we instead create a new formula \( \forall x : \xi_i(x) \). We denote the conjunction of the resulting set of sentences by \( \Gamma \) and set the weight function to be \( w(\xi) = \exp(w_i) \) and \( \bar{w}(\xi) = 1 \), and for all other predicates we set both \( w \) and \( \bar{w} \) to be 1. It is easy to check that \( \text{WFOMC}(\Gamma, w, \bar{w}, \Delta) = Z \).

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.

### 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 \( \Delta \) be a set of constants and \( \Psi = (\alpha_1, \ldots, \alpha_m) \) be a set of formulas. The relational marginal polytope of \( \Psi \) on \( \Delta \) is defined as RMP

\[
\text{RMP}(\Psi, \Delta) := \{ (x_1, \ldots, x_m) \in \mathbb{R}^m \mid \exists \text{dist. on } \Omega_\Delta \text{ s.t. } \mathbb{E}[N(\alpha_1, \omega)] = x_1 \land \cdots \land \mathbb{E}[N(\alpha_m, \omega)] = x_m \},
\]

where \( \Omega_\Delta \) is the set of all possible worlds on the domain \( \Delta \) using the predicates in \( \Psi \).

#### 3.1 RMPs as Convex Polytopes

The definition of 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 \( N(\Psi, \omega) = (N(\alpha_1, \omega), \ldots, N(\alpha_m, \omega)) \).

**Example 1.** Consider “friends-smokers” formulas \( \alpha_1 := \text{sm}(x), \alpha_2 := \text{fr}(x, y) \land \text{sm}(x) \Rightarrow \text{sm}(y) \) and \( \alpha_3 := \text{friends}(x, y) \) and denote \( \Psi_1 = \{ \alpha_1, \alpha_2 \} \) and \( \Psi_2 = \{ \alpha_1, \alpha_2, \alpha_3 \} \). Let \( \Delta \) be a domain of size 7. The two relational marginal polytopes RMP \((\Psi_1, \Delta)\) and 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 \( \langle \alpha_1, x \rangle \leq b_1, \ldots, \langle a_M, x \rangle \leq b_M \), called the H-representation [Fukuda, 2018].

¹Kuzelka 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 \( \langle a, x \rangle \leq b \) is called valid for \( P \) if \( \langle a, x \rangle \leq b \) holds for all \( x \in P \). A half-space \( \langle a, x \rangle \leq b \) is a bounding half-space of the convex polytope \( P \), if it is valid for \( P \) and the intersection of its hyperplane \( \langle a, x \rangle = b \) with \( P \) 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 \( P \) is a 0-dimensional intersection of \( P \) 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 \( \text{RMP}(\Psi, \Delta) \). Since \( \mathbb{N}(\Psi, \omega) \) can only take values in \( \times_{\alpha \in \Psi} \{1, 2, \ldots, \text{vars}(\alpha)\} \), the number of normal vectors to be checked is at most \( |\Delta|^m \sum_{\alpha \in \Psi} \text{vars}(\alpha) \). For each candidate normal vector \( a \), they use a version\(^3\) of Lemma 1 below to find the integer \( b \) for the bounding half-space \( \langle a, x \rangle \leq b \) by computing the partition function of an MLN.

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

When all bounding half-spaces of \( \text{RMP}(\Psi, \Delta) \) 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 \( \text{RMP}(\Psi, \Delta) \) can be done in time polynomial in \( |\Delta| \).

However, as we discussed above, the method that they proposed requires \( |\Delta|^m \sum_{\alpha \in \Psi} \text{vars}(\alpha) \) WFOMC oracle calls, which is usually infeasible in practice. Although the complexity was later improved to \( (|\Delta| + 1)^m \sum_{\alpha \in \Psi} \text{vars}(\alpha) + 1 \) 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.

### 4 A Faster Algorithm

In this section, we propose \textbf{Fast-RMP}, a faster algorithm for the RMP construction problem.

#### 4.1 Intuition

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

For each vertex \( v \) of \( P \), we know it must be the intersection of a set of \( m \) linearly independent bounding half-spaces of \( \text{RMP}(\Psi, \Delta) \). Let \( \{ \langle a_1, x \rangle \leq b_1, \ldots, \langle a_m, x \rangle \leq b_m \} \) be the corresponding bounding half-spaces. To check if \( v \) is a true vertex of \( \text{RMP}(\Psi, \Delta) \), we construct a new normal vector \( a' = \sum_{i=1}^m a_i \) and call the WFOMC oracle to obtain a new bounding half-space \( \langle a', v \rangle \leq b' \) of \( \text{RMP}(\Psi, \Delta) \). By Lemma 2, if \( b' = \sum_{i=1}^m b_i \), the candidate \( v \) is a true vertex of \( \text{RMP}(\Psi, \Delta) \).

**Lemma 2.** Let \( P \) be a convex polytope in \( k \)-dimensional space, and let \( H := \{ \langle a_1, x \rangle \leq b_1, \ldots, \langle a_k, x \rangle \leq b_k \} \) be \( k \) non-redundant half-spaces that are valid for \( P \). Denote \( H \) the corresponding hyperplanes of \( H \). Then the intersection of the hyperplanes in \( H \) is a vertex of \( P \), if and only if the half-space \( H := \{ \sum_{i=1}^k a_i, x \} \leq \sum_{i=1}^k b_i \) is a bounding half-space of \( P \).

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

\[
\sum_{i=1}^k a_i, v' = \sum_{i=1}^k b_i.
\]  

As \( v' \in P \) and every half-space in \( H \) is valid for \( P \), we also have

\[
\langle a_i, v' \rangle \leq b_i, i = 1, \ldots, k.
\]

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

### 4.2 Fast-RMP

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

\[\text{notVisited}(P, visited)\]

returns vertices of \( P \) that are not contained in the set \( visited \).
We use \texttt{HRepresentation}(P, v) to obtain a set of m non-redundant (linearly independent) bounding half-spaces of P related to the vertex v.

We initialize the convex polytope 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 in the half-space satisfying the condition in Lemma 2, we confirm

\begin{align*}
\text{visited} & \leftarrow () \\
\text{while notVisited}(P, \text{visited}) \text{ is not empty do} \\
\text{v} & \leftarrow \text{notVisited}(P, \text{visited})[0] \\
\text{HRRepresentation}(P, v) & \leftarrow \\
\text{visited}.add(v) \\
a' & \leftarrow \sum_{i=1}^{m} a_{i} \\
b' & \leftarrow \text{boundingHalfSpace}(P, \Omega, a') \\
\text{if } b' = \sum_{i=1}^{m} b_{i}, \text{ then} \\
\text{continue} \\
I & \leftarrow \{ x \mid x \in I, \langle a', x \rangle \leq b' \} \\
P & \leftarrow \text{convexHull}(I) \\
\text{return } P
\end{align*}

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 \Phi, denote by \Psi^\infty the set of formulas with infinite weight in \Phi. Let \Phi^\infty be the tuples with finite weight in \Phi and \Omega_{\Delta, \Psi} be the set of all possible worlds on the domain \Delta that satisfy all formulas in some set \Psi. Using the reduction in Section 2.2, we can overload the definition of WFOMC to allow MLNs as input, yielding the partition function of \Phi on a domain \Delta:

\begin{align*}
\text{WFOMC}(\Phi, \Delta) & := \sum_{\omega \in \Omega_{\Delta, \Psi}} \prod_{\phi \in \Phi} \prod_{\theta \in \Theta(\Delta)} \exp(w).
\end{align*}

5.1 ApproxFWOMC

In general, computing \text{WFOMC}(\Phi, \Delta) exactly is a \#P-1-hard problem if \Phi 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 \Phi and a domain \Delta, they define the first-order model counting function as

\begin{align*}
\text{MC}_{\Phi, \Delta}(Z) & = |\{ \omega \in \Omega_{\Delta, \Psi} \mid N(\Psi^\infty, \omega) \in Z \}|
\end{align*}

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

\begin{align*}
\text{WFOMC}(\Phi, \Delta) & = \sum_{k \in \mathcal{K}} \text{MC}_{\Phi, \Delta}([k]) \cdot \prod_{i=1}^{m} \exp(k_i w_i) \tag{3}
\end{align*}

where \( k = (k_1, \ldots, k_m), K = \times_{i=1}^{m} \{0, 1, \ldots, |\Delta|^{\text{vars}(\alpha)}\}. \)

In their algorithm ApproxFWOMC, 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 \( C = \{k^{(1)}, \ldots, k^{(t)}\} \), they compute \( \text{MC}_{\Phi, \Delta} (\mathcal{C}) \) and the minimum and maximum of the product of weights \( lo = \min_{j=1}^{t} \prod_{i=1}^{m} \exp(k^{(j)} w_i) \) and \( hi = \max_{j=1}^{t} \prod_{i=1}^{m} \exp(k^{(j)} w_i) \) respectively. The value of the weighted first-order model count constrained by this box can be bounded by \( \text{MC}_{\Phi, \Delta} (\mathcal{C}) \cdot lo \) and \( \text{MC}_{\Phi, \Delta} (\mathcal{C}) \cdot hi \).

5.2 Improvement on ApproxFWOMC

We present an improved ApproxFWOMC, named FastApproxFWOMC, in Algorithm 2. The main idea is that if \text{RMP}(\Psi, \Delta) is already known, for the decomposition given in (3),

1. we can consider only the summation on \( k \)'s contained in \text{RMP}(\Psi, \Delta) rather than in all of \( \mathcal{K} \); and,
Algorithm 2 Fast-ApproxWFOMC

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

1: \( \text{queue} \leftarrow \text{new priority queue} \)
2: for \( i \in \{1, \ldots, m\} \) do
3:     // Improvement 1
4:     \( \text{constraints}[i] \leftarrow (\min_x, x \in \mathcal{P}, \max_x, x \in \mathcal{P}) \)
5:     \( C \leftarrow \text{integerPoints}(\text{constraints}) \)
6:     \( mc \leftarrow MC_{\Phi, \Delta}(C) \)
7: // Improvement 2
8: \( lb, ub \leftarrow \text{getBounds}(\mathcal{P}, \text{constraints}, \Phi) \)
9: \( LB, UB \leftarrow mc \cdot lb, mc \cdot ub \)
10: Store (\text{constraints}, LB, UB) in queue
11: while \( \frac{LB}{UB} \geq 1 + \tau \) and queue is not empty do
12:     Pop (\text{constraints}, lb, ub) from queue
13:     if \text{constraints} cannot be split further then
14:         continue
15:     \( i \leftarrow \text{the index of optimal formula by heuristic} \)
16:     \( l, u \leftarrow \text{constraints}[i] \)
17:     \( \text{leftConstr} \leftarrow \text{constraints} \)
18:     \( \text{rightConstr} \leftarrow \text{constraints} \)
19:     \( \text{leftConstr}[i] \leftarrow (\lfloor \frac{l + u}{2} \rfloor) \)
20:     \( \text{rightConstr}[i] \leftarrow (\lceil \frac{l + u}{2} \rceil + 1, u) \)
21:     \( LB \leftarrow LB - lb \)
22:     \( UB \leftarrow UB - ub \)
23: for \text{refinedConstr} in \{ \text{leftConstr}, \text{rightConstr} \} do
24: // Improvement 1
25:     if \( \text{refinedConstr} \cap \mathcal{P} = \emptyset \) then
26:         continue
27: // Improvement 2
28:     \( lo, hi \leftarrow \text{getBounds}(\mathcal{P}, \text{refinedConstr}, \Phi) \)
29:     \( C \leftarrow \text{integerPoints}(\text{refinedConstr}) \)
30:     \( mc \leftarrow MC_{\Phi, \Delta}(C) \)
31:     \( LB \leftarrow LB + \text{lo} \cdot \text{mc} \)
32:     \( UB \leftarrow UB + \text{hi} \cdot \text{mc} \)
33:     Push (\text{refinedConstr}, \text{lo} \cdot \text{mc}, \text{hi} \cdot \text{mc}) \text{ to queue} \)
34: return (\( LB, UB \))
35: function getBounds(\( \mathcal{P}, \text{constraints}, \Phi \))
36: Get H-representation \( \langle a_1, x \rangle \leq b_1, \ldots, \langle a_M, x \rangle \leq b_M \) of \( \mathcal{P} \)
37: \( w \leftarrow (w_1 \ldots, w_m) \)
38: Solve the following LPs and obtain the optimal objectives \( \alpha_0 \) and \( \alpha_n \) respectively
39: \( \min(w, x) \quad \text{or} \quad \min(-w, x) \)
40: return (exp(\( \alpha_0 \)), exp(-\( \alpha_n \)))

In improvement 2, to avoid intractable integer programming, we constrain \( k \) to lie in the intersection of \( \text{RMP}(\Psi, \Delta) \) 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 \( \alpha_1 \) and \( \alpha_2 \) are given in Example 1. Figure 2 shows a possible split produced by \textit{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 \((e^{15}, e^{19}), (e^{16}, e^{21})\).

Note our enhancements do not require an exact \( \text{RMP}(\Psi, \Delta) \). 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 \( \text{RMP}(\Psi, \Delta) := \text{RMP}(\{\alpha_{i_1}, \ldots, \alpha_{i_t}\}, \Delta) \cap \mathcal{J} \), where \( \mathcal{J} = \{x \mid 0 \leq x_i \leq |\Delta^{\text{vars}(\alpha_i)}, i = 1, \ldots, m\} \) is the continuous area of \( \mathcal{K} \). Since \( \text{RMP}(\Psi, \Delta) \subseteq \mathcal{J} \), the algorithm with \( \text{RMP}(\Psi, \Delta) \) can still benefit from the improvements proposed above.

5.3 Algorithm Details

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

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 boxes. At each step of the whole 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 \( K \), 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 
\texttt{ApproxWFOMC} are highlighted by comments. In particular, the function \texttt{getBounds}(\texttt{P}, \texttt{constraints, }\Phi) \texttt{realizes improvement \texttt{2}, returning refined lower and upper bounds of the WFOMC within the intersection of }\texttt{P} \texttt{with the continuous area of }\texttt{constraints}.

6 Experiments

We conduct several experiments on different benchmark problems to show the efficiency of our proposed algorithms \texttt{Fast-RMP} and \texttt{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 \texttt{Fast-RMP}, we further adopt a pre-compilation technique [Van Haaren \textit{et al.}, 2016] that only compiles \( \Phi \) 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 \texttt{Fast-RMP} on four problems:

- Friends and smokers: two variants corresponding to \( \Psi_1 \) and \( \Psi_2 \) from Example 1.
- Friends, smokers and drinkers: \( \Psi_3 = \{str(x) \Rightarrow sm(x), sm(x) \land fr(x, y) \Rightarrow sm(y), str(x) \Rightarrow dr(x), dr(x) \land fr(x, y) \Rightarrow dr(y)\} \)
- Infection: \( \Psi_4 = \{\text{disease}(x) \Rightarrow \text{cough}(x), \text{disease}(x) \land \text{contact}(x, y) \Rightarrow \text{disease}(y), \text{disease}(x) \Rightarrow \neg \text{contact}(x, y)\} \) in which the third formula simulates a quarantine on infected people.

In Figure 4a, we show the performance gain of \texttt{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 \( \Psi_3 \) with a domain of size 10, so the domain size we test for \( \Psi_3 \) is up to 10. \texttt{Fast-RMP} is more than a hundred times faster. On all tested problems, the ratio of WFOMC calls steadily decrease with domain size. Overall, \texttt{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 \( \Psi_3 \) is much more complicated than others, while the improvement of \texttt{Fast-RMP} on it is more significant.

6.3 WFOMC Approximation

To evaluate our new approximate WFOMC algorithm \texttt{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 \texttt{ApproxWFOMC} and, for the experiments with domain-liftable MLNs, also with Forclift [Van den Broeck \textit{et al.}, 2011]. Forclift computes the exact WFOMC, but works only on domain-liftable problems. \texttt{ApproxWFOMC} and \texttt{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: \( \Phi_1 = \{(str(x) \Rightarrow sm(x), 1.22), (sm(x) \land fr(x, y) \Rightarrow sm(y), 2.08), \)

\hspace{1cm} \}

\hspace{1cm}
Inference in Liftable MLNs with Binary Evidence

A “smokers and drinkers with evidence” MLN is used here:
\[
\Phi_3 = \{(s(x, y), 1.22), (s(x) \land s(y) \Rightarrow s(x), 2.08),
\]
\[
(f(x, y), 0.69), (d(x) \land f(x, y) \Rightarrow d(y), 1.5), (l_1, +\infty),
\]
\[
\ldots, (l_n, +\infty)\}
\]

where \(l_1, \ldots, l_n\) are randomly generated literals using the binary predicate \(f(x, \cdot)\). 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.

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: \{\text{str}(x) \implies \text{sm}(x), 1.22), (\text{sm}(x) \wedge \text{fr}(x,y) \implies \text{sm}(y), 2.08)\} for domain size 3 is shown in Figure 6a. The distribution of the value of \(M_{\Psi,k}(k) \cdot \prod_{i=1}^{m} \exp(k_i w_i)\) 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.

References


