Efficient Inference for Untied MLNs

Somdeb Sarkhel^{1,3}, Deepak Venugopal^{2}, Nicholas Ruozzi^{3}, Vibhav Gogate^{3}
^{1}Adobe Research, San Jose, CA
^{2}Department of Computer Science, The University of Memphis
^{3}Department of Computer Science, The University of Texas at Dallas
sarkhel@adobe.com, dvngopal@memphis.edu, {nrr150130, vxg112130}@utdallas.edu

Abstract

We address the problem of scaling up local-search or sampling-based inference in Markov logic networks (MLNs) that have large shared sub-structures but no (or few) tied weights. Such untied MLNs are ubiquitous in practical applications. However, they have very few symmetries, and as a result lifted inference algorithms–the dominant approach for scaling up inference–perform poorly on them. The key idea in our approach is to reduce the hard, time-consuming sub-task in sampling algorithms, computing the sum of weights of features that satisfy a full assignment, to the problem of computing a set of partition functions of graphical models, each defined over the logical variables in a first-order formula. The importance of this reduction is that when the treewidth of all the graphical models is small, it yields an order of magnitude speedup. When the treewidth is large, we propose an over-symmetric approximation and experimentally demonstrate that it is both fast and accurate.

1 Introduction

Markov logic networks [Domingos and Lowd, 2009] use weighted first-order logic formulas to specify large Markov networks with repeated sub-structures compactly. The use of first-order logic makes them especially amenable to expressing prior or domain knowledge, which can be easily translated from natural language to first-order logic formulas. As a result, they are routinely used to model prior knowledge in a wide variety of application domains including natural language processing [Venugopal *et al.*, 2014; Riedel and McCallum, 2011], computer vision [Tran and Davis, 2008] and social network analysis [Chen *et al.*, 2013].

Although the MLN representation is compact, it is now well-known that inference in them can be quite challenging and is often a major bottleneck. Specifically, the ground Markov network which is obtained by grounding or propositionalizing the first-order formulas is often so large that even approximate probabilistic inference methods such as Gibbs sampling, Belief Propagation, and MaxWalkSAT are computationally infeasible. To address this issue, several *lifted* inference algorithms that exploit symmetries in the MLN, and avoid constructing the ground Markov network as much as possible, have been proposed in previous work (cf. [Van den Broeck *et al.*, 2012; Singla *et al.*, 2014; Gogate and Domingos, 2011; Venugopal and Gogate, 2014; Kersting *et al.*, 2010]). However, these methods do not function well on arbitrary MLN structures and in the presence of evidence or observations [Van den Broeck and Darwiche, 2013]. As a result, in practice, ground inference is often unavoidable.

Recently there has been growing interest in developing approaches that perform efficient inference over the ground network. Popular approaches include performing lazy inference which constructs the ground network incrementally [Singla and Domingos, 2006b], reducing the size of the network by leveraging evidence [Shavlik and Natarajan, 2009], and using fast, approximate counting approaches that ground the predicates but not the formulas [Venugopal *et al.*, 2015; Sarkhel *et al.*, 2016; Das *et al.*, 2016]. The approach proposed in this paper is related and addresses the following fundamental difficulty associated with the approximate counting approach: it is scalable only on MLNs having both shared sub-structures and weights (MLNs in which all groundings of a first-order formula have the same weight).

In this paper, we consider inference in MLNs having shared sub-structures but *no shared weights* (we call them untied MLNs), namely MLNs in which different groundings of a first-order formula have different weights. Such MLNs are more common in real-world applications [Singla and Domingos, 2006a; Venugopal *et al.*, 2014] since using the same weight for all groundings of a first-order formula often yields a biased, inaccurate model. For example, to specify that a word in a web-page determines the topic of a web-page, we can specify the formula $\texttt{Word}(w, p) \Rightarrow \texttt{Topic}(p, t)$ with a single weight w. However, this model is unrealistic since it assumes that the dependency between every word and every topic is identical. A more practical solution will have $m \cdot n$ different weights, where m is the number of words and n is the number of topics. In the MLN nomenclature, this is represented as a variable preceded by a '$+$' sign. Thus, for our running example, we would write $\texttt{Word}(+w, p) \Rightarrow \texttt{Topic}(p, +t)$, and associate a different weight for each grounding (combination) of the '$+$' variables.

This paper addresses the problem of scaling up inference in untied MLNs and makes the following contributions.

1. We develop a novel graphical model encoding for untied MLN formulas and prove that our new encoding is exact, i.e., the partition function of the graphical model is equal to the sum of the weights of the satisfied groundings in a given world, and exact inference over it is more efficient than the encoding proposed in [Venugopal *et al.*, 2015].
2. We propose an approximate graphical model encoding for untied MLN formulas. This is useful when the exact encoding has large treewidth. The key idea is to cluster together groundings of a formula having similar weights to yield a graphical model with smaller treewidth.

We evaluate our proposed exact and approximate encodings on several MLN benchmarks and compare their performance to [Venugopal *et al.*, 2015]. Our results clearly demonstrate that our new encodings, both exact and approximate, substantially improve the scalability, convergence, and accuracy of Gibbs sampling and MaxWalkSAT.

2 Notation and Background

First-order Logic. The language of first-order logic (cf. [Genesereth and Kao, 2013]) consists of quantifiers ($\forall$ and $\exists$), logical variables, constants, predicates, and logical connectives ($\vee$, $\wedge$, $\neg$, $\Rightarrow$, and $\Leftrightarrow$). A predicate is a relation that takes a specific number of arguments as input and outputs either TRUE (synonymous with 1) or FALSE (synonymous with 0). The arity of a predicate is the number of its arguments. A term is either a logical variable or a constant. We denote predicates by strings in typewriter font (e.g., R, Smokes) followed by a parenthesized list of terms.

A first-order formula is recursively defined as follows:(i) An atomic formula (atom) is a predicate; (ii) Negation of an atom is a formula; (iii) If f and g are formulas then connecting them by binary connectives such as $\wedge$ and $\vee$ yields a formula; and (iv) If f is a formula and x is a logical variable then $\forall x f$ and $\exists x f$ are formulas. A first-order knowledge base (KB) is a set of first-order formulas.

We assume that each argument of each predicate is typed and can only be assigned to a finite set of constants. By extension, each logical variable in each formula is also typed. We assume that our language does not contain the equality symbol and function symbols. We further assume that all first-order formulas are disjunctive (clauses), have no free logical variables, have only universally quantified logical variables (namely, the KB is in conjunctive normal form (CNF)), and have no constants. Note that all first-order formulas can be easily converted to this form.

A ground atom is an atom that contains no logical variables. A ground formula is a formula containing only ground atoms. The grounding of a first-order formula f is the set of all possible ground formulas that can be obtained from f by substituting all the logical variables in it by constants in their domain. A ground KB is obtained from a first-order KB by grounding all of its first-order formulas. A possible world, denoted by ω, is a truth assignment to all possible ground atoms in the first-order KB.

Markov Logic Networks (MLNs). An issue with first-order logic is that it cannot represent uncertainty: all worlds that violate even one ground formula are considered inconsistent. MLNs soften the constraint expressed by each formula, by attaching a weight to it. The higher the weight, the higher the probability of the clause being satisfied, all other things being equal. MLNs can also be seen as a first-order template for generating large Markov networks. Formally, an MLN is a set of pairs (f_i, θ_i) where f_i is a formula in first-order logic and θ_i is a real number. Given a set of constants, an MLN represents a ground Markov network which has one random variable for each grounding of each predicate and one propositional feature for each grounding of each formula. The weight associated with the feature is the weight attached to the corresponding formula. The ground Markov network represents the following probability distribution:

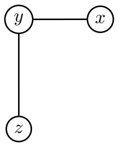
$$P_{\boldsymbol{\theta}}(\omega) = \frac{1}{Z_{\boldsymbol{\theta}}} \exp\left(\sum_i \theta_i N_i(\omega)\right) \quad (1)$$

where $N_i(\omega)$ is the number of groundings of f_i that evaluate to TRUE given ω and $Z_{\boldsymbol{\theta}}$ is the normalization constant. We call this the #SG problem. Important inference queries over MLNs such as computing the partition function, finding the marginal probability of a variable given evidence (where evidence is an assignment to a subset of variables), and finding the most probable assignment to all variables given evidence, also called the MAP inference problem, require solving the #SG problem at every iteration.

Solving the #SG problem. The main computational bottleneck in many inference algorithms for MLNs such as Gibbs sampling for marginal inference and MaxWalkSAT [Kautz *et al.*, 1997] for MAP inference is computing $N_i(\omega)$. Until recently, the #SG problem was solved using the following naive method: given a clause f_i and a world ω, generate all possible ground clauses of f_i and count only those that are satisfied in ω. [Venugopal *et al.*, 2015] showed that the problem could be solved efficiently by reducing it to the problem of computing the number of solutions of a constraint satisfaction problem (i.e., a Markov network in which all potentials have just two values: 0 or 1). Formally, given a first-order clause f_i and a world ω, the corresponding constraint network $\mathcal{C}_i$ has a variable for each (universally quantified) logical variable in f_i. The domain of each variable in $\mathcal{C}_i$ is the set of constants in the domain of the corresponding logical variable. For each atom $\mathtt{R}(x_1, \ldots, x_u)$ in f_i, we have a constraint ϕ in $\mathcal{C}$ defined as follows: $\phi(\overline{\mathbf{x}}_u) = \omega_{\mathtt{R}(X_1,\ldots,X_u)}$ if R is negated in f and $\phi(\overline{\mathbf{x}}_u) = 1 - \omega_{\mathtt{R}(X_1,\ldots,X_u)}$ otherwise. Here, $\overline{\mathbf{x}}_u = (x_1 = X_1, \ldots, x_u = X_u)$ denotes an assignment to the variables in the constraint network and $\omega_{\mathtt{R}(X_1,\ldots,X_u)}$ is the projection of the world ω on the ground atom $\mathtt{R}(X_1, \ldots, X_u)$. Thus, $\omega_{\mathtt{R}(X_1,\ldots,X_u)} = 1$ if $\mathtt{R}(X_1, \ldots, X_u)$ is true in ω and 0 otherwise.

Example 1. *Fig. 1 shows the constraint network for a formula f and a world ω.*

[Venugopal *et al.*, 2015] showed that if $\#\mathcal{C}_i$ denotes the number of solutions of the constraint network $\mathcal{C}_i$ associated with a clause f_i and world ω then: $N_i(\omega) = \prod_{x_j \in V(f_i)} |\Delta(x_j)| - \#\mathcal{C}_i$ where $V(f_i)$ denotes the set of logical variables in f_i and $\Delta(x_j)$ is the set of constants in the domain of x_j. Thus, if a junction tree algorithm is used to



(a)

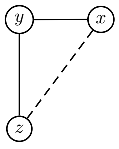
x	y	val
A	A	0
A	B	1
B	A	1
B	B	0

(b) $\phi_1(x, y)$

y	z	val
A	A	1
A	B	0
B	A	1
B	B	1

(c) $\phi_2(y, z)$

Figure 1: (a) Constraint network associated with $f = \forall x, \forall y, \forall z\ \neg \mathtt{R}(x, y) \vee \mathtt{S}(y, z)$. The domain of each logical variable is $\{A, B\}$; (b) and (c): Constraints $\phi_1(x, y)$ and $\phi_2(y, z)$ corresponding to the possible world in which the ground atoms $\mathtt{R}(A, B)$, $\mathtt{R}(B, A)$, and, $\mathtt{S}(A, B)$ are true and the rest are false.



(a) dMRF

x	y	val
A	A	0
A	B	1
B	A	1
B	B	0

(b) $\phi_1(x, y)$

y	z	val
A	A	1
A	B	0
B	A	1
B	B	1

(c) $\phi_2(y, z)$

x	z	val
A	A	$\theta_{A,A}$
A	B	$\theta_{A,B}$
B	A	$\theta_{B,A}$
B	B	$\theta_{B,B}$

(d) $\phi_3(x, z)$

Figure 2: (a) dMRF associated with $f = \forall x, \forall y, \forall z\ \neg \mathtt{R}(+x, y) \vee \mathtt{S}(y, +z)$. The domain of each logical variable is $\{A, B\}$; (b) and (c): Constraints $\phi_1(x, y)$ and $\phi_2(y, z)$ corresponding to the possible world in which the ground atoms $\mathtt{R}(A, B)$, $\mathtt{R}(B, A)$, and, $\mathtt{S}(A, B)$ are true and the rest are false. (d) $\phi_3(x, z)$ corresponds to the parameters $\theta_{x,z}$ associated with the formula

compute $\#\mathcal{C}_i$, then its time and space complexity is exponential in treewidth plus one and treewidth respectively. Since the treewidth can be much smaller than the number of logical variables, the complexity of Venugopal et al.'s method can be exponentially smaller than the naive approach, which is exponential in the number of logical variables.

Untied MLN. An MLN formula represents a template for generating ground formulas (obtained by replacing logical variables with domain objects). Often in the Markov Logic literature, it is assumed the weight of each of these grounding is the same, i.e. they are *tied*. However, in many realistic settings (e.g., for many information extraction tasks such as event extraction [Venugopal *et al.*, 2014]), the parameter tying assumption is too strong. MLNs relax this constraint using the '$+$' operator, which learns a separate weight for each grounding of each logical variable associated with a '$+$' sign.

Example 2. *Let $f = \forall x, \forall y, \forall z\ \neg \mathtt{R}(x, y) \vee \mathtt{S}(y, z)$ be an MLN formula. Then to specify that different weights are attached to different groundings of x and z of f, we use the formula: $f' = \forall x, \forall y, \forall z\ \neg \mathtt{R}(+x, y) \vee \mathtt{S}(y, +z)$*

We will use $\mathcal{X}_+$ to denote the set of logical variable associated with the "$+$" operator. To denote the weights (or parameters) associated with f we will use $\boldsymbol{\theta}_f$. A complete assignment to all the variables in $\mathcal{X}_+$ (denoted by $\overline{\mathbf{x}}_+$) corresponds to exactly one parameter from the parameter set $\boldsymbol{\theta}$. Hence, we will denote each such individual parameter by $\theta_{\overline{\mathbf{x}}_+}$. In the above example if $\Delta(x) = \{A, B\}$ and $\Delta(y) = \{C, D\}$ then the parameter set is, $\boldsymbol{\theta}_f = \{\theta_{A,C}, \theta_{B,C}, \theta_{A,D}, \theta_{B,D}\}$.

For these MLNs we can rewrite Equation 1 as follows:

$$P(\omega) = \frac{1}{Z} \exp\left(\sum_i W_{f_i}(\omega)\right) \tag{2}$$

where, $W_{f_i} = \sum_{\overline{\mathbf{x}}_+} \theta_{\overline{\mathbf{x}}_+} N_{\overline{\mathbf{x}}_+}(\omega)$, is the total weight of the satisfied groundings of a first order formula f_i. We will refer the task of computing the expression W_{f_i} as #WSG (short for, *total weight of satisfied groundings*). Since, #WSG generalizes #SG (which is #P-complete), it is obvious that no efficient algorithm can exist for solving the #WSG problem.

3 Exact Encoding for untied formulas

[Venugopal *et al.*, 2015] and [Sarkhel *et al.*, 2016] used a constraint network to solve the #SG problem. In this section, we extend their approach to untied MLNs. Since untied MLNs have different weights for each combination of the "$+$" variables, we will use a dynamic Markov Random Field (dMRF)[1] instead of constraint network for each first-order formula and show that the partition function of the dMRF corresponds to the #WSG problem. It should be noted that our approach generalizes Venugopal et al.'s approach because a constraint network is an MRF having only 0/1 potentials.

We demonstrate our proposed dMRF encoding on the following clause: $f = \forall x, \forall y, \forall z\ \neg \mathtt{R}(+x, y) \vee \mathtt{S}(y, +z)$. Let the domain of each logical variable be $\{A, B\}$. The formula has a different parameter (instead of only one parameter) for each grounding of the variables x and z (denoted by $\boldsymbol{\theta}_{x,z}$). Our task here is to compute the total weight of $false$ groundings of f given ω. To solve the task, we create a MRF having three random variables (see Fig. 2), one for each logical variable x, y, and z, and having three potentials, $\phi_1(x, y)$, $\phi_2(y, z)$, and, $\phi_3(x, z)$. Here, $\phi_1(x, y)$ and $\phi_2(y, z)$ are exactly same as the functions ϕ_1 and ϕ_2 in the Fig. 1. For $\phi_3(x, z)$ each potential entry indexed by x, z is equal to the parameter $\theta_{x,z}$.

Given the above set up, notice that if we take a product of the three functions $\phi_1(x, y)$, $\phi_2(y, z)$, and, $\phi_3(x, z)$, then the resulting function $\phi(x, y, z)$ will have $\theta_{x,z}$ associated with an entry (x, y, z) iff $\mathtt{R}(x, y)$ is $true$ and $\mathtt{S}(y, z)$ is $false$. Since the weight of a ground formula, indexed by (x, y, z), is given by $\theta_{x,z}$ and it evaluates to $false$ iff $\mathtt{R}(x, y)$ is $true$ and $\mathtt{S}(y, z)$ is $false$, by extension $\phi(x, y, z) = \theta_{x,z}$ implies that the ground formula (x, y, z) is $false$. Therefore, we can compute the total weight of groundings of f that evaluate to $false$, by simply summing over all the entries in $\phi(x, y, z)$, which is the same as computing the partition function of the constructed MRF. To compute the total weight of $true$ groundings we need to compute the total weight of all possible groundings, which in this case, equals to $(|\Delta(y)| \times \sum_{x,z} \theta_{x,z})$. Subtracting the obtained partition function from this quantity will give us the total weight of $true$ groundings.

Next, we will precisely define how to encode the #WSG problem as a partition function computation problem.

MRF Encoding. Given a first-order clause f and a world ω, the corresponding dMRF $\mathcal{G}$ has a variable for each (universally quantified) logical variable in f. The domain of each variable in $\mathcal{G}$ is the set of constants in the domain of the corresponding logical variable. For each atom $\mathtt{R}(x_1, \ldots, x_u)$ in

[1]We refer to the constructed MRF as *dynamic* because the potential entries depend on the given world ω.

f, we have a potential ϕ in $\mathcal{G}$ defined as follows:

$$\phi(\overline{\mathbf{x}}_u) = \begin{cases} \omega_{\mathtt{R}(X_1,\ldots,X_u)} & \text{if R is negated in } f \\ \neg\omega_{\mathtt{R}(X_1,\ldots,X_u)} & \text{Otherwise} \end{cases}$$

where $\overline{\mathbf{x}}_u = (x_1 = X_1, \ldots, x_u = X_u)$ denotes an assignment to the dMRF variables and $\omega_{\mathtt{R}(X_1,\ldots,X_u)}$ is the projection of the world ω on the ground atom $\mathtt{R}(X_1, \ldots, X_u)$, namely the truth-value of the ground atom $\mathtt{R}(X_1, \ldots, X_u)$ in ω. Finally if $x_i, \ldots, x_j$ are variables associated with the '+' operator and if the corresponding parameters are $\theta_{x_i,\ldots,x_j}$ then we have a potential ψ in $\mathcal{G}$ defined as:

$$\psi(x_i, \ldots, x_j) = \theta_{x_i,\ldots,x_j}$$

By generalizing the arguments presented for the example MLN given above, we can show that:

Theorem 1. *Let f be a first-order clause, $\{x\}$ be logical variables in f, $\mathcal{X}_+ \subseteq \{x\}$ be logical variables associated with '+' operators, $\{\theta_{\overline{\mathbf{x}}_+}\}$ be parameters attached to f, ω be a world and let $Z_\mathcal{G}$ denote the partition function of the dMRF $\mathcal{G}$ obtained from (f, ω) using the MRF encoding. Then, $W_f(\omega) = \prod_{x \notin \mathcal{X}_+} |\Delta(x)|. \sum_{\overline{\mathbf{x}}_+} \theta_{\overline{\mathbf{x}}_+} - Z_\mathcal{G}$ where $\Delta(x)$ is the set of constants in the domain of x.*

Since we have reduced the #WSG problem to the partition function calculation problem, we can use any graphical model inference algorithm such as the junction tree algorithm [Lauritzen and Spiegelhalter, 1988] and leverage its advances and guarantees to compute the former efficiently. Thus, the time and space complexity of solving #WSG using the junction tree algorithm is exponential in the treewidth of $\mathcal{G}$.

An important property of our MRF encoding is that junction tree inference over it is guaranteed to have either the same or smaller complexity than inference in the constraint network encoding proposed by [Venugopal *et al.*, 2015]. This is because unlike the constraint network encoding which grounds all the '+' variables, the MRF encoding takes advantage of the structure induced by the '+' variables and this can yield significant reductions in complexity. For example, the treewidth of the MRF encoding of the formula $f = \forall x, \forall y, \forall z\ \neg\mathtt{R}(+x, y) \vee \mathtt{S}(y, z)$ (notice that unlike our running example there is no $+$ sign associated with z) is 1, while one will have to create $O(n)$ constraint networks having treewidth 1 each in order to solve the #WSG problem using Venugopal et al.'s approach. Thus, inference on the MRF encoding will be more efficient by a factor of $O(n)$ over the constraint network encoding. In summary,

Theorem 2. *The time and space complexity for solving #WSG using the junction tree algorithm is no-worse (i.e., at least the same or better) than solving #SG using same algorithm.*

Since, local-search based algorithms such as MaxWalkSAT and Gibbs sampling require solving #WSG at every iteration, we can efficiently implement them using our encoding. This will significantly improve their computational complexity.

4 Approximate Encoding for untied formulas

In the encoding described in the previous section, we added a new potential to account for the fact that different groundings corresponding to the '+' variables have different weights. This encoding is impractical if the treewidth of the graphical model or the number of objects is large. Therefore, in order to improve the computational complexity, we propose to reduce the number of weights associated with each formula by replacing weights which are close to each other by a single weight. We illustrate this idea next.

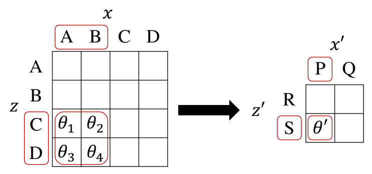


Figure 3: An example of clustering parameters associated with a full assignment to (x, z). The highlighted cells of the parameter matrix shows an example of clustering. After clustering we have $\theta' = (\theta_1 + \theta_2 + \theta_3 + \theta_4)/4$

Consider the formula in our running example: $f = \forall x, \forall y, \forall z\ \neg\mathtt{R}(+x, y) \vee \mathtt{S}(y, +z)$. Assume now that the domain of each logical variable is $\{A, B, C, D\}$. There are 16 different parameters associated with f, each corresponding to a full assignment to (x, z). Now assume that $\theta_{A,C}, \theta_{A,D}, \theta_{B,C}, \theta_{B,D}$, are almost the same. In that case we can combine the four (partially) ground formulas associated with these parameters into a single formula $f' = \forall x', \forall y, \forall z'\ \neg\mathtt{R}(x', y) \vee \mathtt{S}(y, z')$ having a single parameter θ' and $\Delta(x') = \{A, B\}$, $\Delta(z') = \{C, D\}$ and $\Delta(y) = \{A, B, C, D\}$. Similarly we can combine the other formulas yielding the four parameters given in Fig. 3.

The above idea can be set up as a clustering problem. However, in order to reduce the complexity, care must be taken to ensure that each cluster is the Cartesian product of subsets of domains of the '+' variables. In two dimensions, this is equivalent to biclustering [Hartigan, 1972]. For instance, in the above example, putting the diagonal elements of the matrix in Fig. 3 into one cluster will not yield reductions in the computational complexity. This is because all constants in the domain of x and z will be involved in the computations. On the other hand, consider the cluster given by $x \in \{A, B\}$ and $z \in \{C, D\}$ shown in Fig. 3. In this case, only two constants in each domain will be involved in computations.

Formally, we are interested in finding the optimal joint clustering over all the '+' variables. Since the parameters associated with an untied formula can be arranged in a tensor (i.e., a *multi-dimensional array*), where each logical variable represents one dimension or *order*, we define this problem as a *tensor clustering* problem.

Definition 1. *[Tensor Quantizer]: Given two order-p tensors, $\boldsymbol{A} = \{(a_{i_1 i_2 \ldots i_p}) \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_p}\}$ and $\boldsymbol{B} = \{(b_{j_1 j_2 \ldots j_p}) \in \mathbb{R}^{k_1 \times k_2 \times \ldots \times k_p}\}$, a function $\mathcal{Q}$, called a tensor quantizer, is a surjective function $\mathcal{Q} : \boldsymbol{A} \twoheadrightarrow \boldsymbol{B}$, such that $|\boldsymbol{A}| \geq |\boldsymbol{B}|$.*

A tensor qunatizer $\mathcal{Q}$ induces a partition $(\mathcal{P}^{(l)})$ on the set of values $(\Delta(i_l))$ for each dimension (i_l) of the tensor, and hence reduces its size. For example, in Fig. 3, a quantizer reduced the 4×4 matrix into a 2×2 matrix. Here the induced partitions are $\mathcal{P}^{(x)} = \{P, Q\} = \{\{A, B\}, \{C, D\}\}$

Algorithm 1 Cluster-Param
(Parameter tensor $\boldsymbol{\theta}_{\mathcal{X}_+}$, Cluster sizes $\boldsymbol{k}_{\mathcal{X}_+)}$

```
Select a random ordering $O$ of $\mathcal{X}_+$.
for each variable $x \in \mathcal{X}_+$ chosen according to $O$ do
    Create a random partition $\mathcal{P}^{(x)}$ of size $k_x$ of $\Delta(x)$
/* The set $\{\mathcal{P}^{(x)}\}$ defines the initial cluster assignment */
while the cluster assignment changes do
    for each variable $x \in O$ do
        Find the optimum partition $\mathcal{P}^{(x)}$ that minimizes Eq. (3)
        assuming that partitions associated with other variables
        are fixed.
```

and $\mathcal{P}^{(z)} = \{R, S\} = \{\{A, B\}, \{C, D\}\}$. We are interested in finding a quantizer having minimum quantization error:

$$\sum_{a \in \boldsymbol{A}} ||a - \mathcal{Q}(a)||_2^2 \tag{3}$$

where $\mathcal{Q}(a)$ is the cluster mean.

Solving the joint clustering problem is computationally hard. A naive approach which searches for all possible partitions of a given domain is not scalable since the number of possible partitions of size k for a set of size n (also known as the *Stirling numbers of the second kind*) grows exponentially with n. Hence, to solve this optimization problem we propose a greedy approach. Our method is described in Algorithm 1. It begins by selecting an ordering O of variables. For each variable, the algorithm assigns each of its domain values to a random cluster. Then for each variable x along the ordering O, it determines the best possible cluster assignment for x assuming that the assignments of the other variables are fixed. The algorithm repeats the assignment process until convergence. Since the cluster assignment step for each variable always reduces the objective function given in Eq. (3), Algorithm 1 is guaranteed to reach a local optima. Formally,

Theorem 3. *Algorithm 1 is guaranteed to converge to a local minima.*

Once we have obtained the joint-clusters, we can use Algorithm 2 to create a collection of Markov networks for each formula f, each of which has smaller treewidth as well as variables with smaller domain size than the Markov network obtained using the encoding described in the previous section. This reduces the complexity of the junction tree algorithm. The basic idea in Algorithm 2 is to create one first-order formula f' for each cluster and attach to it a weight equal to the cluster center. The formula represents a subset of groundings of f, the partition function of which can be computed efficiently by ignoring the weight potential. We summarize the computational complexity of using the junction tree algorithm for inference on the encoding returned by Algorithm 2 in the following theorem:

Theorem 4. *Let f be a MLN formula having n $+$-variables, d be the number of values in the domain of each $+$ variable, k be the number of clusters, and w be the treewidth of the Markov network associated with f' in Algorithm 2. Then, the time and space complexity of junction tree inference on the encoding returned by Algorithm 2 is $O\left(k^n \left(\frac{d}{k}\right)^{w+1}\right)$.*

Algorithm 2 Partition-Network
(MLN function f, Set of partitions $\{\mathcal{P}^{(l)}\}$)

```
Let, $\mathcal{G} = \{\}$.
Let, $\mathcal{X}_+ = \{x_1, \ldots, x_m\}$ be the set of '+' variables
Create a set of ordered tuples $\mathcal{O} = \{(\boldsymbol{P}_{i_1}^{(1)}, \ldots, \boldsymbol{P}_{i_m}^{(m)})\}$
  from $\mathcal{P}^{(1)} \times \ldots \times \mathcal{P}^{(m)}$.
foreach tuple $(\boldsymbol{P}_{i_1}^{(1)}, \ldots, \boldsymbol{P}_{i_m}^{(m)})$ in $\mathcal{O}$ do
    Create a formula $f'$ by replacing each $x_l \in \mathcal{X}_+$
      by a new logical variable $x'_l$ such that $\Delta(x'_l) = \boldsymbol{P}_{i_l}^{(l)}$.
    The weight of $f'$ is given by $\mu_{\boldsymbol{C}_{(\boldsymbol{P}_{i_1}^{(1)}, \ldots, \boldsymbol{P}_{i_m}^{(m)})}}$
    Encode $f'$ as a Markov network $G$, and Add $G$ to $\mathcal{G}$.
return $\mathcal{G}$.
```

5 Experiments

5.1 Setup

We evaluate the graphical model encodings proposed in our paper by using them within two inference algorithms: (1) Gibbs sampling to compute marginal probabilities and (2) MaxWalkSAT for MAP inference. Specifically, we perform the counting sub-step within both inference algorithms using our new graphical model encodings for the counting problem. We compare our exact encoding as well as approximate encoding methods with the encoding approach proposed in Venugopal et al. (we refer to this as '#SG'). We conducted our experiments on the following three datasets:

(i) **Student** MLN having the formula $\neg\texttt{Student}(x, +p) \vee \neg\texttt{Publish}(x, z) \vee\texttt{Cited}(z, +u)$

(ii) **WebKB** MLN from the Alchemy web page

(iii) **Citation Information-Extraction** (IE) MLN form the Alchemy web page

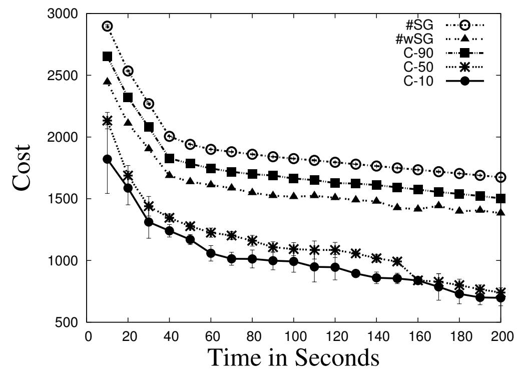
The primary goal of our experimental evaluation is to test the accuracy and scalability of our new encodings.

In our experiments, we implemented our system on top of the publicly available Magician system [Venugopal *et al.*, 2016] that uses #SG. That is, we used the same counting subroutines as the ones used by Magician, but replaced the graphical model encoding used in Magician with our proposed exact and approximate encoding methods respectively.

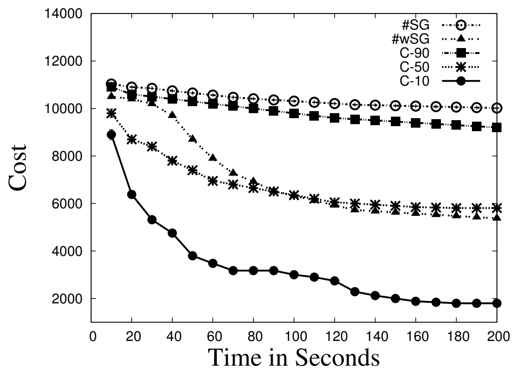
5.2 Results for MaxWalkSAT.

Fig. 4 shows the 'cost' of the MAP solution returned by each solver (the cost is typically reported when the MAP problem, which is a maximization problem, is solved as a minimization problem) as a function of time. The smaller the cost, the better the solver. Each solver was given 200 seconds for each dataset. We tested four versions of our solvers against '#SG.' They are '#WSG,' which is the exact encoding approach and three different 'C-p's which are obtained by using the clustering approach described in section 4, where the number of clusters equals $p\%$ of the number of constants in the domain.

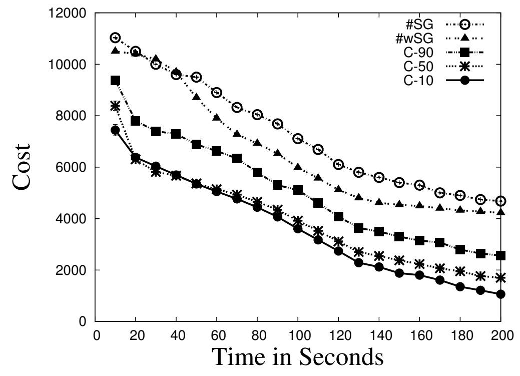
We observe that our new approaches are better than '#SG' and 'C-10' which compresses the parameter space to 10% of the original is the best performing solver. We conjecture that this is because, when p is small, MaxWalkSAT performs more flips/second which results in a better exploration of the state-space, which in turn improves its accuracy.



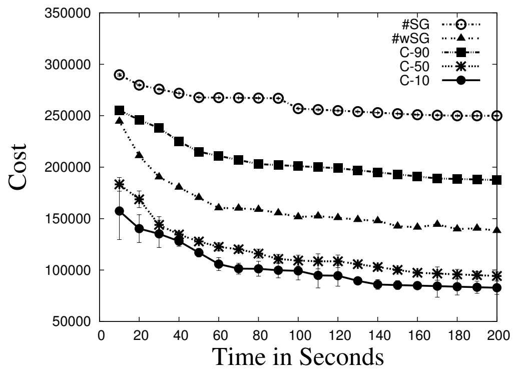
(a) Student(2.7K,0.8M)



(b) Student(30K,100M)



(c) WebKb(3.2K,1M)



(d) IE(3.2K,1M)

Figure 4: Cost vs Time: Cost of unsatisfied clauses(smaller is better) vs time for different domain sizes. Notation used to label each figure: MLN(numvariables, numclauses). Note: the quantities reported are for ground Markov network associated with the MLN. Standard deviation is plotted as error bars.

5.3 Results for Gibbs Sampling

For Gibbs sampling, we evaluate the convergence of the sampler and the accuracy of the computed marginals.

For measuring convergence of the Gibbs sampler, we use the Gelman-Rubin (G-R) Statistic [Gelman and Rubin, 1992]. For a well-mixed sampler, the G-R statistic should ideally decrease over time illustrating that the MCMC chain has mixed. To compute the G-R statistics, we set up five Gibbs samplers from random initialization and measure within chain and across chain variances for the marginal probabilities for 1000 randomly chosen ground query atoms. We compute the G-R statistics for each of these query atoms and report the mean G-R statistic.

For measuring accuracy, we use the KL-divergence between the average true marginal probabilities of ground query atoms and the average approximate marginal probability computed by the Gibbs sampler using our encodings. Note that obtaining the true marginal probabilities is infeasible for arbitrary evidence. Therefore, for this experiment, we did not use evidence, in which case, the average true marginal probability is equal to 0.5.

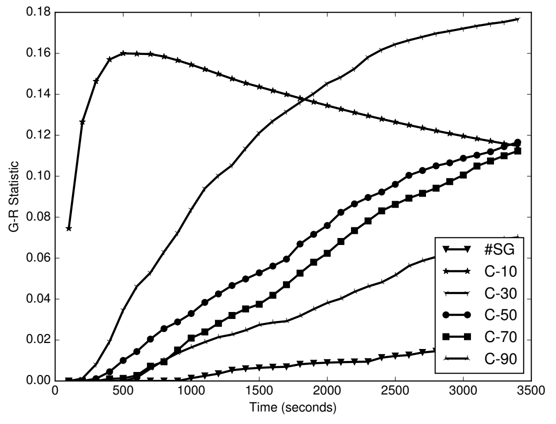
Fig. 5 compares the convergence of the Gibbs sampler for #SG as well as different clustering based approximations. We observe that C-10 has the best convergence, and as we increase the number of clusters, the convergence gradually becomes worse. When we have fewer clusters, the complexity of weighted counting is smaller, which helps us draw more samples and achieve faster convergence.

Fig. 6 compares the accuracy of Gibbs sampler when used with #SG as well as different clustering based approximations. As before, for a smaller number of clusters, we get higher accuracy within a fixed time (since we can collect more samples/second). As we increase the number of clusters, for the same time interval we collect fewer samples, and therefore, the accuracy of computing the marginals is reduced.

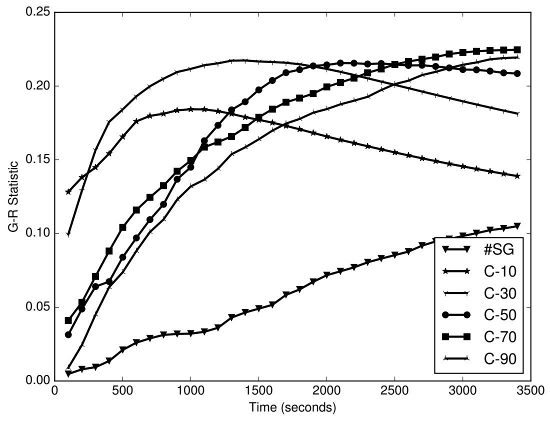
In both Fig. 5 and Fig. 6, the performance of #WSG was almost identical to C-90. Therefore, for readability, we do not show the curves for #WSG.

6 Conclusion

In this paper, we developed two novel graphical model encodings for MLN formulas that have different weights for different subsets of groundings. Both these encodings are useful in solving a computationally complex counting problem that manifests itself in several sampling and local-search based MLN inference algorithms. Namely, counting the sum of weights of the groundings of an MLN formula that are sat-

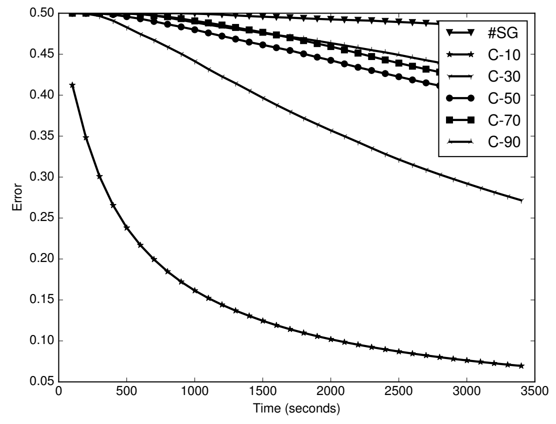


(a) Student(2.7K,0.8M)

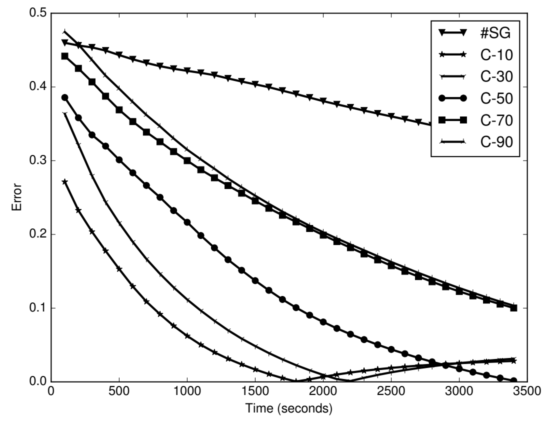


(b) WebKb(3.2K,1M)

Figure 5: Convergence of the Gibbs Sampler.



(a) Student(2.7K,0.8M)



(b) WebKb(3.2K,1M)

Figure 6: Accuracy of the Gibbs Sampler.

isfied in a given world. The first encoding was an exact encoding such that the partition function of the graphical model corresponds exactly to the number of satisfied groundings in the MLN formula. The second was an approximate encoding that clustered formula groundings with approximately similar weights together, which in several cases, reduces the treewidth of the encoded graphical model. We incorporated our encoding in two well-known inference algorithms, the Gibbs sampling, and the MaxWalkSAT. We demonstrated through experiments that both of our encoding techniques result in more scalable and accurate inference algorithms.

Although our approach achieves scalability for the inference task of the untied MLNs, the greatest challenge is to learn these models from the data. Since inference is a substep of many popular learning methods, in future, we will focus on developing a scalable learning algorithm for untied MLNs by extending our current work. We will also explore other efficient encodings (e.g., variational inference based encoding) for solving the counting task. Finally, since our cluster-based encoding introduces symmetry in the otherwise asymmetric untied MLNs, in future, we will implement novel lifted inference algorithms for these MLNs by leveraging our clustering idea.

Acknowledgments

This work was supported by the DARPA Probabilistic Programming for Advanced Machine Learning Program under AFRL prime contract number FA8750-14-C-0005. This work was also supported in part by National Science Foundation grants III-1527312, IIS-1652835 and IIS-1528037. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of DARPA, AFRL, ARO or the US government.

References

[Chen *et al.*, 2013] Chao Chen, Vladimir Kolmogorov, Yan Zhu, Dimitris Metaxas, and Christoph Lampert. Computing the m most probable modes of a graphical model. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS-13)*, 2013.

[Das *et al.*, 2016] Mayukh Das, Yuqing Wu, Tushar Khot, Kristian Kersting, and Sriraam Natarajan. Scaling lifted probabilistic inference and learning via graph databases. In *SIAM International Conference on Data Mining*, 2016.

[Domingos and Lowd, 2009] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool, San Rafael, CA, 2009.

[Gelman and Rubin, 1992] A. Gelman and D. B. Rubin. Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 7(4):457–472, 1992.

[Genesereth and Kao, 2013] Michael R. Genesereth and Eric Kao. *Introduction to Logic, Second Edition*. Morgan & Claypool Publishers, 2013.

[Gogate and Domingos, 2011] V. Gogate and P. Domingos. Probabilistic Theorem Proving. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 256–265. AUAI Press, 2011.

[Hartigan, 1972] John A Hartigan. Direct clustering of a data matrix. *Journal of the american statistical association*, 67(337):123–129, 1972.

[Kautz *et al.*, 1997] H. Kautz, B. Selman, and Y. Jiang. A General Stochastic Approach to Solving Problems with Hard and Soft Constraints. In D. Gu, J. Du, and P. Pardalos, editors, *The Satisfiability Problem: Theory and Applications*, pages 573–586. American Mathematical Society, New York, NY, 1997.

[Kersting *et al.*, 2010] K. Kersting, Y. E. Massaoudi, F. Hadiji, and B. Ahmadi. Informed Lifting for Message-Passing. In Maria Fox and David Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 1181–1186, 2010.

[Lauritzen and Spiegelhalter, 1988] S. L. Lauritzen and D. J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2):157–224, 1988.

[Riedel and McCallum, 2011] Sebastian Riedel and Andrew McCallum. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 46–50, 2011.

[Sarkhel *et al.*, 2016] Somdeb Sarkhel, Deepak Venugopal, Tuan Anh Pham, Parag Singla, and Vibhav Gogate. Scalable training of markov logic networks using approximate counting. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[Shavlik and Natarajan, 2009] Jude W. Shavlik and Sriraam Natarajan. Speeding up inference in markov logic networks by preprocessing to reduce the size of the resulting grounded network. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 1951–1956, 2009.

[Singla and Domingos, 2006a] P. Singla and P. Domingos. Entity Resolution with Markov Logic. In *Proceedings of the Sixth IEEE International Conference on Data Mining*, pages 572–582, Hong Kong, 2006. IEEE Computer Society Press.

[Singla and Domingos, 2006b] P. Singla and P. Domingos. Memory-Efficient Inference in Relational Domains. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, Boston, MA, 2006. AAAI Press. This volume.

[Singla *et al.*, 2014] Parag Singla, Aniruddh Nath, and Pedro Domingos. Approximate lifting techniques for belief propagation. In *Proceedings of the Twenty-Eigth AAAI Conference on Artificial Intelligence*, pages 2497–2504, 2014.

[Tran and Davis, 2008] Son D. Tran and Larry S. Davis. Event modeling and recognition using markov logic networks. In *10th European Conference on Computer Vision*, pages 610–623, 2008.

[Van den Broeck and Darwiche, 2013] G. Van den Broeck and Adnan Darwiche. On the complexity and approximation of binary evidence in lifted inference. In *Proceedings of the Twenty-Seventh Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2868–2876, 2013.

[Van den Broeck *et al.*, 2012] G. Van den Broeck, A. Choi, and A. Darwiche. Lifted relax, compensate and then recover: From approximate to exact lifted probabilistic inference. In *Proceedings of the Twenty-Eigth Conference on Uncertainty in Artificial Intelligence*, pages 131–141, 2012.

[Venugopal and Gogate, 2014] D. Venugopal and V. Gogate. Evidence-based Clustering for Scalable Inference in Markov Logic. In *Machine Learning and Knowledge Discovery in Databases*. 2014.

[Venugopal *et al.*, 2014] Deepak Venugopal, Chen Chen, Vibhav Gogate, and Vincent Ng. Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 831–843. ACL, 2014.

[Venugopal *et al.*, 2015] Deepak Venugopal, Somdeb Sarkhel, and Vibhav Gogate. Just count the satisfied groundings: Scalable local-search and sampling based inference in mlns. In *Proceedings of the Twenty-Nineth AAAI Conference on Artificial Intelligence*, pages 3606–3612, 2015.

[Venugopal *et al.*, 2016] D. Venugopal, S. Sarkhel, and V. Gogate. Magician: Scalable Inference and Learning in Markov logic using Approximate Symmetries. Technical report, Department of Computer Science, The University of Memphis, Memphis, TN, 2016. https://github.com/dvngp/CD-Learn.