SCALABLE INFERENCE TECHNIQUES FOR MARKOV LOGIC

by

Deepak Venugopal

APPROVED BY SUPERVISORY COMMITTEE:

Vibhav Gogate, Chair

Gopal Gupta

Sanda M. Harabagiu

Raymond J. Mooney

Vincent Ng

SCALABLE INFERENCE TECHNIQUES FOR MARKOV LOGIC

by

DEEPAK VENUGOPAL, BE, MS

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

August 2015

ACKNOWLEDGMENTS

I wish to thank my advisor Vibhav Gogate without whom this dissertation would not have been possible. Vibhav has been a great advisor who inspired me to work hard and maintain high standards of research by setting a fine example. He has shown me how to think and write with clarity, and always seemed to have an idea whenever I have been struck with seemingly unsolvable problems. Best of all, he has always been patient, friendly and approachable. I would certainly hope to emulate some of his traits as I embark on my own academic career. Next, I wish to thank members of my dissertation committee, Dr. Gopal Gupta, Dr. Sanda Harabagiu, Dr. Ray Mooney and Dr. Vincent Ng, for the time they have taken not only for my dissertation but also in my job search process. Particularly, Dr. Mooney was kind enough to serve on my committee from UT-Austin and I am grateful for this. I also wish to thank Somdeb, Chen, Dr. Parag Singla and Dr. Vincent Ng for collaborating with me on various projects due to which I gained a lot of insight into several different research areas.

Needless to say, my family has supported me enormously in completing this dissertation. It is hard to thank Krithika enough for the love, patience and faith that she has shown in me during numerous ups and downs associated with Ph.D. life. Were it not for her support, I certainly would not have made it this far. Special thanks to Esha for all the laughs and joy she has given me when writing this dissertation. My parents have given me every possible opportunity and let me pursue my interests at all times. I am extremely thankful for their wishes, love and encouragement at all stages of my life. I am also deeply indebted to my extended family in India that has given me much needed support during my Ph.D.

I will also cherish the friendships that I made at UT-Dallas. Somdeb, David, Tahrima, Li, Chen and several others have been great friends and I have enjoyed countless interesting

conversations with them about work and life in general. I thank them and will certainly miss their company as I finish my studies.

Last but not least, I thank various funding agencies: ARO, AFRL and DARPA for providing me financial support through grant numbers W911NF-08-1-0242, FA8750-14-C-0021 and FA8750-14-C-0005.

June 2015

SCALABLE INFERENCE TECHNIQUES FOR MARKOV LOGIC

Publication No. _____

Deepak Venugopal, PhD
The University of Texas at Dallas, 2015

Supervising Professor: Vibhav Gogate

In this dissertation, we focus on Markov logic networks (MLNs), an advanced modeling language that combines first-order logic, the cornerstone of traditional Artificial Intelligence (AI), with probabilistic graphical models, the cornerstone of modern AI. MLNs are routinely used in a wide variety of application domains including natural language processing and computer vision, and are preferred over propositional representations because unlike the latter they yield compact, interpretable models that can be easily modified and tuned. Unfortunately, even though the MLN representation is compact and efficient, inference in them is notoriously difficult and despite great progress, several inference tasks in complex real-world MLNs are beyond the reach of existing technology. In this dissertation, we greatly *advance the state-of-the-art in MLN inference*, enabling it to solve much harder and larger problems than existing approaches. We develop *several domain-independent principles, techniques and algorithms for fast, scalable and accurate inference that fully exploit both probabilistic and logical structure.*

This dissertation makes the following five contributions. First, we propose two approaches that respectively address two fundamental problems with Gibbs sampling, a popular approx-

imate inference algorithm: it does not converge in presence of determinism and it exhibits poor accuracy when the MLN contains a large number of strongly correlated variables. Second, we *lift* sampling-based approximate inference algorithms to the first-order level, enabling them to take full advantage of symmetries and relational structure in MLNs. Third, we develop novel approaches for exploiting *approximate symmetries*. These approaches help scale up inference to large, complex MLNs, which are not amenable to conventional lifting techniques that exploit only exact symmetries. Fourth, we propose a new, efficient algorithm for solving a major bottleneck in all inference algorithms for MLNs: counting the number of true groundings of each formula. We demonstrate empirically that our new counting approach yields orders of magnitude improvements in both the speed and quality of inference. Finally, we demonstrate the power and promise of our approaches on Biomedical event extraction, a challenging real-world information extraction task, on which our system achieved state-of-the-art results.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Over the past few decades, two core theories have contributed immensely to the success and popularity of Artificial Intelligence (AI) and machine learning (ML): *logic* and *probability*. Each theory enables the user to compactly and faithfully model different facets of the real-world. Specifically, first-order logic and its various subsets, enable the user to compactly encode *background relational knowledge* while probabilistic models and methods enable him or her to represent and reason about *uncertainty* in a principled manner.

In the past, research on logic and probability in AI/ML was carried out by researchers who affiliated themselves with different sub-communities and conferences. These researchers seldom interacted or shared ideas with each other. A problem with this "to each his own" approach is that many real-world problems are of such size and complexity that they cannot be solved accurately until and unless we marry logical and probabilistic representation and reasoning techniques. Although, researchers as far back as Leibniz in the 17^{th} century and relatively recently Carnap (Carnap, 1950) and Gaifman (Gaifman, 1964) had emphasized the need and worked on unifying logic and probability, it was not until recently that these efforts have gained significant momentum, coinciding with the rise of a new sub-field of AI/ML called *statistical relational learning* (SRL) (Getoor and Taskar, 2007).

In this dissertation, we focus on Markov logic (Domingos and Lowd, 2009), a modeling tool for statistical relational learning that combines probabilistic graphical models (Pearl, 1988) with first-order logic. Although, other languages that combine the two do exist (e.g., `Blog` (Milch et al., 2005), `IBAL` (Pfeffer, 2001), `Problog` (De Raedt et al., 2007), `BALP` (Raghavan and Mooney, 2011), `PSL` (Broecheler et al., 2010), etc.), we focus on Markov logic because it is more intuitive and more widely used than the aforementioned languages.

At a high level, a Markov logic network (MLN) is a set of weighted first-order logic formulas. The formulas encode background knowledge and the weights, which are real numbers, express the belief in the truth of the corresponding formula. Higher the weight, higher the probability of the corresponding formula being true in a possible world, all other things being equal. Given a set of constants that model objects in the real-world domain, the MLN represents a Markov network, an undirected probabilistic graphical model. The Markov network represents a joint probability distribution over the ground atoms – all possible propositional atoms obtained by substituting each argument of each predicate with a constant in the domain – and thus an MLN can be used to answer any query over the ground atoms via probabilistic inference.

The key advantage of MLNs is that the user can easily write concise, interpretable models. As a result, they are widely used by application designers for modeling complex problems in a variety of domains including natural language understanding, robotics, computer vision, information extraction, and social networks. Unfortunately, even though great progress has been made in inference and learning algorithms, several real-world MLNs are so large and complex that inference (over them) remains out of reach of even the most advanced methods.

The main contribution of this dissertation is a suite of novel methods that significantly scale up and advance the state-of-the-art in MLN inference, and can solve much harder and more complex tasks than is possible today. Specifically, we develop several fundamental techniques for *fast, scalable and more accurate inference that fully exploit logical structure and symmetries* encoded in the MLN representation. In developing these techniques, we draw on concepts from diverse fields such as sampling theory, machine learning, database theory, constraint satisfaction, SAT and discrete optimization. We demonstrate the wide applicability of our novel methods by using them to solve a Biomedical event extraction task, a challenging task in the information extraction domain.

The road map of this dissertation is as follows. In the next section, we present a brief overview of our contributions. Chapter 2 provides the necessary background to follow the

rest of our work. Chapters 3-7 present our original research contributions and chapter 8 presents possible future directions.

1.1 Contributions

Our contributions can be best explained using Figure. 1.1, which schematically illustrates the space of MLN inference problems. The problems are divided into several overlapping regions, which we call classes. Each class is labeled with the type of inference techniques that can be used to accurately and efficiently solve problems in the class. The inner-most class corresponds to exact propositional inference algorithms, namely, propositional algorithms that work on the probabilistic graphical model represented by the MLN and output exact answers. Unfortunately, these algorithms can solve only a tiny fraction of the inference problems in practice, even if one uses advanced (exact) methods such as Bucket elimination (Dechter, 1999), AND/OR search (Dechter and Mateescu, 2007), recursive conditioning (Darwiche, 2001) and junction trees (Lauritzen and Spiegelhalter, 1988). Approximate propositional inference algorithms (e.g., MCMC sampling (Liu, 2001), Belief Propagation (Yedidia et al., 2000), etc.) output approximate answers and are naturally more scalable; they can accurately solve a much larger class of inference problems than exact methods. The term lifted inference algorithms refers to algorithms that take advantage of symmetries in the relational representation (cf. (Poole, 2003; de Salvo Braz, 2007; Gogate and Domingos, 2011b)). These algorithms include exact propositional inference algorithms as special cases and when symmetries are present, are much more scalable than their propositional counterparts. Similar to propositional inference, exact lifted inference is subsumed by *unbiased* approximate lifted inference. We have used the term *unbiased* to denote conventional approximation inference techniques such as importance sampling, Belief propagation and Gibbs sampling which output estimates having strong guarantees (e.g., convergence to the correct distribution, consistency, etc.). Finally, there is a vast landscape of extremely challenging MLNs which

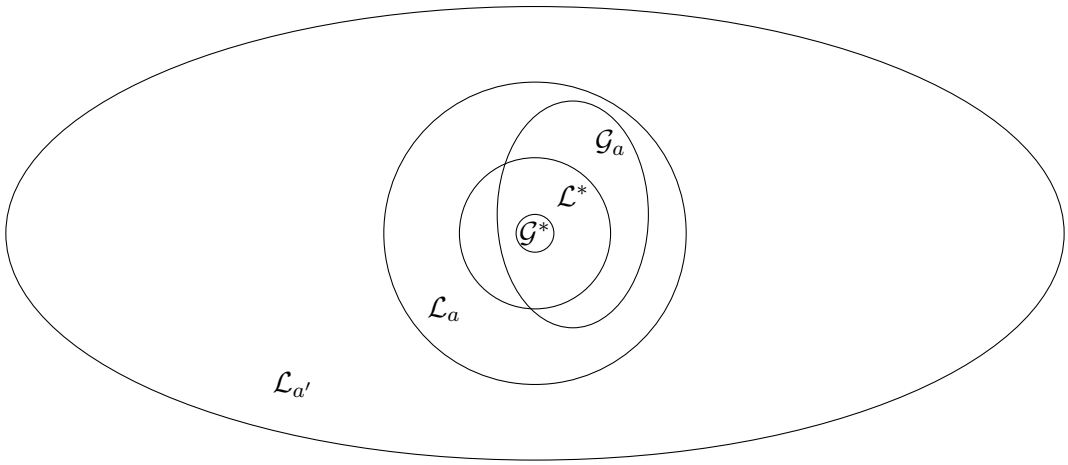


Figure 1.1. The space of inference problems in MLNs. The space is divided into several overlapping regions. The label associated with a region gives the name of the general inference technique that is capable of efficiently solving problems in the region. $\mathcal{G}^*$ denotes the class of exact graphical model inference algorithms (or exact propositional inference), $\mathcal{G}_a$ denotes the class of approximate graphical model inference algorithms (approximate propositional inference), $\mathcal{L}^*$ denotes the class of exact lifted inference algorithms, $\mathcal{L}_a$ denotes the class of lifted approximate inference algorithms (*unbiased* approximations with strong provable guarantees), $\mathcal{L}_{a'}$ denotes the class of approximate lifted inference algorithms with new types of approximations where we may need to sacrifice strong theoretical guarantees to achieve practical scalability.

both exact and approximate, propositional as well as lifted methods with strong guarantees are unable to handle. In such cases, in order to achieve scalability, we require novel approximations and sometimes need to sacrifice strong theoretical guarantees in lieu of weaker guarantees. We call such approximations biased approximations.

In this dissertation, we propose algorithms corresponding to several of the general inference techniques illustrated in Figure. 1.1 and in each case, we push the boundaries of these techniques outwards, significantly expanding their range (schematically, our new techniques increase the area of each oval in the figure). We describe our specific contributions in each subsequent chapter next.

Chapter 3 describes two novel approaches for scaling up Gibbs sampling (Geman and Geman, 1984), arguably the most widely used Markov Chain Monte Carlo (MCMC) algorithm. Our two approaches respectively address the following issues with Gibbs sampling: (1) it

performs poorly in presence of logical dependencies and determinism, and (2) it exhibits slow convergence when several correlated variables are present. These issues are especially problematic in the context of MLNs because most real-world MLNs have strong logical and deterministic dependencies (e.g., formulas having high weights) and a large number of correlated variables.

Our first technique called `GiSS` (Venugopal and Gogate, 2013a) improves the performance of Gibbs sampling on models having hard deterministic constraints. Deterministic constraints fracture the support (the subset of assignments having non-zero probability) of the probability space into multiple regions. On such models, Gibbs sampling gets struck in a local cluster in the state space and fails to converge to the desired posterior probability distribution (the distribution represented by the MLN given evidence). As a result, it may yield inaccurate answers. `GiSS` combines Gibbs sampling with SampleSearch (Gogate and Dechter, 2011), an importance sampling algorithm that generates high quality samples from deterministic spaces. At a high level, SampleSearch samples the clusters, hopping from one cluster to another, and Gibbs sampling samples the assignments within each cluster. As a result, the entire support of the probability space is reachable, and the algorithm (`GiSS`) converges to the desired distribution.

Our second approach (Venugopal and Gogate, 2013b) improves the performance of Gibbs sampling on models having several strongly correlated variables. It is known that the two strategies of collapsing and blocking (Liu, 2001), when applied separately, can improve the convergence of Gibbs sampling on such models. We show that the convergence can be further improved by cleverly combining the two strategies. The combination is non-trivial because collapsing and blocking are orthogonal – using one diminishes the performance gains achieved by the other and vice versa. We solve this problem by formulating it as a multi-objective optimization problem and show that a so-called *Pareto optimal solution* to the optimization problem yields a dynamic/adaptive MCMC sampler that is superior in terms of accuracy

and convergence to Gibbs sampling and its various advanced variations on a wide variety of challenging inference problems.

Chapter 4 presents sampling-based *lifted inference* algorithms for MLNs. Unlike propositional (graphical model) inference algorithms that operate on the Markov network obtained by grounding the MLN, lifted algorithms (Poole, 2003; de Salvo Braz, 2007; Gogate and Domingos, 2011b) exploit symmetries in the relational representation and operate as much as possible on the compact first-order representation, grounding only when necessary. As a result, they are far more scalable than propositional approaches. We lift two widely used sampling algorithms to the first-order level: (i) Blocked Gibbs sampling (Jensen et al., 1993) which is an advanced variant of Gibbs sampling and (ii) Importance Sampling (Rubinstein, 1981). We briefly describe the two algorithms, in turn, next.

Unlike Gibbs sampling, which samples just one randomly selected variable at each iteration, the blocked Gibbs sampling algorithm partitions the variables into blocks, and jointly samples all variables in a randomly selected block at each iteration. As the number of variables in each block (block size) is increased, the accuracy typically improves but so does the computational complexity and thus there is a trade-off. Our main idea in Lifted Blocked Gibbs sampling (LBG) (Venugopal and Gogate, 2012) is to cluster first-order atoms rather than propositional variables. We show that unlike propositional blocking which improves accuracy at the expense of increased complexity, in some cases, increasing the lifted block size may increase the accuracy and reduce the complexity. This is because increasing the block size gives the lifted algorithms more opportunities to exploit symmetries. We show that LBG has smaller variance and is therefore likely to be more accurate than blocked Gibbs sampling. We also show via a detailed empirical evaluation on large real-world MLNs that LBG is far superior to propositional approaches in terms of scalability and convergence.

Our second contribution in Chapter 4 is lifting the importance sampling algorithm. Unlike Gibbs sampling, which uses dependent unweighted samples for estimating various statistics,

importance sampling (IS) uses independent weighted samples. Since it is hard to generate independent samples from the posterior distribution (the distribution represented by the MLN), importance sampling draws independent samples from a so-called proposal distribution and corrects the bias by weighting each sample appropriately. The accuracy of IS depends on how close the proposal is to the posterior distribution. Constructing a good proposal is challenging in practice, and is the main sub-task in IS.

In lifted importance sampling (LIS) (Gogate et al., 2012), we exploit symmetries in the MLN representation to design a proposal distribution that is not only highly accurate but also *lifted*. Specifically, we leverage what are known as *lifting rules* (Jha et al., 2010), namely, rules that identify symmetries from the first-order representation, to group together symmetric ground atoms. We show that our grouping strategy provably reduces the variance of importance sampling because each sample from the lifted proposal distribution is worth several distinct samples from a propositional proposal distribution (it is known that the accuracy improves as the number of samples is increased). We also develop new lifting rules which identify several new symmetries that previous methods (Gogate and Domingos, 2011b) are unable to detect, which further improves the quality of our estimates. We illustrate how these rules can be used to construct the lifted proposal distribution in a principled, tractable manner and further, develop an adaptive approach that improves the parameters of the proposal distribution over time. We demonstrate empirically that our approach significantly improves the accuracy and scalability of importance sampling in MLNs.

Lifted inference algorithms are extremely powerful when they are able to find "exploitable symmetries" in the first-order representation. On such MLNs, they work directly on the compact first-order representation without consulting the much larger ground representation (Markov network). Unfortunately, the class of MLNs that have exploitable symmetries (according to our current knowledge of lifted inference) is extremely restrictive (Jha et al., 2010; Van den Broeck, 2011). Moreover, as the amount of evidence, namely information that

some atoms in the MLN are observed to be either true or false, is increased, the performance of lifted inference algorithms, even on MLNs having exploitable symmetries, deteriorates considerably. This is because in contrast to propositional inference where evidence can be used to prune a large portion of the graphical model, evidence breaks symmetries and increases the size of the lifted network. Because of these two issues, in practice, lifted inference algorithms often ground a large portion of the MLN and have the same scalability problems as propositional inference.

Chapter 5 addresses the aforementioned problems and presents a novel, practical approach that exploits *approximate symmetries* (Venugopal and Gogate, 2014a,b) and easily scales to large problems defined over MLNs having arbitrary *non-liftable* structure and large amount of (symmetry-breaking) evidence. In order to find approximate symmetries, we formulate a clustering problem and utilize standard unsupervised machine learning algorithms such as *K-Means* to partition objects into clusters, such that objects that are approximately symmetrical (exchangeable) from an inference perspective are in the same cluster. Given a clustering, we generate a *compressed* representation of the original MLN by replacing all objects in each cluster by a new (meta) object. Then, we perform lifted inference over the compressed MLN and use these results over the full MLN under the assumption that all objects within a cluster have the same statistics.

To learn approximate symmetries effectively, we develop a novel distance measure (which will be used by the clustering algorithm) between objects that is based on the evidence presented to the MLN. Specifically, our premise in designing the distance measure is that if evidence presented to the MLN affects domain objects in a "similar" manner, then such objects can typically be exchanged with each other without affecting the inference results. Thus, our method dynamically adapts itself as the observed evidence changes. We perform a detailed evaluation of our approach on several real-world MLNs with arbitrary evidence utilizing different clustering and different inference algorithms to demonstrate the generality

of our approach. Our results clearly show that our approach can accurately scale up inference to far larger and more complex MLNs, which is not possible using existing inference methods.

An issue with approximate symmetries is that it introduces bias; the distribution represented by compressed MLN can be quite far from the one represented by the original MLN. To control the bias, we develop a novel importance sampler that utilizes approximate symmetries for scalability but has provable guarantees on the estimates (Venugopal and Gogate, 2014b) in that as the number of samples is increased, the bias will go down, and disappear in the limit of infinite samples (in the statistics literature, such schemes are called *asymptotically unbiased*). Specifically, we scale up the two key steps in importance sampling as follows. First, use the compressed MLN to design an accurate proposal distribution and sample from this proposal in a tractable, lifted manner. Each sample from our proposal corresponds to multiple approximately-symmetric worlds in the original MLN. Second, we develop a novel weighting scheme that approximates the importance weights tractably since computing the exact importance weight of each lifted sample is computationally expensive. To reduce the variance of our sampler, we develop a method to perform lifted *Rao-Blackwellisation* (Casella and Robert, 1996), namely combine exact lifted inference with lifted sampling in order to improve the accuracy.

In chapter 6, we introduce a novel strategy for scaling up MLN inference that is quite distinct from the typical "symmetry-exploiting" techniques developed in the previous three chapters (Venugopal et al., 2015). Our idea is based on the observation that several inference algorithms, whether lifted or not, needs to solve the following task: given a possible world (a truth assignment to all ground atoms), find the number of groundings of each first-order formula that evaluate to true. Existing inference systems use a naive method which is very inefficient and slow for solving this problem and is one of the main reasons for their poor scalability. We develop a new, efficient approach for solving this problem. The key idea in our approach is to encode the counting problem as the problem of counting the number of

solutions of a constraint satisfaction problem (CSP) (Dechter, 2003). The main advantage of our encoding is that all guarantees, approximations and algorithms from decades of research in CSPs can be easily leveraged to solve the counting problem efficiently. We apply our approach to two classical approximate inference algorithms: Gibbs sampling and MaxWalkSAT (Kautz et al., 1997) (a local search solver). We show that in both these algorithms, the key computational steps involve solving the encoded CSP where the constraints change over time (dynamic CSP). We develop a junction tree (Lauritzen and Spiegelhalter, 1988; Dechter, 1999) based algorithm (a classic approach for counting the number of solutions of a CSP) for efficiently computing the number solutions of the dynamic CSP. Our detailed experiments on several real-world MLNs with large domains clearly show that our approach is orders of magnitude more scalable than existing state-of-the-art MLN systems such as Alchemy (Kok et al., 2008) and Tuffy (Niu et al., 2011).

In Chapter 7, we propose a new MLN model for automatically extracting biomedical events from raw text (Venugopal et al., 2014).[1] MLNs are well-suited for this task because unlike traditional machine learning algorithms (e.g., SVMs), MLNs can perform *joint inference*, i.e., exploit relational dependencies that typically exist between different events. We model these dependencies by constructing an MLN that has both hard (infinite weight) and soft formulas (we learn these weights from data). In practice, joint inference is extremely challenging and exhibits poor scalability. Specifically, it turns out that certain rich linguistic features (Chen and Ng, 2012; Huang and Riloff, 2012b; Li et al., 2012) that are essential for accurate event extraction are extremely high dimensional and learning their weights reliably in our joint model is computationally infeasible (it will also require a large amount of labeled data for producing reliable estimates). SVMs on the other hand, lack the ability to perform joint inference but due to their simpler formulation can easily and effectively learn high-dimensional features from limited data. To get the best of both worlds, we learn high

[1] Joint work with Prof. Vincent Ng and Chen Chen.

dimensional linguistic features using SVMs and embed these features as low-dimensional soft-evidence in the MLN. However, even with this hybrid MLN, joint inference (and consequently weight-learning) is still infeasible using off-the-shelf inference methods (ILPs (Roth and Yih, 2005), MaxWalkSAT (Kautz et al., 1997), etc.) since the ground Markov network is extremely large. Instead, we utilize first-order structure, namely lifted inference in our MLN and drastically reduce the space/time complexity of joint inference in our event extraction system. Our approach yields promising results on three BioNLP datasets; our system was better or on par with the best published results and significantly outperformed all previous MLN-based systems.[2]

[2]Our system did not participate in the BioNLP competition. Results compared against both competition winners as well other systems that later tested against the same datasets

CHAPTER 2

BACKGROUND

In this chapter, we present a brief background on Markov networks and Markov logic. Specifically, we briefly review the main concepts in representation, inference and learning. We also provide an overview of sampling based statistical techniques which we have utilized heavily throughout this dissertation. For more details on first-order logic, refer to (Russell and Norvig, 2003; Genesereth and Kao, 2013), on Markov networks and probabilistic graphical models, refer to (Koller and Friedman, 2009; Darwiche, 2009), on Markov logic, refer to (Domingos and Lowd, 2009) and on sampling methods, refer to (Liu, 2001).

2.1 Representation

2.1.1 Propositional Logic

The language of propositional logic consists of atomic sentences called propositions or atoms, and logical connectives such as $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation), $\Rightarrow$ (implication) and $\Leftrightarrow$ (equivalence). Each proposition takes values from the binary domain $\{\texttt{False}, \texttt{True}\}$ (or $\{0, 1\}$). A propositional formula f is an atom, or any complex formula that can be constructed from atoms using logical connectives. For example, $\texttt{A}$, $\texttt{B}$ and $\texttt{C}$ are propositional atoms and $f = \texttt{A} \vee \neg\texttt{B} \wedge \texttt{C}$ is a propositional formula. A *knowledge base* (KB) is a set of formulas. A *world* is a truth assignment to all atoms in the KB.

2.1.2 First-Order Logic

First-order logic (FOL) generalizes propositional logic by allowing relational structure between atoms. Throughout this dissertation, we assume a strict subset of first-order logic,

called *finite Herbrand logic* (cf. (Genesereth and Kao, 2013)). Thus, we assume that we have no function constants and finitely many object constants. A first-order knowledge base (KB) is a set of first-order formulas. A formula in first-order logic is made up of quantifiers ($\forall$ and $\exists$), logical variables, constants, predicates and logical connectives ($\vee$, $\wedge$, $\neg$, $\Rightarrow$, and $\Leftrightarrow$). We denote logical variables by lower case letters (e.g., x, y, z, etc.) and constants by strings that begin with an upper case letter (e.g., A, Ana, Bob, etc.). Constants model objects in the real-world domain. A predicate is a relation that takes a specific number of arguments (called its arity) as input and outputs either `True` (synonymous with 1) or `False` (synonymous with 0). We denote predicates by strings in typewriter font (e.g., `R`, `S`, `Smokes`, etc.). We denote the i-th argument of a predicate `R` as $i_{\texttt{R}}$. A term is either a logical variable or a constant that can substitute an argument of a predicate. The domain of a logical variable x refers to the set of constants that can be substituted for x and we represent this as Δ_x. An atom is an instance of a predicate in some formula. Throughout this dissertation, we follow the convention that whenever we refer to an atom, i.e., we write the predicate symbol followed by a parenthesized set of terms, for instance, $\texttt{R}(x_1, x_2 \ldots x_n)$, we use this notation in a general sense to refer to any atom in the first-order KB that unifies with $\texttt{R}(x_1, x_2 \ldots x_n)$. This gives us an easy way to refer to all atoms that correspond to a specific predicate symbol, without complicating notation. We denote all possible groundings that can be generated from a predicate `R` using $\Delta_{\texttt{R}}$. Any atom with arity equal to one is called a singleton. That is, $\texttt{R}(x)$ is a singleton and $\texttt{S}(x, A)$ is also a singleton since it can be renamed as $\texttt{S}_A(x)$. Clearly, for a singleton $\texttt{R}(x)$, $\Delta_x = \Delta_{\texttt{R}}$.

A first-order formula is recursively defined as follows: (i) An atomic formula contains a single predicate; (ii) Negation of an atomic formula 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.

We assume that each argument of each predicate is typed and can only be assigned to a fixed subset of constants. By extension, each logical variable in each formula is also typed.

We further assume that all first-order formulas are disjunctive (clauses), have no free logical variables (namely, each logical variable is quantified), have only universally quantified logical variables (CNF). Note that all first-order formulas can be easily converted to this form. To simplify notation, we at times skip explicitly writing $\forall$ for each variable. Throughout this dissertation, any variable for which no quantifier is specified is implicitly assumed to be a universally quantified variable. A ground atom is an atom that contains no logical variables, i.e., each variable is substituted with a constant. A possible world, denoted by ω, is a truth assignment to all possible ground atoms that can be formed from the constants and the predicates.

2.1.3 Discrete Graphical Models

Graphical models refer to models that integrate classical graph theory and automated reasoning methods. In this dissertation, we focus only on discrete graphical models, namely models that describe a joint distribution with discrete random variables. From here on, when we use the term random variable, we implicitly mean a discrete random variable. The two most widely used probabilistic graphical models (PGMs) are Bayesian networks and Markov networks. Bayesian networks are directed or causal models (Pearl, 1988), where the joint distribution is represented as a product of conditional probability distributions. Specifically, each conditional distribution also called a conditional probability table (CPT) is specified for a variable given all its parents in the Bayesian network. Markov networks, on the other hand were originally derived from Markov Random Fields that were used to model physical phenomena in statistical physics (cf. (Kindermann and Snell, 1980)). Markov networks are un-directed models and represent the joint probability distribution as a product of potentials/functions, where each function is defined over a subset of random variables and maps each assignment to these variables to a non-negative real number. In this dissertation, for the most part, we focus our discussions on Markov networks as they form the basis for Markov logic.

We use the following notation to describe graphical models. We represent variables by capital letters (e.g., X), values in the domain of a variable by corresponding small letters (e.g., x) and an assignment of a value x to a variable X by $\overline{x}$. We represent sets of variables by bold capital letters (e.g., $\mathbf{X}$) and assignment of values to all variables in the set $\mathbf{X}$ by $\overline{\mathbf{x}}$. For simplicity of exposition, we assume that all variables are bi-valued or binary.

Markov Networks

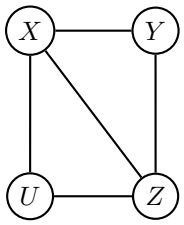
Definition 1. *A (discrete) PGM or a Markov network, denoted by $\mathcal{M}$ is a pair $\langle \boldsymbol{X}, \Phi \rangle$ where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of discrete variables (i.e., they take values from a finite domain) and $\Phi = \{\phi_1, \ldots, \phi_m\}$ is a set of positive real-valued functions (or potentials). Each function is defined over one or more variables and the scope of a function, $S(\phi)$, is the union of all the variables occurring in ϕ. $\mathcal{M}$ represents a probability distribution called the Gibbs distribution which is the normalized product of all its potentials as given by the following equation.*

$$P(\overline{\mathbf{x}}) = \frac{1}{Z} \prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}_{S(\phi)}) \tag{2.1}$$

where $\overline{\mathbf{x}}$ is an assignment of values to all variables in $\mathbf{X}$, $\overline{\mathbf{x}}_{S(\phi)}$ is the projection of $\overline{\mathbf{x}}$ on the scope $S(\phi)$ of ϕ, and Z is a normalization constant called the partition function. We will often abuse notation and write $\phi(\overline{\mathbf{x}}_{S(\phi)})$ as $\phi(\overline{\mathbf{x}})$.

The primal (or interaction) graph associated with $\mathcal{M} = \langle \mathbf{X}, \Phi \rangle$, denoted by $\mathcal{G}$, is an undirected graph which is defined as follows.

Definition 2. *Given $\mathcal{M} = \langle \boldsymbol{X}, \Phi \rangle$ where $\mathbf{X} = \{X_1, \ldots, X_n\}$, the primal graph, $\mathcal{G}$, of $\mathcal{M}$ is an un-directed graph, ($\mathbf{V}$, $\mathbf{E}$), that contains n vertices, V_1, V_2 ... V_n, where V_i corresponds to variable X_i. For each function $\phi \in \Phi$, $\mathcal{G}$ contains a clique between a set of vertices $\mathbf{V}' \subseteq \mathbf{V}$, where $V_k \in \mathbf{V}' \Rightarrow X_k \in S(\phi)$.*



(a) $\mathcal{G}$

X	Y	Z	ϕ_1
0	0	0	4
0	0	1	1.25
0	1	0	2.3
0	1	1	5
1	0	0	4
1	0	1	3
1	1	0	1
1	1	1	2.3

(b) ϕ_1

X	U	ϕ_2
0	0	1.2
0	1	2.3
1	0	1.2
1	1	2

(c) ϕ_2

U	Z	ϕ_3
0	0	2
0	1	3
1	0	2.5
1	1	4

(d) ϕ_3

X	Y	Z	U	$P()$
0	0	0	0	$\frac{1}{Z} \times 4 \times 1.2 \times 2$
0	0	0	1	$\frac{1}{Z} \times 4 \times 2.3 \times 2.5$
...	...	...	...	...
...	...	...	...	...
1	1	1	1	$\frac{1}{Z} \times 2.3 \times 2 \times 4$

(e) Gibbs Distribution

Figure 2.1. An example Markov network. (a) shows the primal graph, (b) - (d) show the potentials defined on cliques in the primal graph and (e) shows the joint distribution represented by the Markov network.

An example Markov network is shown in Figure 2.1. (a) shows the primal graph of the Markov network. The joint distribution which is actually exponential in 4 variables as shown in (e) is stored in a factored form across three potentials shown in (b), (c) and (d). Further, to convert the weights, i.e., product of potentials into a probability distribution, we need to plug-in the partition function as shown in (e).

D-Separation

D-Separation is a graphical test used compute conditional independencies in a PGM. A conditional independence relation between three sets of variables, $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ is represented as I($\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$) to mean that $\mathbf{X}$ is conditionally independent of $\mathbf{Y}$ given $\mathbf{Z}$ or $\mathbf{Z}$ d-separates $\mathbf{X}$ and $\mathbf{Y}$. In a Markov network, the conditional independence relation is obtained using graph separation, i.e., I($\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$) holds if removing the nodes corresponding to $\mathbf{Z}$ in the primal graph results in a graph where there is no path between any node in $\mathbf{X}$ to any node in $\mathbf{Y}$.

A	B	ϕ
0	0	10
0	1	10
1	0	1
1	1	10

(a) Potential

f	w
$\neg A \wedge \neg B$	log(10)
$\neg A \wedge B$	log(10)
$A \wedge \neg B$	log(10)
$A \wedge B$	log(1)

(b) Weighted Formulas

Figure 2.2. Example to illustrate conversion of the Markov network potential in (a) to weighted formulas in a log-linear model as shown in (b).

In Figure 2.1, $\{Y\}$ and $\{U\}$ are d-separated by $\{X, Z\}$. The Markov blanket (MB) of a variable X is a set of variables $\mathbf{M}$ such that I(X,$\mathbf{X}\setminus \mathbf{M}$,$\mathbf{M}$) holds. For a Markov network, $MB(X)$ is the set of nodes in the primal graph that are directly connected to X.

Log-Linear Models

Markov networks are frequently described as log-linear models where each entry in a potential is converted to a *feature* (propositional formula) that has an associated weight. The weight is computed as the logarithm of the function value. Figure 2.2 shows the conversion of an example Markov network potential to weighted propositional formulas.

For a log-linear model, the joint distribution is represented by the following equation.

$$P(\overline{\mathbf{x}}) = \frac{1}{Z} exp\left(\sum_{f_i} w_i \mathbb{I}_{\overline{\mathbf{x}}}(f_i)\right) \tag{2.2}$$

where Z is the normalization constant or partition function, f_i is the i-th *feature*, w_i is the weight of the feature and $\mathbb{I}_{\overline{\mathbf{x}}}(f_i)$ is equal to 1 if $\overline{\mathbf{x}}$ satisfies the logical formula f_i and 0 otherwise.

It is important to note that encoding potentials of Markov networks in logic has desirable properties. Namely, it allows us to compactly represent the Markov network in many cases. For instance, in the example shown in Figure 2.2, all formulas with equal weights can be merged and we can represent the entire potential table which is exponential in the number of variables, by a single logical expression, i.e., $A \Rightarrow B$. Further, using a logical encoding also

allows us to leverage a vast amount of research in the logical inference and SAT communities (e.g., DPLL (Davis and Putnam, 1960)) for efficient probabilistic inference.

2.1.4 Markov Logic Networks

Markov logic networks (MLNs) (Richardson and Domingos, 2006; Domingos and Lowd, 2009) can be viewed as lifted log linear models that combine Markov networks with first-order logic. MLNs soften the hard constraints expressed by formulas in first-order logic by attaching weights to each formula. The weights lie between $-\infty$ and ∞, and have an intuitive meaning as follows. If formula f has weight w, for $0 < w < \infty$, it implies that worlds that satisfy f are more likely than worlds that do not satisfy f. For $-\infty < w < 0$, it implies that worlds that do not satisfy f are more likely. At the extreme, a *hard formula* or a formula with weight equal to ∞ (or $-\infty$) implies that f is always true (or false).

An MLN can be regarded as a template to specify extremely large PGMs. That is, given a set of constants that represent real-world objects in a domain-of-interest, an MLN specifies a Markov network as follows. We ground each formula of the MLN with all possible combinations of constants. Each ground atom represents a binary random variable of the Markov network and each ground formula specifies a potential of the Markov network.

Formally, an MLN is a set of pairs (f_i, w_i) where f_i is a formula in first-order logic and w_i is a real number. Given a set of constants, the probability distribution represented by the MLN is given by the following equation.

$$\Pr(\omega) = \frac{1}{Z} \exp\left(\sum_i w_i N_{f_i}(\omega)\right)$$

where ω is a world, $N_{f_i}(\omega)$ is the number of groundings of f_i that evaluate to `True` given ω and Z is the normalization constant or the partition function of the MLN given by the following formula.

$$Z = \sum_{\omega} \exp\left(\sum_{i} w_i N_{f_i}(\omega)\right)$$

Figure 2.3 (a) shows an example MLN where the first formula encodes knowledge about the relationship between smoking and friendship while the second formula encodes the relationship between smoking and cancer. The example is shown for just two objects in the domain. As shown in Figure 2.3 (b), the Markov network underlying the MLN has 8 random variables (ground atoms) and 6 potentials (ground formulas). The joint distribution represented in (c) is defined over 2^8 unique worlds.

Normal MLN

Jha et.al (Jha et al., 2010) introduced a canonical form of MLNs called normal MLNs. Normal MLNs are related to other canonical forms in statistical relational learning such as the specification in Blog (Milch et al., 2008), parfactors (Poole, 2003) and substitution constraints (Gogate and Domingos, 2011b; Van den Broeck et al., 2011). Formally, the class of normal MLNs is defined as follows.

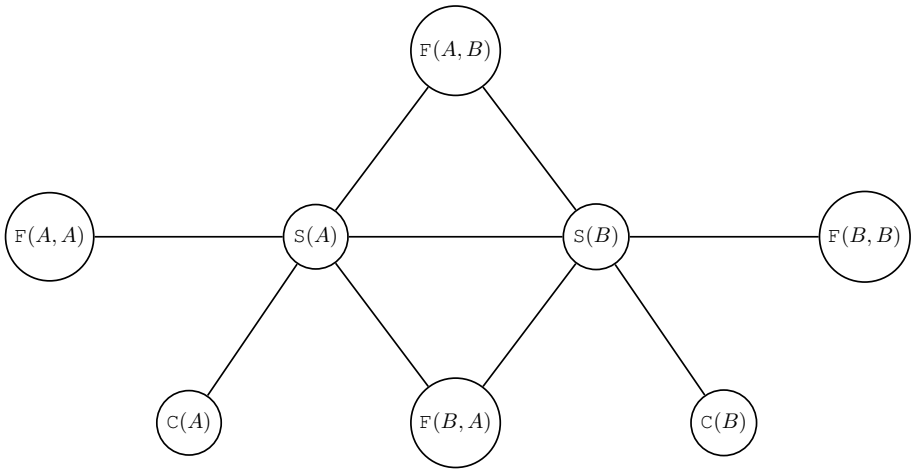
Definition 3. *A* normal *MLN is an MLN that satisfies the following two properties: (1) There are no constants in any formula, and (2) If two distinct atoms have the same predicate symbol, say* `R` *and if x and y are distinct logical variables that substitute the same argument position of* `R`*, then their domains are identical, i.e., $\Delta_x = \Delta_y$.*

Even though there are no constants in a normal form MLN (or normal MLN), throughout this dissertation, to simplify notation and make ground atoms and constants explicit, we slightly abuse the definition of normal forms with the following rule. Whenever we have a logical variable has the domain-size equal to 1, we denote it explicitly with its constant.

Example 1. *Consider an MLN having two formulas (*`Smokes`*$(x) \Rightarrow$ `Asthma`(x), w) and (*`Smokes`*(Ana), ∞) (evidence). Its normal form has three formulas:* `Smokes`*$(x') \Rightarrow$* `Asthma`*(x'),*

MLN : $\forall xy\ \neg\texttt{Smokes}(x) \vee \neg\texttt{Friends}(x,y) \vee \texttt{Smokes(y)}\,; w_1$
$\forall xy\ \neg\texttt{Smokes}(x) \vee \texttt{Cancer}(x); w_2$
Domain : $\Delta_x = \Delta_y = \{A, B\}$

(a) MLN



(b) Markov Network

S(A)	S(B)	F(A,A)	F(A,B)	F(B,A)	F(B,B)	C(A)	C(B)	$P(\omega)$
0	0	0	0	0	0	0	0	$\frac{1}{Z}\exp(4w_1 + 2w_2)$
...	...	...	...	...	...	...	...	...
1	1	0	0	0	0	0	1	$\frac{1}{Z}\exp(0w_1 + 1w_2)$
...	...	...	...	...	...	...	...	...
1	1	1	1	1	1	1	1	$\frac{1}{Z}\exp(4w_1 + 2w_2)$

(c) Distribution

Figure 2.3. (a) shows an example MLN. (b) shows the Markov network corresponding to the MLN. Each clique in the graph shown in (b) represents a potential or equivalently a ground formula. (c) shows the joint probability distribution represented by the MLN. Note that in (b) and (c), the predicate names are shortened to their first letter.

w ***Smokes1****(y)* $\Rightarrow$ ***Asthma1****(y), w and* ***Smokes1****(y),* ∞*, where* $\Delta_{x'} = \Delta_x \setminus \{Ana\}$ *and* $\Delta_y = \{Ana\}$*. In our simplified notation, we write the constants explicitly as* ***Smokes1****(Ana)* $\Rightarrow$ ***Asthma1****(Ana), w and* ***Smokes1****(Ana),* ∞*.*

2.2 Inference

The standard inference tasks in MLNs (and also Markov networks) are as follows.

(i) Computing the partition function, i.e.,

$$Z = \sum_{\omega} \exp\left(\sum_{i} w_i N_{f_i}(\omega)\right) \tag{2.3}$$

(ii) Computing probability of a query given evidence, i.e., $P(Q|E)$. In this dissertation, we consider 1-variable marginal probability computations. That is, Q is a single ground atom in an MLN (or equivalently a single random variable in the Markov network). For example, given a MLN ($\texttt{Smokes}(x) \Rightarrow \texttt{Asthma}(x)$, w), and evidence that *Ana* smokes, i.e., ($\texttt{Smokes}(Ana)$, ∞), we might be interested in knowing the posterior probability that *Ana* has Asthma, namely $P(\texttt{Asthma}(Ana)|\texttt{Smokes}(Ana))$.

(iii) Finding the most probable state of the world ω, i.e., finding a complete assignment to all ground atoms which has maximum probability in the joint distribution. This task is known as *Maximum a Posteriori* (MAP) inference and can be formally stated as follows,

$$\arg\max_{\omega} \frac{1}{Z} \exp\left(\sum_{i} w_i N(f_i, \omega)\right) \tag{2.4}$$

Clearly, inference in MLNs can simply be reduced to inference on PGMs. That is, we simply perform inference on the ground Markov network of the MLN. Therefore, all graphical model inference algorithms are directly applicable to MLNs. However, this does not utilize the relational structure of the MLN. Specifically, the distribution represented by MLNs is somewhat unique compared to standard distributions. Namely, a large number of variables have implicit/explicit dependencies between each other and several potentials are symmetric to each other. For example, consider a simple MLN with one formula $\texttt{Strong}(x) \Rightarrow \texttt{Wins}(x)$ w. Let Δ_x have 1 million constants. Clearly, the Markov network associated with the MLN has 1 million potentials. However, every potential is identical to each other and only depends upon the value of the parameter w. Generally, graphical model inference algorithms leverage statistical structure (e.g., conditional independencies), but are typically oblivious to

relational/logical structure. Therefore, traditional graphical model inference algorithms need to be adapted to take advantage of the unique properties in the MLN distribution. In this section, we first review some standard graphical model inference algorithms, specifically those based on sampling techniques and then outline some of the recent advances in specialized inference for relational models, typically referred to as *lifted inference*, which leverage both logical as well as statistical structure.

2.2.1 Exact Inference in Markov Networks

The inference tasks of computing the partition function as well as computing marginal probabilities are equivalent to each other. This is because, the marginal probability can be expressed as a ratio of partition functions. The MAP inference task is slightly different since it is an optimization problem. Here, we discuss standard techniques for the partition function and marginal probability computation tasks. For more details on MAP inference in graphical models, refer to (Koller and Friedman, 2009).

Computing the partition function is known to be a $\#P$-complete problem, where $\#P$ is the complexity class of counting problems. A general approach called variable elimination can be used to compute the partition function exactly. Several advances to the basic variable elimination algorithm have also been proposed in methods such as bucket elimination (Dechter, 1999) and junction trees (Lauritzen and Spiegelhalter, 1988). The basic idea in all these algorithms is to use dynamic programming to perform computations efficiently. Specifically, variable elimination consists of two basic operations, namely, product and sum-out. An example of both these operations is shown in Figure 2.4. To compute the partition function, variable elimination *eliminates* one variable from the Markov network at a time as follows. It creates a new potential which is a product of all the potentials that the variable occurs in. It then sums-out the variable from this new potential. In the example shown in Figure 2.4, suppose we want to eliminate X, we first take a product of all the potentials that

X	Y	ϕ_1
0	0	1.2
0	1	2.3
1	0	1.2
1	1	2

(a) ϕ_1

Y	Z	ϕ_2
0	0	1
0	1	4
1	0	2
1	1	1

(b) ϕ_2

X	Y	Z	ϕ
0	0	0	1×1.2
...	...	...	...
...	...	...	...
1	0	0	1.2×1
...	...	...	...

(c) Product

Y	Z	ϕ
0	0	$1.2 + 1.2$
...	...	...
...	...	...

(d) Sum-out

Figure 2.4. Example for variable elimination. (a), (b) are the original potentials from which we want to eliminate X. (c) is the product of the two potentials and (d) sums-out X from the product.

mention X, i.e., ϕ_1 and ϕ_2 and create a new potential as shown in (c) and finally sum-out X from this new potential as shown in (d).

The complexity of inference using variable elimination is determined by the order in which the variables are summed-out. Specifically, it can be shown that the inference complexity is exponential in the minimum width tree-decomposition (also called *treewidth*) of the primal graph of the Markov network. The treewidth of the primal graph of a Markov network is of particular interest because it allows us to analyze inference in Markov networks from a graph based perspective. Several exact inference algorithms (e.g., the junction tree algorithm (Lauritzen and Spiegelhalter, 1988), AND/OR graph search (Dechter and Mateescu, 2007), variable (bucket) elimination (Zhang and Poole, 1994; Dechter, 1999), etc.) are exponential in the *treewidth* of the primal graph. Thus, the primal graph can be used to quantify the complexity of these algorithms regardless of the underlying probability distribution. Unfortunately, computing the treewidth of a graph itself is a $\mathcal{NP}$-complete problem (Arnborg et al., 1987). Therefore, in practice, we often employ heuristic approaches such as the *min-*

fill heuristic and *min-degree* heuristic to find an upper-bound on the treewidth. Hereafter, whenever we refer to the treewidth of a graph, we implicitly assume that we have access to a close upper-bound to the treewidth.

In practice, exact inference very rarely scales-up to practical problems which more often than not have larger treewidths. Thus, approximate inference is the method-of-choice for inference in most practical problems. Dominant approximate inference approaches include variational methods such as belief propagation (Yedidia et al., 2000; Wainwright et al., 2003) and sampling based approaches (cf. (Liu, 2001)). Next we provide a brief review of popular sampling based inference techniques which we use extensively throughout this dissertation.

2.2.2 Sampling Based Approximate Inference

The main idea in all sampling algorithms is to estimate the expected value of a function using a sample average. Specifically, we draw samples from the distribution-of-interest and compute the expected value of a function w.r.t that distribution by averaging the function value computed from the samples. Sampling can be used for probabilistic inference as follows. We formulate the summation in the inference task as an expectation and estimate the expected value of the summation from the sample average. It is well-known from sampling theory that as the number of samples tend to ∞, the sample average approaches the true expected value. Further, as the number of samples increase, the variance of the estimates derived from the samples reduces. Thus, the aim of sampling based inference is to collect as many samples as possible from the target distribution. However, here, we are immediately faced with a problem, i.e., how can we collect samples from the distribution represented by the MLN/Markov network? That is, to compute the Gibbs distribution of a Markov network, we need the exact partition function. However, computing the partition function exactly is an infeasible problem due to which we are using approximate inference in the first place. Therefore, we require sampling methods that allow us to sample from a Markov network even

though we do not know its complete true distribution, i.e., we only know its distribution up to the normalization constant. Two of the most popular approaches for sampling from such hard-to-compute distributions are Markov Chain Monte Carlo (MCMC) (Metropolis et al., 1953) based sampling and Importance Sampling (Geweke, 1989). We provide a brief overview of both these methods next.

MCMC methods

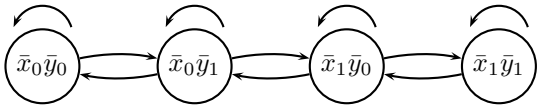
Markov chains are stochastic processes that transition from one state to the next state in a given state space. The key property of Markov chains is that they are *memoryless*, i.e., the transition to a state only depends upon the previous state. Formally, let $\mathbf{X}$ be the set of variables in a state-space. The transition probability is given by,

$$P(\mathbf{X}^{(i)}|\mathbf{X}^{(i-1)}, \ldots, \mathbf{X}^{(0)}) = P(\mathbf{X}^{(i)}|\mathbf{X}^{(i-1)}) \tag{2.5}$$

where $\mathbf{X}^{(0)}$ is the starting state and $\mathbf{X}^{(i)}$ is the state after exactly i transitions.

For finite state spaces, the probabilities of moving from one state to the next is described as a matrix called the *transition matrix* represented by $\mathbf{T}$. Clearly, if the state space has n dimensions (variables) where each dimension can take m distinct values (states), the transition matrix T is a square matrix of size $n^m \times n^m$. P_{ij} refers to the entry in the transition matrix which gives the probability of moving from state i to state j. A simple Markov chain with 2 dimensions and 4 possible states is shown in Figure 2.5. As shown, the transition matrix is a 4×4 matrix and each row of the matrix is a distribution, i.e., it sums to 1.

The Markov chain defines a joint probability distribution on the set of possible states. Clearly, this distribution depends upon the transition probabilities. However, the interesting property of some specific Markov chains is that irrespective of the starting state, after a certain number of transitions, the Markov chain always converges to the exact same distribution on the states. This distribution is called the *stationary distribution* represented by π.



(a) Markov Chain

$$\begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0.35 & 0.45 & 0.2 & 0 \\ 0 & 0.1 & 0.15 & 0.75 \\ 0 & 0 & 0.5 & 0.5 \end{bmatrix}$$

(b) Transition Matrix

Figure 2.5. Example of a 2-dimensional Markov chain, both the dimensions, X and Y are binary. The 4 possible states in the state-space are labeled in (a) and its corresponding 4×4 transition matrix is shown in (b).

Once the Markov chain reaches its stationary distribution, applying the transition matrix leaves this distribution invariant. Formally,

$$\pi \mathbf{T} = \pi \tag{2.6}$$

Stationary distributions of a Markov chain has been well-studied to describe numerous physical and computational processes (e.g., thermodynamics, queuing theory, chemical processes, etc.). However, typically, in these systems, the task is to compute the stationary distribution given transition probabilities. For the case that we are interested in, i.e., in Markov Chain Monte Carlo (MCMC) methods, this task is flipped. That is, we are interested in computing the transition probabilities that yield a specific stationary distribution. Specifically, if we can construct a Markov chain that has a unique stationary distribution that is exactly the same as the distribution represented by the Markov network, we are guaranteed that after certain number of transitions, the Markov chain will reach this stationary distribution. Thus, to sample from the Markov network's distribution, we can equivalently sample from the Markov chain once it reaches its stationary distribution. The key challenge though, is to design the transition matrix such that the stationary distribution is guaranteed to be equal to the distribution represented by the Markov network. Next, we provide conditions under which a Markov chain has a unique stationary distribution.

Definition 4. *[Reversibility] A Markov chain is said to be reversible w.r.t distribution π if it satisfies the detailed balance condition, i.e., $\pi P_{ij} = \pi P_{ji}$ and consequently the stationary distribution of this Markov chain is π.*

Note that for a reversible Markov chain, a stationary distribution exists. However, this condition alone is not sufficient to generate samples from the Markov chain. Specifically, we require guarantees that the Markov chain will converge to this distribution. This is given by the following definitions and the accompanying theorem.

Definition 5. *[Irreducibility] A Markov chain is said to be irreducible if any state i can be reached from any other state j in the state space.*

Definition 6. *[Aperiodicity] An irreducible Markov chain is aperiodic if for any state i, 1 is the GCD of all integers $l \geq 1$ such that $P_{ii}^{(l)} > 0$. In other words, we should be able to come back to the start state in arbitrary time steps.*

Theorem 1. *A finite state space Markov chain that is irreducible and aperiodic has a unique stationary distribution and converges to its unique stationary distribution for all starting states.*

Theorem 1 guarantees that there exists exactly one stationary distribution for the Markov chain if the irreducibility and aperiodicity properties are satisfied. Further, it also guarantees that we will reach this stationary point regardless of the starting state. This condition allows us to sample from the Markov chain starting from any random point in the state space. Markov chains that satisfy Theorem 1 are often referred to as *ergodic* Markov chains.

Thus, from Theorem 1, it is clear that for MCMC based inference, we want to design transition functions that yield ergodic Markov chains. Then, we just simulate the Markov chain until it reaches its stationary distribution at which time the chain is said to have *mixed*. Once the chain has mixed, sampling from the chain guarantees that we are sampling from

the target distribution. However, note that samples generated from the Markov chain are *dependent* samples and not strictly independent and identically distributed (i.i.d) samples. To reduce the dependencies between samples, typically, only every k-th sample from the chain is considered during estimation. Another issue with MCMC is that analyzing mixing time is an extremely hard problem. Therefore, in practice, a fixed number of iterations called the *burn-in* period is used after which the Markov chain is assumed to have mixed and samples that are collected after this are assumed to be from the target distribution.

Next, we briefly discuss some popular MCMC based algorithms. The earliest method that introduced MCMC was the Metropolis algorithm (Metropolis et al., 1953). The Metropolis algorithm used a symmetric transition function. Specifically, the transition function is defined using a symmetric proposal distribution. In each step, the Metropolis sampler draws a new state from the proposal distribution and jumps to this new state based on an *acceptance probability*. The acceptance probability is computed by observing the gain obtained from moving to the new state. Specifically, the gain for the move from state $\mathbf{X}_{i+1}$ from $\mathbf{X}_i$ is the ratio $\frac{P(\mathbf{X}_{i+1})}{P(\mathbf{X}_i)}$. Note that the ratio need not be computed exactly, i.e., we only need to know it up to a normalization constant. Therefore, $P(\mathbf{X}_i)$ is the un-normalized probability value for the assignment $\mathbf{X}_i$ in the Markov network's distribution. Several variants of the basic Metropolis algorithm have been developed. For example, the Metropolis-Hastings algorithm (Hastings, 1970) extends the original Metropolis method by allowing a wider range of proposal distributions that are non-symmetric. Gibbs sampling (Geman and Geman, 1984) which is a special case of the Metropolis-Hastings sampler is arguably the most popular MCMC method in practical applications. We next review Gibbs sampling.

Gibbs Sampling

Gibbs sampling is one of the most widely used MCMC algorithms to date. Here, when sampling from a n-dimensional state space, we sample exactly one dimension at a time from the conditional distribution of that dimension as described below.

Given a PGM $\mathcal{M} = \langle \mathbf{X}, \Phi \rangle$, Gibbs sampling begins by initializing all variables randomly, denoted by $\overline{\mathbf{x}}^{(0)}$. Then, at each iteration j, it randomly chooses a variable $X_i \in \mathbf{X}$ and samples a value $\overline{x}_i$ for it from the conditional distribution $P(X_i|\overline{\mathbf{x}}_{-i}^{(j-1)})$, where $\overline{\mathbf{x}}_{-i}^{(j-1)}$ denotes the projection of $\overline{\mathbf{x}}^{(j-1)}$ on all variables in the PGM other than X_i. The new sample is $\overline{\mathbf{x}}^{(j)} = (\overline{x}_i, \overline{\mathbf{x}}_{-i}^{(j-1)})$. The computation of the conditional distribution can be simplified by observing that in a PGM, a variable X_i is conditionally independent of all other variables given its neighbors (or its *Markov blanket*) denoted by $MB(X_i)$. Formally, $P(X_i|\overline{\mathbf{x}}_{-i}) = P(X_i|\overline{\mathbf{x}}_{MB(X_i)})$. Thus, generating each sample is efficient in Gibbs sampling and therefore it is the arguably the method-of-choice for MCMC based inference in PGMs.

The Gibbs sampling procedure just described is called random-scan Gibbs sampling in literature. Another variation is systematic-scan Gibbs sampling in which we draw samples along a particular ordering of variables. It is known that random-scan Gibbs sampling is statistically more efficient than systematic-scan Gibbs sampling (cf. (Liu, 2001)).

Gibbs sampling is usually used to estimate the marginal probabilities. However, it can also be used to compute an estimate for the partition function as well. This is slightly more tricky and discussed in detail in the next chapter (Section 3.1.2). After T samples are generated, the 1-variable marginal probabilities can be estimated using the following equation.

$$\widehat{P}_T(\overline{x}_i) = \frac{1}{T} \sum_{t=1}^{T} P(\overline{x}_i|\overline{\mathbf{x}}_{-i}^{(t)}) \tag{2.7}$$

The estimator presented in Eq. (2.7) to compute the 1-variable marginal probabilities is commonly referred to as the *mixture estimator*. We provide more details on the different types of estimators used in Gibbs sampling and their implications later in this section.

As $T \to \infty$, $\widehat{P}_T(\overline{x}_i)$ will converge to the $P(\overline{x}_i)$ if the underlying Markov chain is ergodic. Recall that for ergodicity of the chain, every full assignment (state) is reachable from every other full assignment (state) with probability greater than zero. Thus, if the PGM contains

no deterministic functions, then the Markov chain is guaranteed to be ergodic. Unfortunately, when the PGM encodes hard constraints, the joint probability distribution has 0 probabilities. In such a case, Gibbs sampling is no longer guaranteed to be ergodic and the estimate given in Eq. (2.7) may not converge to $P(\overline{x}_i)$. In chapter 3, we develop a new algorithm to handle such cases.

Next, we discuss two advanced variants of Gibbs sampling, namely, blocking and collapsing.

Blocking

Blocked/Blocking Gibbs sampling (Jensen et al., 1993) is variant of Gibbs sampling where instead of sampling one variable at a time as in the standard Gibbs sampler, variables are sampled jointly in blocks given assignments to other variables in the PGM. Formally, let the variables of the PGM be partitioned into disjoint groups or *blocks*, denoted by $\mathbb{B} = \{\mathbf{B}_i\}_{i=1}^{k}$, where $\mathbf{B}_i \subseteq \mathbf{X}$ and $\cup_i \mathbf{B}_i = \mathbf{X}$. Then, starting with a random assignment $\overline{\mathbf{x}}^{(0)}$ to all variables in the PGM, in each iteration j of blocked Gibbs sampling, we create a new sample $\overline{\mathbf{x}}^{(j)}$ by replacing the assignment to all variables in a randomly selected block $\mathbf{B}_i$ in $\overline{\mathbf{x}}^{(j-1)}$ by a new assignment that is sampled (jointly) from the distribution, $P(\mathbf{B}_i|\overline{\mathbf{x}}^{(j-1)}_{\mathbf{X}\setminus\mathbf{B}_i})$, where $\overline{\mathbf{x}}^{(j-1)}_{\mathbf{X}\setminus\mathbf{B}_i}$ is the projection of $\overline{\mathbf{x}}^{(j-1)}$ on all variables not in $\mathbf{B}_i$. We define the Markov blanket (MB) of a block $\mathbf{B}_i$ as all other blocks that contain at least one variable in $MB(X_i)$, where $X_i \in \mathbf{B}_i$. Similar to Gibbs sampling, an assignment to all variables in $MB(\mathbf{B}_i)$ makes $\mathbf{B}_i$ conditionally independent of all other variables. Note that blocked Gibbs sampling is feasible only when every block $\mathbf{B}_i$ is tractable given an assignment to $MB(\mathbf{B}_i)$. These tractability constraints are often imposed in practice by putting a limit on the treewidth of the primal graph projected on the block.

Collapsing

Collapsing is an alternative technique that improves the accuracy and convergence of Gibbs sampling. Collapsing operates by eliminating or marginalizing out a subset of variables

from the Markov network. That is, given a Markov network $\mathcal{M}\ \langle \mathbf{X}, \mathbf{\Phi} \rangle$, we choose $\mathbf{C} \subseteq \mathbf{X}$ and sum-out all variables in $\mathbf{C}$ from the joint distribution of $\mathcal{M}$ to obtain a collapsed Markov network $\mathcal{M}'\ \langle \mathbf{X} \backslash \mathbf{C}, \mathbf{\Phi}' \rangle$. Gibbs sampling is then performed on this collapsed Markov network and this improves its accuracy and convergence because only a sub-space of the original state space is sampled (cf. (Liu, 2001)). However, collapsing implicitly changes the structure of the Markov network as follows. Whenever we collapse a variable, X_i, implicitly, we form a clique connecting all the neighbors of X_i in the primal graph of the Markov network. Therefore, for collapsing to be feasible in practice, it should be tractable to marginalize/eliminate the collapsed variables, i.e., the size of the maximum clique formed when the variables are collapsed should be bounded.

Estimators

Given N Gibbs samples $\{\overline{\mathbf{x}}^{(i)}\}_{i=1}^{N}$ from the distribution P, we can estimate the 1-variable marginal probabilities using one of the following three different estimators.

1. **Histogram estimator:**
$$\widehat{P}(\overline{x}_i) = \frac{1}{N} \sum_{j=1}^{N} \mathbb{I}_{\overline{x}_i}(\overline{\mathbf{x}}^{(j)})$$
where $\mathbb{I}_{\overline{x}_i}(\overline{\mathbf{x}}^{(j)})$ is an indicator function which equals 1 if $\overline{x}_i$ appears in $\overline{\mathbf{x}}^{(j)}$ and 0 otherwise.

2. **Mixture Estimator:**
$$\widehat{P}(\overline{x}_i) = \frac{1}{N} \sum_{j=1}^{N} P(\overline{x}_i | \overline{\mathbf{x}}^{(j)}_{MB(X_i)}) \tag{2.8}$$

3. **Rao-Blackwell Estimator:** This estimator is a more advanced estimator that generalizes the mixture estimator and is given by
$$\widehat{P}(\overline{x}_i) = \frac{1}{N} \sum_{j=1}^{N} P(\overline{x}_i | \overline{\mathbf{x}}^{(j)}_{\mathbf{R}}) \tag{2.9}$$
where $\mathbf{R} \subseteq \mathbf{X}$.

The above three estimators are all unbiased. However, it has been shown that the Rao-Blackwell estimator has smaller variance than the mixture estimator which in turn has smaller variance than the histogram estimator (Liu, 2001) and thus the Rao-Blackwell and the mixture estimators should always be preferred. However, the Rao-Blackwell estimator requires more computation since we are essentially "ignoring" the samples on certain variables (non-sampled variables). The non-sampled variables, $\mathbf{X} \setminus \mathbf{R}$ should now be marginalized out to obtain the estimate $\widehat{P}(\overline{x}_i)$. Therefore, as the set of non-sampled variables grows larger, estimation becomes more accurate but also computationally more expensive.

Note that all the three estimators can be used with blocked as well as collapsed Gibbs sampling. To use the Rao-Blackwell estimator with blocked Gibbs sampling, we simply find the block, say $\mathbf{B}$, in which the variable resides, set $\mathbf{R}$ equal to $\mathbf{X} \setminus \mathbf{B}$ and compute $P(\overline{x}_i|\overline{\mathbf{x}}_{\mathbf{R}}^{(j)})$ by marginalizing out all variables other than X_i in the block. These computations are tractable if the block is assumed to be tractable. In collapsed Gibbs sampling, we can use the Rao-Blackwell estimator to estimate the marginals over all the collapsed variables which is again guaranteed to be tractable by imposing tractability constraints on the collapsed variables.

Importance Sampling

Importance Sampling (Geweke, 1989) is a sampling strategy that is a popular alternative to MCMC based sampling. Just like MCMC based inference, importance sampling can be used to approximately compute the partition function as well as marginal probabilities in Markov networks. The main idea in importance sampling is to sample from a probability distribution Q, called the proposal distribution instead of the target distribution P. Q is carefully chosen such that it is easy to sample from Q even though sampling from P is hard. Unlike Gibbs sampling, using importance sampling, we obtain independent samples for estimation. However, to account for fact that we sample from the wrong distribution,

each sample is weighted with its *importance weight* which is the ratio of the probability of the sample in the true distribution to its probability in the proposal. Just as before, we can reformulate the inference task as computing the expected value of a function and using weighted samples drawn from Q, we approximate the expected value by the sample mean (average over the samples).

Formally, we can express the partition function of a Markov network $\mathcal{M}\langle \mathbf{X}, \Phi \rangle$ using the following equation.

$$Z = \sum_{\overline{x}} \frac{\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}) Q(\overline{\mathbf{x}})}{Q(\overline{\mathbf{x}})} = \mathbb{E}_Q \left[\frac{\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}})}{Q(\overline{\mathbf{x}})} \right] \tag{2.10}$$

where the notation $\mathbb{E}_Q[x]$ denotes the expected value of x w.r.t. Q. Q should be such that it is easy to generate samples from it. It should also satisfy the constraint: $\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}) > 0 \Rightarrow Q(\overline{\mathbf{x}}) > 0$. Given T samples $(\overline{\mathbf{x}}^{(1)}, \ldots, \overline{\mathbf{x}}^{(T)})$ drawn from Q, we can estimate Z using the following sample mean:

$$\widehat{Z} = \frac{1}{T} \sum_{t=1}^{T} \left[\frac{\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}^{(t)})}{Q(\overline{\mathbf{x}}^{(t)})} \right] = \frac{1}{T} \sum_{t=1}^{T} w(\overline{\mathbf{x}}^{(t)}) \tag{2.11}$$

where $w(\overline{\mathbf{x}}^{(t)})$ is called the importance weight (or simply weight) of $\overline{\mathbf{x}}^{(t)}$. It is well known that the quality (accuracy) of $\widehat{Z}$ is highly dependent on how close the proposal distribution Q is to P. Note that we cannot use P as the proposal distribution because it is hard to generate samples from it.

Typically, Q is expressed in product form or as a Bayesian network so that it is easy to generate samples from it. Formally, given an ordering of variables $(X_1, \ldots, X_n)$, the proposal distribution is expressed using a collection of n conditional probability tables (CPTs) of the form $Q_i(X_i|\mathbf{Y}_i)$ where $\mathbf{Y}_i \subseteq \{X_1, \ldots, X_{i-1}\}$ (i.e., $Q(\overline{\mathbf{x}}) = \prod_{i=1}^{n} Q_i(\overline{x}_i|\overline{\mathbf{y}}_i)$). To ensure polynomial complexity, we ensure that $|\mathbf{Y}_i|$ is bounded by a constant. We can generate samples from this Bayesian network using logic sampling (Pearl, 1988), namely, by sampling variables one by one along the order $(X_1, \ldots, X_n)$.

Similar to approximating the partition function, we can approximate the 1-variable marginal probabilities using importance sampling as follows.

$$\widehat{P}_T(\overline{x}) = \frac{\sum_{t=1}^{T} \mathbb{I}_{\overline{x}}(\overline{\mathbf{x}}^{(t)}) w(\overline{\mathbf{x}}^{(t)})}{\sum_{t=1}^{T} w(\overline{\mathbf{x}}^{(t)})} \tag{2.12}$$

where $\mathbb{I}_{\overline{x}}(\overline{\mathbf{x}})$ is equal to 1 iff $\overline{\mathbf{x}}$ contains the assignment $\overline{x}$ and 0 otherwise.

We can also perform *Rao-Blackwellisation* in importance sampling to reduce the variance of estimates derived from the samples. For this, we partition the set of variables $\mathbf{X}$ in the distribution into two sets, say $\mathbf{X}_e$ and $\mathbf{X}_s$. When computing the sample estimate, we use the sampled assignment on $\mathbf{X}_s$ and conditioned on the sampled assignment, we perform exact computations on $\mathbf{X}_E$. Specifically, after Rao-Blackwellisation, the importance sampler estimate for 1-variable marginals is equal to

$$\widehat{P}_T(\overline{x}) = \frac{\sum_{t=1}^{T} P(\overline{x}|\overline{\mathbf{x}}_s^{(t)}) w(\overline{\mathbf{x}}^{(t)})}{\sum_{t=1}^{T} w(\overline{\mathbf{x}}^{(t)})} \tag{2.13}$$

In practice, for Rao-Blackwellisation, $\mathbf{X}_s \subseteq \mathbf{X}$ should be carefully chosen such that $P(\overline{x}|\overline{\mathbf{x}}_s^{(t)})$ in Eq. (2.13) should be tractable to compute. For more details of Rao-Blackwellisation in importance sampling, please refer to (Liu, 2001).

Both $\widehat{Z}$ and $\widehat{P}_T(\overline{x}_i)$ are **consistent estimates**, namely, as the number of samples get large, they converge to the correct answer with high probability. However, $\widehat{Z}$ is an unbiased estimate of Z, namely $\mathbb{E}_Q[\widehat{Z}] = Z$, while $\widehat{P}_T(\overline{x})$ is an asymptotically unbiased estimate of $P(\overline{x})$, namely $\lim_{T\to\infty} \mathbb{E}_Q[\widehat{P}_T(\overline{x})] = P(\overline{x})$. $\widehat{P}_T(\overline{x})$ is also called the normalized estimate of $P(\overline{x})$ because *we only need to know each sample weight up to a normalizing constant.*

2.2.3 Lifted Inference

At a high level, lifted inference in MLNs can be viewed as the probabilistic equivalent of theorem proving in first-order logic. That is, just as theorem proving in first-order logic does not

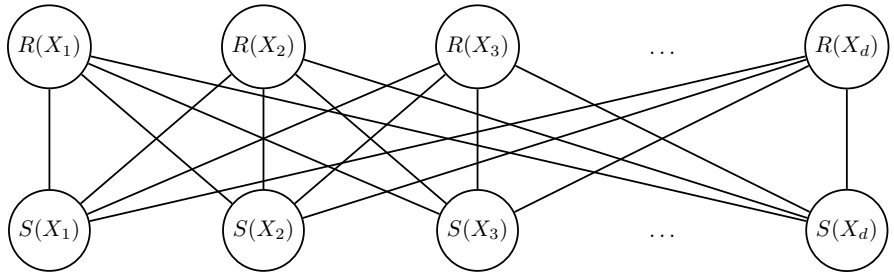
ground formulas in a knowledge base but instead reasons on the first-order representation, lifted inference in statistical relational models aims to perform probabilistic reasoning at the level of first-order formulas instead of the ground Markov network. Over the last few years, several lifted inference algorithms have been proposed starting with the pioneering work by Poole (Poole, 2003). Popular exact inference methods include, FOVE (de Salvo Braz, 2007), WFOMC (Van den Broeck et al., 2011), Probabilistic Theorem Proving (PTP) (Gogate and Domingos, 2011b) and lifted inference with soft evidence (Bui et al., 2012). Popular approximate lifted inference methods include (Milch and Russell, 2006; Milch et al., 2008; Singla and Domingos, 2008; Kersting et al., 2009; Gogate et al., 2012; Niepert, 2012; Venugopal and Gogate, 2012; Bui et al., 2013; Ahmadi et al., 2013). Almost all lifted inference algorithms, whether exact or approximate have the same basic idea, namely, they exploit symmetries in the relational representation and perform efficient inference over groups of symmetrical ground atoms rather than treating each ground atom as a separate entity. The key challenge in lifted inference is to identify these symmetrical groups efficiently from the relational structure without explicitly constructing the ground representation.

We illustrate the power of leveraging symmetries in lifted inference through a simple example. Consider a MLN with one formula, $\forall x \forall y \neg \mathtt{R}(x) \vee \mathtt{S}(y);\ w$. Let $|\Delta_x| = |\Delta_y| = d$. Figure 2.6 (a) shows the primal graph of the ground Markov network corresponding to the MLN. As seen in the figure, this is a complete bipartite graph. The tree-width of the graph is clearly d, therefore, if we run variable elimination or any other exact inference algorithm to compute the partition function, it will have time/space complexity exponential in $d + 1$. Figure 2.6 (b) shows how Probabilistic Theorem Proving (PTP) (Gogate and Domingos, 2011b), a well-known lifted inference algorithm, computes the exact same partition function much more efficiently by leveraging symmetries in the MLN representation. First, we observe the following symmetry. Consider two distinct assignments to all groundings of $\mathtt{R}$ denoted by $\overline{\mathtt{R}}^i$ and $\overline{\mathtt{R}}^j$. If in both $\overline{\mathtt{R}}^i$ and $\overline{\mathtt{R}}^j$, the same number groundings of $\mathtt{R}$ have an assignment

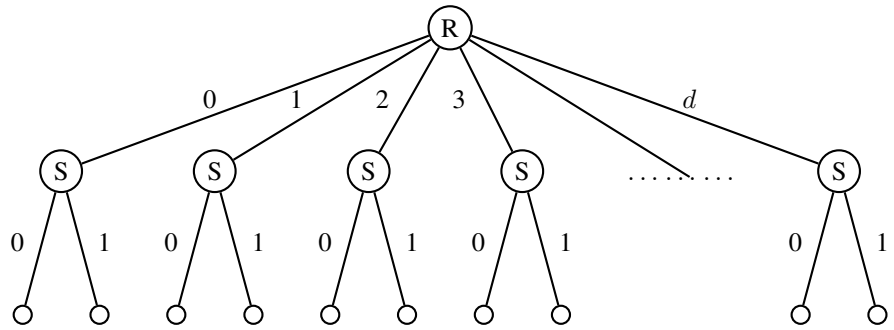
true (or 1), then it turns out that conditioning the MLN on $\overline{\mathtt{R}}^i$ is equivalent to conditioning the MLN on $\overline{\mathtt{R}}^j$. Therefore, instead of conditioning over 2^d possible assignments for $\mathtt{R}$, we need to condition over just $d+1$ groups of assignments. From each group, we pick a single assignment and then project the same results to all the other assignments in that group. The second symmetry that we can leverage is as follows. After conditioning on any of the aforementioned d+1 distinct assignments, the conditional distribution over all the groundings of $\mathtt{S}$ can be decomposed into d independent and identical distributions. Thus, we need to only evaluate one of these distributions and project the same results over all equivalent distributions. Figure 2.6 (b) shows these operations schematically. Computing the partition function is equivalent to generating the full tree shown in Figure 2.6 (b) and has time/space complexity that is linear in $d+1$, an exponential reduction in complexity when compared to propositional inference.

To summarize the above example, by utilizing symmetries, treewidth is not a limiting complexity bound for lifted inference, i.e., lifted inference can in many cases perform tractable inference on high-treewidth Markov networks by leveraging symmetrical computations and is thus more general and scalable as compared to propositional inference.

To formalize the notion of lifted inference, (Van den Broeck, 2011; Jaeger, 2015) introduced the concept of *domain liftability*. Specifically, an MLN is domain liftable if exact inference on the MLN has time and space complexity that is polynomial in the size of the domain. The above example MLN ($\mathtt{R}(x) \vee \mathtt{S}(y)$) is domain liftable. However, there are several MLN structures which are not known to be domain liftable. For instance, the transitive relation $\mathtt{Friends}(x, y) \wedge \mathtt{Friends}(y, z) \Rightarrow \mathtt{Friends}(z, x)$ is not domain liftable as per our current knowledge. Currently, our understanding is that the structure of the MLN can be used to determine if that MLN is domain liftable or not. Finding new lifted inference rules based on the structure of the MLN to push the limits on domain liftability is an area of active research (Jha et al., 2010; Gogate and Domingos, 2011b; Gogate et al., 2012; Van den Broeck, 2011; Jaeger, 2015; Taghipour et al., 2013).



(a) Ground Markov Network



(b) Illustrating inference using PTP

Figure 2.6. Illustrating the power of lifted inference on high treewidth models. (a) is the primal graph of the ground Markov network corresponding to $\forall x \forall y \neg \mathtt{R}(x) \vee \mathtt{S}(y)$; w with each variable having a domain-size equal to d. Computing the partition function for this Markov network is exponential in d. (b) is a schematic illustration of how PTP would compute the partition function using lifted inference. The complexity of lifted inference is linear in d.

Niepert and Broeck (Niepert and Van den Broeck, 2014) recently contributed to a deeper understanding of lifted inference by connecting inference tractability with the concept of *finite partial exchangeability* in statistics (Diaconis and Freedman, 1980). Specifically, distributions with partially exchangeable random variables can be expressed more succinctly using sufficient statistics. It turns out that most lifted inference algorithms implicitly use finite exchangeability to perform domain lifted inference. In the next section, we provide a brief background on one such lifted inference algorithm, PTP (Gogate and Domingos, 2011b), that we use in this dissertation. For a detailed survey of lifted inference techniques please refer to (Kimmig et al., 2014). For an in-depth formal treatment of lifted inference theory and complexity results, refer to (Van den Broeck, 2013).

Probabilistic Theorem Proving (PTP)

PTP is a popular lifted inference algorithm that computes the partition function of an MLN exactly. It defines two *lifting rules*, i.e., rules that identify symmetries from first-order structure and applies these rules recursively on the input MLN. We briefly describe the two rules below.

Power Rule: The power rule identifies identical and independent components in the underlying Markov network. It is based on the concept of a decomposer as defined next.

We first introduce some notation in order to formally define a decomposer. Let $i_{\mathtt{R}}$ denote the i-th argument of predicate $\mathtt{R}$. Two arguments $i_{\mathtt{R}}$ and $j_{\mathtt{S}}$ are said to be unifiable if they share a logical variable in some MLN formula. Clearly, the binary relation for unification $U(i_{\mathtt{R}}, j_{\mathtt{S}})$ is symmetric and reflexive. Let $\mathcal{U}$ denote the transitive closure of U. Given an argument $i_{\mathtt{S}}$, let $\mathcal{U}(i_{\mathtt{S}})$ denote its equivalence class under $\mathcal{U}$.

Definition 7 (Decomposer)**.** *Given a normal MLN $\mathcal{M}$ having m formulas denoted by $f_1, \ldots, f_m$, $\mathbf{d} = \mathcal{U}(i_{\mathtt{R}})$ where $\mathtt{R}$ is a predicate in $\mathcal{M}$, is called a decomposer iff the following conditions are satisfied: (i) there is exactly one argument of $\mathtt{R}$, $i_{\mathtt{R}}$ that is an element of $\mathbf{d}$; and (ii) in each formula f_i, there exists a variable x such that x appears in all atoms of f_i and for a predicate $\mathtt{S}$ in f_i, x substitutes the argument $i_{\mathtt{S}}$, where $i_{\mathtt{S}} \in \mathbf{d}$.*

If an MLN $\mathcal{M}$ has a decomposer $\mathbf{d}$, then we can generate a new MLN $\mathcal{M}|\mathbf{d}$ where we replace all the logical variables of $\mathcal{M}$ that substitute the arguments in $\mathbf{d}$ with a constant. The partition function of $\mathcal{M}$ is given by the following recursive formula.

$$Z(\mathcal{M}) = \left(Z(\mathcal{M}|\mathbf{d})\right)^{D(\mathbf{d})} \tag{2.14}$$

where $Z(\mathcal{M}|\mathbf{d})$ denotes that all elements of $\mathbf{d}$ have been replaced by the same constant and $D(d)$ is the "size" of the decomposer, i.e., the domain-size of the variables that can substitute a decomposer argument.

Example 2. *Consider the MLN $\mathcal{M}$ with one formula,* $\texttt{Strong}(x) \Rightarrow \texttt{Wins}(x, y)$*, w with $\Delta_x = \Delta_y = \{A, B, C\}$. Here, the decomposer $\mathbf{d} = \{1_{\texttt{Strong}}, 1_{\texttt{Wins}}\}$. Applying the power rule, we replace $\mathbf{d}$ with a logical variable whose domain has exactly one constant. The new MLN denoted by $\mathcal{M}|\mathbf{d}$ has the formula* $\texttt{Strong}(A) \Rightarrow \texttt{Wins}(A, y)$*, w. $D(\mathbf{d}) = 3$ since $|\Delta_x| = 3$, therefore, $Z(\mathcal{M}) = (Z(\mathcal{M}|\mathbf{d}))^3$.*

Generalized Binomial Rule: The generalized binomial rule identifies the cases where it is possible to efficiently condition on all possible assignments to all possible groundings an atom. In general, this conditioning is exponential in the number of groundings. Specifically, given $\texttt{R}(x)$, the total number of possible assignments is equal to $2^{|\Delta_x|}$. However, in some cases, it is possible to group the set of assignments such that all the assignments within a group are equivalent to each other. That is, we can condition on any assignment within a group and project the same results to all assignments of that group while guaranteeing the correctness of the partition function computation. The generalized binomial rule provides a sufficient condition to identify these groups from first-order structure as follows.

Definition 8. *Given a normal MLN $\mathcal{M}$ and a singleton $\texttt{S}(x)$ that does not participate in self-joins, the possible assignments to the ground atoms of $\texttt{S}$ can be partitioned into $|\Delta_x| + 1$ groups, where the i-th group is the set of all the assignments with exactly i ground atoms of $\texttt{S}$ assigned to 1. All assignments in a group are symmetrical to each other, i.e., for any two assignments in the same group, $\overline{S}^i$ and $\overline{S}^j$, $\mathcal{M}|\overline{S}^i \equiv \mathcal{M}|\overline{S}^j$*

The partition function for $\mathcal{M}$, when conditioned on a singleton $\texttt{S}(x)$ is computed using

$$Z(\mathcal{M}) = \sum_{i=0}^{|\Delta_x|} \binom{|\Delta_x|}{i} Z(\mathcal{M}|\bar{\texttt{S}}^i) w(i) 2^{p(i)} \tag{2.15}$$

where $\mathcal{M}|\bar{\texttt{S}}^i$ denotes the reduced MLN obtained after conditioning on $\texttt{S}$ by assigning any i ground atoms corresponding to $\texttt{S}$ as `True` and $d - i$ ground atoms as `False`, $p(i)$ is the number of ground atoms (apart from the ones corresponding to the conditioned predicate)

that are completely removed as a result of reducing $\mathcal{M}$ and $w(i)$ is the total weight of the formulas that are removed since they are satisfied by the assignment.

Example 3. *Consider an MLN $\mathcal{M}$ with a single formula, $\texttt{Smokes}(x) \Rightarrow \texttt{Cancer}(y)$, w with $\Delta_x = \Delta_y = \{A, B, C\}$. Conditioning on $\texttt{Smokes}(x)$ implies that we need to condition on all possible groundings of $\texttt{Smokes}(x)$, i.e., 2^3 different possible assignments. We can group these 8 assignments into 4 groups containing 1, 3, 3 and 1 assignments respectively. Only one assignment from each group needs to be considered. An assignment where no groundings of $\texttt{Smokes}$ is set to 1, an assignment where 1 grounding is set to 1, an assignment where 2 groundings are set to 1 and an assignment where all three groundings of $\texttt{Smokes}$ are set to 1. Let $\mathcal{M}_0$, $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ be the MLNs after conditioning on each of the 4 assignments. The overall partition function of $\mathcal{M}$ is given by,*

$$Z(\mathcal{M}) = \binom{3}{0} Z(\mathcal{M}_0) 9w + \binom{3}{1} Z(\mathcal{M}_1) 6w + \binom{3}{2} Z(\mathcal{M}_2) 3w + \binom{3}{3} Z(\mathcal{M}_3)$$

Algorithm 1: PTP

Input: Normal MLN, $\mathcal{M}$
Output: Exact Partition function, $Z(\mathcal{M})$

```
if M is empty then
    return 1
if M contains a decomposer d then
    return Z(M|d)^{D(d)}
if there exists a singleton S(x) then
    return Σ_{i=0}^{|Δs|} C(|Δs|, i) Z(M|S̄^i) w(i) 2^{p(i)}
else
    Partially ground M until a singleton can be selected for conditioning
```

Algorithm 1 illustrates the recursive PTP algorithm. The input is an MLN in normal form. The base case checks for an empty MLN, i.e., if all formulas have either been removed (since they are satisfied) or each formula is empty (all atoms in the formulas have been conditioned), then we return 1. For all other cases, PTP first tries to apply the power rule

to the input MLN recursively. If no decomposer can be found, then a singleton predicate is chosen heuristically (cf. (Gogate and Domingos, 2011b)). Once a singleton is selected, the MLN is recursively conditioned using Eq. 2.15. If no singletons can be found for conditioning, then PTP partially grounds a predicate until a singleton is found. In such a case, PTP is no longer considered tractable or in other words, the MLN is not domain-liftable.

Apart from the two lifting rules shown in Algorithm 1, PTP leverages standard SAT techniques such as unit propagation and caching to greatly improve the performance of lifted inference in practice. For more details on these extensions, refer to (Gogate and Domingos, 2011b).

2.3 Learning

The two main types of learning in MLNs are, *structure learning* and *weight learning*. In structure learning, we learn the MLN formulas as well as the weights while in weight learning, we assume that the formulas are given and learn the weights for each formula. Notable structure learning algorithms for MLNs are described in (Mihalkova and Mooney, 2007), (Natarajan et al., 2012) and (Domingos and Lowd, 2009). Here, we give a brief overview on weight learning in MLNs.

2.3.1 Weight Learning in MLNs

The problem of weight learning assumes that the MLN structure is known and given data, the task is to learn a weight for each MLN formula. Here, we focus on *Max-likelihood learning* which is most popular approach for MLNs.[1]. In Max-likelihood learning, we choose the weights for the formulas to maximize the likelihood or the probability of the data. Weights can be learned either *generatively* or *discriminatively*. In generative learning, we maximize

[1]Other methods such as *Max-margin learning* have also been proposed for MLNs (Huynh and Mooney, 2009)

the log-likelihood of the data while in discriminative learning, we maximize the conditional log-likelihood, i.e., a set of variables are assumed as evidence variables whose truth value is known. We briefly review discriminative learning in MLNs below.

We assume that the structure of the MLN is given along with complete data. That is, we assume that there is no missing data. One easy way to implement this assumption is to assume a *closed world* universe. That is, data that is not given is assumed to be false. Given this complete data, we use gradient ascent to maximize the conditional log-likelihood. For this, we start with random weights for each formula and in each iteration of gradient ascent, we update the weight of the i-th formula using the following equation (for a derivation see (Domingos and Lowd, 2009)).

$$w_i = w_i - \alpha \mathbb{E}_{\mathbf{w},y}[N_i(\mathbf{x}, \mathbf{y})] - N_i(\mathbf{x}, \mathbf{y}) \tag{2.16}$$

where w_i is the i-th formula's weight, $\mathbf{w}$ is the current weight vector, $\mathbf{x}$ represents the non-evidence variables, $\mathbf{y}$ represents the evidence variables, $N_i(\mathbf{x}, \mathbf{y})$ is the number of satisfied groundings of the i-th formula based on the data, α is the learning rate and $E_{\mathbf{w},y}[N_i(\mathbf{x}, \mathbf{y})]$ is the expected number of satisfied groundings of the i-th formula based on the current weight vector w.r.t the conditional distribution $P(\mathbf{x}|\mathbf{y})$.

Computing the expected value in Eq. (2.16) requires exact inference over the MLN which is infeasible in practice. Typically, approximate inference is used to compute the expected value in Eq. (2.16). Two popular approximate inference based approaches are *voted perceptron* (Collins, 2002; Singla and Domingos, 2005) and *contrastive divergence* (Hinton, 2002; Lowd and Domingos, 2007a). In voted perceptron, we estimate $\mathbb{E}_{\mathbf{w},\mathbf{y}}[N_i(\mathbf{x}, \mathbf{y})]$ by counting the satisfied groundings in the MAP assignment. In contrastive divergence, we estimate $\mathbb{E}_{\mathbf{w},\mathbf{y}}[N_i(\mathbf{x}, \mathbf{y})]$ by counting the satisfied groundings from Gibbs samples.

Max-likelihood learning even with approximate inference is computationally expensive because inference has to be performed at every iteration and typically the gradient ascent

algorithm takes several hundreds of iterations to converge. Therefore, *pseudo likelihood* learning is a popular alternative that has been adopted because it is computationally inexpensive, i.e., it does not require inference in each sub-step. Here, the likelihood is approximated as a product of conditional probabilities and computing the weight-update simply requires counting over the data. For more details on pseudo likelihood learning, please refer to (Domingos and Lowd, 2009).

CHAPTER 3

HANDLING LOGICAL DEPENDENCIES IN MCMC BASED INFERENCE

Deterministic dependencies or logical constraints are ubiquitous when we consider real-world applications. For example, application domains such as linkage analysis (Fishelson and Geiger, 2004), stereo vision (Scharstein and Szeliski, 2002), medical diagnosis (Shwe et al., 1991) and entity resolution (McCallum and Wellner, 2004) often contain both probabilistic and deterministic dependencies. MLNs typically model such deterministic dependencies with very large (or very small) weights in its formulas. Unfortunately, inference which is already a hard problem becomes much more challenging when the model encodes these dependencies. Specifically, popular approximate inference methods such as Gibbs sampling (Geman and Geman, 1984) and Belief propagation (Murphy et al., 1999; Yedidia et al., 2005) often perform poorly in the presence determinism (0 or very small probabilities in the joint distribution). Further, a second problem is that logical constraints implicitly lead to highly correlated variables in the distribution which drastically slows down convergence of approximate inference algorithms, especially sampling based ones. In this chapter, we address both problems in the context of Gibbs sampling. Specifically, we propose two algorithms, (i) `GiSS`, an algorithm that combines Gibbs sampling with SampleSearch (Gogate and Dechter, 2011), an advanced importance sampling algorithm that can sample from hard deterministic state spaces, (ii) an adaptive Gibbs sampler that tractably and effectively combines blocking and collapsing, two strategies that help improve convergence in the presence of correlations. We next describe in detail the theory and experimental evaluations for each of the two algorithms.

X	Y	ϕ_1
0	0	0.5
0	1	0
1	0	0
1	1	0.5

(a)

Y	Z	ϕ_2
0	0	0.25
0	1	0.25
1	0	0.25
1	1	0.25

(b)

X	Y	Z	P
0	0	0	0.25
0	0	1	0.25
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0.25
1	1	1	0.25

(c)

Figure 3.1. A Markov network with determinism. (a) and (b) are the potentials of the Markov network. (c) is the full joint distribution.

3.1 GiSS: Sampling in PGMs with determinism

Gibbs sampling performs poorly in the presence of determinism and converges to incorrect results as its Markov chain is no longer ergodic (Gilks et al., 1996). This is because, determinism fractures the support (assignments that have non-zero probability) of the state space into disconnected clusters. Gibbs sampling will get trapped in one cluster and its Markov chain will converge to the marginal probability distribution over that cluster, which is clearly incorrect. We illustrate this with the following example.

Example 4. *Consider a graphical model with 3 variables X,Y and Z with 2 potentials as shown in Figure 3.1. X,Y share 2 deterministic constraints (given by the 0 probability values in the table). Assume that the Gibbs sampler starts at state (0,0,0) and X, Y and Z are sampled in that order. Since $P(X = 0|Y = 0) = 1$, the next Gibbs sample is necessarily (0,0,0). Similarly, in sampling Y, since $P(Y = 1|X = 0, Z = 0) = 0$, the state of the sampler stays at (0,0,0). Further, $P(Z = 0|Y = 0) = P(Z = 1|Y = 0) = 0.5$ and therefore, the next sample is either (0,0,0) or (0,0,1) with equal probability. Thus, we converge to the distribution where the states (0,0,0) and (0,0,1) each have probability 0.5 which as seen in Figure 3.1 (c) is clearly not the correct distribution.*

Many solutions have been proposed in the past to address the aforementioned problem. Notable examples are Blocking (Jensen et al., 1993), Rao-Blackwellisation (Casella and Robert, 1996; Liu, 2001) and the recently proposed scheme by (Gries, 2011). However, none of them are scalable to large PGMs because they require the PGM projected over the deterministic variables to be tractable and thus amenable to exact inference. This is not always possible, especially when the number of deterministic variables (or their treewidth) is large. Here, we propose a hybrid approach for solving this problem: combine Gibbs sampling with SampleSearch (Gogate and Dechter, 2011, 2007b), a state-of-the-art importance sampling algorithm that leverages complete CSP/SAT solvers to generate high quality samples from hard deterministic spaces. We call our new algorithm `GiSS`.

The key idea in `GiSS` is the following: use SampleSearch to sample the variables involved in hard constraints (henceforth called deterministic variables) and then use Gibbs sampling to sample the remaining variables (henceforth called non-deterministic variables) given the sampled assignment to the deterministic variables. In essence, the use of SampleSearch partitions the state-space into multiple clusters, one corresponding to each unique sample generated by SampleSearch. SampleSearch samples the clusters independently while Gibbs sampling collects dependent samples within each cluster. Since `GiSS` uses importance sampling as a sub-step, its samples need to be weighted appropriately in order to guarantee that the estimates derived from them converge to the correct answer. It turns out that computing these weights involves computing the probability of the sampled (partial) assignment to all deterministic variables, a problem that is in #P. We therefore propose several approximate methods, which estimate the weights using the samples on the non-deterministic variables generated via Gibbs sampling. We show that our approximate weighting schemes are correct in that they yield consistent (asymptotically unbiased) estimates of one-variable marginals – a standard inference task in PGMs.

SampleSearch

Importance sampling performs poorly in presence of determinism because it suffers from the *rejection problem* (Gogate and Dechter, 2011), namely, the proposal distribution may be such that the probability of generating a sample having *zero weight*[1] from it is arbitrarily close to one. Assuming finite sample size, which is the case in practice, with high probability, all samples drawn from such a proposal distribution will have zero weight. This is problematic because the estimate of the partition function ($\widehat{Z}$) will equal zero while the estimate of the marginal probabilities ($\widehat{P}_T(\mathbf{x})$) will be an undefined number $0/0$.

One can solve the rejection problem by modifying the proposal distribution such that it is *backtrack-free* along an ordering. One approach to make the proposal distribution backtrack-free is to write it as a probability tree and remove all sub-trees that contain only zero weight assignments, normalizing to ensure that it represents a valid probability distribution. This approach is illustrated in Figure 3.2. Unfortunately, this approach is infeasible in practice because it has exponential time and space complexity. Moreover, the problem of constructing a backtrack-free proposal distribution is #P-complete (Roth, 1996; Yu and van Engelen, 2012) and thus there is no hope of solving it using other methods.

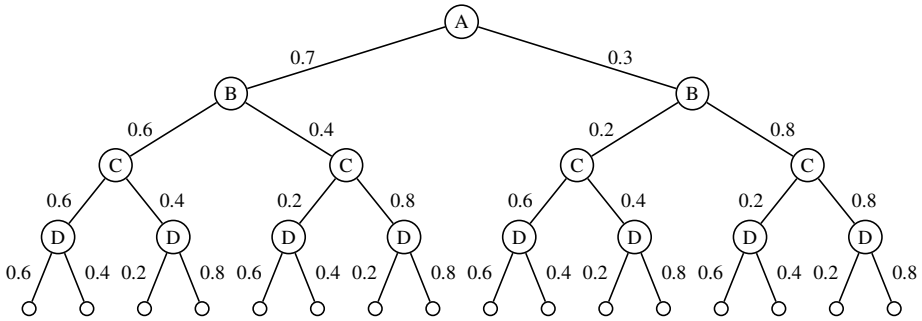
SampleSearch (Gogate and Dechter, 2007b, 2011) solves the rejection problem by interleaving backtracking search with sampling. It can be understood as a randomized or probabilistic depth-first search (DFS) for an assignment having non-zero weight over the probability tree. At each non-leaf node in the probability tree, the DFS algorithm selects one of its child nodes as the next node to visit with probability attached to the edge between the node and the child. (Gogate and Dechter, 2007b) proved that this randomized DFS algorithm generates independent samples from the backtrack-free distribution Q_{BF} of Q. Thus, given a set of samples generated via SampleSearch, we can weight them as

[1] A sample $\overline{\mathbf{x}}$ has zero weight if $Q(\overline{\mathbf{x}}) > 0$ but $\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}) = 0$.

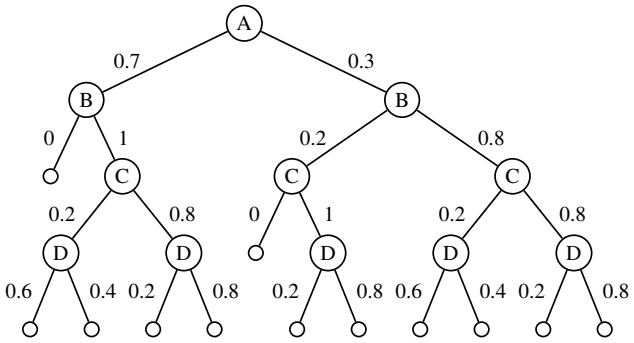
A	B	C	ϕ_1
0	0	0	0
0	0	1	0
0	1	0	0.7
0	1	1	0.3
1	0	0	0
1	0	1	1
1	1	0	0.25
1	1	1	0.75

B	D	ϕ_2
0	0	0.25
0	1	0.75
1	0	0.3
1	1	0.7

(a) A PGM



(b) Proposal Distribution as a probability tree



(c) Backtrack-free probability tree of (b)

Figure 3.2. (a) shows two functions ϕ_1 and ϕ_2 defining a PGM over four binary random variables (ϕ_1 is a deterministic function). Let $Q(A, B, C, D) = Q(A)\ Q(B|A)\ Q(C|B)\ Q(D|C)$ be the proposal distribution where $Q(A = 0) = 0.7$, $Q(B = 0|A = 0) = Q(C = 0|B = 0) = Q(D = 0|C = 0) = 0.6$ and $Q(B = 0|A = 1) = Q(C = 0|B = 1) = Q(D = 0|C = 1) = 0.2$. (b) shows the probability tree of Q. In the tree, the left and the right branches of a node labeled by X denote the assignment of 0 and 1 to X respectively. (c) shows the backtrack-free probability tree derived from the proposal distribution by removing all sub-trees that contain only zero weight assignments, and normalizing. We have removed two sub-trees from the probability tree given in (b): the sub-tree rooted at $A = 0, B = 0$ and the sub-tree rooted at $A = 1, B = 0, C = 0$. Any samples extending these two partial assignments will have zero weight.

$w(\overline{\mathbf{x}}^{(t)}) = \frac{\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}^{(t)})}{Q_{BF}(\overline{\mathbf{x}}^{(t)})}$ and use Equations (2.11) and (2.12) to estimate the partition function and one-variable marginals respectively.

In practice, we can speed-up SampleSearch by using advanced backtracking (DFS) schemes developed in the SAT/CSP literature over the past few decades. These advanced search procedures and their implementations are quite fast and can consistently solve problems having

millions of variables in a few seconds (Eén and Sörensson, 2003). We leveraged these advanced schemes in our experiments.

3.1.1 The GiSS Algorithm

In this section, we describe Algorithm GiSS, which combines Gibbs sampling with SampleSearch. We propose several weighting schemes that trade computational complexity with accuracy for it and show that all of them are correct in the sense that they yield consistent estimates.

Before describing the algorithm, we introduce some additional notation. Given a PGM $\mathcal{M} = \langle \mathbf{X}, \Phi \rangle$, let $\mathbf{X}_d$ and $\mathbf{X}_p$ respectively denote the sets of deterministic and non-deterministic variables ($\mathbf{X}_p = \mathbf{X} \setminus \mathbf{X}_d$) in $\mathcal{M}$. Also, let $\mathcal{M}|\overline{\mathbf{x}}_d$ denote the PGM obtained by instantiating $\overline{\mathbf{x}}_d$ in $\mathcal{M}$.

Algorithm 2 gives the pseudo-code of GiSS. The algorithm takes as input a PGM $\mathcal{M}$, two integers T and U and outputs an estimate of all one-variable marginals. The algorithm begins by initializing the estimates of all one-variable marginals to zero. It then constructs a proposal distribution $Q(\mathbf{X}_d)$ over the deterministic variables and uses SampleSearch to generate T samples, one by one, from the backtrack-free distribution $Q_{BF}(\mathbf{X}_d)$ of $Q(\mathbf{X}_d)$. For each sampled assignment $\overline{\mathbf{x}}_d^{(i)}$ generated by SampleSearch, the algorithm generates U samples over the non-deterministic variables by performing Gibbs sampling over $\mathcal{M}|\overline{\mathbf{x}}_d^{(i)}$. The algorithm then weights the samples appropriately and updates the running estimate of all one-variable marginals. Finally, after all the samples are generated, the algorithm normalizes the estimates and returns them.

3.1.2 Computing the Sample Weights

Unlike Gibbs sampling in which every sample has the same weight, each sample generated by GiSS needs to be weighted properly in order to guarantee convergence to the correct answer. The weighting scheme of GiSS is formalized by the following theorem.

Algorithm 2: `GiSS`

Input: Graphical Model $\mathcal{M} = \langle \mathbf{X}, \Phi \rangle$, Integers T and U
Output: An estimate of all one-variable marginals

for *each value* x *in the domain of* $X \in \mathbf{X}$ **do**
  $\widehat{P}(\overline{x}) = 0$ `// Initialize the estimates`
Construct the proposal distribution $Q(\mathbf{X}_d)$ over the deterministic variables;
for $t = 1$ *to* T **do**
  Use SampleSearch to generate a sample $\overline{\mathbf{x}}_d^{(t)}$ from the backtrack-free distribution $Q_{BF}(\mathbf{X}_d)$ of $Q(\mathbf{X}_d)$;
  `// Sample the non-deterministic variables`
  for $j = 1$ *to* U **do**
    $\overline{\mathbf{x}}_p^{(j)} = \text{Gibbs-iteration}(\mathcal{M}|\overline{\mathbf{x}}_d^{(t)})$;
    Compute the sample weight $w(\overline{\mathbf{x}}_d^{(t)}, \overline{\mathbf{x}}_p^{(j)})$;
    for *each value* x *in the domain of* $X \in \mathbf{X}$ **do**
      $\widehat{P}(\overline{x}) = \widehat{P}(\overline{x}) + w(\overline{\mathbf{x}}_d^{(t)}, \overline{\mathbf{x}}_p^{(j)}) \mathbb{I}_{\overline{x}}(\overline{\mathbf{x}}_d^{(t)}, \overline{\mathbf{x}}_p^{(j)})$;
Normalize the estimates and **return** $\{P(\overline{x})\}$;

Theorem 2. *Given* $\mathcal{M}$ *and a complete* `GiSS` *sample* $(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p)$ *for* $\mathcal{M}$*, where* $\overline{\mathbf{x}}_d$ *is sampled from* $Q_{BF}(\mathbf{X}_d)$*, and* $\overline{\mathbf{x}}_p$ *is sampled from* $\mathcal{M}|\overline{\mathbf{x}}_d$*, the importance weight of the* `GiSS` *sample (up to a normalization constant) is given by,*

$$w(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p) = \frac{Z(\mathcal{M}|\overline{\mathbf{x}}_d)}{Q_{BF}(\overline{\mathbf{x}}_d)} \tag{3.1}$$

where $Z(\mathcal{M}|\overline{\mathbf{x}}_d)$ *is the partition function of* $\mathcal{M}|\overline{\mathbf{x}}_d$*.*

Proof. Since $\overline{\mathbf{x}}_d$ is sampled from the proposal distribution Q_{BF}, while $\overline{\mathbf{x}}_p$ is sampled from the true distribution, namely, $P(\overline{\mathbf{x}}_p|\overline{\mathbf{x}}_d)$, the importance weight of the full sample is given by,

$$w(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p) = w(\overline{\mathbf{x}}_d) = \frac{P(\overline{\mathbf{x}}_d)}{Q_{BF}(\overline{\mathbf{x}}_d)} \tag{3.2}$$

Substituting for $P(\overline{\mathbf{x}}_d)$, in Eq. (3.2), we obtain the following equation,

$$w(\overline{\mathbf{x}}_d) = \frac{1}{Z} \frac{\sum_{\overline{\mathbf{x}}_p} \prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p)}{Q_{BF}(\overline{\mathbf{x}}_d)} \propto \frac{Z(\mathcal{M}|\overline{\mathbf{x}}_d)}{Q_{BF}(\overline{\mathbf{x}}_d)} \tag{3.3}$$

□

The following Theorem now follows directly from to the correctness of the importance weight for each sample.

Theorem 3. *Assuming that the Gibbs sampling sub-step in Algorithm* `GiSS` *generates samples from* $\mathcal{M}|\overline{\mathbf{x}}_d^{(t)}$*, where* $\overline{\mathbf{x}}_d^{(t)}$ *is sampled from* $Q_{BF}(\mathbf{X}_d)$*, if the weight of the sample* $w(\overline{\mathbf{x}}_d^{(t)}, \overline{\mathbf{x}}_p^{(j)})$ *is computed as specified in Eq* (3.3)*, the estimate* $\widehat{P}(\overline{x})$ *output by Algorithm* `GiSS` *converges almost surely to* $P(\overline{x})$ *as* $T \to \infty$*. Namely, Algorithm* `GiSS` *yields an asymptotically unbiased estimate of one-variable marginals.*

Approximating the Sample Weights

The weighting scheme given in Eq. (3.1) requires computing the partition function of $\mathcal{M}|\overline{\mathbf{x}}_d$. However, the latter is intractable in general, i.e., computing the partition function is in the complexity class #P. Therefore, we consider three sampling-based approximate estimators for it.

The first estimator that we consider is the harmonic mean estimator (Newton and Raftery, 1994). The main advantage of this method is that it uses the samples generated by Gibbs sampling and thus requires no additional computation. Formally, the harmonic mean estimate of $Z(\mathcal{M}|\overline{\mathbf{x}}_d)$ is

$$\widehat{Z}_h(\mathcal{M}|\overline{\mathbf{x}}_d) = \frac{2^{|\mathbf{X}_p|} \times U}{\sum_{j=1}^{U} \frac{1}{\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p^{(j)})}} \tag{3.4}$$

Substituting the harmonic mean estimate of $Z(\mathcal{M}|\overline{\mathbf{x}}_d)$ given in Eq. (3.4) in Eq. (3.1), we get

$$\widehat{w}_h(\overline{\mathbf{x}}_d) = \frac{2^{|\mathbf{X}_p|} \times U}{\sum_{j=1}^{U} \frac{1}{\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p^{(j)})} Q_{BF}(\overline{\mathbf{x}}_d)} \tag{3.5}$$

Notice that in computing the marginals using the ratio estimator, since the set of deterministic variables never changes, we can eliminate $2^{|\mathbf{X_P}|}$ from Eq. (3.5). Despite its ease of use and asymptotic convergence properties, the harmonic mean estimator has large variance and can often yield highly inaccurate answers (Neal, 2008). To reduce its variance, we propose

to store (cache) the weight of each partial assignment $\mathbf{x}_d$ generated by SampleSearch. If the same assignment is generated again, we simply replace its weight by the *running average* of the current weight and the new weight. The variance is reduced because as more samples are drawn the stored weights will get closer to the desired exact weights.

The second estimator that we consider is the product estimator (Jerrum et al., 1986; Chib, 1995). Here, we pick a random assignment, say $\overline{\mathbf{x}}_p^*$, and compute $P(\overline{\mathbf{x}}_p^*)$ as a product of $|\mathbf{X}_p|$ conditional distributions:

$$P(\overline{\mathbf{x}}_p^*) = \prod_{k=1}^{|\mathbf{X}_p|} P(\overline{x}_k^* | \overline{x}_1^*, \ldots, \overline{x}_{k-1}^*)$$

To estimate each component of the form $P(\overline{x}_k^* | \overline{x}_1^*, \ldots, \overline{x}_{k-1}^*)$, we use Gibbs sampling, running it over $\mathcal{M}|\mathbf{x}_d$ with $\overline{x}_1^*$, ..., $\overline{x}_{k-1}^*$ as evidence (in principle, we can also use other inference approaches such as loopy Belief propagation (Murphy et al., 1999) instead of Gibbs sampling). Let $\widehat{P}(\overline{\mathbf{x}}_p^*)$ denote the estimate of $P(\overline{\mathbf{x}}_p^*)$. Then, we can estimate the partition function $Z(\mathcal{M}|\overline{\mathbf{x}}_d)$ using:

$$\widehat{Z}_c(\mathcal{M}|\overline{\mathbf{x}}_d) = \frac{\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p^*)}{\widehat{P}(\overline{\mathbf{x}}_p^*)} \tag{3.6}$$

Substituting the product estimate of $Z(\mathcal{M}|\overline{\mathbf{x}}_d)$ given in Eq. (3.6) in Eq. (3.1), we get

$$\widehat{w}_c(\overline{\mathbf{x}}_d) = \frac{\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p^*)}{\widehat{P}(\overline{\mathbf{x}}_p^*) Q_{BF}(\overline{\mathbf{x}}_d)} \tag{3.7}$$

Assuming that U samples generated by Gibbs sampling are used to estimate each component $P(\overline{x}_k^* | \overline{x}_1^*, \ldots, \overline{x}_{k-1}^*)$, the product estimator is $|\mathbf{X}_p|$ times more expensive to compute than the harmonic mean estimator. However, its variance is likely to be much smaller and thus there is a trade-off.

The third estimator that we consider is based on annealed importance sampling (Neal, 1993) (we will call the resulting weighting scheme annealed weighting scheme). To compute the estimate, we define a family of PGMs by parameterizing the original PGM using an "inverse temperature" setting. Specifically, given a series of inverse temperatures $\beta_0 = 0 <$

$\beta_1 \ldots < \beta_{k+1} = 1$, we define $k+2$ intermediate PGMs $(\mathcal{M}|\overline{\mathbf{x}}_d)^{\beta_i}$, $0 \leq i \leq (k+1)$ where $(\mathcal{M}|\overline{\mathbf{x}}_d)^{\beta_i}$ denotes the PGM obtained by replacing each function ϕ in $\mathcal{M}|\overline{\mathbf{x}}_d$ by ϕ^{β_i}. Note that the PGM corresponding to β_{k+1} is the same PGM as $\mathcal{M}|\overline{\mathbf{x}}_d$ (the original PGM). Rewriting $Z(\mathcal{M}|\overline{\mathbf{x}}_d)$,

$$Z(\mathcal{M}|\overline{\mathbf{x}}_d) = \prod_{i=1}^{k+1} \frac{Z\left((\mathcal{M}|\overline{\mathbf{x}}_d)^{\beta_i}\right)}{Z\left((\mathcal{M}|\overline{\mathbf{x}}_d)^{\beta_{i-1}}\right)} \tag{3.8}$$

Given U samples generated from $(\mathcal{M}|\overline{\mathbf{x}}_d)^{\beta_{i-1}}$ using Gibbs sampling, we can estimate the i^{th} ratio in the right hand side of Eq. (3.8) using the following quantity (Neal, 1993):

$$\frac{1}{U}\sum_{j=1}^{U}\prod_{\phi\in\Phi}(\phi(\overline{\mathbf{x}}^{(j)}))^{\beta_i-\beta_{i-1}} \tag{3.9}$$

Substituting the ratio estimates obtained using Eq. (3.9) in Eq. (3.8), we obtain an estimate for $Z(\mathcal{M}|\overline{\mathbf{x}}_d)$. Substituting this estimate in Eq. (3.1), we get

$$\widehat{w}_a(\overline{\mathbf{x}}_d) = \frac{1}{U}\frac{\prod_{i=1}^{k+1}\sum_{j=1}^{U}\prod_{\phi\in\Phi}(\phi(\overline{\mathbf{x}}^{(j)}))^{\beta_i-\beta_{i-1}}}{Q_{BF}(\overline{\mathbf{x}}_d)} \tag{3.10}$$

We can show that all three weighting schemes are correct in the sense that they yield consistent estimates .

Theorem 4. *Assuming that the Gibbs sampling algorithm generates samples from* $\mathcal{M}|\overline{\mathbf{x}}_d^{(t)}$ *and the weight of each sample is given by either Eq.* (3.5)*, Eq.* (3.7) *or Eq.* (3.10)*, the estimate output by Algorithm* `GiSS` *converges almost surely to* $P(\overline{x})$ *as* $T \to \infty$*.*

Proof. Consider the harmonic mean estimator in Eq. (3.5). Substituting the approximate weight $\widehat{w}_h(\overline{\mathbf{x}}_d^{(t)})$ in the asymptotically unbiased estimator specified in Theorem 3, we have,

$$\widehat{P}(\overline{x}) = \frac{\sum_{t=1}^{T}\sum_{j=1}^{U}\widehat{w}_h(\overline{\mathbf{x}}_d^{(t)},\overline{\mathbf{x}}_p^{(j)})\mathbb{I}_{\overline{x}}(\overline{\mathbf{x}}_d^{(t)},\overline{\mathbf{x}}_p^{(j)})}{\sum_{i=1}^{T}\sum_{j=1}^{U}\widehat{w}_h(\overline{\mathbf{x}}_d^{(t)},\overline{\mathbf{x}}_p^{(j)})} \tag{3.11}$$

The harmonic mean estimator yields an unbiased estimate of $Z(\mathcal{M}|\overline{x}_d)$. This means that, as $T \to \infty$, for every partial sample $\overline{\mathbf{x}}_d^{(t)}$, $\widehat{w}_h(\overline{\mathbf{x}}_d) \to w(\overline{\mathbf{x}}_d)$. Therefore, Eq. 3.11 is a ratio of asymptotically unbiased quantities and is thus an asymptotically unbiased estimator for $P(\overline{x})$.

□

3.1.3 Related Work and Discussion

As mentioned earlier, a number of approaches have been proposed in the past to solve the convergence problem of Gibbs sampling in presence of determinism. The two conventional, popular solutions are Blocking (Jensen et al., 1993) and Rao-Blackwellisation (or collapsing) (Casella and Robert, 1996; Liu, 2001). Unlike `GiSS`, these methods are not scalable to large PGMs because in order to guarantee convergence to the correct answer, they require $Z(\mathcal{M}|\overline{\mathbf{x}}_p)$ to be tractable, where $\overline{\mathbf{x}}_p$ is a full assignment to the non-deterministic variables. Another related work is the MC-SAT algorithm (Poon and Domingos, 2006) which combines slice sampling (Neal, 2000) with SAT solution samplers (Wei et al., 2004). Unlike `GiSS`, MC-SAT is a local-search procedure and as a result is unable to make large moves in the state-space. Large moves often promote rapid mixing. Recently (Gries, 2011) proposed a modified Gibbs sampling algorithm for inference in probabilistic logic models with deterministic constraints. His method assumes that the deterministic portion of the PGM is tractable and can be succinctly expressed using a subset of description logic. `GiSS` does not make this assumption and therefore is more widely applicable. Finally, `GiSS` is related to (Hajishirzi and Amir, 2008) who first sample deterministic paths in a probabilistic sequence to improve accuracy. However, unlike `GiSS`, their approach is applicable specifically to dynamic first order models.

`GiSS` is based on the premise that whenever possible, it is better to use Gibbs sampling (MCMC) rather than importance sampling (SampleSearch); we stop performing importance sampling when the underlying Markov chain is guaranteed to be ergodic. Although this premise is debatable, Gibbs sampling and other MCMC techniques are preferred by practitioners over importance sampling because it is often hard to derive a good proposal distribution for large PGMs. In `GiSS`, we only have to construct the proposal distribution over a fraction of the variables in the PGM – on variables involved in deterministic functions. Further, generating independent importance samples is computationally more expensive than

generating Gibbs samples. Therefore, GiSS can generate more samples than SampleSearch and therefore reduce the variance of its estimates.

It is also possible to view GiSS as a general algorithm for combining Gibbs sampling and importance sampling. To see this, notice that all we have to do is partition the set of variables in the PGM into two sets, $\mathbf{X}_d$ and $\mathbf{X}_p$ and use GiSS as before. This general method can especially useful when we have domain knowledge or access to a (provably) good proposal distribution over $\mathbf{X}_d$. The key problem here is finding a partitioning that is likely to yield a good accuracy in practice.

3.1.4 Experiments

We compared GiSS with four approximate inference algorithms from literature: MC-SAT (Poon and Domingos, 2006), Belief Propagation (BP) (Murphy et al., 1999), SampleSearch (SS) (Gogate and Dechter, 2011) and Gibbs sampling. We used the implementations of MC-SAT and BP available in the Alchemy system (Kok et al., 2006). We implemented GiSS on top of the code base for SampleSearch, available from the authors (Gogate and Dechter, 2011). For a fair comparison, in GiSS, the proposal is constructed using the the same scheme as SampleSearch. The only difference is that in Samplesearch, the proposal distribution is defined on all the variables whereas in GiSS the proposal is defined only on the deterministic variables. The proposal is constructed for both algorithms using the output of BP. In GiSS, we set $U = 25$ and used 25 samples for burn-in. We performed our experiments on a quad-core machine with 8GB RAM, running CentOS Linux and ran each algorithm for 500 seconds on each benchmark PGM. Note that both SampleSearch (co-winner of the UAI 2010 and the PASCAL 2011 competitions) and MC-SAT are state-of-the-art solvers that explicitly reason about and exploit determinism in PGMs. Therefore, our main comparison is with them.

We evaluated the algorithms on *mixed deterministic and probabilistic graphical models* from four benchmark domains: Grids, linkage analysis, statistical relational learning, and

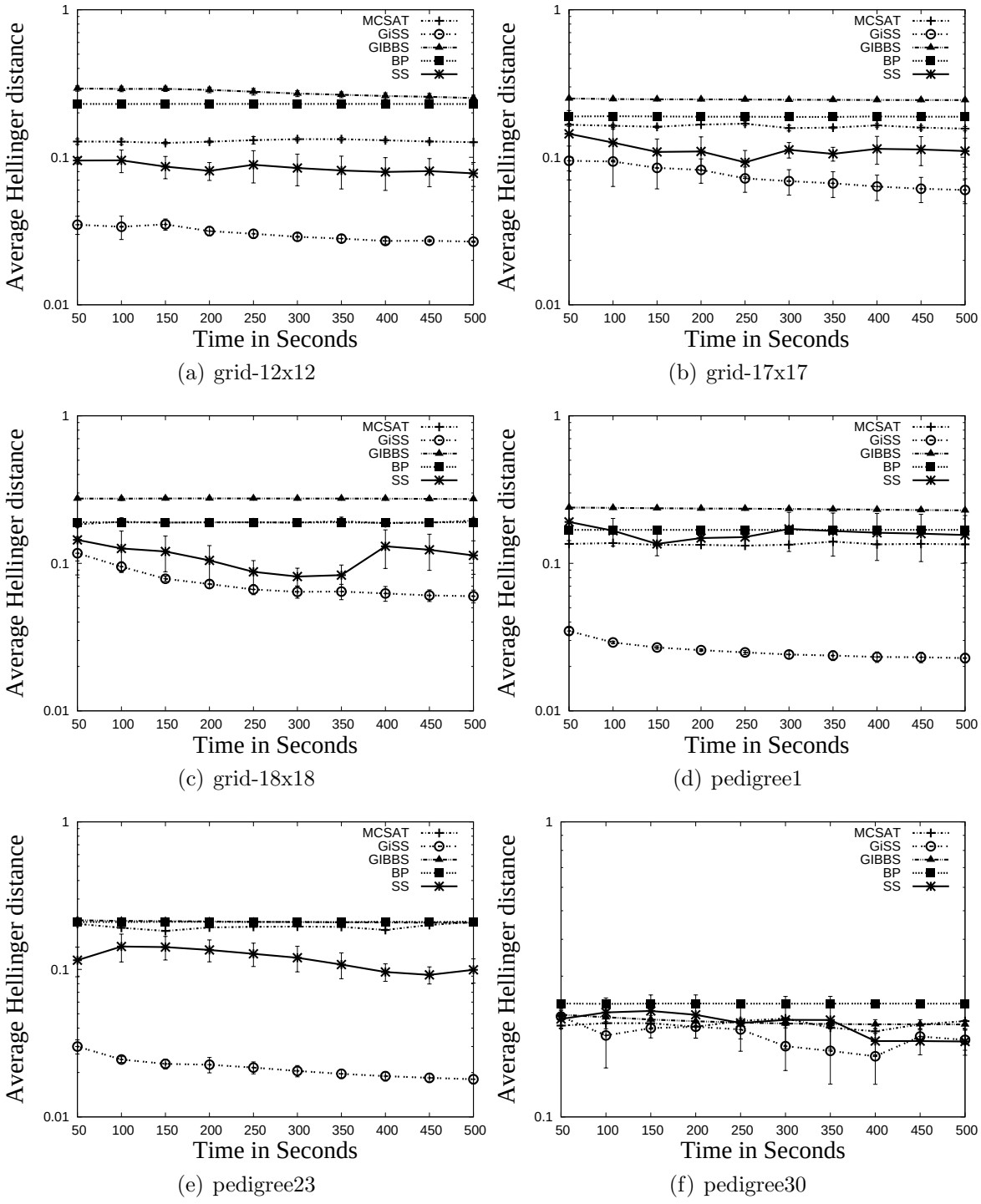
medical diagnosis. All the PGMs and test cases are available from the UAI 2008 repository (http://graphmod.ics.uci.edu/uai08/). Note that in order to fairly evaluate approximate inference, we need to compute the exact marginal probabilities on these benchmarks. Thus, the benchmarks have relatively small treewidth where exact inference can be performed in a computationally feasible manner. However, they correspond to several varied applications as described in the next section. In our experiments, we measured performance using the average Hellinger distance (Kokolakis and Nanopoulos, 2001) between the exact and the approximate one-variable marginals. We implemented and experimentally evaluated all three weighting schemes described in the previous section. We found that the harmonic mean weighting scheme is the most cost-effective. Therefore, we will first compare our best algorithm, `GiSS` equipped with the harmonic mean weighting scheme, with BP, MC-SAT, SampleSearch and Gibbs sampling. After that, we will describe our results comparing the three weighting schemes.

Results Comparing `GiSS` with Other Techniques

Figures 3.3 and 3.4 illustrate our results. We can see that on all instances, `GiSS` is either better than or competitive with the other algorithms in terms of accuracy.

Grids. We experimented with three grid PGMs having sizes 12×12, 17×17 and 18×18 respectively. These networks were generated by (Sang et al., 2005). Almost 75% of the functions in these PGMs are deterministic. Figure 3.3(a)-(c) show that `GiSS` is the best performing algorithm on Grids. Specifically, on all the three grids, `GiSS` had smaller error as well as variance (indicated by the small error bars) compared to the other algorithms.

Linkage PGMs are generated by converting data used for linkage analysis in computational Biology (Fishelson and Geiger, 2004) into a PGM. We experimented with three linkage PGMs: pedigree1 (300 variables), pedigree23 (300 variables) and pedigree30 (1000 variables). Almost 85% of the functions in these PGMs are deterministic. Figure 3.3(d)-(f)



(a) grid-12x12 (b) grid-17x17

(c) grid-18x18 (d) pedigree1

(e) pedigree23 (f) pedigree30

Figure 3.3. Average Hellinger distance between the exact and the approximate one-variable marginals plotted as a function of time along with error bars (indicating the standard deviation taken over 5 runs) for `GiSS`, MC-SAT, BP, SampleSearch (SS) and Gibbs sampling. (a)-(c):Grids; (d)-(f):Linkage;

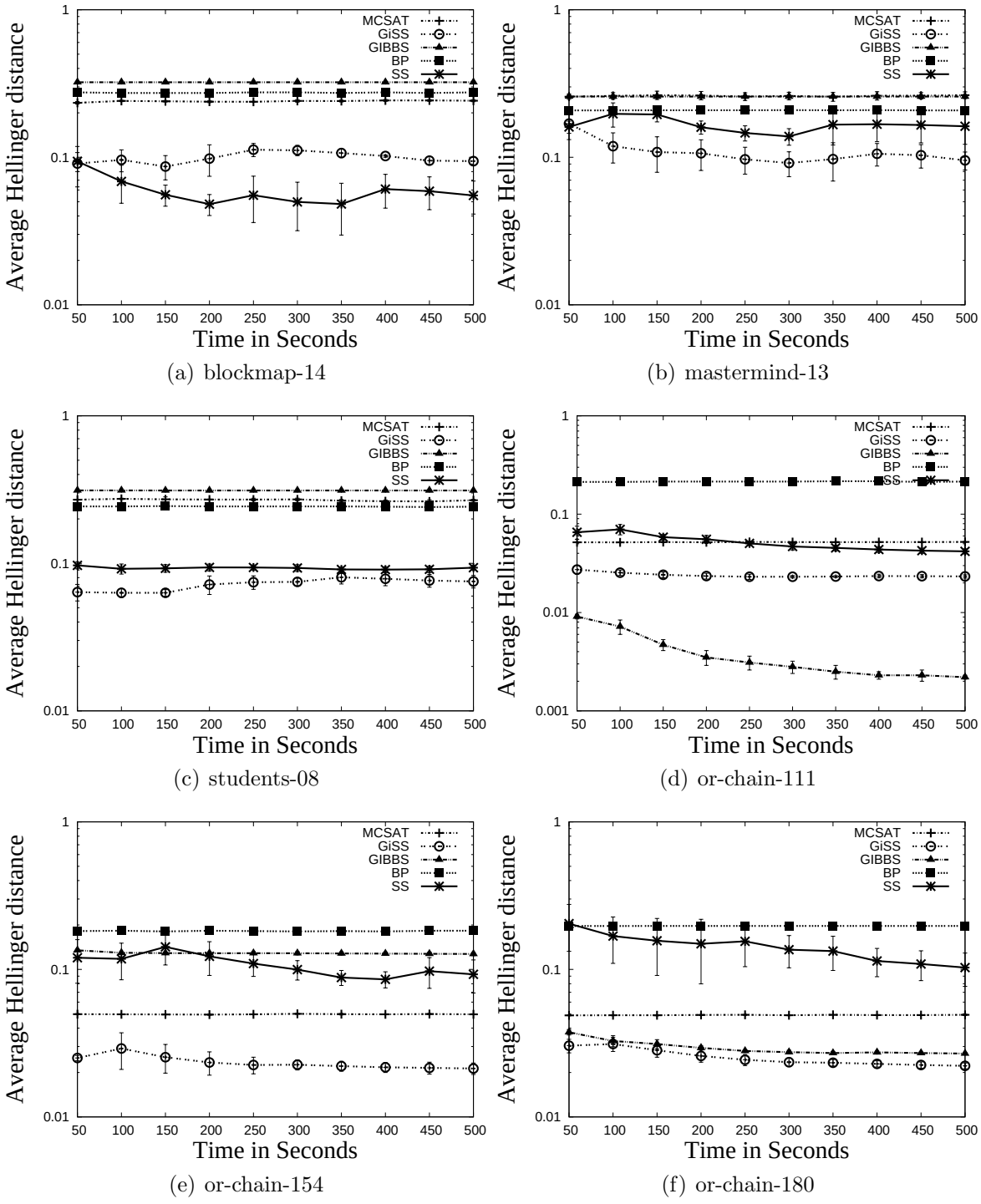
show the results. On pedigree1 and pedigree23, GiSS is clearly superior while on pedigree30, GiSS is slightly better than than the other algorithms (however, SampleSearch has smaller variance than GiSS).

Relational PGMs are obtained by *grounding* statistical relational models (Getoor and Taskar, 2007; Domingos and Lowd, 2009). Statistical relational models often have large amount of determinism and GiSS is ideal for such models because it combines logical (SampleSearch) and probabilistic inference (Gibbs sampling). We experimented with three relational PGMs: mastermind-13 (1200 variables), blockmap-14 (700 variables) and students-08 (400 variables). Almost 90% of the dependencies in these networks are deterministic. From Figure 3.4(a)-(c), we see that on the mastermind and student networks, GiSS is slightly better than SampleSearch and clearly superior to the others. On blockmap-14, SampleSearch is the best performing scheme followed by GiSS.

Promedas PGMs are noisy-OR networks generated by the Promedas system for medical diagnosis (Wemmenhove et al., 2007). In our experiments, we used three networks from this class, or-chain-111, or-chain-154 and or-chain-180. These PGMs have around 500 variables and almost 50% of the functions in them are deterministic. From Figure 3.4(d)-(f), we can see that on or-chain-154, GiSS is the best performing algorithm. On or-chain-180, GiSS is marginally better than Gibbs sampling but significantly better than the others. However, on or-chain-111, Gibbs sampling performs better than GiSS. We suspect that this is because the Markov chain associated with Gibbs sampling for this network is ergodic (note that determinism is necessary but not sufficient for breaking ergodicity of the Markov chain underlying the Gibbs sampler). Moreover, the proposal used by GiSS is a poor approximation of P as evidenced by the poor performance of BP on this network.

Results Comparing the Weighting Schemes

Next, we compared the efficacy of our different weighting schemes. Figure 3.5 shows the impact of varying time on the accuracy of the three weighting schemes. For these experi-



(a) blockmap-14

(b) mastermind-13

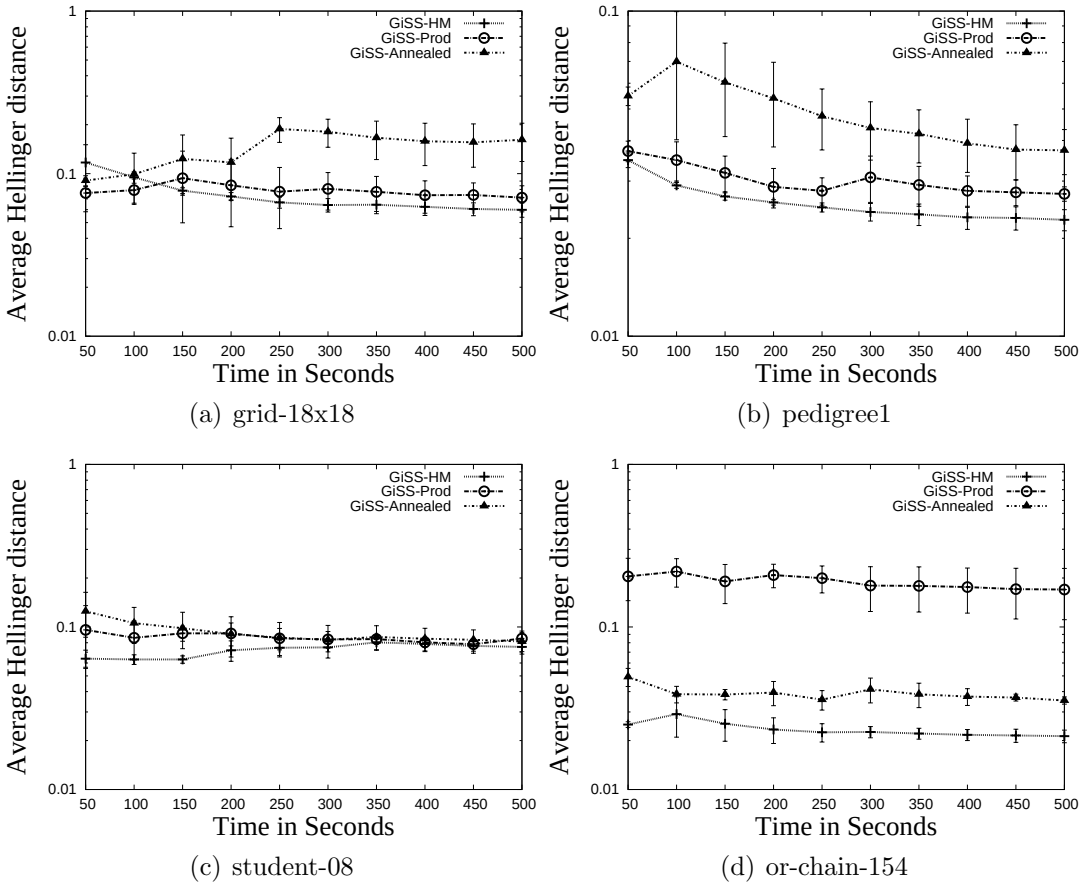
(c) students-08

(d) or-chain-111

(e) or-chain-154

(f) or-chain-180

Figure 3.4. Average Hellinger distance between the exact and the approximate one-variable marginals plotted as a function of time along with error bars (indicating the standard deviation taken over 5 runs) for `GiSS`, MC-SAT, BP, SampleSearch (SS) and Gibbs sampling.(a)-(c):Relational; (d)-(f):Promedas.



(a) grid-18x18 (b) pedigree1

(c) student-08 (d) or-chain-154

Figure 3.5. Average Hellinger distance between the exact and the approximate one-variable marginals plotted as a function of time along with error bars (standard deviation taken over 5 runs) for `GiSS`-HM (`GiSS` with the harmonic mean weighting scheme), `GiSS`-Prod: (`GiSS` with the product weighting scheme) and `GiSS`-Annealed: (`GiSS` with the annealed weighting scheme).

ments, we used $U = 25$ for the harmonic mean and the product weighting schemes. For the annealed weighting scheme, we used a linear schedule $\beta_k = \beta_{k+1} \times 0.95$ to vary the inverse temperature. We see that the harmonic mean scheme dominates others in terms of average accuracy as well as the variance. The harmonic mean scheme is computationally more efficient than the other schemes and as a result its estimates are based on a larger sample size. However, its weights are less accurate. Our results suggest that *larger sample size is more critical to improving the accuracy than high quality weights.*

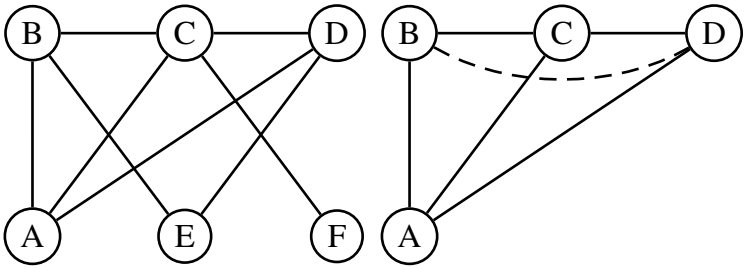
3.2 Dynamic Blocking and Collapsing

We now present our second MCMC algorithm that combines blocking (Jensen et al., 1993; Liu et al., 1994) and collapsing (Liu et al., 1994), two of the most popular strategies for improving the statistical efficiency of Gibbs sampling, particularly when the distribution under consideration contains highly correlated variables. Both these strategies trade sample quality with sample size. Thus, the hope is that users will achieve the right balance between the two for the specific PGM at hand, improving the estimation accuracy as a result. Here, we show that striking this balance is non-trivial and systematically develop an approach to combine these two techniques into a unified adaptive MCMC sampler.

Unlike Gibbs sampling which samples each variable individually given others, blocked Gibbs sampling partitions the variables into disjoint groups or blocks and then *jointly samples* all variables in each block given an assignment to all other variables not in the block. Joint sampling is more expensive than sampling variables individually but the samples are of higher quality in that for a fixed sample size, the estimates based on blocked Gibbs sampling have smaller variance than the ones based on Gibbs sampling (Liu et al., 1994). A collapsed Gibbs sampler operates by *marginalizing out* a subset of variables (collapsed variables) and then generating dependent samples from the marginal distribution over the remaining variables via conventional Gibbs sampling.[2] Marginalizing out variables is more expensive than sampling them. However, since only a sub-space is sampled, the samples are of higher quality.

Although, it is provably better to collapse a variable rather than block (group) it with other variables (Liu et al., 1994), collapsing is computationally more expensive than blocking and in practice, in many cases, the latter is feasible while the former is not. Therefore, an

[2]Collapsing is often called Rao-Blackwellisation. Technically, the latter is an advanced estimator while blocking and collapsing are advanced sampling strategies. In principle, we can also use the Rao-Blackwell estimator in blocked Gibbs sampling (Hamze and de Freitas, 2004). Here, we will separate sampling from estimation.



(a) A PGM (b) Collapsed PGM

Figure 3.6. Example to illustrate trade-off between blocking and collapsing.

obvious idea is to combine blocking and collapsing, and we aim to investigate this combination in the context of PGMs. Specifically, the key question we seek to answer is: find a k-way partitioning of the variables in the PGM where each of the first $k-1$ subsets is a block and the k-th subset contains all the collapsed variables, such that the estimation error is minimized and the resulting algorithm is tractable. This problem is non-trivial because of the complex interplay between collapsing and blocking. For example,

Example 5. *Consider the pair-wise Markov network (potentials defined on edges) given in Figure 3.6(a). Let us assume that each variable in the network has d values in its domain and our time and memory resource constraints dictate that we cannot incur more than* $O(d^3)$ *complexity. Let us further assume that we have prior knowledge that A, B, C, and D should be blocked in order to improve the estimation accuracy (for instance, they are highly correlated or involved in deterministic constraints). Notice that we can only collapse (eliminate) E and F from the PGM . Otherwise, we will violate the complexity constraints. However, eliminating both E and F yields a clique over* A, B, C, D *(see Figure 3.6(b)) and we can no longer block these variables because the complexity of computing a joint distribution over them (using junction tree propagation) and then sampling from it is* $O(d^4)$*. A much better solution in this case is to collapse F, create two blocks* $\{A, B, C, D\}$ *and* $\{E\}$ *and perform blocked Gibbs sampling over this sub-space.*

As seen from the above example, in computation-limited settings, in many cases, variables that can be blocked in the original PGM can no longer be blocked in the collapsed PGM. In other words, there is a trade-off between blocking and collapsing which needs to be taken into account while combining the two schemes. We model this tradeoff by (i) defining two integer parameters α and β which bound the complexity of collapsing and blocking respectively and thus allow the user to control the number of blocked versus collapsed variables; (ii) defining two scoring functions, one each for blocking and collapsing, which favor blocks that contain variables that are highly correlated with each other and the collapsed set that contains variables which are highly correlated with other variables in the network; and (iii) casting the problem of finding the k-way partitioning into blocked and collapsed variables as a multi-objective optimization problem. This problem seeks to simultaneously maximize the scoring functions subject to the tractability constraints enforced by α and β.

The optimization problem is $\mathcal{NP}$-hard in general and therefore we propose a dynamic, greedy algorithm to solve it approximately. We integrate this algorithm with blocked-collapsed Gibbs sampling yielding a dynamic sampling algorithm. The algorithm begins by generating samples from the PGM using a feasible k-way partitioning computed using the (primal) graph associated with the PGM. It then periodically updates the partitioning after every M samples by leveraging the correlations computed from the generated samples and performs blocked-collapsed Gibbs sampling using the new partitioning. As more samples are drawn and as the accuracy of the measured correlation increases, the underlying Markov chain is likely to mix rapidly because highly correlated variables will be either blocked together or collapsed out.

3.2.1 Combining Blocking and Collapsing

As we seek to combine blocking and collapsing, the obvious question to ask is: can we prove theoretically whether such a combination will be beneficial. The answer to this question is "yes" and we briefly describe the theoretical justification for this next.

Let $\mathbf{F}$ denote the *forward operator* (Liu, 2001) for a Markov chain induced by the Gibbs sampler, let $\mathbf{x} = \{x_1 \dots x_d\}$ be the variables of the distribution (d dimensional state space), let $\mathbf{x}_{-a_1,\dots,-a_n}$ indicate $\mathbf{x} \setminus \{x_{a_1}, \dots, x_{a_n}\}$. (Liu, 2001) shows that the convergence rate w.r.t a finite-variance function h is given by,

$$||\mathbf{F}h|| = \sum_{i=1}^{d} \alpha_i var\{\mathbb{E}_P[h(\mathbf{x})|\mathbf{x}_{-i}]\} \tag{3.12}$$

P is the stationary distribution of the chain, α_i is the probability of sampling the i-th dimension of the state space and *var* is the variance.

Let $\mathbf{F}_B$ be the forward operator for the blocked sampler where x_1 and x_2 have been blocked, $\mathbf{F}_C$ be the forward operator for the collapsed sampler where x_1 has been collapsed, $\mathbf{F}_{BC}$ be the forward operator for the combined blocked-collapsed sampler where x_1 has been collapsed and x_2,x_3 have been blocked and $\mathbf{F}_S$ be the forward operator for the random scan Gibbs sampler. Then, we have the following relation,

Theorem 5. $||\boldsymbol{F}_{BC}h|| \leq ||\boldsymbol{F}_C h|| \leq ||\boldsymbol{F}_B h|| \leq ||\boldsymbol{F}_S h||$.

Proof. Since $\mathbf{x}_1$ is collapsed and $\mathbf{x}_2$, $\mathbf{x}_3$ are blocked, we can re-write Eq. (3.12) as,

$$\begin{aligned} ||\mathbf{F}_{BC}h|| = \alpha_1 var\{\mathbb{E}_P[h(\mathbf{x})|\mathbf{x}_{-1}]\} + (\alpha_2 + \alpha_3) var\{\mathbb{E}_P[h(\mathbf{x})|\mathbf{x}_{-1,-2,-3}]\} \\ + \sum_{i=4}^{d} var\{\mathbb{E}_P[h(\mathbf{x})|\mathbf{x}_{-i,-1}]\} \end{aligned} \tag{3.13}$$

Similarly, we have,

$$\begin{aligned} ||\mathbf{F}_C h|| = \alpha_1 var\{\mathbb{E}_P[h(\mathbf{x})|\mathbf{x}_{-1}]\} + \alpha_2 var\{\mathbb{E}_P[h(\mathbf{x})|\mathbf{x}_{-1,-2}]\} + \alpha_3 var\{\mathbb{E}_P[h(\mathbf{x})|\mathbf{x}_{-1,-3}]\} \\ + \sum_{i=4}^{d} var\{\mathbb{E}_P[h(\mathbf{x})|\mathbf{x}_{-i,-1}]\} \end{aligned} \tag{3.14}$$

Further we have,

$$\mathbb{E}_P[h(\mathbf{x})|\mathbf{x}_{-1,-2,-3}] = \mathbb{E}_P[\mathbb{E}_P[h(\mathbf{x})|\mathbf{x}_{-1,-2}]|\mathbf{x}_{-3}] \tag{3.15}$$

From the law of total variance, we can directly show that,

$$var\{\mathbb{E}_P[h(\mathbf{x})|\mathbf{x}_{-1,-2,-3}]\} \leq var\{\mathbb{E}_P[h(\mathbf{x})|\mathbf{x}_{-1,-2}]\} \tag{3.16}$$

Similarly, we have

$$var\{\mathbb{E}_P[h(\mathbf{x})|\mathbf{x}_{-1,-2,-3}]\} \leq var\{\mathbb{E}_P[h(\mathbf{x})|\mathbf{x}_{-1,-3}]\} \tag{3.17}$$

Using the inequalities in Eq. (3.16) and Eq. (3.17), we directly have Eq. (3.13) $\leq$ Eq. (3.14). Using the same reasoning as above, we can show the rest of the inequalities in the theorem. □

3.2.2 Optimally Selecting Blocked and Collapsed Variables

Integrating blocking and collapsing is tricky because they interact with each other. Moreover, we cannot collapse and block indiscriminately because for our algorithm to be practical we need to ensure that both blocking and collapsing are computationally tractable. We model this complex interplay as a constrained optimization problem. We start with some definitions.

Given a Markov network $\mathcal{M}$, let $\mathcal{G}$ be its primal graph. Recall that eliminating any node in $\mathcal{G}$ changes the structure of $\mathcal{G}$ by adding edges to it. For any (partial or full) ordering of vertexes in $\mathcal{G}$, say, $\pi = (X_1, \ldots, X_n)$, the *width* of the ordering is defined as follows.

Definition 9. *The width of an ordering (either partial or full) $\pi = (X_1, \ldots, X_n)$ of nodes in $\mathcal{G}$ denoted by $w(\pi, \mathcal{G})$, is the maximum degree of X_i in $\mathcal{G}_{i-1}$, where $\mathcal{G} = \mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_n$ is a sequence of graphs such that $\mathcal{G}_i$ is obtained from $\mathcal{G}_{i-1}$ by adding edges so as to make the neighbor set of X_i in $\mathcal{G}_{i-1}$ a clique, and then removing X_i from $\mathcal{G}_i$ (i.e., eliminating X_i from $\mathcal{G}_{i-1}$).*

The treewidth of $\mathcal{G}$ ($\mathbf{V}$,$\mathbf{E}$), denoted by $tw(\mathcal{G})$ equals the *minimum width* over all possible orderings of $\mathbf{V}$. The example in Figure 3.6(b) shows the graph obtained after eliminating E and F from the graph in Figure 3.6(a). The width of the partial order (E, F) is equal to 2. The width of the total order (E, F, A, B, C, D) is equal to 3. For this example, the treewidth of the graph shown in Figure 3.6(a) is also equal to 3.

We now capture these constraints and the complex interplay between blocking and collapsing in a principled manner by formulating the problem of selecting the blocks and collapsed variables as an optimization problem, as defined next.

Definition 1. *Given a PGM $\mathcal{M} = \langle \boldsymbol{X}, \Phi \rangle$, two scoring functions ω and ψ for blocking and collapsing respectively (defined in the next sub-section), and integer parameters α, and β, find a k-way partition of $\mathbf{X}$ denoted by $\mathbb{X} = \mathbb{B} \cup \mathbf{C}$, where $\mathbb{B} = \{\boldsymbol{B}_i\}_{i=1}^{k-1}$ is a set of $k-1$ blocks and $\mathbf{C}$ is the set of collapsed variables such that both $\omega(\mathbb{B})$ and $\psi(\mathbf{C})$ are maximized, subject to two tractability constraints: (i) The minimum width of $\mathbf{C}$ in the primal graph $\mathcal{G}$ is bounded by α; and (ii) The treewidth of $\mathcal{G}_{\setminus \mathbf{C}}$ (the graph obtained by eliminating $\mathbf{C}$ from $\mathcal{G}$) projected on each block $\mathbf{B}_i$ is bounded by β, namely, $\forall\ \mathbf{B}_i \in \mathbb{B}$, $tw(\mathcal{G}_{\setminus \mathbf{C}}(\mathbf{B}_i)) \leq \beta$.*

The optimization problem just presented requires maximizing two functions and is thus an instance of a multi-objective optimization problem (Marler and Arora, 2004; Hwang and S., 1979). As one can imagine, this problem is much harder than typical optimization problems in machine learning which require optimizing just one objective function. In general, there may not exist a feasible solution that simultaneously optimizes each objective function. Therefore, a reasonable approach is to find a *Pareto optimal solution* , i.e., a solution which is not dominated by any other solution in the solution space. A Pareto optimal solution cannot be improved with respect to any objective without worsening another objective.

To find Pareto optimal solutions, we will use the *lexicographic method* – a well-known approach for managing the complexity of multi-objective optimization problems. In this

method, the objective functions are arranged in order of importance and we solve a sequence of single objective optimization problems. Since collapsing changes the structure of the primal graph while blocking does not, it is obvious that we should first find the collapsed variables (i.e., give more importance to the objective function for collapsing) and then compute the blocks. We will use this approach. To reduce the sensitivity of the final solution to the objective-function for collapsing, we introduce a hard penalty which penalizes solutions that result in small block sizes (since the accuracy typically increases with the block size). We describe our proposed scoring (objective) functions and the hard penalty used next.

Scoring Functions

We wish to design scoring functions such that they improve mixing time of the underlying Markov chain. However, it is well known that computing the exact mixing time analytically is an extremely hard problem (Liu, 2001). Therefore, we use a heuristic scoring function that uses correlations between the variables measured periodically from the generated samples. In general, collapsing variables is much more effective when the collapsed variables exhibit high correlation with other variables in the PGM. For instance, a variable X that is involved in a deterministic dependency (or constraint) with another variable Y (e.g., $Y = y \rightarrow X = x$) is a good candidate for collapsing; sampling such variables likely causes the Markov chain to get stuck and hinders mixing. Similarly, blocking is effective when we jointly sample variables which are tightly correlated because sampling them separately may cause the sampler to get trapped. Moreover, we also want to minimize the number of blocks or maximize the number of variables in each block because sampling a variable jointly with other variables in a block is better than or at least as good as sampling the variables individually (Liu, 2001). We quantify these desirable properties using the following scoring functions:

$$\omega(\mathbb{B}) = \frac{1}{|\mathbb{B}|} \sum_{\mathbf{B}_i \in \mathbb{B}} \sum_{X_j, X_k \in \mathbf{B}_i} D(X_j, X_k) \tag{3.18}$$

where $D(X_i, X_j)$ is any distance measure between the joint distribution $P(X_i, X_j)$ and the product of the marginal distributions $P(X_i)P(X_j)$.

$$\psi(\mathbf{C}) = \sum_{i=1}^{p} \frac{1}{|\mathbf{X} \setminus \mathbf{C}_{i-1}|} \sum_{X \in \mathbf{X} \setminus \mathbf{C}_{i-1}} D(C_i, X) \tag{3.19}$$

where $(C_1, \ldots, C_p)$ is a user-defined order on variables in $\mathbf{C}$, $\mathbf{C}_i = \{C_1, \ldots, C_i\}$ and $C_0 = \emptyset$. We use the Hellinger distance, which is a symmetric measure to compute $D(X_i, X_j)$. Formally, this distance is given by:

$$D(X_i, X_j) = \frac{1}{\sqrt{2}} \sqrt{\sum_{\overline{x}_i, \overline{x}_j} \left(\sqrt{P(\overline{x}_i, \overline{x}_j)} - \sqrt{P(\overline{x}_i)P(\overline{x}_j)} \right)^2}$$

$D(X_i, X_j)$ measures the statistical dependence (correlation) between variables. Higher values indicate that the variables are statistically dependent while smaller values indicate that the variables are statistically independent. Notice that in order to compute $D(C_i, C_j)$, we need to know the 1-variable and 2-variable marginals. Their exact values are clearly not available and therefore we propose to estimate them from the generated samples.

As mentioned above, since we choose the collapsed variables before constructing the blocks, we have to penalize the feasible solutions that are likely to yield small blocks. We impose this penalty by using a hard constraint. The hard constraint disallows all feasible solutions $\mathbf{C}$ such that eliminating all variables in $\mathbf{C}$ along the ordering $(C_1, \ldots, C_p)$ adds more than γ edges to the primal graph. Thus, γ controls the relative importance of blocking versus collapsing. When γ is infinite or sufficiently large, the optimal solution to the objective function for collapsing is further refined to construct the blocks. On the other hand, when γ is small, a suboptimal solution to the objective function for collapsing, which can in turn enable higher quality blocking, is refined to construct the blocks.

3.2.3 Dynamic Blocked-Collapsed Gibbs Sampling

Although splitting the multi-objective optimization problem into two single objective optimization problems makes it comparatively easier to handle, it turns out that the resulting

Algorithm 3: Greedy-Collapse

Input: A PGM $\mathcal{M} = \langle \mathbf{X}, \Phi \rangle$, Integers α, and γ
Output: The collapsed PGM $\mathcal{M}_{\mathbf{X}/\mathbf{C}}$ obtained by eliminating $\mathbf{C}$ from $\mathcal{M}$

$E = 0$; $\mathbf{C} = \emptyset$;
repeat
  // Let $\mathcal{G}$ be the primal graph associated with $\mathcal{M}$
  Compute the value of the heuristic evaluation function for each vertex in $\mathcal{G}$ (see Eq. (3.21));
  Select a variable X with the maximum heuristic value such that the degree $deg(X, \mathcal{G}) \leq \alpha$ where $deg(X, \mathcal{G})$ is the degree of X in $\mathcal{G}$;
  // Let $E(X, \mathcal{G})$ be the number of new edges added to $\mathcal{G}$ by forming a clique over neighbors of X
  $E = E + E(X, \mathcal{G})$;
  Eliminate X from $\mathcal{M}$;
  $\mathbf{C} = \mathbf{C} \cup \{X\}$;
until *all vertices in* $\mathcal{G}$ *have degree larger than* α *or* $E > \gamma$;
return $\mathcal{M}$;

single objective optimization problems are $\mathcal{NP}$-hard. For instance, the problem of computing the set of collapsed variables includes the $\mathcal{NP}$-hard problem of computing the (weighted) treewidth (cf. (Arnborg et al., 1987)) as a special case. We therefore solve them using greedy methods.

Solving the optimization problem for Collapsing

Our greedy approach for computing the collapsed variables is given in Algorithm 3. The algorithm takes as input the PGM $\mathcal{M}$, two integer parameters α and γ which constrain the width of the collapsed variables (tractability constraints) and the total number of edges added to the primal graph after eliminating the collapsed variables (penalty) respectively, selects the collapsed variables, and outputs a PGM obtained by eliminating the collapsed variables.

Algorithm 3 heuristically selects variables one by one for collapsing until no variables can be selected because they will violate either the tractability constraints or the (penalty)

constraint on the total number of edges added. For maximizing the objective function, we want to collapse as many highly correlated variables as possible. Thus, a simple greedy approach would be to select, at each iteration, the variable X with the maximum correlation score $\psi(X)$ where $\psi(X)$ is given by

$$\psi(X) = \frac{1}{|\mathbf{X}|} \sum_{X_i \in \mathbf{X}} D(X, X_i) \tag{3.20}$$

However, this approach is problematic because a highly correlated variable may add several edges to the primal graph, potentially increasing its treewidth. This will in turn constrain future selections and may yield solutions which are far from optimal. In other words, at each iteration, we have to balance locally maximizing the scoring function with the number of edges added in order to have a better chance of hitting the optimum or getting close to it. We therefore use the following heuristic evaluation function to evaluate the various choices:

$$\chi(X) = \psi(X) + \left(\frac{\binom{\alpha}{2} - E(X, \mathcal{G})}{\binom{\alpha}{2}} \right) \tag{3.21}$$

where $\psi(X)$ is defined in Eq. (3.20) and $E(X, \mathcal{G})$ is the number of new edges that will be added to $\mathcal{G}$ by forming a clique over X. Note that since the maximum degree of any eliminated variable is bounded by α, the maximum number of edges that can be added is bounded by $\binom{\alpha}{2}$. Therefore, the quantity in the brackets in Eq. (3.21) lies between 0 and 1 and high values for this quantity are desirable since very few edges will be added by eliminating the particular variable ($\psi(X)$ also lies between 0 and 1 and high values for it are desirable too).

Solving the optimization problem for Blocking

Algorithm 4 presents the pseudo-code for our greedy approach for constructing the blocks. The algorithm takes as input a PGM $\mathcal{M}$ and an integer parameter β which bounds the treewidth of the primal graph of $\mathcal{M}$ projected on each block, and outputs a partitioning of

Algorithm 4: Greedy-Block

Input: A PGM $\mathcal{M} = \langle \mathbf{X}, \Phi \rangle$ and Integer β
Output: A partition of $\mathbf{X}$ denoted by $\mathbb{B}$

Initialize $\mathbb{B} = \{\{X\} | X \in \mathbf{X}\}$ (each block contains just one variable);
repeat
    `// Let` $\mathbb{B}_{i,j}$ `denote the partitioning formed from` $\mathbb{B}$ `by merging two blocks` $\mathbf{B}_i, \mathbf{B}_j$ `in` $\mathbb{B}$
    Merge two blocks $\mathbf{B}_i$ and $\mathbf{B}_j$ in $\mathbb{B}$ such that:
    1. they are in the Markov blanket of each other,
    2. $tw(\mathcal{G}(\mathbf{B}_i \cup \mathbf{B}_j)) \leq \beta$
    3. there does not exist another pair $\mathbf{B}_k$, $\mathbf{B}_m$ in $\mathbb{B}$ which satisfies the above two constraints and $\omega(\mathbb{B}_{k,m}) > \omega(\mathbb{B}_{i,j})$
until $\forall \mathbf{B}_i, \mathbf{B}_j \in \mathbb{B},\ tw(\mathcal{G}(\mathbf{B}_i \cup \mathbf{B}_j)) > \beta$;
return $\mathbb{B}$;

the variables of $\mathcal{M}$ into blocks. The algorithm begins by having $|\mathbf{X}|$ blocks, each containing just one variable. Then it greedily merges two blocks such that they will yield the maximum increase in the score $\omega(\mathbb{B})$ under the constraint that the treewidth of the merged block is bounded by β. (Note that computing the treewidth is $\mathcal{NP}$-hard (Arnborg et al., 1987) and therefore in our implementation we use the min-fill algorithm to compute an upper bound on it.) To guard against merging blocks which are far away from each other in the primal graph (and thus likely to be statistically independent), we merge two blocks only if they are in the Markov blanket of each other.

Dynamic Blocked Collapsed Gibbs sampling

Next, we describe how to use the greedy blocking and collapsing algorithms within a Gibbs sampler, yielding an advanced sampling technique. Our proposed method is summarized in Algorithm 5. The algorithm takes as input a PGM $\mathcal{M}$, parameters α, β and γ for performing blocking and collapsing, and two integers T and M which specify the sample size

Algorithm 5: Dynamic Blocked-Collapsed Sampling

Input: A PGM $\mathcal{M} = \langle \mathbf{X}, \Phi \rangle$; integers T, M; integers α, β and γ
Output: An estimate of marginal probabilities for all $X \in \mathbf{X}$

```
Initialize all 1-variable P(x̄_i) and 2-variable marginals P(x̄_i, x̄_j) to zero;
for t = 1 to T do
    M_{X\C} = Greedy-Collapse(M,α,γ);
    B = Greedy-Block(M_{X\C},β);
    Generate M samples from M_{X\C} using Blocked Gibbs sampling with B as blocks;
    Update all 1-variable P(x̄_i) and 2-variable marginals P(x̄_i, x̄_j) using the
    Rao-Blackwell estimator (see Eq. (2.9)).
return P(x̄_i) for all variable-value combinations.
```

and the interval at which the statistics are updated. At termination, the algorithm outputs an estimate of all 1-variable marginal probabilities.

The algorithm maintains an estimate of 1-variable and 2-variable marginals. The 2-variable marginals are used for computing the scoring functions. At each iteration, given a k-way partitioning of the variables into blocked and collapsed variables, denoted by $\mathbb{B}$ and $\mathbf{C}$ respectively, the algorithm generates M samples via blocked Gibbs sampling over $\mathcal{M}_{\mathbf{X}\setminus\mathbf{C}}$. After every M samples, the algorithm updates the blocks and collapsed variables using the greedy procedures outlined in the previous two subsections. The 1-variable and 2-variable marginals are updated using the Rao-Blackwell estimator (Eq. (2.9)).

Next, we describe how to update the 1-variable marginals (2-variable marginals can be updated analogously). At each iteration t where $t \in \{1, T\}$, let $\{\overline{\mathbf{x}}^{(i,t)}\}_{i=1}^{M}$ be the set of M samples generated via Blocked Gibbs sampling and let $\widehat{P}_t(\overline{x})$ denote the estimate of $P(X = x)$ at iteration t. Then $\widehat{P}_t(\overline{x})$ is given by:

$$\widehat{P}_t(\overline{x}) = \frac{(t-1)\widehat{P}_{t-1}(\overline{x}) + Q_t(\overline{x})}{t} \tag{3.22}$$

where $Q_t(\overline{x})$ is computed as follows. If $X \in \mathbf{C}$ is a collapsed variable, then without loss of generality, let $\mathbf{B}_k$ denote the largest block in $\mathbb{B}$. Similarly, If X is a blocked variable, then without loss of generality, let $\mathbf{B}_k$ denote the block in $\mathbb{B}$ in which X is present. Let $\overline{\mathbf{x}}_{-k}^{(i,t)}$

denote the projection of $\overline{\mathbf{x}}^{(i,t)}$ on all variables in $\mathbb{B} \setminus \mathbf{B}_k$. Then Q_t is given as follows:

$$Q_t(\overline{x}) = \frac{1}{M} \sum_{i=1}^{M} P(\overline{x}|\overline{\mathbf{x}}_{-k}^{(i,t)}) \tag{3.23}$$

To compute $P(\overline{x}|\overline{\mathbf{x}}_{-k}^{(i,t)})$ we have to marginalize out all variables in $\mathbf{B}_k \cup \mathbf{C} \setminus \{X\}$. Computing this is tractable because according to our assumptions marginalizing out $\mathbf{C}$ is tractable. After marginalizing out $\mathbf{C}$, marginalizing out $\mathbf{B}_k$ is tractable because its treewidth is bounded by β.

Note that when the correlation statistics are not available, i.e., when $t = 0$, the blocked and collapsed variables are computed by consulting the primal graph of the PGM. Thus, the blocks are constructed by randomly merging variables which are in the Markov blanket of each other; ties broken randomly. Similarly, the collapsed variables are selected along a constrained min-fill ordering (constrained by α). Thus, if we use a time bound, namely we stop sampling after the time bound has expired, and set M to be sufficiently large, Algorithm 5 is equivalent to a static graph-based blocked-collapsed Gibbs sampling procedure.

Convergence

Clearly, the sampler in Algorithm 5 is *non-Markovian*. That is, since we change the blocks and collapsed variables dynamically, the state of the Markov chain is not *memoryless* and depends upon the previous samples that were generated from the chain. Thus, it is an instance of what is referred to as *adaptive MCMC* techniques (Roberts and Rosenthal, 2009). Analyzing the convergence such adaptive samplers is known to be a hard problem and is an area of active research in statistics. In fact it is known that some well-known adaptive samplers do not converge (cf. (Roberts and Rosenthal, 2007)). Gonzalez et al. (Gonzalez et al., 2011) prove that if a Markov chain depends upon the state of a sampler, then the chain is not guaranteed to be ergodic even in a finite state space where each variable is sampled

infinitely often. The solution commonly used to ensure convergence is the concept of *diminishing adaptation* (Roberts and Rosenthal, 2007). This means that the rate of adaptation must tend towards 0 as the samples tend towards ∞. In our case, this is straightforward to achieve since M controls the rate at which the blocks and collapsed variables are updated. By using a policy in which M is progressively increased as t increases, using the results from adaptive MCMC (Roberts and Rosenthal, 2007), it directly follows that the estimates output by Algorithm 5 will converge to $P(\overline{x}_i)$ as T tends to infinity.

3.2.4 Related Work

A number of earlier papers have investigated blocking and collapsing in the context of PGMs. Table 3.1 summarizes some notable ones and how they are related to our work. Blocked Gibbs sampling was first proposed by Jensen et al. (Jensen et al., 1993). The key idea in their algorithm was to create a "single block" by removing variables one by one from the primal graph until the treewidth of the remaining network is bounded by a constant and then sample this block using the junction tree algorithm. Unlike Jensen et al.'s work, we allow multiple blocks, combine collapsing with blocking and use the Rao-Blackwell estimator for computing the marginals (Jensen et al. use the histogram estimator).

Our algorithm is related to the Rao-Blackwellised blocked Gibbs sampling (RBBG) algorithm proposed by Hamze and de Freitas (Hamze and de Freitas, 2004). RBBG operates by dividing the network into two tractable tree-structured blocks and then performing Rao-Blackwellised estimation in each block. Unlike our algorithm, RBBG is applicable to grid Markov networks only. Also, unlike our algorithm, RBBG does not use multiple blocks and does not update the blocks dynamically. Moreover, RBBG does not use collapsing.

Another related work is that of Bidyuk and Dechter (Bidyuk and Dechter, 2007) in which the authors propose a collapsed Gibbs sampling algorithm. The key idea in their work is similar to Jensen et al.: remove variables one by one until the treewidth is bounded by a

Table 3.1. Comparing prior work and our work along different dimensions. Blocking (1: uses a single block, 2: uses 2 blocks, M: uses multiple blocks, N: not blocked), collapsing (Y/N), Rao-Blackwell Estimation (RB) (Y/N) and Dynamic (N: Static,Y: Dynamic).

Algorithm	Blocked	Collapsed	RB	Dynamic
Geman & Geman (Geman and Geman, 1984)	N	N	N	N
Jensen et al. (Jensen et al., 1993)	1	N	N	N
Bidyuk & Dechter (Bidyuk and Dechter, 2007)	N	Y	Y	N
Hamze & de Freitas (Hamze and de Freitas, 2004)	2	N	Y	N
Paskin (Paskin, 2003)	M	Y	Y	N
Our work	M	Y	Y	Y

constant w (the removed variables form a w-cutset). However, unlike Jensen et al., they use the junction tree to sample the w-cutset variables. Formally, let $\mathbf{W}$ be the set of w-cutset variables and $\mathbf{V} = \mathbf{X} \setminus \mathbf{W}$ be the set of remaining variables. Then, the junction tree is used to compute the distribution $P(W_i|\mathbf{w}_{-i})$ and sample from it. Effectively, the set $\mathbf{V}$ is always collapsed out. A key drawback of this algorithm is that the junction tree algorithm must be run from scratch for sampling each w-cutset variable and as a result the algorithm can be quite slow. In our approach, we save time by marginalizing out a subset of variables before running the junction tree algorithm (i.e., marginalization is a pre-processing step before sampling). Also, unlike our work, the Bidyuk and Dechter algorithm does not use blocking and is not dynamic.

The sample propagation algorithm of Mark Paskin (Paskin, 2003) is the only blocked-collapsed algorithm for PGMs that we are aware of. The algorithm integrates sampling with message passing in a junction tree. The key idea is to walk the clusters of a junction tree, sampling some of the current cluster's variables and then passing a message to one of its neighbors. The algorithm designates a subset of variables for sampling and marginalizes out the remaining variables by performing message passing over the junction. In that sense, sample propagation is similar to (but more efficient than) Bidyuk and Dechter's algorithm. The only difference is that variables within each cluster are sampled jointly (or blocked) if

the cluster size is small enough or sampled using Metropolis-Hastings otherwise. Since the blocks in sample propagation are confined to the clusters of a junction tree, they can be much smaller than the blocks used in our algorithm. Also, this algorithm is not dynamic.

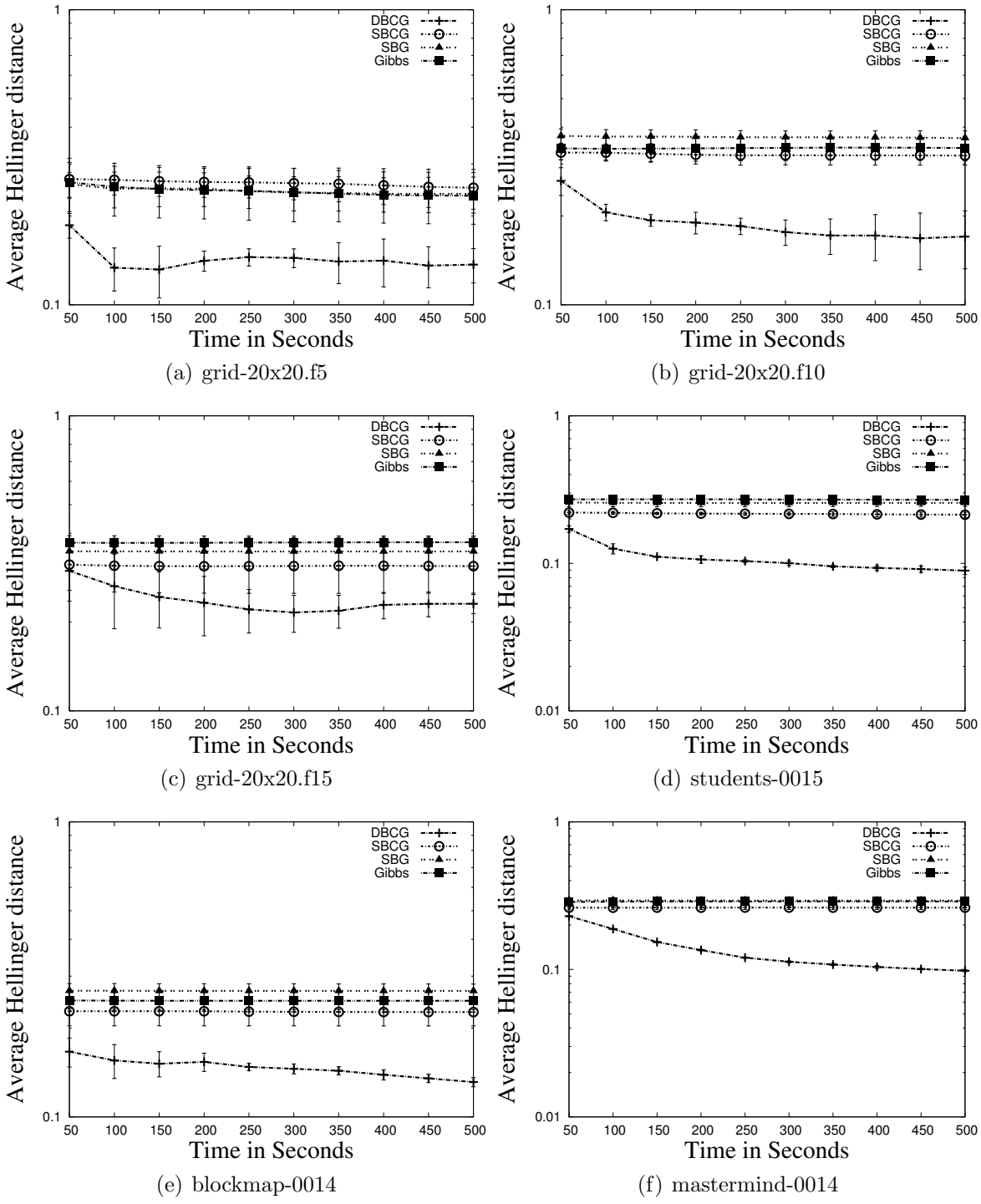
Finally, our work is related to parallel Gibbs sampling by Gonzalez et al. (Gonzalez et al., 2011) who use likelihood estimates to compute the blocks.

3.2.5 Experiments

In this section, we experimentally evaluate the performance of the following algorithms on several benchmark PGMs: (*a*) Naive Gibbs sampling (`Gibbs`); (*b*) Static Blocked Gibbs sampling (`SBG`); (*c*) Static blocked collapsed Gibbs sampling (`SBCG`); and (*d*) Dynamic blocked collapsed Gibbs sampling (`DBCG`) . `SBG` is similar to the algorithm of Hamze and de Freitas (Hamze and de Freitas, 2004) except that we allow multiple blocks and do not constrain the blocks to be tree structured. `SBCG` is an advanced version of Paskin's sample propagation algorithm (Paskin, 2003). We implemented `SBG` and `SBCG` by setting M to a sufficiently large value i.e., these methods consult only the primal graph of the PGM to choose the blocks and collapsed variables. To compute marginals, we use the Rao-Blackwell estimator in `SBG`, `SBCG` and `DBCG`, and the mixture estimator in `Gibbs`. In `DBCG` we set $\alpha = \beta = 8$, $\gamma = 50 \times \alpha$ and $M = 1000$ (progressively increasing it). We evaluate the impact of α, β and γ in the next sub-section.

Figures 3.7 and 3.8 illustrate our results. We see that `DBCG` is more accurate than all other algorithms on almost all the PGMs, often outperforming the competition by an order of magnitude.

Ising models. Figures 3.7(a)-(c) show the performance of various algorithms on three Ising models of size 20×20 with evidence on 5, 10 and 15 randomly selected nodes respectively. `DBCG` is the best algorithm on all three PGMs. `SBCG` performs better than the other two algorithms on grid20x20.f10 and grid20x20.f15 and its performance is almost similar to `SBG` and `Gibbs` on grid20x20.f5.



(a) grid-20x20.f5

(b) grid-20x20.f10

(c) grid-20x20.f15

(d) students-0015

(e) blockmap-0014

(f) mastermind-0014

Figure 3.7. Average Hellinger distance between the exact and the approximate 1-variable marginals plotted as a function of time. (a)-(c): Grids, (d)-(f): Relational

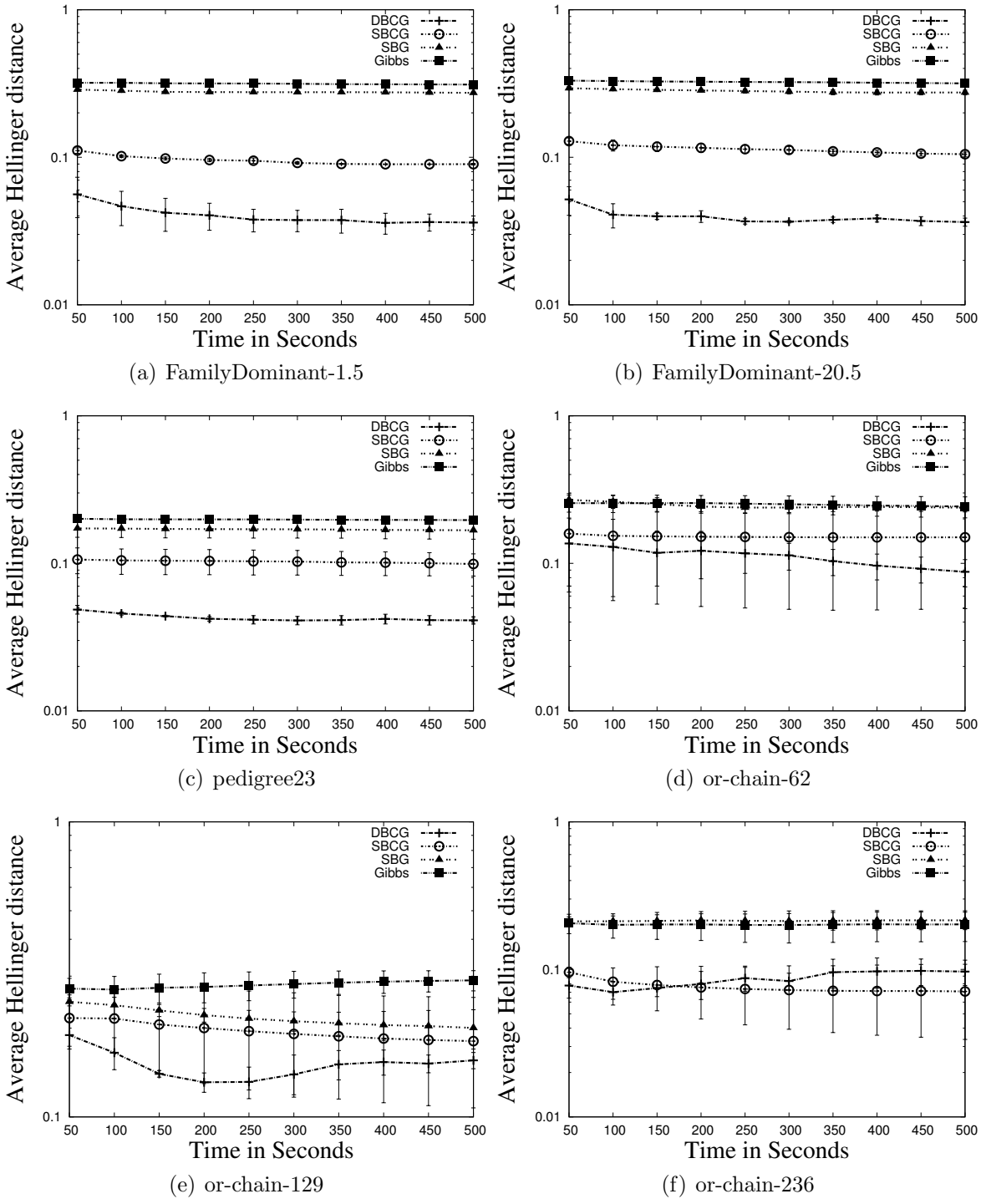
Relational PGMs are obtained by grounding statistical relational models which often have large number of correlated variables as well as deterministic dependencies. Our dynamic approach is beneficial on such models because it has the ability to learn correlations and adjust the partitions accordingly. We experimented with three relational PGMs available from the UAI-08 repository: students-0015, blockmap-0014 and mastermind-0014. Figures 3.7 (d)-(f) show the results. Again, we see that `DBCG` is the best performer followed by `SBCG`.

Linkage PGMs are used for performing genetic linkage analysis (Fishelson and Geiger, 2004). Figures 3.8 (a)-(c) show results on three linkage PGMs. Again, on all three PGMs, `DBCG` is the best performing algorithm and `SBCG` is the second best.

Promedas PGMs are noisy-OR medical diagnosis networks generated by the Promedas medical diagnosis system (Wemmenhove et al., 2007). The networks are two-layered bipartite graphs in which bottom layer has the symptoms and the top layer has the diseases. We experimented with three PGMs: or-chain-62, or-chain-129 and or-chain-236. Figures 3.8 (d)-(f) show the results. `DBCG` performs better than all other algorithms in two out of the three PGMs. On or-chain-236, `SBCG` is slightly better than `DBCG`, but has larger variance.

Impact of varying the parameters α and β

Figure 3.9 shows the impact of changing the parameters α and β on the performance of `DBCG`. Figures 3.9 (a)-(c) show the impact of increasing α with β set to a constant while Figures 3.9 (d)-(f) show the impact of increasing β with α set to a constant. We see that increasing α or β typically increases the accuracy and reduces the variance as a function of time. However, in some cases (e.g., Figure 3.9(c) and Figure 3.9(f)), we see that the accuracy goes down as we increase α and β, which indicates that there is a trade-off between blocking and collapsing. In summary, α and β help us explore the region between a completely collapsed and a completely blocked sampler, and in turn help us achieve the right balance between blocking and collapsing.



(a) FamilyDominant-1.5

(b) FamilyDominant-20.5

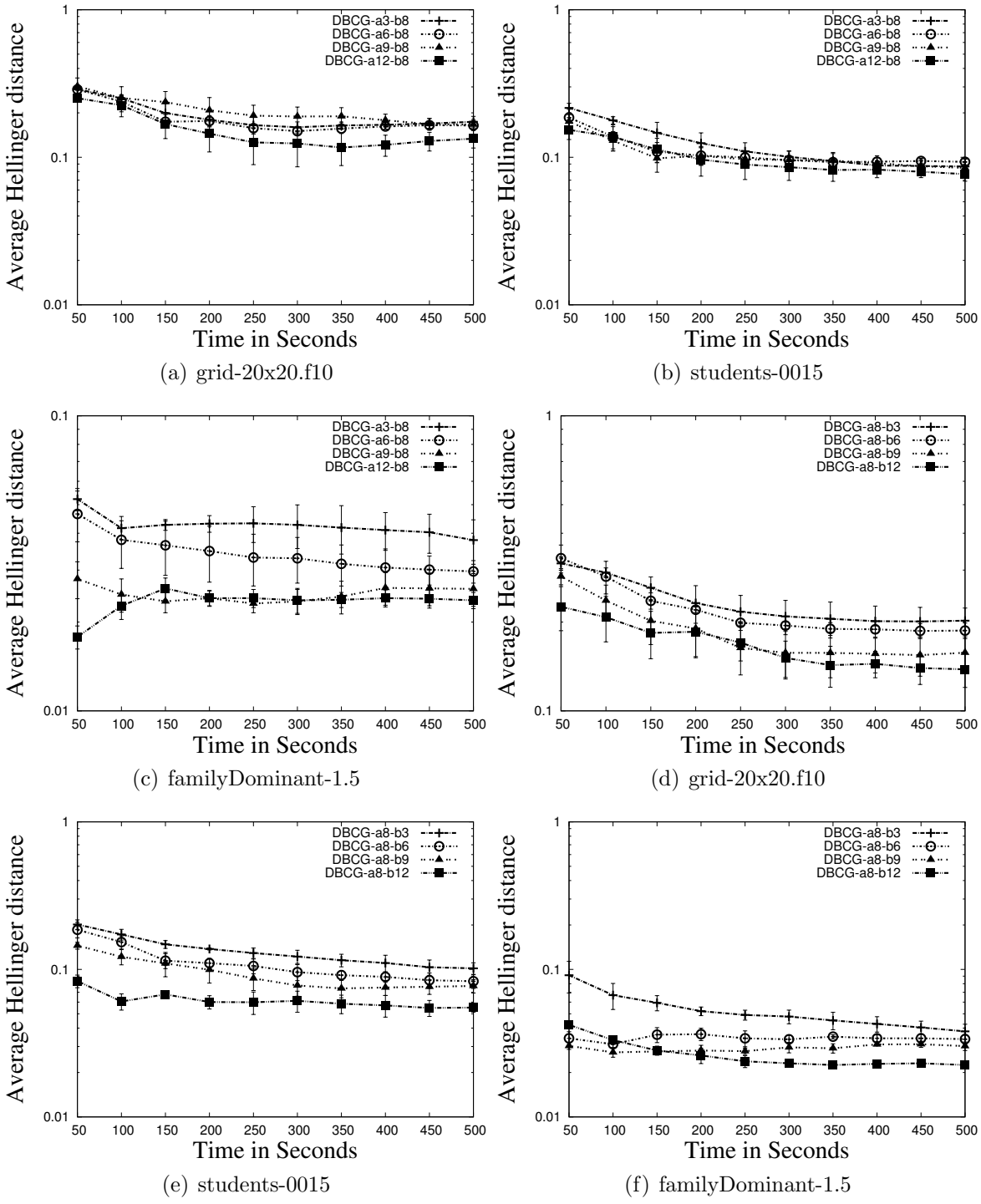
(c) pedigree23

(d) or-chain-62

(e) or-chain-129

(f) or-chain-236

Figure 3.8. Average Hellinger distance between the exact and the approximate 1-variable marginals plotted as a function of time. (a)-(c): Linkage, (d)-(f): Promedas.



(a) grid-20x20.f10 (b) students-0015

(c) familyDominant-1.5 (d) grid-20x20.f10

(e) students-0015 (f) familyDominant-1.5

Figure 3.9. Blocking vs. Collapsing tradeoff. (a)-(c): Impact of varying α with β set to a constant value. (d)-(f): Impact of varying β with α set to a constant value. We use $\gamma = 50 \times \alpha$. In all the plots, we plot the average Hellinger distance between the exact and the approximate 1-variable marginals as a function of time. The notation shown in the plots is as follows. DBCG-ax-by indicates that $\alpha = x$ and $\beta = y$.

3.3 Summary

Gibbs sampling performs poorly in PGMs that encode logical dependencies. In this chapter we presented two algorithms that improve the performance of Gibbs sampling in such PGMs. First, we introduced `GiSS`, a hybrid algorithm that combines Gibbs sampling with SampleSearch to perform inference in models with determinism. Determinism fractures the state space into multiple clusters and Gibbs sampling get struck in a local cluster converging to the wrong distribution. The main virtue of `GiSS` is that unlike Gibbs sampling which does not converge to the correct answers in presence of determinism, `GiSS` yields provably correct (asymptotically unbiased) estimates of various inference tasks. We proposed three schemes for correctly weighting the samples generated by `GiSS` and showed that they trade computational complexity with accuracy. We performed experiments on benchmark PGMs from several domains and found that `GiSS` is often better in terms of accuracy than state-of-the-art algorithms such as MC-SAT and SampleSearch.

Next, we improved the performance of Gibbs sampling in the presence of correlations. Specifically, we presented an approach that combines two widely used techniques that provably improve the convergence of Gibbs sampling, namely, blocking and collapsing. We showed that combining these two tractably and effectively is a hard problem and formulated it as a multi-objective optimization problem. We proposed a greedy algorithm to solve this problem which assumes access to correlations between all pairs of variables. Since the exact value of these correlations is not available, we proposed to estimate them from the generated samples, and update the greedy solution periodically. This yields a dynamic/adaptive blocked collapsed Gibbs sampling algorithm which iterates between two steps: partitioning and sampling. In the partitioning step, the algorithm uses the current estimate of correlations between variables to partition the variables in the PGM into blocked and collapsed subsets and constructs the collapsed PGM. In the sampling step, the algorithm uses the blocks constructed in the previous step to generate samples from the collapsed PGM. As

the dynamic sampler learns correlations, the mixing of the sampler is improved. Our experiments showed that this dynamic method has much better accuracy than using static blocks and collapsed variables.

CHAPTER 4

LIFTING SAMPLING BASED INFERENCE ALGORITHMS

Graphical model inference algorithms ignore the first-order structure of MLNs and treat it as a regular Markov network. Thus, they are *propositional* in the sense that they work on the ground representation of the MLN. Since the ground representation is typically several orders of magnitude larger than the first-order specification of the MLN, propositional inference techniques have limited scalability. On the other hand, *lifted inference* algorithms can be viewed as algorithms with "relational-awareness". That is, lifted inference algorithms exploit symmetries in the MLN's relational structure and work at the first-order level grounding the MLN only as needed. The central idea behind most lifted inference algorithms is the same, namely, identify and perform efficient inference over groups of symmetric variables instead of treating each variable as a separate entity, which is typically the assumption made by propositional inference algorithms. The challenge though is to identify such symmetries efficiently from first-order structure without consulting the ground representation. As is the case with propositional inference, algorithms that yield exact results, namely, exact lifted inference algorithms (e.g., FOVE (de Salvo Braz, 2007), PTP (Gogate and Domingos, 2011b), WFOMC (Van den Broeck et al., 2011), etc.) are typically computationally infeasible for several real-world MLNs. Therefore, approximate inference is the method-of-choice in these cases. In this chapter, we introduce two sampling based lifted approximate inference algorithms, namely, (i) Lifted Blocked Gibbs (LBG), which lifts an advanced variant of Gibbs sampling and (ii) Lifted Importance Sampling (LIS). Just like exact lifted inference, both LBG and LIS leverage relational structure for inference, however, they produce approximate results but with strong, provable convergence guarantees. At a high level, the central theme

in both LBG and LIS is the notion of sampling from a *lifted* state space instead of a propositional state space. That is, the sampling algorithm explores a state space where symmetric variables are grouped together. To find such groups efficiently while preserving correctness and convergence guarantees of the sampler, we exploit symmetries in the first-order representation of MLNs. Next, we describe the LBG and LIS algorithms in detail.

4.1 Lifted Blocked Gibbs

Gibbs sampling is an instance of the well-known *Markov Chain Monte Carlo* (MCMC) based sampling technique. Several earlier papers have attempted to exploit relational or first-order structure in various MCMC based sampling algorithms. Notable examples are MC-SAT (Poon et al., 2008), Metropolis-Hastings MCMC for Bayesian logic (BLOG)(Milch and Russell, 2006), typed MCMC (Liang et al., 2010) and orbital MCMC (Niepert, 2012). Unfortunately, none of the aforementioned techniques are truly lifted. In particular, they do not exploit first-order structure to the fullest extent. Upon close observation, in fact, lifting a generic MCMC technique is difficult because at each point, in order to ensure convergence to the desired stationary distribution, one has to maintain an assignment to all random variables in the distribution. For instance, if we wish to apply Gibbs sampling to MLNs, each iteration can sample exactly one ground atom and thus, it is necessary to maintain an assignment over every ground atom in the MLN. In other words, the Gibbs sampler implicitly works in a propositional state space. We circumvent these issues by lifting an advanced variant of Gibbs sampling called blocked Gibbs sampling and show that it is more amenable to lifted inference.

Blocking or blocked Gibbs sampling (Jensen et al., 1993; Liu et al., 1994) is an advanced technique that improves upon the Gibbs sampling algorithm by grouping variables (each group is called a block) and then jointly sampling all variables in the block. Blocking improves the mixing time and as a result improves both the accuracy and convergence of

Gibbs sampling (cf. (Liu, 2001)). The difficulty is that to jointly sample variables in a block, we need to compute the exact joint distribution over them. This is typically exponential in the treewidth of the ground Markov network projected on the block.

Our main idea in applying the blocking strategy to MLNs is the following. Instead of blocking over propositional variables, we partition the set of first-order atoms in the model into a set of disjoint *liftable* clusters, i.e., the MLN projected on each cluster can be solved tractably using exact lifted inference. Given such a set of liftable clusters, we show that Gibbs sampling is essentially a message passing algorithm over the graph that is formed by connecting clusters where an edge between clusters $\mathbf{C}_i$ and $\mathbf{C}_j$ indicates that at least one atom in $\mathbf{C}_i$ and one atom in $\mathbf{C}_j$ are in the Markov blanket of each other. A message from $\mathbf{C}_i$ to $\mathbf{C}_j$ in this case is the current state (truth assignment) of all ground atoms in $\mathbf{C}_i$ that are in the Markov blanket of some atom in $\mathbf{C}_j$. We then show how to *lift* each such message while maintaining invariance of the MLN distribution by exploiting symmetries that can be identified from the first-order structure. Surprisingly, we show that in some cases more blocking can in fact reduce the complexity of inference. This is a counter-intuitive result since, in propositional inference, more blocking always increases inference complexity. However, with lifted inference, in some cases more blocking preserves more symmetries within a block which in-turn reduces the complexity of inference.

We present experimental results comparing the performance of lifted blocked Gibbs sampling with (propositional) blocked Gibbs sampling, MC-SAT (Poon and Domingos, 2006; Poon et al., 2008) and Lifted BP (Singla and Domingos, 2008) on various MLN benchmarks. Our experiments show that lifted Gibbs sampling is superior to blocked Gibbs sampling and MC-SAT in terms of convergence, accuracy and scalability. It is also more accurate than lifted BP on some instances.

4.1.1 Our Approach

We illustrate the key ideas in our approach using an example MLN having two weighted formulas: $\mathtt{R}(x,y) \vee \mathtt{S}(y,z), w_1$ and $\mathtt{S}(y,z) \vee \mathtt{T}(z,u), w_2$. Note that the problem of computing the partition function of this MLN for arbitrary domain sizes is non-trivial; it cannot be polynomially solved using existing exact lifted approaches such as PTP (Gogate and Domingos, 2011b), WFOMC (Van den Broeck et al., 2011) and lifted VE (de Salvo Braz, 2007).

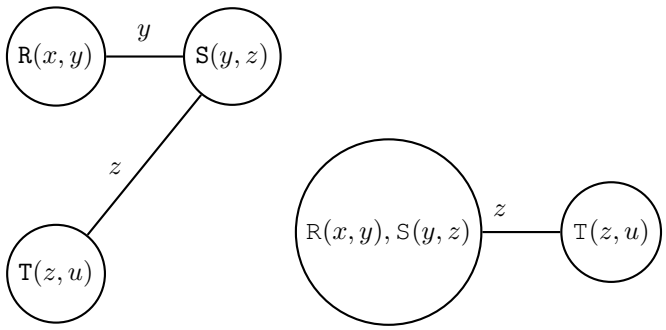
Our main idea is to partition the set of first-order atoms into disjoint blocks (clusters) such that exact lifted inference using PTP is tractable in each cluster. We then sample all the ground atoms in a cluster jointly, given assignments to all other clusters. Recall that PTP is tractable if we can apply its two rules, namely, the *power rule* and the *generalized binomial rule* (Section 2.2.3) recursively or at least until we can reduce the MLN to a size that is "small-enough" such that it can be solved by any exact inference algorithm (e.g. variable elimination). For this, the treewidth of the remaining ground network of the MLN after applying the PTP operations should be bounded by a constant.

Now, let us apply the clustering idea to our example MLN. Let us put each first-order atom in a cluster by itself, namely we have three blocks/clusters: $\mathtt{R}(x,y)$, $\mathtt{S}(y,z)$ and $\mathtt{T}(z,u)$ (see Figure 4.1(a)). Note that each (first-order) cluster represents all groundings of all atoms in that cluster. To perform Gibbs sampling over this clustering, we pick any cluster and sample all the atoms in that cluster given assignments to all other clusters. Thus, in the above example, we need to compute three conditional distributions: $P(\mathtt{R}(x,y)|\bar{\mathtt{S}}(y,z), \bar{\mathtt{T}}(z,u))$ (where $\bar{\mathtt{T}}(z,u)$ denotes an assignment to all groundings of $\mathtt{T}$), $P(\mathtt{S}(y,z)|\bar{\mathtt{R}}(x,y), \bar{\mathtt{T}}(z,u))$ and $P(\mathtt{T}(z,u)|\bar{\mathtt{R}}(x,y),\ \bar{\mathtt{S}}(y,z))$ where $\bar{\mathtt{R}}(x,y)$ denotes a truth assignment to all possible groundings of $\mathtt{R}$. Let the domain size of each variable be d. Naively, given an assignment to all other atoms not in the cluster, we will need $O(2^{d^2})$ time and space for computing and specifying the joint distribution at each cluster. This is because there are d^2 ground atoms

associated with each cluster. However, PTP has a smaller complexity by taking advantage of conditional independence. Specifically, consider the application of PTP to compute $P(\mathtt{S}(y,z)|\bar{\mathtt{R}}(x,y),\bar{\mathtt{T}}(z,u))$. When we instantiate the evidence $\bar{\mathtt{R}}(x,y),\bar{\mathtt{T}}(z,u)$ and normalize the MLN, we obtain $2\times d^3$ ground formulas. Each ground formula is either empty or satisfied or has exactly one unit ground clause of the form $\mathtt{S}(X,Y)$. The normal MLN now has d^2 atoms and conditioning on each can be done independently in constant time. Therefore, the overall complexity is equal to $O(d^3)$. Note that the complexity of sampling all variables using propositional Gibbs sampling is also $O(d^3)$ because grounding the entire MLN produces $2\times d^3$ ground formulas.

Now, let us consider an alternative clustering in which we have two clusters as shown in Figure 4.1(b). Intuitively, this clustering is likely to yield better accuracy than the previous one because more atoms will be sampled jointly. For instance, consider the degenerate case where every predicate is in a single cluster, this is the ideal blocking because it produces exact results. Also, it is easy to see that, in graphical model inference, more blocking will increase the complexity of exact inference, since implicitly, this adds more variables and edges into the graphical model projected on a block. However, since lifted inference works on the relational representation and not the ground Markov network, it turns out that, counter-intuitively, as we show next, Clustering 2 will yield a blocked sampler having smaller complexity than the one based on Clustering 1.

To perform blocked Gibbs sampling over Clustering 2, we need to compute two distributions $P(\mathtt{R}(x,y),\mathtt{S}(y,z)|\bar{\mathtt{T}}(z,u))$, $P(\mathtt{T}(z,u)|\bar{\mathtt{R}}(x,y),\bar{\mathtt{S}}(y,z))$. Let us see how PTP will compute $P(\mathtt{R}(x,y),\mathtt{S}(y,z)|\bar{\mathtt{T}}(z,u))$. If we instantiate all groundings of $\mathtt{T}$, we get the following reduced, normal MLN $\{\mathtt{R}(x,y)\vee\mathtt{S}(y,Z_i),w_1\}_{i=1}^d$ and $\{\mathtt{S}(y,Z_i),k_iw_2\}_{i=1}^d$ where $Z_i\in\Delta_z$ and k_i is the number of False groundings of $\mathtt{T}(y,Z_i)$. This MLN contains a decomposer y. PTP will now apply the decomposer rule, yielding formulas of the form $\{\mathtt{R}(x,Y)\vee\mathtt{S}(Y,Z_i),w_1\}_{i=1}^d$ and $\{\mathtt{S}(Y,Z_i),k_iw_2\}_{i=1}^d$ where $Y\in\Delta_y$. $\mathtt{R}(x,Y)$ is a singleton atom and therefore applying



(a) Clustering1 (b) Clustering2

Figure 4.1. Two possible clusterings for lifted blocked Gibbs sampling on the example MLN having two weighted formulas. : $\mathtt{R}(x,y) \vee \mathtt{S}(y,z), w_1$ and $\mathtt{S}(y,z) \vee \mathtt{T}(z,u), w_2$.

the generalized binomial rule, we will get $d+1$ reduced MLNs, each containing d atoms of the form $\{\mathtt{S}(Y, Z_i)\}_{i=1}^{d}$. These atoms are conditionally independent of each other and a distribution over them can be computed in $O(d)$ time. Thus, the complexity of computing $P(\mathtt{R}(x,y), \mathtt{S}(y,z) | \bar{\mathtt{T}}(z,u))$ is $O(d^2)$. Samples for $\mathtt{R}$ and $\mathtt{S}$ can be generated from $P(\mathtt{R}(x,y), \mathtt{S}(y,z) | \bar{\mathtt{T}}(z,u))$ in $O(d^2)$ time as well. Notice that $P(\mathtt{T}(z,u) | \bar{\mathtt{R}}(x,y), \bar{\mathtt{S}}(y,z)) = P(\mathtt{T}(z,u) | \bar{\mathtt{S}}(y,z))$ because $\mathtt{R}$ is not in the Markov blanket of $\mathtt{T}$. This distribution can also be computed in $O(d^2)$ time. Therefore, the complexity of sampling all atoms using the clustering shown in Figure 4.1(b) is $O(d^2)$.

Space Complexity: For Clustering 2, notice that to compute the conditional distribution $P(\mathtt{R}(x,y), \mathtt{S}(y,z) | \bar{\mathtt{T}}(z,u))$, we only need to know how many groundings of $\mathtt{T}(Z_i, u)$ are True in $\bar{\mathtt{T}}(z,u)$ for all $Z_i \in \Delta_z$. Cluster $\mathtt{T}(z,u)$ can share this information with its neighbor using only $O(d)$ space. Similarly, to compute $P(\mathtt{T}(z,u) | \bar{\mathtt{S}}(y,z))$ we only need to know how many groundings of $\mathtt{S}(y, Z_i)$ are True in $\bar{\mathtt{S}}(y,z)$ for all $Z_i \in \Delta_z$. This requires $O(d)$ space and thus the overall space complexity of Clustering 2 is $O(d)$. On the other hand, the space complexity of Gibbs sampling over Clustering 1 is $O(d^2)$.

4.1.2 PTP-Tree

Before, we formalize the example presented in the previous section, we define a canonical structure from which we obtain *lifted* samples. Specifically, just as advanced search based inference algorithms such as AND/OR search (Dechter and Mateescu, 2007) represent the probability distribution as a conditioning tree/graph by taking advantage of independent sub-problems, PTP can be visualized as algorithm that explores a *lifted conditioning tree* during its execution. We refer to this tree as a *PTP-tree*

Each node in PTP-tree represents a lifted operation of PTP (Section 2.2.3). Thus, a PTP-tree can be viewed as a lifted representation of the input MLN's joint distribution. However, note that we have assumed a tree structure for ease of exposition. Using extensions of PTP, we can easily compile a PTP-tree into a more compact graph through techniques such as caching that exploit repeated sub-steps. However, for the purpose of this chapter, we strictly assume a tree structure.

Formally, a PTP-tree $\mathcal{T}$ for an MLN $\mathcal{M}$ has three distinct types of nodes, each of which is defined as below.

1. **Power Node** represents the application of the power rule in PTP. A power node $\mathcal{W}$ has exactly one child denoted by $C(\mathcal{W}, 1)$. The distribution represented at $\mathcal{W}$ denoted by $P_{\mathcal{W}}$ is the product of $D(\mathbf{d})$ independent and identical distributions, where $\mathbf{d}$ is the decomposer and $D(\mathbf{d})$ is size of the decomposer.

$$P_{\mathcal{W}} = \prod_{j=1}^{D(\mathbf{d})} P_{C(\mathcal{W},1)}$$

2. **Decomposition Node** represents the operation where an MLN is decomposed into disjoint parts. The distribution at a decomposition node $\mathcal{D}$ is the product of the K independent distributions, where K is the number of children of $\mathcal{D}$.

$$P_{\mathcal{D}} = \prod_{i=1}^{K} P_{C(\mathcal{D},i)}$$

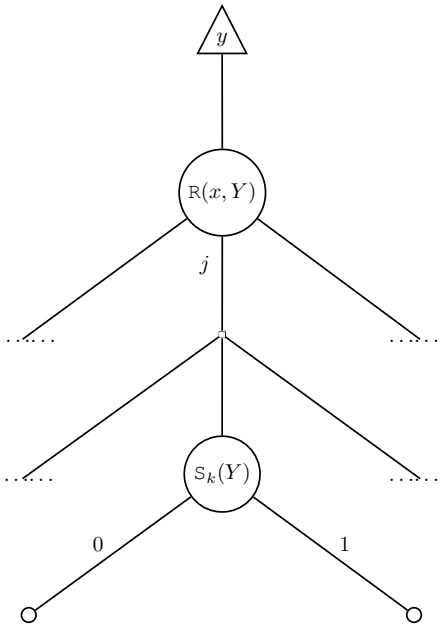


Figure 4.2. The PTP tree for the MLN which contains formulas $\{\mathtt{R}(x,y) \vee \mathtt{S}_i(y), w\}_{i=1}^{K}$ and $\{\mathtt{S}_i(y), w_i'\}_{i=1}^{K}$. The triangle represents a power node where y is the decomposer variable. The circle represents a conditioning node corresponding to an atom. For the atom $\mathtt{R}(x,Y)$ which represents Δ_x groundings, the conditioning node has $\Delta_x + 1$ children, where the j-th child represents conditioning over all groundings of $\mathtt{R}(x,Y)$ by setting any j groundings to true and $\Delta_x - j$ groundings to false. After conditioning on $\mathtt{R}(x,Y)$, we can split the remaining MLN into k independent parts, where each part contains a single atom of the form $\mathtt{S}_k(Y)$. This is represented as a decomposition node by the small square. We then condition on each of these atoms as represented by the conditioning node.

3. **Binomial Node** represents the application of the generalized binomial rule in PTP. The distribution at a node that conditions on all groundings of $\mathtt{R}(x)$ denoted by $P_{\mathtt{R}}$ is defined in a lifted manner as follows.

$$P_{\mathtt{R}}(i) = \frac{\binom{|\Delta_x|}{i} Z(P_{C(\mathtt{R},i+1)})}{\sum_{j=0}^{|\Delta_x|} \binom{|\Delta_x|}{j} Z(P_{C(\mathtt{R},j+1)})}$$

where $P_{\mathtt{R}}(i)$ denotes the probability that any i groundings of $\mathtt{R}$ are assigned true.

An example PTP tree is illustrated in Figure 4.2. To obtain lifted samples from the joint distribution represented by a PTP-tree, we simply visit every node of the PTP-tree and sample the distribution corresponding to that node.

4.1.3 Lifted Blocked Gibbs

We now formalize the lifted blocked Gibbs sampling (LBG) algorithm. We assume that the input to our algorithm is a normal MLN (Section 2.1.4). First, we begin with some required definitions.

Definition 10 (Gibbs cluster graph)**.** *Given an MLN $\mathcal{M}$, a* Gibbs cluster graph*, G, is an undirected graph where the set of nodes represent a partition of the atoms of $\mathcal{M}$ and an edge between two nodes $\mathbf{C}_i$ and $\mathbf{C}_j$ exists iff there exist two ground atoms, one in $\mathbf{C}_i$ and the other in $\mathbf{C}_j$ which are in the Markov blanket of each other.*

Definition 11 (Ground Messages)**.** *Given a Gibbs cluster graph G, corresponding to each edge ($\mathbf{C}_i$, $\mathbf{C}_j$), there exists two ground messages defined as follows. $GM(\mathbf{C}_i, \mathbf{C}_j)$ is an assignment to each ground atom in $\mathbf{C}_i$ that is in the Markov blanket of some ground atom in $\mathbf{C}_j$ and $GM(\mathbf{C}_j, \mathbf{C}_i)$ is an assignment to each ground atom in $\mathbf{C}_j$ that is in the Markov blanket of some ground atom in $\mathbf{C}_i$*

Example 6. *The two graphs shown in Figure 4.1 (a) and (b) are both two different Gibbs cluster graphs. Consider Figure 4.1 (a). The message $GM(\mathtt{S}, \mathtt{R})$ contains an assignment to all groundings corresponding to $\mathtt{S}$ and the message $GM(\mathtt{R}, \mathtt{S})$ contains an assignment to all groundings of $\mathtt{R}$.*

Algorithm 6 illustrates LBG. Given a normal MLN $\mathcal{M}$ and its Gibbs cluster graph G, LBG computes the marginal probabilities for a given set of query atoms. To start, we initialize all the ground messages in G randomly. In each iteration, t, we choose a random cluster in G, say $\mathbf{C}_i$ and instantiate $\mathcal{M}$ with every incoming message to $\mathbf{C}_i$, i.e., $\bigcup_{\mathbf{C}_j} GM(\mathbf{C}_j, \mathbf{C}_i)$ to obtain a reduced-MLN, $\mathcal{M}^{(t)}$. We then sample $\mathbf{C}_i$ from $\mathcal{M}^{(t)}$ in a lifted manner. Specifically, we first run PTP on $\mathcal{M}^{(t)}$ to construct its PTP-tree, $\mathcal{T}^{(t)}$. We then sample $\mathbf{C}_i$ by traversing $\mathcal{T}^{(t)}$ top-down and at each node, we sample the distribution associated with that node as

Algorithm 6: LBG

Input: A normal MLN $\mathcal{M}$, a Gibbs cluster graph G, integers T, b and query atoms $\mathbf{Q}$
Output: $\widehat{P}_{\mathcal{M}}(Q)$, where $Q \in \mathbf{Q}$

```
begin
    for each edge (C_i, C_j) ∈ G do
        Randomly initialize GM(C_i, C_j) and GM(C_j, C_i)
    for t = 1 to T do
        Choose a cluster C_i uniformly at random
        M^(t) = Instantiate ∪_{C'} GM(C', C_i) in M
        T^(t) = PTP(M^(t))
        Sample C_i from T^(t)
        Update all outgoing messages from C_i
        if t ≥ b then
            Update P̂_M(Q) for all query atoms in C_i
```

specified in section 4.1.2. Using the sampled value for $\mathbf{C}_i$, we update all the outgoing ground messages from $\mathbf{C}_i$. After the *burn-in* period, i.e., when we assume that the sampler has mixed, we use the sampled assignment to $\mathbf{C}_i$ to update the probability estimates for the query atoms using the following mixture estimator (see Section 2.2.2).

$$\widehat{P}_{\mathcal{M}}(Q) = \frac{1}{T}\sum_{t=1}^{T} P_{\mathcal{M}}(Q|\bar{\mathbf{S}}^{(t)}_{-Q}) \tag{4.1}$$

where $\bar{\mathbf{S}}^{(t)}_{-Q}$ is the t-th sample projected on all atoms except Q. The mixture estimator shown in Eq. (4.1) is unbiased, i.e., $\widehat{P}_{\mathcal{M}}(Q)$ approaches $P_{\mathcal{M}}(Q)$ as $T \to \infty$, if Algorithm 6 draws each sample correctly from the distribution represented by $\mathcal{M}$. We prove this next.

Theorem 6. *The Markov chain induced by LBG (Algorithm 6) is ergodic, i.e., for any input normal MLN $\mathcal{M}$ and its Gibbs cluster graph G, the Markov chain induced by LBG converges to $P_{\mathcal{M}}$ for any random initialization of ground messages in G.*

Proof. First, we begin by showing that the Markov chain underlying LBG is a reversible Markov chain. Note that the transition function of LBG is given by the following equation.

$$P_{\mathcal{M}}(A_{i1}, A_{i2} \ldots A_{iN}|\bar{\mathbf{S}}^{(t)}_{-i}) \tag{4.2}$$

where $\{A_{ij}\}_{j=1}^{N}$ are the ground atoms in cluster $\mathbf{C}_i$ and $\bar{\mathbf{S}}_{-i}^{(t)}$ is the t-th sample projected on all atoms of $\mathcal{M}$ except those in $\mathbf{C}_i$.

Consider the transition from $A_{i1}^{(t)} \dots A_{iN}^{(t)} \cup \bar{\mathbf{S}}_{-i}^{(t)} \rightarrow A_{i1}^{(t+1)} \dots A_{iN}^{(t+1)} \cup \bar{\mathbf{S}}_{-i}^{(t)}$ according to the transition function defined above. This occurs with probability equal to

$$P(A_{i1}^{(t+1)}, A_{i2}^{(t+1)} \dots A_{iN}^{(t+1)} | \bar{\mathbf{S}}_{-i}^{(t)})$$

We now show that the above transition function satisfies the detailed balance condition.

$$\begin{aligned} & P_{\mathcal{M}}(A_{i1}^{(t+1)} \dots A_{iN}^{(t+1)} | \bar{\mathbf{S}}_{-i}^{(t)}) P_{\mathcal{M}}(A_{i1}^{(t)} \dots A_{iN}^{(t)} \cup \bar{\mathbf{S}}_{-i}^{(t)}) \\ = & \frac{P_{\mathcal{M}}(A_{i1}^{(t+1)} \dots A_{iN}^{(t+1)} \cup \bar{\mathbf{S}}_{-i}^{(t)})}{P_{\mathcal{M}}(\bar{\mathbf{S}}_{-i}^{(t)})} (P_{\mathcal{M}}(\bar{\mathbf{S}}_{-i}^{(t)}) P_{\mathcal{M}}(A_{i1}^{(t)} \dots A_{iN}^{(t)} | \bar{\mathbf{S}}_{-i}^{(t)}) \\ = & P_{\mathcal{M}}(A_{i1}^{(t+1)} \dots A_{iN}^{(t+1)} \cup \bar{\mathbf{S}}_{-i}^{(t)}) P_{\mathcal{M}}(A_{i1}^{(t)} \dots A_{iN}^{(t)} | \bar{\mathbf{S}}_{-i}^{(t)}) \end{aligned}$$

Thus, LBG generates a reversible Markov chain. Further, since $\mathcal{M}$ represents a positive distribution (no zero values in the distribution), the distribution in Eq. (4.2) is guaranteed to be positive. Thus, the Markov chain is irreducible and since it is reversible w.r.t $P_{\mathcal{M}}$, $P_{\mathcal{M}}$ is a unique stationary distribution of the Markov chain. Further, the probability of staying in any state is non-zero and therefore the chain is aperiodic. Thus, the Markov chain is ergodic and converges to $P_{\mathcal{M}}$ for any starting state obtained by randomly initializing the messages in G.

□

4.1.4 Lifted Messages

Even though Algorithm 6 samples each cluster in a lifted manner, its messages are still propositional. Recall that, a cluster $\mathbf{C}_i$ sends a message to $\mathbf{C}_j$ over all ground atoms in $\mathbf{C}_i$ that are in the Markov blanket of some atom in $\mathbf{C}_j$. Therefore, technically, we need to maintain an assignment over the propositional state space, i.e., a 0/1 assignment to all

ground atoms in $\mathcal{M}$. Here, we show how to modify the sampler in Algorithm 6 such that it works in a *lifted state space*. Specifically, by taking advantage of symmetries in the MLN representation, we group together symmetric atoms and send an aggregate message over the state of these atoms while maintaining invariance of the MLN distribution.

We first illustrate the idea of lifted messages with our running example. In our example MLN ($\mathcal{M}$), consider the clustering in Figure 4.1 (b), i.e., $\mathbf{C}_1 = \{\mathtt{R}, \mathtt{S}\}$ and $\mathbf{C}_2 = \{\mathtt{T}\}$. Let us instantiate a ground message from $\mathbf{C}_2$ in $\mathcal{M}$. After instantiating the message, let the reduced normal MLN, $\mathcal{M}'$, contain the following $d+1$ sets of formulas.

$$\mathbf{F}' = \{\mathtt{R}(x,y) \vee \mathtt{S}_i(y, Z_i)\}_{i=1}^{d}; w_1$$

$$\mathbf{F}_1 = \{\mathtt{S}_1(y, Z_1); w_2(\top), \mathtt{S}_1(y, Z_1); w_2, \ldots\}$$

$$\mathbf{F}_2 = \{\mathtt{S}_2(y, Z_2); w_2, \mathtt{S}_2(y, Z_2); w_2(\top), \ldots\}$$

$$\ldots$$

$$\mathbf{F}_d = \{\mathtt{S}_d(y, Z_d); w_2(\top), \mathtt{S}_d(y, Z_d); w_2(\top), \ldots\}$$

where the symbol ($\top$) denotes that the formula is satisfied due to one or more assignments to atoms of $\mathtt{T}$.

Now, consider a grouping where all the ground atoms of $\mathtt{T}$ that have the same constant substituting their first argument are placed in the same group. That is, all possible groundings of $\mathtt{T}_j(Z_j, u)$ are in the same group. We now sum the assignments in each of these groups. Each summation produces the total number of ground atoms that have a value equal to 1 (or true) in that group. Now, let us see how this number can be used to represent $\mathcal{M}|\bar{\mathtt{T}}$. Observe that in the set $\mathbf{F}_j$, the number of satisfied groundings is exactly equal to the number of groundings of $\mathtt{T}(Z_j, u)$ that have an assignment 1. Thus, we can schematically represent all the unsatisfied formulas in each of the d sets $\mathbf{F}_1 \ldots \mathbf{F}_d$ with d formulas by re-parameterizing the weight of the formulas. Specifically, the unsatisfied formulas can be

represented as $\mathtt{S}_1(y, Z_1); (d - \theta_{Z_1})w_2, \ldots \mathtt{S}_d(y, Z_d); (d - \theta_{Z_d})w_2$, where θ_{Z_j} is the number of true assignments in $\mathtt{T}(Z_j, u)$. Thus, instead of sharing d^2 ground assignments, $\mathtt{T}$ can simply share d lifted assignments $\theta_{Z_1} \ldots \theta_{Z_d}$ with cluster $\mathbf{C}_1$ and this information sufficiently specifies the MLN that represents the conditional distribution $\mathcal{M}|\bar{\mathtt{T}}$.

We now formalize the above example to define lifted messages in the Gibbs cluster graph. We say that a representation of truth assignments to the groundings of an atom is lifted if we only specify the number of true (or false) assignments to its full or partial grounding.

Example 7. *Consider an atom* $\mathtt{R}(x, y)$*, where* $\Delta_x = \{X_1, X_2\}$ *and* $\Delta_y = \{Y_1, Y_2\}$*. We can represent the truth assignment* $(\mathtt{R}(X_1, Y_1) = 1, \mathtt{R}(X_1, Y_2) = 0, \mathtt{R}(X_2, Y_1) = 1, \mathtt{R}(X_2, Y_2) = 0)$ *in a lifted manner using either an integer* 2 *or a vector* $([\mathtt{R}(x, Y_1), 2], [\mathtt{R}(x, Y_2), 0])$*. The first representation says that* 2 *groundings of* $\mathtt{R}(x, y)$ *are true while the second representation says that* 2 *groundings of* $\mathtt{R}(x, Y_1)$ *and* 0 *groundings of* $\mathtt{R}(x, Y_2)$ *are true.*

Next, we state sufficient conditions for representing a message in a lifted manner while ensuring correctness as summarized in Theorem 6. For any formula f in MLN $\mathcal{M}$, we define a logical variable x that occurs in f as a *shared variable* of f iff x occurs in more than one atom in f. Further, given an MLN $\mathcal{M}$ and its Gibbs cluster graph, $x \in \mathbf{x}$ is a shared term of $\mathtt{R}(\mathbf{x})$ w.r.t cluster $\mathbf{C}_j$ iff there exists a formula f such that an atom of $\mathbf{C}_j$ occurs with $\mathtt{R}(\mathbf{x})$ in f and x is a shared variable of f.

Example 8. *Consider the formula* $R(x) \vee S(x, y)$*. x is a shared variable in the formula. Further, consider a clustering where* $\mathbf{C}_1 = \{R\}$ *and* $\mathbf{C}_2 = \{S\}$*. x is a shared term of $S(x, y)$ w.r.t* $\mathbf{C}_2$*.*

Definition 12 (Lifted Message)**.** *The lifted message from* $\mathbf{C}_i$ *to* $\mathbf{C}_j$ *denoted by* $LM(\mathbf{C}_i, \mathbf{C}_j)$ *contains the number of true groundings corresponding to every atom in* $\mathbf{C}_i$ *for each possible grounding of its shared terms w.r.t* $\mathbf{C}_j$*.*

Example 9. *In the same formula* $R(x) \vee S(x, y)$*, let* $|\Delta_x| = |\Delta_y| = d$ *and let* $\mathbf{C}_1 = \{R\}$ *and* $\mathbf{C}_2 = \{S\}$*,* $LM(\mathbf{C}_2, \mathbf{C}_1)$ *is a vector of* d *values,* $([S(X_1, y), \theta_1], \ldots [S(X_d, y), \theta_d])$ *where* θ_j *is an integer in the range* $[0, d]$ *and corresponds to the number of true ground atoms in* $S(X_j, y)$ *according to the current assignment.*

We now show that LBG with lifted messages is correct, i.e., its Markov chain converges to the distribution represented by the input MLN.

Lemma 1. *Given a normal MLN* $\mathcal{M}$ *with atom* $\mathtt{R}(x)$ *(singleton), if* $\overline{\mathtt{R}}$ *is an assignment to all groundings of* $\mathtt{R}$*, the number of groundings of* $\mathtt{R}$ *assigned to true in* $\overline{\mathtt{R}}$ *is sufficient to compute the conditional distribution* $P(\mathcal{M}|\overline{\mathtt{R}})$*.*

Proof. The proof follows directly from the generalized binomial rule in PTP (Section 2.2.3). That is, there are $\binom{|\Delta_x|}{i}$ distinct ways to define the assignment $\overline{\mathtt{R}}$. However, conditioning on each of these assignments yields the exact same distribution $P(\mathcal{M}|\bar{\mathtt{R}})$. Thus, i is sufficient to specify $P(\mathcal{M}|\bar{\mathtt{R}})$. □

Lemma 2. *Given a normal MLN* $\mathcal{M}$ *representing the joint distribution* $P_{\mathcal{M}}$*, if there exists* $\mathtt{R}(\mathbf{y})$ *such that no terms in the set* $\mathbf{y}$ *are shared variables in any formula containing the predicate symbol* $\mathtt{R}$ *and we replace* $\mathtt{R}(\mathbf{y})$ *by a singleton* $\mathtt{R}'(y)$ *with* Δ_y *equal to the cross-product of the domains of the variables in* $\mathbf{y}$ *to obtain MLN* $\mathcal{M}'$*, then* $P_{\mathcal{M}'} \equiv P_{\mathcal{M}}$*.*

Proof. We form a bijection $B : \omega \to \omega'$, where ω is a world of $\mathcal{M}$ and ω' is a world in $\mathcal{M}'$. Let $\zeta : \bar{R} \to \bar{R}'$ denote a bijection between a ground atom of $\mathtt{R}(\mathbf{y})$ and a ground atom in $\mathtt{R}'(y)$. Notice that since Δ_y is equal to the cross-product of the domains of the variables in $\mathbf{y}$, we can form ζ directly. B is now easy to define as follows. For any world ω, if the groundings of $\mathtt{R}$ say $\bar{R}_1 \ldots \bar{R}_n$ have assignments $a_1 \ldots a_n$, then in ω' $\zeta(\bar{R}_1) \ldots \zeta(\bar{R}_n)$ have assignments equal to $a_1 \ldots a_n$. Importantly, since $\mathbf{y}$ is not shared in any formula containing $\mathtt{R}$, replacing the variables in $\mathbf{y}$ with a single term y does not change the terms in any other

atom of $\mathcal{M}$. This means ω projected on all ground atoms apart from the groundings of $\mathtt{R}$ is identical to ω' projected on all ground atoms apart from the groundings of $\mathtt{R}'$. Further, it is easy to see that if ω makes a ground formula containing $\mathtt{R}$ true (or false), $B(\omega)$ will make the corresponding ground formula where the atoms of $\mathtt{R}$ have been replaced by ground atoms defined by ζ true (or false). Thus, the probability of ω and $B(\omega)$ is identical. Therefore, the theorem holds. □

Theorem 7 (**Sufficient Conditions for a Lifted Message Representation).** *If each ground message in LBG is substituted by a lifted message as given by definition 12, then the stationary distribution of the Markov chain induced by LBG is the distribution represented by the input normal MLN.*

Proof. Let G be the Gibbs cluster graph of the input normal MLN $\mathcal{M}$. Let $\mathbf{C}_i$ and $\mathbf{C}_j$ be the neighbors of each other in G. Let $\mathtt{R}$ be an element of $\mathbf{C}_i$ and let $\mathbf{x}$ be its shared terms and $\mathbf{y}$ its non-shared terms. Let $\Delta_{\mathbf{x}}$ be the Cartesian product of the domains of all terms in $\mathbf{x}$, $\mathbf{X}_k \in \Delta_{\mathbf{x}}$ is the k-th element in $\Delta_{\mathbf{x}}$ and r_k is the number of groundings of $\mathtt{R}(\mathbf{X}_k, \mathbf{y})$ that are true in the current assignment. By definition 12, $LM(\mathbf{C}_i, \mathbf{C}_j)$ contains a vector of size $|\Delta_{\mathbf{x}}|$ for $\mathtt{R}$ where the k-th element of the vector is equal to $[\mathtt{R}(\mathbf{X}_k, \mathbf{y}), r_k]$. Next, we show that each $\mathtt{R}(\mathbf{X}_k, \mathbf{y})$ is equivalent to a singleton.

Consider the MLN $\mathcal{M}'$ which is obtained from $\mathcal{M}$ by first removing all formulas that do not mention atoms in C_j and then (partially) grounding all the shared terms of $\mathtt{R}$. Let y' be a logical variable such that its domain $\Delta_{y'} = \Delta_{\mathbf{y}}$, where $\Delta_{\mathbf{y}}$ is the Cartesian product of the domains of all variables in $\mathbf{y}$ and let $\mathtt{R}'_k(y') = \mathtt{R}(\mathbf{X}_k, \mathbf{y})$ where $\mathbf{X}_k \in \Delta_{\mathbf{x}}$ is the k-th element in $\Delta_{\mathbf{x}}$. By Lemma 2, $\mathtt{R}(\mathbf{X}_k, \mathbf{y})$ in $\mathcal{M}'$ can be replaced by $\mathtt{R}'_k(y')$ without changing the associated distribution. Moreover, each atom $\mathtt{R}'_k(y')$ is a singleton and therefore it follows from Lemma 1 that in order to compute the distribution associated with $\mathcal{M}'$ conditioned on $\mathtt{R}'_k(y')$, we only need to know how many of its possible groundings are true. Since C_i sends

precisely this information to C_j using the message defined in definition 12, it follows that LBG with lifted messages is equivalent to the algorithm that uses a propositional representation (Algorithm 6). Since Algorithm 6 converges to the distribution represented by the MLN (Theorem 6), the proof follows. □

Lifted State Space

We now take a closer look at how lifted messages influence the Markov chain induced by LBG. Theorem 7 provides a method for representing the messages in LBG succinctly by taking advantage of the symmetry in MLN structure which reduces the space complexity of messages in LBG. Formally,

Proposition 1 (**Space Complexity of a Message).** *Given a Gibbs cluster graph G and an MLN $\mathcal{M}$, let the outgoing message from cluster C_i to cluster C_j in G be defined over the set of atoms, $\{\mathtt{R}_1, \ldots, \mathtt{R}_k\}$. Let $\mathbf{x}_i$ denote the set of shared terms of $\mathtt{R}_i$ w.r.t $\mathbf{C}_j$. Then, the space complexity of $LM(\mathbf{C}_i, \mathbf{C}_j)$ is $O(\sum_{i=1}^{k} |\Delta_{\mathbf{x}_i}|)$.*

Apart from reducing the space complexity, by specifying each message succinctly in a lifted manner, LBG explores a *lifted state space* which means that each sample drawn from this space is equivalent to a number of distinct propositional samples. We illustrate this with the following example.

Example 10. *Figure 4.3 shows a 12-dimensional propositional state space (each dimension is represented by a binary variable) and a corresponding 3-dimensional lifted state space. Assume that both spaces represent the exact same probability distribution. The lifted space is defined by grouping together the variables that form the propositional space. Specifically, variables representing dimensions 1 through 4, 4 through 8 and 8 through 12 are grouped together. For each of the three groups, we have one* meta-variable *in the lifted state space that specifies the number of 1's assigned to the propositional variables in that group. Importantly,*

Propositional State Space	Lifted State Space
0000 0000 0000	0 0 0
0000 0000 0001	0 0 1
0000 0000 0010	0 0 1
0000 0000 0011	0 0 2
0000 0000 0100	0 0 1
0000 0000 0101	0 0 2
...	...
1111 1111 1111	4 4 4

Figure 4.3. Propositional vs Lifted State Space

the number of possible samples in the propositional space is 2^{12} *while in the lifted space this number is equal to* 5^3*. Thus, a single lifted sample is virtually equivalent to* $\frac{2^{12}}{5^3} \approx 38$ *propositional samples.*

To formalize the above example, let Q be a probability distribution defined over nd binary random variables X_{11}, X_{12} ... X_{1d} ... X_{nd}. Let ω and ω' be two full assignments to X_{11}, X_{12} ... X_{1d} ... X_{nd}. Let ω assign v_i variables among $\{X_{ij}\}_{j=1}^{d}$ to 1. Let ω' assign v_i' variables among $\{X_{ij}\}_{j=1}^{d}$ to 1. $Q(\omega) = Q(\omega')$ if $\forall i$, $v_i = v_i'$. Let Q' be defined over n random variables each with a integer range $[0, d]$, specifying the count of 1-assignments to each of the groups $\{X_{1j}\}_{j=1}^{d}$, $\{X_{2j}\}_{j=1}^{d}$ ... $\{X_{nj}\}_{j=1}^{d}$. $Q'(\bar{x}') = \sum_{\bar{x} \sim \bar{x}'} Q(\bar{x})$, where $\bar{x} \sim \bar{x}'$ denotes that assignment $\bar{x}$ in Q is consistent with the counts to its variables specified in assignment $\bar{x}'$ in Q'. Suppose, we wish to estimate a function f with expected value equal to,

$$\mathbb{E}_Q[f] = \sum_{\bar{x}} f(\bar{x}) Q(\bar{x})$$

We can estimate f using a Gibbs sampler whose stationary distribution is Q. The sample mean for f derived from these Gibbs samples is clearly *unbiased*, i.e., the expected value of the sample mean is equal to the expected value of f. However, instead of sampling from Q, we can construct a Gibbs sampler whose stationary distribution is Q', where we estimate f by projecting each sample from Q', say $\bar{x}^{(t)'}$ to $\bar{x}^{(t)}$, where $\bar{x}^{(t)} \sim \bar{x}^{(t)'}$. Let us denote

the sample means of the two samplers as $\bar{E}_Q[f]$ an $\bar{E}_{Q'}[f]$ respectively, and the variances as $Var_Q(f)$ and $Var_{Q'}(f)$.

Theorem 8. *$\bar{E}_{Q'}[f] = \mathbb{E}_Q[f]$ and if $\frac{d}{\log(d+1)} \geq \frac{n+1}{n}$ then $Var_{Q'}(f) \leq Var_Q(f)$*

Proof. First, it is easy to see that the sample means are identical. That is, since $Q'(\bar{x}') = \sum_{\bar{x} \sim \bar{x}'} Q(\bar{x})$ and $Q(\bar{x})$ is the same for all $\bar{x} \sim \bar{x}'$, it follows that $\bar{E}_Q[f]$ and $\bar{E}_{Q'}[f]$ are the same and are both unbiased estimates for $\mathbb{E}_Q[f]$.

Let K Gibbs samples be drawn from Q. The expected equivalent number of samples from Q' is $K\frac{1}{d+1}$, since to generate each Gibbs sample from Q', we need to compute a distribution of size $(d+1)$. However, each sample from Q' is in expectation worth $\frac{2^{nd}}{(d+1)^n}$ samples of Q, since Q is a distribution over 2^{nd} states while Q' is defined over $(d+1)^n$ states. Thus the equivalent number of samples from Q' is $K\frac{2^{nd}}{(d+1)^{n+1}}$. It is known that the variance of the sample estimate is equal to $\frac{\sigma^2}{N}$, where σ^2 is the variance of f and N is the number of samples used to derive the sample estimate. Thus, $Var_{Q'}(f) \leq Var_Q(f)$ if,

$$K\frac{2^{nd}}{(d+1)^{n+1}} \geq K$$

$$\frac{2^{nd}}{(d+1)^{n+1}} \geq 1$$

Re-arranging terms and taking log, we have the required condition,

$$\frac{d}{\log(d+1)} \geq \frac{n+1}{n}$$

□

Assuming that $d \gg \log(d)$ and $\frac{n+1}{n} \approx 1$, the condition in Theorem 8 is always satisfied. Thus, the variance of estimates derived from samples of Q' is smaller than the estimates derived from Q. Since lifted messages group assignments on ground atoms, from the above theorem we immediately have the following result.

Corollary 1. *The marginal probability estimates derived from LBG with lifted messages has lower variance than the estimates derived from an equivalent propositional Gibbs sampler.*

4.1.5 Clustering

Next, we present a heuristic algorithm to construct the Gibbs cluster graph. From a computational perspective, we want to compute each conditional distribution tractably. From a mixing perspective, we want to place correlated variables within a block. We first discuss the complexity of our sampler and then describe the clustering algorithm that constructs graphs that ensure tractability while improving mixing in the sampler.

Complexity

The sampling complexity is the complexity of running PTP over the MLN projected on each cluster. We quantify this complexity with the *lifted width* of an MLN. Note that in propositional inference, the treewidth of the graphical model bounds the complexity of inference. However, this is not a sufficient complexity measure for lifted inference (Section 2.2.3).

Definition 13 (Lifted Width)**.** *The lifted width of an MLN $\mathcal{M}$ is defined as follows*

$$LW^*(\mathcal{M}) = \min_{\mathcal{T}_\mathcal{M}}(N_c(\mathcal{T}_\mathcal{M})tw(\mathcal{G}_{\mathcal{T}_\mathcal{M}}))$$

where $\mathcal{T}_\mathcal{M}$ is a possible PTP-tree for $\mathcal{M}$, $N_c(\mathcal{T}_\mathcal{M})$ is the number of conditioning nodes in $\mathcal{T}_\mathcal{M}$ and $tw(\mathcal{G}_{\mathcal{T}_\mathcal{M}})$ is an upper bound on the treewidth of the remaining ground Markov network after applying the operations specified in $\mathcal{T}_\mathcal{M}$

Clearly computing the lifted width is NP-hard since a special case is the NP-complete problem of computing the treewidth for the ground Markov network. Therefore, we can only compute a heuristic estimate for the lifted width. Specifically, we execute a symbolic trace of the PTP algorithm and obtain an upper-bound on the lifted width. This is illustrated in Algorithm 7. Algorithm 7 is a recursive procedure where the input is a normal MLN and the output is an estimate of the lifted width. As shown here, we first try to apply the power rule of PTP. We then recursively compute the lifted width on the reduced MLN after

Algorithm 7: LW

Input: A normal MLN $\mathcal{M}$
Output: An upper-bound for the lifted width of $\mathcal{M}$

```
1 begin
2     if M contains a decomposer d then
3         return LW(M|d)
4     if there exists a singleton S(x) then
5         Remove all instances of S from M
6         return (|Δ_x| + 1)LW(M)
7     else
          // tree-width of the ground Markov network
8         return tw(M)
```

applying the power rule. If we cannot apply the power rule, we find a singleton and apply the generalized binomial rule. If we find such a singleton say $\mathtt{S}(x)$, we simple remove all atoms with predicate symbol `S` and multiply the lifted with with $|\Delta_x| + 1$. Note that, this is an approximation that can overestimate the lifted width because the complexity of exploring each of these branches can be different. For example, consider a simple formula $\mathtt{R(x)} \vee \mathtt{S}(x)$, if all groundings of `R(x)` are assigned to 1, then we do not need to condition on any groundings of $\mathtt{S}(x)$ at all since they essentially become "don't-care" variables. However, if no groundings of `R(x)` are true, then we need to condition on the groundings of $\mathtt{S}(x)$ as they are no longer don't-care variables. Thus, logical structure contributes to the complexity of exploring each of the $\mathtt{S}(x) + 1$ branches. However, for a heuristic upper-bound of lifted width, we simply ignore this structure. Finally, if we can apply neither rule, then we compute an upper-bound for the treewidth of the remaining ground Markov network using standard heuristics such as *min-fill* and *min-degree*.

Constructing the Gibbs Cluster Graph

Given a cluster graph G corresponding to $\mathcal{M}$, let the message complexity $M(G)$ be given by the following equation.

$$M(G) = \sum_{(\mathbf{C}_i, \mathbf{C}_j) \in E(G)} |LM(\mathbf{C}_i, \mathbf{C}_j)| + |LM(\mathbf{C}_j, \mathbf{C}_i)|$$

where $E(G)$ is the set of edges in G. Note that minimizing $M(G)$ passes fewer messages across clusters. This implicitly places more atoms which are in the Markov blanket of each other inside the same cluster. These are the atoms that are likely to have correlations and thus, the mixing time is likely to improve. Formally, the optimization problem is defined as follows.

$$\begin{aligned} &\arg\min_{G_{\mathcal{M}}} M(G_{\mathcal{M}}) \\ &\text{s.t.} \max_{\mathbf{C} \in G_{\mathcal{M}}} \mathrm{LW}(\mathcal{M}_{\mathbf{C}}) \leq \alpha \end{aligned} \tag{4.3}$$

where $\mathbf{C}$ is a cluster defined by $G_{\mathcal{M}}$ and $\mathcal{M}_{\mathbf{C}}$ is the MLN $\mathcal{M}$ projected on cluster $\mathbf{C}$.

The above optimization problem is NP-hard since a special case is the bin packing problem. That is, we can re-formulate an equivalent bin-packing problem, where each cluster has a capacity equal to α and we need minimize a function over these clusters, where, adding/removing to/from the clusters alters its capacity. To solve this optimization problem, we start with an initial set of clusters and then iteratively improve the clustering. Specifically, for the initial solution, we place each first-order atom of $\mathcal{M}$ in a separate cluster. We assume that this initial solution is a feasible solution that satisfies the constraints. Note that, we may need to pre-process the original MLN by partially grounding it to ensure that the tractability constraint is satisfied for the specified α. In the degenerate case, a completely ground MLN guarantees a feasible solution for any $\alpha \geq 1$. That is, in normal form, each ground atom is in a separate cluster and therefore our algorithm becomes equivalent to regular Gibbs sampling on the ground Markov network.

Given an initial Gibbs cluster graph, we define edge weights on the graph, where for any edge $(\mathbf{C}_i, \mathbf{C}_j)$, its weight is equal to $|LM(\mathbf{C}_i, \mathbf{C}_j)| + |LM(\mathbf{C}_j, \mathbf{C}_i)|$. We construct a new graph by collapsing edges (merging the clusters connected by this edge) from this initial graph such that the function defined in Eq. (4.3) is minimized. Specifically, we maintain a set of feasible edges, i.e., all the edges that whose end-nodes can be merged while satisfying the tractability constraint. In each iteration, we choose a single feasible edge which maximizes the objective function and construct a new graph where that edge is collapsed. After each merge, we update the feasible edge set. That is, we re-evaluate the feasibility for every edge that is incident on the merged nodes. If the feasible edge-set becomes empty or we reach a maximum number of iterations, the algorithm terminates.

4.1.6 Experiments

Setup

In this section, we compare the performance of lifted blocked Gibbs sampling (LBG) with (propositional) blocked Gibbs sampling (BG), lazy MC-SAT (Poon and Domingos, 2006; Poon et al., 2008) and lifted belief propagation (LBP) (Singla and Domingos, 2008). We experimented with four MLNs whose formulas are given below.

(i) Student : $\texttt{Student}(x) \Rightarrow \texttt{Takes}(x, c)$, $\texttt{Jobs}(p, x) \Rightarrow \texttt{Takes}(x, c)$

(ii) Asthma : $\texttt{Asthma}(x) \rightarrow \neg\texttt{Smokes}(x)$, $\texttt{Asthma}(x) \wedge \texttt{Friends}(x, y) \Rightarrow \neg\ \texttt{Smokes}\ (y)$, $\texttt{Smokes}(x) \Rightarrow \texttt{Cancer}(x)$

(iii) Topics : $\texttt{Word}(w, p) \Rightarrow \texttt{Topic}(p, t)$, $\texttt{Topic}(p, t) \Rightarrow \texttt{Class}(t, c)$

(iv) WebKB : The WebKB MLN used in (Lowd and Domingos, 2007b).

Note that the first two MLNs can be solved in polynomial time using PTP while PTP is exponential on Topics and WebKB. For each MLN, we set 10% randomly selected ground

atoms as evidence. We varied the number of objects in the domain from 5 to 200. We used a time-bound of 1000 seconds for all algorithms.

We implemented LBG and BG in C++ and used alchemy (Kok et al., 2006) to implement MC-SAT and LBP. For LBG, BG and MC-SAT, we used a burn-in of 100 samples to negate the effects of initialization.

For Student and Asthma, we measure the accuracy using the KL divergence between the estimated marginal probabilities and the true marginal probabilities computed using PTP. Since computing exact marginals of Topics and WebKB is not feasible, we perform convergence diagnostics for LBG and BG using the Gelman-Rubin statistic (Gelman and Rubin, 1992), denoted by R. R measures the disagreement between multiple Markov chains generated by the algorithm. Specifically, it compares the between-chain variances with the within-chain variances as follows.

$$\hat{V} = (1 - \alpha)W + \alpha B \tag{4.4}$$

where B measures the between-chain variance, W measures the within chain variance and α is a parameter in the weighted average which is normally set to the reciprocal of the total number of parallel chains in the simulation.

$$R = \sqrt{\frac{\hat{V}}{W}} \tag{4.5}$$

Note that in Eq. (4.5), the closer the value of R is to 1, the better the mixing.

Results

Accuracy: Figure 4.4 illustrates the accuracy of LBG on the tractable MLNs. Specifically, we plot the average KL divergence between the true marginal probabilities and the approximate marginal probabilities as a function of time for the MLNs Student and Asthma respectively. In both cases, LBG converges much faster than both BG and MC-SAT and

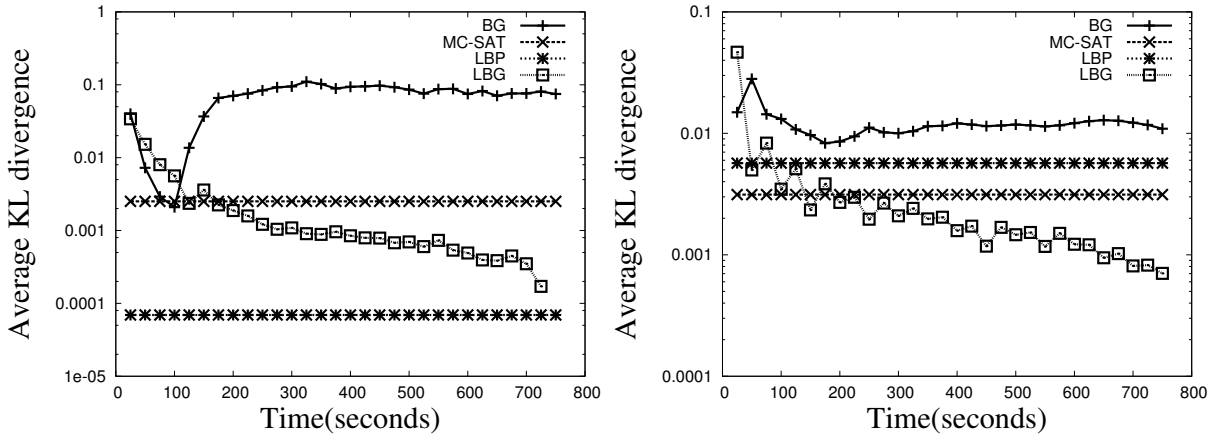
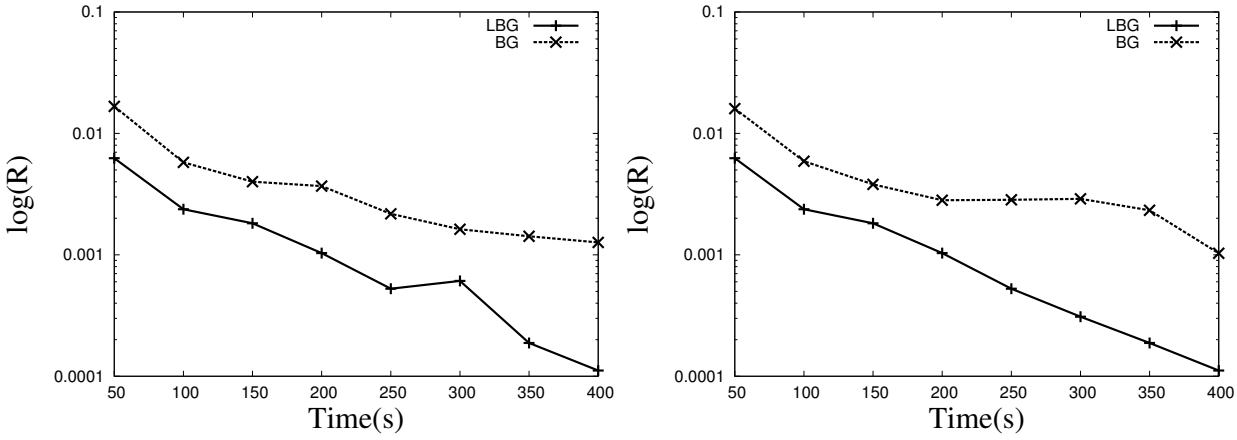


Figure 4.4. Illustrating accuracy. Plots show average KL Divergence between the true marginal probabilities and the approximate marginal probabilities as a function of time for: (a) Student with 50 objects and (b) Asthma with 50 objects.

also has a smaller error. LBP is more accurate than LBG on Student while LBG is more accurate than LBP on Asthma.

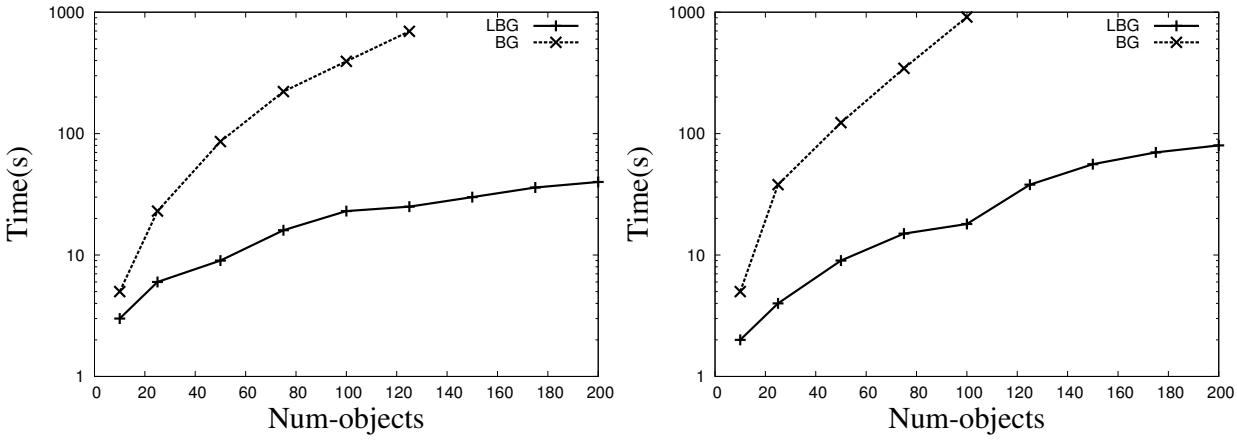
Mixing Rate: Figure 4.5 illustrates the convergence of the Markov chain induced by LBG. We run this experiment for the MLNs, Topics and WebKB, which are larger MLNs on which inference is intractable. Here, for both LBG and BG, we plot $\log(R)$ as a function of time for Topics and WebKB respectively. We see that the Markov chain associated with LBG mixes much faster than the one associated with BG for both MLNs. This can be attributed to the fact that LBG works in a lifted state space and therefore samples more effectively, i.e., each sample from LBG is worth several samples from BG.

Scalability: Finally, Figures 4.6(a) and (b) compare the scalability of LBG with BG. Here, we use running time per Gibbs iteration as a performance metric. A short running time per sample means that given a finite amount of time, we are able to generate a larger number of samples which typically results in faster convergence and higher quality estimates. Figures 4.6(a) and 4.6(b) show the time required by 100 Gibbs iterations as a function of number of objects for Topics and WebKB respectively. As clearly illustrated in these figures, LBG



(a) (b)

Figure 4.5. Convergence diagnostics using Gelman-Rubin statistic (R) as a function of time for (a) Topics and (b) WebKB.



(a) (b)

Figure 4.6. Scalability of lifted blocked Gibbs. Time required by 100 Gibbs iterations as a function of the number of objects for (a) Topics and (b) WebKB.

is much more scalable than BG. Also, with increasing domain-size LBG scales up almost linearly while the time taken by BG increases exponentially. This clearly illustrates the advantage of lifted inference over propositional inference because as the domain gets larger, while the ground MLN gets larger, LBG utilizes the symmetries that it can identify directly from relational structure without requiring to process the full ground MLN.

4.2 Lifted Importance Sampling

We now present our next approximate inference algorithm, namely lifted importance sampling (LIS). In contrast to Gibbs sampling where we obtain samples directly from the MLN distribution, importance sampling (IS) samples from a different distribution called the proposal distribution and corrects for this by weighting each sample. Its main advantage is that, unlike Gibbs sampling where each of our samples were not strictly independent samples, using IS, we obtain independent samples, which is beneficial in statistical estimation. However, in IS, the estimates are highly dependent on the quality of the proposal distribution. The closer the proposal distribution is to the true distribution, the higher the quality of estimates obtained from IS. In propositional IS, the proposal distribution (Q) is defined over the same space as the original distribution (P). That is, suppose P is defined over n binary random variables, then the joint distribution of Q has size 2^n. The difference though is that Q is carefully designed such that Q is easy to sample while P may be an arbitrary distribution that is hard to sample (e.g., a Markov network). In LIS, we design an accurate proposal distribution that is also lifted, i.e., considering our previous example with n variables, the size of the lifted proposal Q_L is $\ll 2^n$. Implicitly, every sample from Q_L corresponds to virtually many samples from Q. Importantly, the increase in samples improves the accuracy of LIS as compared to propositional IS.

Previously, (Gogate and Domingos, 2011b) introduced a PTP based lifted importance sampling algorithm for the inference task of approximating the partition function of an MLN. However, it has two limitations. First, it does not use lifting (relational structure) to the fullest extent and as a result it can be needlessly inefficient and inaccurate on some problems. Second, it uses an uninformative proposal distribution and therefore produces poor estimates in many cases. We remedy these issues as follows.

First, we propose a new lifting rule that identifies additional symmetries and using this, reduces the sampling space exponentially in many instances. We show how to perform

importance sampling in this reduced space and prove that our new sampling algorithm has smaller variance. Second, we propose an adaptive, structured approach for constructing and dynamically updating the proposal distribution. Given an MLN and evidence, the main idea here is to apply various lifting and probability propagation rules in an approximate manner by relaxing their pre-conditions, yielding a polynomially specifiable proposal distribution. Then, we initialize it to the prior distribution and dynamically update its parameters via adaptive importance sampling (Cheng and Druzdzel, 2000; Ortiz and Kaelbling, 2000).

We present experiments comparing the quality of estimation of our advanced LIS scheme with the LIS scheme of (Gogate and Domingos, 2011b) for the task of computing the partition function on MLNs from various domains. Our experiments clearly demonstrate that our advanced algorithm is always superior.

4.2.1 PTP-based Importance Sampling

Recall that the partition function of an MLN is given by the following equation.

$$Z(\mathcal{M}) = \sum_{\omega} \exp\left(\sum_{i} w_i N(f_i, \omega)\right) \tag{4.6}$$

The main idea in IS is to reformulate the summation problem in Eq. (4.6) as an expectation problem using a probability distribution Q, called the proposal or the importance distribution. Q should be such that it is easy to generate independent samples from it. Also, in order to apply IS to MLNs, Q should satisfy the constraint: $\exp(\sum_i w_i N(f_i, \omega)) > 0 \Rightarrow Q(\omega) > 0$. Formally, using Q, we can rewrite Eq. (4.6) as

$$Z(\mathcal{M}) = \sum_{\omega} \exp\left(\sum_{i} w_i N(f_i, \omega)\right) \frac{Q(\omega)}{Q(\omega)} \tag{4.7}$$

$$= \mathbb{E}_Q\left[\frac{\exp\left(\sum_i w_i N(f_i, \omega)\right)}{Q(\omega)}\right] \tag{4.8}$$

where $\mathbb{E}_Q[x]$ denotes the expected value of the random variable x w.r.t. Q. Given N worlds $(\omega^{(1)}, \ldots, \omega^{(N)})$, sampled independently from Q, we can estimate $Z(\mathcal{M})$ using:

$$\widehat{Z}(\mathcal{M}) = \frac{1}{N} \sum_{j=1}^{N} \frac{\exp\left(\sum_i w_i N(f_i, \omega^{(j)})\right)}{Q(\omega^{(j)})} \tag{4.9}$$

It is known that $\mathbb{E}[\widehat{Z}(\mathcal{M})] = Z(\mathcal{M})$ (i.e., it is unbiased) and therefore the mean squared error between $\widehat{Z}(\mathcal{M})$ and $Z(\mathcal{M})$ can be reduced by reducing its variance. The variance can be reduced by using a proposal distribution Q that is as close as possible to the distribution $P_{\mathcal{M}}$. Thus, a majority of research on importance sampling is focused on finding a good Q. For more details, see (Liu, 2001).

The PTP based lifted importance sampling algorithm (Gogate and Domingos, 2011a) operates by grouping symmetric random variables, sampling just one member from each group and using the sampled member to estimate quantities defined over the group. It uses the generalized binomial rule from PTP to sample singleton atoms efficiently. Specifically, given a normal MLN $\mathcal{M}$ having a singleton atom $\mathtt{R}(x)$ that is not involved in self-joins, we can compute the partition function as given by the following equation.

$$Z(\mathcal{M}) = \sum_{i=0}^{|\Delta_x|} \binom{|\Delta_x|}{i} Z(\mathcal{M}|\bar{\mathtt{R}}^i) w(i) 2^{p(i)}$$

where $\bar{\mathtt{R}}^i$ is a truth-assignment to all groundings of $\mathtt{R}$ such that exactly i groundings of $\mathtt{R}$ are set to `True` (and the remaining are set to `False`). $\mathcal{M}|\bar{\mathtt{R}}^i$ is the MLN obtained from $\mathcal{M}$ by performing the following steps in order: (i) Ground all $\mathtt{R}(x)$ and set its groundings to have the same assignment as $\bar{\mathtt{R}}^i$, (ii) Delete all formulas that evaluate to either `True` or `False`, (iii) Delete all groundings of $\mathtt{R}(x)$ and (iv) Convert the resulting MLN to a normal MLN. $w(i)$ is the exponentiated sum of the weights of formulas that evaluate to `True` and $p(i)$ is the number of ground atoms that are removed from the MLN as a result of removing formulas (these are essentially don't care propositional atoms which can be assigned to either `True` or `False`).

Algorithm 8: LIS

Input: A normal MLN $\mathcal{M}$ and a proposal distribution Q
Output: An unbiased estimate of the partition function of $\mathcal{M}$
if $\mathcal{M}$ is empty **then return** 1
if *there exists a singleton atom* $\mathtt{R}(x)$ *that does not appear more than once in the same formula* **then**
  Use Q to sample an integer i from the range $[0, |\Delta_x|]$
  return $\frac{\mathrm{LIS}(\mathcal{M}|\bar{\mathtt{R}}^i)w(i)2^{p(i)}}{Q(i)}\binom{|\Delta_x|}{i}$
Choose an atom $\mathtt{A}$ and sample all of its groundings from Q. Let $\bar{\mathtt{A}}$ be the sampled assignment.
return $\frac{\mathrm{LIS}(\mathcal{M}|\bar{\mathtt{A}})w(\bar{\mathtt{A}})2^{p(\bar{\mathtt{A}})}}{Q(\bar{\mathtt{A}})}$

Algorithm 8 provides a schematic description of LIS. It takes as input a normal MLN $\mathcal{M}$ and a proposal distribution Q. If the MLN is empty, the algorithm returns 1. The algorithm then checks if there exists a singleton atom $\mathtt{R}(x)$. If there exists one, then the algorithm samples an integer i from Q and recurses on $\mathcal{M}|\bar{\mathtt{R}}^i$ according to the generalized binomial rule. If not, the algorithm selects an atom $\mathtt{A}$, samples all of its groundings from Q and recurses on the MLN obtained by instantiating the sampled assignment $\bar{\mathtt{A}}$ (denoted by $\mathcal{M}|\bar{\mathtt{A}}$). $w(\bar{A})$ denotes the exponentiated sum of the weights of formulas that evaluate to $\mathtt{True}$ because of the assignment $\bar{\mathtt{A}}$ and $p(\bar{\mathtt{A}})$ denotes the number of ground atoms that are removed from the MLN as a result of removing formulas.

4.2.2 New Lifting Rule

The proposal distribution Q is lifted over all the singleton atoms that are encountered in Algorithm 8. However, for all other atoms, the proposal is defined over a propositional space. It turns out that Algorithm 8 does not fully exploit the symmetries in the MLN. We now introduce a new lifting rule that leverages these additional symmetries not handled in Algorithm 8 and using this, we define a much more advanced lifted importance sampler with reduced variance.

MLN: $(\mathtt{R}(x,y) \vee \mathtt{S}(y,z) \vee \mathtt{T}(z,u), w)$
Domains:
$\Delta_x = \{X_1, X_2, X_3\}$, $\Delta_y = \{Y_1, Y_2, Y_3\}$
$\Delta_z = \{Z_1, Z_2, Z_3\}$, $\Delta_u = \{U_1, U_2, U_3\}$

(a)

Sampled goundings of $\mathtt{R}(x,y)$
$\mathtt{R}(X_1,Y_1), \neg\mathtt{R}(X_1,Y_2), \neg\mathtt{R}(X_1,Y_3)$
$\mathtt{R}(X_2,Y_1), \mathtt{R}(X_2,Y_2), \neg\mathtt{R}(X_2,Y_3)$
$\neg\mathtt{R}(X_3,Y_1), \mathtt{R}(X_3,Y_2), \mathtt{R}(X_3,Y_3)$

(b)

Reduced MLN after instantiating $\mathtt{R}$
$(\mathtt{S}(Y_1,z) \vee \mathtt{T}(z,u), 2w)$
$(\mathtt{S}(Y_2,z) \vee \mathtt{T}(z,u), 2w)$
$(\mathtt{S}(Y_3,z) \vee \mathtt{T}(z,u), w)$

(c)

Figure 4.7. Illustration of lifted importance sampling. (a) An example MLN. (b) Sampled groundings of $\mathtt{R}(x,y)$. (c) Reduced MLN obtained by instantiating the sampled groundings of $\mathtt{R}$.

Isolated Variables Rule

We first describe the isolated variables rule using a non-trivial MLN having just one weighted formula $f = \mathtt{R}(x,y) \vee \mathtt{S}(y,z) \vee \mathtt{T}(z,u)$ and then generalize it. Note that none of the existing exact techniques (de Salvo Braz, 2007; Gogate and Domingos, 2011b; Van den Broeck et al., 2011) that we are aware of can compute $Z(\{(f,w)\})$ in time that is polynomial in the domain sizes of x, y, z and u.

We begin by demonstrating how LIS will estimate the partition function of $\{(f,w)\}$ (see Figure 4.7). LIS will first select an atom, either $\mathtt{R}$, $\mathtt{S}$ or $\mathtt{T}$, and check if it can be sampled in a lifted manner. For the given f, this is not possible. Therefore, it will define an importance distribution over all groundings of the selected atom and sample all of its groundings from it. Let us assume that LIS selected $\mathtt{R}$, which has $n_x n_y$ possible groundings, assuming that $|\Delta_x| = n_x$ and $|\Delta_y| = n_y$. Sampling $\mathtt{R}$ has the effect of removing it from all groundings of f, yielding an MLN having possibly $n_x n_y$ formulas of the form $\mathtt{S}(Y_i,z) \vee \mathtt{T}(z,u)$. Note that some of the formulas in the resulting MLN can be deleted because they will evaluate to `False`. Also, we can further reduce the representation by merging identical formulas; the weight of the new formula equals the sum of the weights of the merged formulas. Figure 4.7(c)

Lifted sampling of groundings of $\mathtt{R}(x, y)$:
For $i = 1, 2, 3$ **do**
    Select an index j_i from $Q_{j_i|j_1,\ldots,j_{i-1}}$
    Randomly set j_i of $\mathtt{R}(x, Y_i)$ to `True`, remaining to `False`.
Let the sampled indices be $j_1 = 2$, $j_2 = 2$ and $j_3 = 1$.

Figure 4.8. Illustration of the advanced grouping strategy for lifted importance sampling. Here we sample indices of j_i for each $Y_i \in \Delta_y$. Let the sampled indices j_i be as shown. Then, we will get the same MLN as the one in Figure 4.7(c).

shows the reduced MLN obtained by instantiating the sampled assignments of $\mathtt{R}(x, y)$ given in Figure 4.7(b). We now show how we can group instantiations of $\mathtt{R}(x, y)$ yielding an estimate having smaller variance than LIS. Let $\Delta_y = \{Y_1, \ldots, Y_{n_y}\}$ and $\Delta_x = \{X_1, \ldots, X_{n_x}\}$. For $i = 1$ to n_y, let $j_i \in \{0, \ldots, n_x\}$, yielding a vector $(j_1, \ldots, j_{n_y})$. Consider the set of truth-assignments to the groundings of $\mathtt{R}(x, y)$ such that exactly j_i of $R(x, Y_i)$ are instantiated to `True` and the remaining to `False`. For each such group of truth assignments we have the following reduced MLN $\mathcal{M}_{j_1,\ldots,j_{n_y}} = \bigcup_{i=1}^{n_y} \{(\mathtt{S}(Y_i, z) \wedge \mathtt{T}(z, u), j_i w)\}$. Moreover, there are $\prod_{i=1}^{n_y} \binom{n_x}{j_i}$ members in this group (since for each j_i, there are $\binom{n_x}{j_i}$ ways in which j_i of $\mathtt{R}(x, Y_i)$ can be made `True`). Therefore, $Z(\mathcal{M})$ can be expressed as a sum over all possible vectors $(j_1, \ldots, j_{n_y})$:

$$Z(\mathcal{M}) = \sum_{j_1=0}^{n_x} \cdots \sum_{j_{n_y}=0}^{n_x} Z(\mathcal{M}_{j_1,\ldots,j_{n_y}}) \prod_{i=1}^{n_y} \binom{n_x}{j_i} \tag{4.10}$$

In LIS, we sampled a truth assignment to all groundings of $\mathtt{R}(x, y)$ from a state space of size $O(2^{n_x n_y})$. If we sample from the grouping described above, the state space size reduces exponentially from $O(2^{n_x n_y})$ to $O((n_x + 1)^{n_y})$.

Next, we describe how to define an importance distribution Q over this space. From Eq. (4.10), it is easy to see that we can define it sequentially as $\prod_{i=1}^{n_y} Q_{j_i|j_1,\ldots,j_{i-1}}$ along an order $(Y_1, \ldots, Y_n)$ of Δ_y, where each $Q_{j_i|j_1,\ldots,j_{i-1}}$ gives the conditional probability of sampling the index $j_i \in \{0, \ldots, n_x\}$ given an assignment to all previous indices.

Figure 4.8 shows how to use our advanced grouping strategy to generate samples from the MLN given in Figure 4.7.

The ideas presented in this section can be generalized using the following *isolated variables* rule in LIS. For a predicate symbol $\mathtt{R}$ of an MLN $\mathcal{M}$, we partition its arguments into two sets, $\mathbf{A}_1$ and $\mathbf{A}_2$. $\mathbf{A}_1$ contains all those arguments such that in any formula of $\mathcal{M}$ where $\mathtt{R}$ occurs, the logical variables that substitute the arguments in $\mathbf{A}_1$ are exclusive to $\mathtt{R}$. Let $\mathbf{x}$ be a set of logical variables where each $x \in \mathbf{x}$ substitutes a unique argument in $\mathbf{A}_1$. Similarly, let $\mathbf{y}$ be variables that substitute $\mathbf{A}_2$. We refer to $\mathbf{x}$ as isolated variables for $\mathtt{R}$. Let $\Delta_{\mathbf{y}}$ denote the Cartesian product of the domains of variables in $\mathbf{y}$ and let $\mathbf{Y}_i$ denote the i-th element in $\Delta_{\mathbf{y}}$. Let $\mathcal{M}[\mathtt{R}, \mathbf{x}]$ be an MLN obtained from $\mathcal{M}$ by applying the following steps in order: (i) for $i = 1$ to $|\Delta_{\mathbf{y}}|$, sample j_i from a distribution $Q_i(j_i|j_1, \ldots, j_{i-1})$ and set j_i arbitrarily selected groundings of $\mathtt{R}(x, \mathbf{Y}_i)$ to $\mathtt{True}$ and the remaining to $\mathtt{False}$, (ii) Delete all formulas that evaluate to either $\mathtt{True}$ or $\mathtt{False}$, (iii) Delete all groundings of $\mathtt{R}$ and (iv) Convert the MLN to a normal MLN. Let $w(\mathtt{R})$ be the exponentiated sum of the weights of formulas that evaluate to $\mathtt{True}$ and let $p(\mathtt{R})$ be the number of ground atoms that are removed from the MLN as a result of removing formulas. The unbiased estimate of $Z(\mathcal{M})$ is given by:

$$\widehat{Z}(\mathcal{M}) = \widehat{Z}(\mathcal{M}[\mathtt{R}, \mathbf{x}]) w(\mathtt{R}) 2^{p(\mathtt{R})} \prod_{i=1}^{|\Delta_{\mathbf{y}}|} \frac{\binom{|\Delta_{\mathbf{x}}|}{j_i}}{Q_i(j_i|j_1, \ldots, j_{i-1})} \tag{4.11}$$

where $\widehat{Z}(\mathcal{M}[\mathtt{R}, \mathbf{x}])$ is the unbiased estimate of $Z(\mathcal{M}[\mathtt{R}, \mathbf{x}])$.

Note that in general, the isolated variables rule is only applicable to atoms not involved in self-joins. However, if the isolated variables appear in the same position in all instances of $\mathtt{R}$ that are involved in self-joins, we can safely apply it to $\mathtt{R}$. For efficiency reasons, the isolated variables rule should be applied only if the generalized binomial rule is not applicable. In other words, we should apply the the generalized binomial rule followed by the isolated variables rule. In summary,

Theorem 9. *LIS augmented with the isolated variables rule yields an unbiased estimate of the partition function of its input MLN.*

Proof. We first show the correctness of Eq. (4.10). Let R be the predicate where the isolated variable rule is applied, let Δ_x be the possible groundings of isolated variables and Δ_y be the possible groundings of non-isolated variables of R. Grounding R with any $y \in \Delta_y$ results in a singleton atom with Δ_x groundings. From the generalized binomial rule, we can specify conditioning over this singleton using $|\Delta_x| + 1$ values. Thus, conditioning over R can be specified using all possible vectors of size $|\Delta_y|$, where each vector element varies from 0 to $|\Delta_x|$ as in Eq. (4.10). Since, $\prod_{i=1}^{|\Delta_y|} Q(j_i|j_1 \dots j_{i-1})$ is a distribution over these $|\Delta_y|(|\Delta_x|+1)$ assignments, it follows that Eq. (4.11) is an unbiased estimator of the partition function. Therefore, the theorem holds. □

Variance Reduction

Intuitively, the scheme that utilizes the most grouping is likely to have better accuracy because it samples a smaller (sub)space. We formalize this notion using the following grouping lemma:

Lemma 3 (**Grouping Lemma**). *Let Z be a sum over M numbers grouped into k groups such that all numbers in each group are identical. Let $(m_{1,1}, \dots, m_{1,g_1}, \dots, m_{k,1}, \dots, m_{k,g_k})$ denote an arbitrary ordering of the M numbers such that $\forall\ a, b, c$, $m_{a,b} = m_{a,c}$, where $a \in \{1, \dots, k\}$, $b, c \in \{1, \dots, g_a\}$ and g_a is the number of numbers in group a. Let Q be a proposal distribution defined over all the M numbers and R be a proposal distribution defined over the k groups such that $R(i) = \sum_{j=1}^{g_i} Q(m_{i,j})$. Then, the variance of the importance sampling estimate of Z defined with respect to R is smaller than the variance of the estimate of Z defined with respect to Q.*

Proof. Let f_Q and f_R be two estimators defined as follows.

$$f_Q = \frac{1}{T} \sum_{i=1}^{T} \left(\frac{m^{(t)}}{Q(m^{(t)})} \right)$$

where $m^{(t)}$ is drawn from Q

$$f_R = \frac{1}{T}\sum_{i=1}^{T}\left(\frac{m^{(t)}}{R(g^{(t)})}\right)$$

where $g^{(t)}$ is a group drawn from R and $m^{(t)} \in g^{(t)}$

Clearly, $\mathbb{E}_Q[f_Q] = Z$ and $\mathbb{E}_R[f_R] = Z$. That is, both Q and R yield unbiased estimates of Z. R is defined over k values while Q is defined over $\sum_{j=1}^{k} g_j$ values. Thus, each sample from R in expectation, is equivalent to $\frac{\sum_{j=1}^{k} g_j}{k}$ samples from Q. Therefore, drawing T samples from S is equivalent to drawing $\frac{T\sum_{j=1}^{k} g_j}{k}$ if we sample from R. Since the sample mean (expected value) for both f_Q and f_R is equal, we have,

$$\frac{Var(f_Q)}{Var(f_R)} = \frac{T\sum_{j=1}^{k} g_j}{kT}$$

.

Therefore, it follows that $Var(f_R) \leq Var(f_Q)$.

□

Since the isolated variables rule groups atoms (see Eq. (4.10)) that were not grouped in the original LIS algorithm, it directly follows from the grouping lemma that:

Theorem 10. *The variance of LIS (Algorithm 8) is reduced by augmenting it with the isolated variables rule.*

4.2.3 Constructing the Proposal Distribution

As mentioned earlier, the accuracy of any importance sampling algorithm depends on how close the proposal distribution is to the target distribution (the one represented by the MLN). Often, practical constraints dictate that the importance distribution should be polynomially specifiable (i.e., tractable) as well as easy to sample from. To construct such a tractable

distribution for MLNs, a natural choice is to use the generalized binomial rule approximately by relaxing the requirement that the atom must be a singleton. For example,

Example 11. *Consider our example MLN,* $\mathcal{M} = \{(\mathtt{R}(x, y) \wedge \mathtt{S}(y, z) \wedge \mathtt{T}(z, u), w)\}$. *Applying the approximate generalized binomial rule to* $\mathtt{R}(x, y)$*, we can rewrite the partition function as* $\sum_{i=0}^{|\Delta_x \times \Delta_y|} Z(\mathcal{M}|\bar{\mathtt{R}}^i)$. *Each MLN,* $\mathcal{M}|\bar{\mathtt{R}}^i$ *is tractable and therefore we can associate a tractable probability distribution, say* $Q(\mathcal{M}|\bar{\mathtt{R}}^i)$ *with each. The full proposal distribution is* $Q(i)Q(\mathcal{M}|\bar{\mathtt{R}}^i)$ *where* $Q(i)$ *is the distribution defined over* $|\Delta_x \times \Delta_y| + 1$ *points, where each* i*-th point corresponds to setting exactly* i *groundings of* $\mathtt{R}(x, y)$ *to* ***True*** *and the remaining to* ***False****.*

Although the approximate rule reduces the branching factor (of the search space) from $2^{|\Delta_R|}$ to $|\Delta_R| + 1$ for an atom $\mathtt{R}$, it is still infeasible when the number of atoms is large. In particular, we will assume that the proposal distribution is specified in the product form, i.e., a relational Bayesian network (Jaeger, 1997). Formally, given an ordered set of atoms $(\mathtt{R}_1, \ldots, \mathtt{R}_m)$, the proposal distribution is given by $\prod_{i=1}^{m} Q_i(\mathtt{R}_i|\mathtt{R}_1, \ldots, \mathtt{R}_{i-1})$. The space required by this product form will be $O(m[\max_i(|\Delta_{\mathtt{R}_i}|)]^m)$, where $\Delta_{\mathtt{R}_i}$ is the Cartesian product of arguments of $\mathtt{R}_i$. Therefore, in order to achieve polynomial complexity, we make the following conditional independence assumption: $\mathtt{R}_i$ is conditionally independent of all other atoms given k atoms from the set $\{\mathtt{R}_1, \ldots, \mathtt{R}_{i-1}\}$, where k is a constant. Thus, each component of the proposal distribution is of the form: $Q_i(\mathtt{R}_i|pa(\mathtt{R}_i))$ where $pa(\mathtt{R}_i) \subseteq \{\mathtt{R}_1, \ldots, \mathtt{R}_{i-1}\}$ and $|pa(\mathtt{R}_i)| \leq k$. We will refer to $pa(\mathtt{R}_i)$ as the parents of $\mathtt{R}_i$.

Algorithm 9 describes a recursive approach for constructing the proposal distribution using the ideas discussed above. The algorithm takes as input an MLN $\mathcal{M}$, a constant k that limits the parent size for each atom (in our experiments, we used $k = 2$), and the potential parent set $\mathbf{R}$. The algorithm first checks the base condition: if the MLN is empty, it returns a 1. Then, the algorithm checks if the MLN can be decomposed into (multiple)

Algorithm 9: Construct Proposal (CP)

Input: An normal MLN $\mathcal{M}$, an integer k and a set of atoms $\mathbf{R}$
Output: The structure of the proposal distribution Q

```
if M is empty then return 1
if M can be decomposed into m MLNS M_1, ..., M_k such that no two MLNs share any atoms then
    for i = 1 to m do
        CP(M_i, k, R)
    return 1
Heuristically select an atom R from M
Heuristically select k atoms from R as parents of R
// Construct Proposal over R
for every assignment to the groundings of pa(R) index by i do
    if R contains no isolated variables then
        Use the approximate generalized binomial rule to construct Q_i(R)
    else
        Use the isolated variables rule to construct Q_i(R)
Add R to R
Ground R and then remove it from all formulas of M
return CP(M, k, R)
```

independent MLNs (if two MLNs do not share any atoms, they are independent). If it can be decomposed, the algorithm recurses on the independent MLNs and exits. Then the algorithm heuristically selects an atom $\mathtt{R}_i$ and selects k atoms from the potential parent set $\mathbf{R}$ as parents of $\mathtt{R}$. It then constructs the proposal distribution component for $\mathtt{R}$ (described below), adds $\mathtt{R}$ to $\mathbf{R}$, reduces the MLN by removing $\mathtt{R}$ from all formulas and recurses on the reduced MLN.

The proposal distribution component for $\mathtt{R}$ is computed as follows. Given an assignment to all groundings of the parents, denoted by $\overline{pa(\mathtt{R})}$ each conditional marginal distribution $Q(\mathtt{R}|\overline{pa(\mathtt{R})})$ is constructed as follows. If $\mathtt{R}$ contains a set $\mathbf{x}$ of isolated variables, we use the following method. Let $\mathbf{y}$ denote the set of variables which are not isolated in $\mathtt{R}$. Note that to effectively utilize the isolated variables rule, we have to sample a number in the range $[0, |\Delta_{\mathbf{x}}|]$, for each value $\mathbf{Y} \in \Delta_{\mathbf{y}}$. We propose to express this distribution using a product

of $|\Delta_{\mathbf{y}}|$ marginal distributions, each defined over $|\Delta_{\mathbf{x}}| + 1$ points. Namely, using notation from the previous section, we define $Q_i(j_1, \ldots, j_{|\Delta_{\mathbf{y}}|}) = \prod_{a=1}^{|\Delta_{\mathbf{y}}|} Q_{i,a}(j_a)$. If $\mathtt{R}$ has no isolated atoms then we use the approximate generalized binomial rule and define a distribution over $|\Delta_A| + 1$ points. To limit the number of assignments $\overline{pa(\mathtt{R})}$ (see line 10 of Algorithm 9), we group the assignments to each atom $\mathtt{A} \in pa(\mathtt{R})$ into $|\Delta_{\mathtt{A}}| + 1$ groups, where the j-th group has j groundings of $\mathtt{A}_i$ set to `True` and the remaining to `False`. This helps us polynomially bound the space required by the proposal distribution component at $\mathtt{R}$. In particular, the space complexity of each component is $O(|\Delta_{\mathtt{R}}|(\sum_{\mathtt{A} \in pa(\mathtt{R})} |\Delta_A|))$.

We use the following heuristics to select the atom $\mathtt{R}$: Select any singleton atom. Otherwise, select an atom that participates in most formulas, ties broken randomly. This heuristic is inspired by the max-degree conditioning heuristic which often yields a smaller search space. To select parents of $\mathtt{R}$, we first select atoms, say $\mathbf{R}_1$, that are mentioned in the same formula that $\mathtt{R}$ participates in, followed by atoms which participate in formulas that atoms in $\mathbf{R}_1$ participate in and so on. Again, ties are broken randomly.

Until now, we have described an algorithm that outputs the structural form of the proposal distribution. To use it in LIS, we have to define its parameters. Moreover, we should define its parameters in such a way that the resulting distribution is as close as possible to the target distribution. In principle, we can use any approximate inference method such as lifted BP (Singla and Domingos, 2008) to compute the parameters. However, because of the relatively high time-complexity of lifted BP, this approach is not likely to be cost effective.

Therefore, we use the following adaptive importance sampling approach (Cheng and Druzdzel, 2000; Ortiz and Kaelbling, 2000) that dynamically updates the proposal distribution Q based on the generated samples. The updating step is performed every l samples. Since the initial proposal distribution, no matter how well chosen, is often very different from the target distribution, dynamic updating can substantially improve the accuracy of importance sampling. The hope is that as more and more samples are drawn, the updated

proposal distribution gets closer and closer to the target distribution. We initialize the proposal distribution to the prior distribution Q^0, defined by a collection of components Q_i^0 (for each atom `R` chosen in Algorithm 9). After every l samples, we update each component Q_i^m using the expression $Q_i^{m+1}(j) = Q_i^m(j) + \alpha(m)(\Pr(j) - Q_i^m(j))$ where $0 \leq \alpha(m) \leq 1$ is the learning rate and $\Pr(j)$ is the estimate of the probability of j based on the last l samples. In our experiments, we set $\alpha(m) = 0.1$ and $l = 10^3$.

4.2.4 Experiments

In this section, we compare the performance of LIS (based purely on PTP shown in Algorithm 8) with two advanced versions: (i) LIS augmented with the new lifting rule and (ii) LIS augmented with the new lifting rule and the adaptive structured method for constructing the proposal distribution described in the previous section. We will refer to PTP based LIS as LIS, LIS with isolated variables' rule as ILIS and adaptive LIS as ALIS respectively. Note that both LIS and ILIS use the same proposal distribution as the one used in (Gogate and Domingos, 2011a) while ALIS uses the structured, adaptive approach described in the previous section. We experimented with three MLNs: the example `R,S,T` MLN shown in Figure 4.7, the WEBKB MLN used in (Lowd and Domingos, 2007a) and the Entity resolution MLN used in (Singla and Domingos, 2006). The last two MLNs are publicly available from `www.alchemy.cs.washington.edu`. We set the weights of each formula in each MLN arbitrarily by sampling a value from the range $(-1, 1)$. For each MLN, we set 10% randomly selected ground atoms as evidence. We varied the number of objects in the domain from 100 to 300.

Because computing the partition function of the MLNs used is not feasible, evaluating the accuracy of this inference task is a hard problem. We use the following approach for evaluating the algorithms that is quite often employed in practice. Specifically, we use the sampling algorithms to compute a probabilistic lower bound on the partition function.

The higher the lower bound the better the sampling algorithm. For computing the lower bound, we combine our sampling algorithms with the *Markov inequality based minimum lower bounding scheme* presented in (Gogate et al., 2007). This lower bounding scheme, see also (Gomes et al., 2007), takes as input a set of unbiased estimates of the partition function and a real number $0 < \alpha < 1$, and outputs a lower bound on the partition function that is correct with probability greater than α. Formally,

Theorem 11. *(Gomes et al., 2007; Gogate et al., 2007) Let $\widehat{Z}_1, \ldots, \widehat{Z}_m$ be the unbiased estimates of Z computed over m independent runs of an importance sampling algorithm. Let $0 < \alpha < 1$ be a constant and let $\beta = \frac{1}{(1-\alpha)^{1/m}}$. Then $Z_{lb} = \frac{1}{\beta}\left[\min_{i=1}^{m}(\widehat{Z}_m)\right]$ is a lower bound on Z with probability greater than α.*

In our experiments, we set $\alpha = 0.99$ and $m = 7$, namely, we run each sampling algorithm 7 times and each lower bound is correct with probability greater than 0.99.

Figure 4.9 shows the impact of varying time and number of objects on the performance of the three algorithms. Note that the Entity Resolution MLN has no isolated variables and as a result LIS is equivalent to ILIS. Therefore, for this domain, we only compare LIS with ALIS. Also, note that we are plotting the log partition function as a function of time and therefore the Y-axis is in log-scale. From Figure 4.9, it is easy to see that ALIS is superior to ILIS which in turn is superior to LIS. Moreover, from the error bars in Figure 4.9, we see that the variance of ALIS and ILIS is typically smaller than that of LIS.

4.3 Summary

In this chapter, we developed two sampling based lifted approximate inference algorithms, namely, lifted blocked Gibbs (LBG) and lifted importance sampling (LIS). The common theme underlying both these algorithms was to exploit symmetries to sample from a *lifted state space*, i.e., a state space where symmetrical variables are sampled as a group to improve the scalability, accuracy and convergence of the samplers.

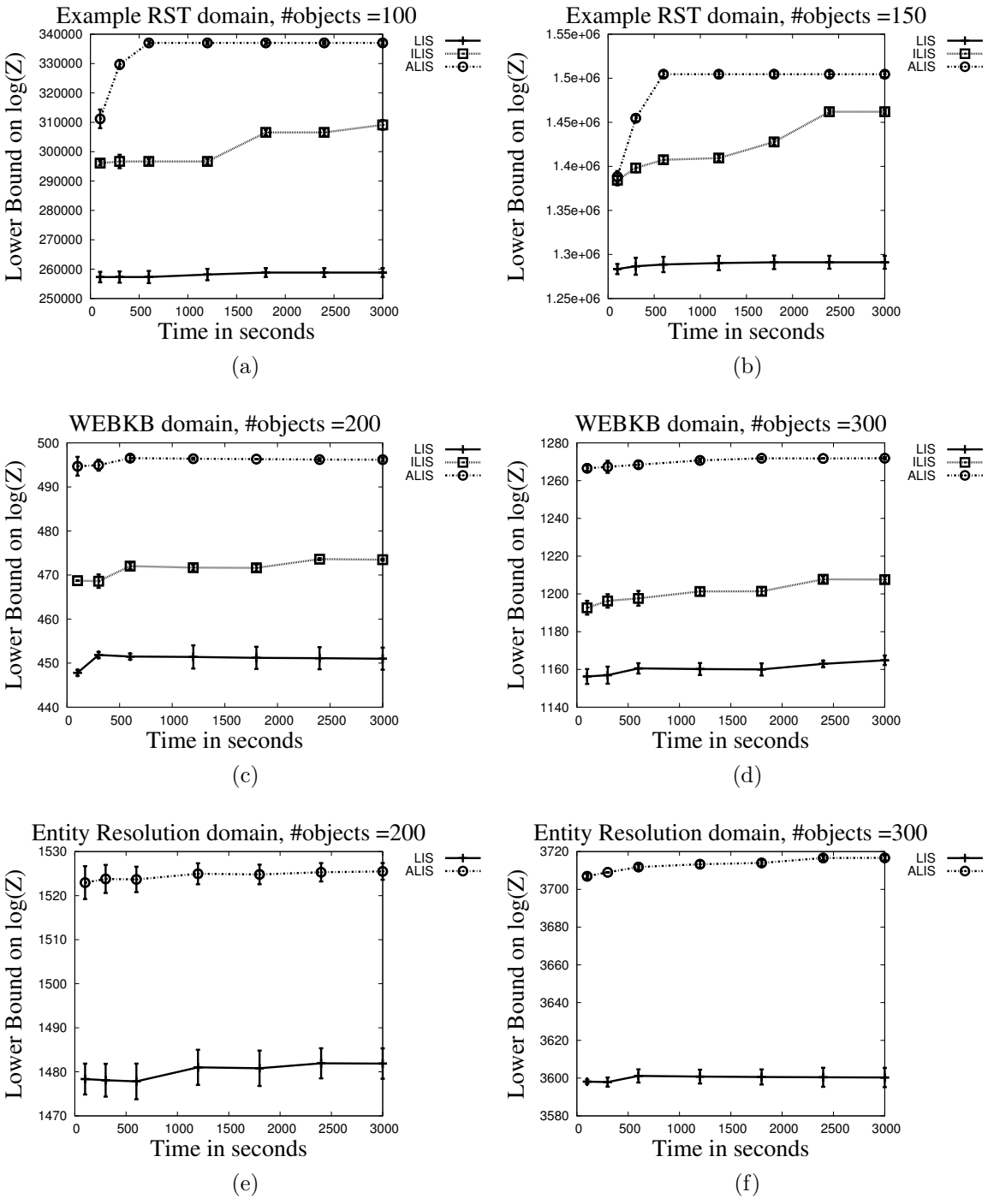


Figure 4.9. Lower bound on the partition function computed using LIS, ILIS and ALIS as a function of time. (a) The example $\mathtt{R}, \mathtt{S}, \mathtt{T}$ domain with 100 objects. (b) The example $\mathtt{R}, \mathtt{S}, \mathtt{T}$ domain with 150 objects. (c) WEBKB MLN with 200 objects, (d) WEBKB MLN with 300 objects, (e) Entity Resolution MLN with 200 objects and (f) Entity resolution MLN with 300 objects. Note that for each point, we have plotted error bars showing the standard deviation. When the standard deviation is small, the error bars are not visible in the plots.

LBG is a message-passing algorithm over clusters/blocks of first-order atoms carefully designed such that the MLN projected on each cluster is tractable for exact lifted inference. In sharp contrast to blocking over propositional variables, we showed that larger first-order blocks in some cases retain more symmetries and therefore reduces the complexity of LBG while improving accuracy. Further, we showed how to represent messages in our sampler in a lifted manner by taking advantage of relational structure in the MLN thereby reducing variance in our sampler. Our experiments on MLN benchmarks showed that LBG is much more accurate and scalable than propositional samplers.

In LIS, we constructed an accurate, lifted proposal distribution where each sample constitutes several distinct samples from an equivalent propositional proposal distribution. We showed that importance sampling using a lifted proposal distribution provably reduces the variance of estimates derived from the samples. To construct the lifted proposal distribution, we grouped together symmetric variables in the MLN by applying lifting rules from PTP. We pushed the boundaries of these rules by identifying a new lifting rule and showed that it can in some cases reduce the size of the lifted proposal distribution exponentially. We showed how to construct the lifted proposal distribution by tractably applying the lifting rules and improved the accuracy of our system by adaptively learning the parameters of the distribution. Our results on various benchmarks showed that our approach is much more accurate and scalable than previous approaches.

CHAPTER 5

EXPLOITING APPROXIMATE SYMMETRIES FOR SCALABLE INFERENCE

Lifted inference is extremely powerful when symmetries in the MLN can be gleaned using the first-order structure. For example, consider a simple MLN with a single unit-clause, $\mathtt{R}(x);w$. Here, irrespective of the domain that x can take, the full joint distribution of the MLN can be reduced to a distribution over a single object of the domain because every object is symmetrical with every other object of the domain. Using this, any kind of inference task (partition function, marginal inference or MAP inference) can be performed in constant time since we perform inference over the reduced distribution specific to a single object and project the same results to every other object. Much more importantly, we know how to reduce the size of the joint distribution by simply observing the syntax/structure of the formula. Most existing lifted inference algorithms focus on discovering such first-order syntax rules that can detect *exactly symmetrical* structures in the underlying ground representation of the MLN. Unfortunately, not all MLNs have first-order structures that allow the application of such rules. Particularly, since first-order logic is extremely expressive, application designers often create MLNs with arbitrary structures to model intricate background knowledge and it turns out that on such structures lifted inference is more or less as scalable as propositional inference. To address this problem, in this chapter, we present a much more general, practical framework where we use approximate symmetries to scale up inference on those MLNs that do not show exploitable exact symmetries in its first-order structure. We refer to this as *approximate lifting*. To place this approach into context, the lifted inference techniques presented in the previous chapter can be regarded as methods

that did not change the underlying distribution of the MLN. That is, even though they yield approximate results, they keep the underling MLN distribution invariant. In contrast, using approximate symmetries modifies the underlying distribution in lieu of scalability.

The rest of this chapter is organized as follows. We first describe common problems with most existing lifted inference algorithms. We then present our general approach where we learn approximate symmetries by leveraging standard machine learning algorithms and use these symmetries to develop a family of approximately lifted, scalable inference algorithms. Finally, we describe an importance sampler that utilizes approximate symmetries for scalability and its main virtue is that it controls the bias induced by approximate lifting yielding provable asymptotic guarantees on its inference results.

5.1 Grounding and Evidence Problems

With our current understanding of lifted inference, the types of MLNs from which exact symmetries can be deciphered is fairly limited. That is, most lifted inference algorithms impose certain restrictions or rules on the first-order structure of the MLN and only those MLNs whose structure satisfies these rules can be processed in a lifted manner. For instance, (i) Singla and Domingos (Singla and Domingos, 2008) lift belief propagation by identifying indistinguishable messages based on certain structural properties (ii) Jha et al. (Jha et al., 2010) introduced rules that can perform exact conditioning efficiently on singleton atoms, (iii) Broeck (Van den Broeck, 2011) proved rules that the class of theories where each formula has at most 2 logical variables is domain liftable for exact inference and (iv) Sarkhel et al.(Sarkhel et al., 2014b) proved that a *non-shared* MLN is liftable for MAP inference. In general, all the aforementioned rules work extremely well in specific cases but on general MLN structures, where such rules are not applicable, lifted inference ends up grounding a large part of the MLN and thus has the same scalability problems as propositional inference. We call this the *grounding problem*.

A second, even more serious problem with lifted inference is the *evidence problem*, i.e., the power of lifted inference diminishes with evidence (Van den Broeck and Darwiche, 2013). Specifically, unlike propositional inference where evidence helps prune the model, from the perspective of lifted inference, evidence breaks symmetries and is therefore detrimental to lifting. Thus, in many cases, even though the MLN is liftable without evidence, upon observing evidence, the MLN no longer remains liftable due to the absence of exact symmetries. As a concrete example, consider the MLN in Figure 5.1. The marginal probabilities of the ground atoms before any evidence is given to the MLN is shown in Figure 5.1 (a). As seen here, the marginal probabilities are symmetrical to each other. However, when presented with the evidence shown in (b), the symmetries are broken as shown in (c). Therefore, a lifted algorithm that could potentially exploit the symmetry in (a) can no longer do so in (c). In general, in the presence of evidence, lifted inference algorithms often resort to grounding the MLN affecting their scalability. This is extremely problematic from a practical perspective because most interesting inference problems are almost always of the form $P(Q|E)$, i.e., computing the probability of a query given evidence.

To summarize, for arbitrarily structured MLNs or arbitrarily structured evidence which are both cases often encountered in practice, typical lifted inference approaches are just as scalable as propositional inference. In this chapter, we address both these problems. That is, both MLN as well as evidence structure *does not significantly affect* the scalability of our proposed inference methods.

5.2 Approximate Lifting using Evidence-based Clustering

The evidence and grounding problems suggest that existing lifted inference methods are in some ways overly restrictive and to scale up inference over real-world problems, we need to generalize the "symmetry-exploiting" idea of lifted inference. Here, we do this by leveraging approximate symmetries in the MLN. Specifically, we perform inference over groups

Wins(A,A)	0.56
Wins(A,B)	0.56
Wins(A,C)	0.56
Wins(B,A)	0.56
Wins(B,B)	0.56
Wins(B,C)	0.56
Wins(C,A)	0.56
Wins(C,B)	0.56
Wins(C,C)	0.56

(a) Original Marginals

Strong(C)
Wins(A,C)
Wins(B,B)
Wins(B,C)
Wins(C,A)

(b) Evidence

Wins(A,A)	0.6
Wins(A,B)	0.6
Wins(B,A)	0.63
Wins(C,B)	0.85
Wins(C,C)	0.85

(c) New Marginals

Figure 5.1. Effect of evidence on an MLN with one formula, 1.75 $\texttt{Strong}(x) \Rightarrow \texttt{Wins}(x,y)$. The marginal probabilities which were equal in (a) become unequal in (c) due to evidence (b).

of approximately symmetrical objects that we learn using unsupervised machine learning techniques. In doing so, we approximate the original MLN with a *compressed MLN* that approximates the original distribution and thus, results obtained by performing inference on the smaller MLN are as close as possible to the ones obtained by running an expensive inference algorithm on the original MLN. To achieve this compression, we pre-process the MLN utilizing standard clustering algorithms such as K-Means to merge approximately symmetrical objects, namely, objects that are similar to each other from an inference perspective. Importantly, this pre-processing step allows us to plug-in any existing inference algorithm and control the complexity of inference thereby allowing us to develop a family of approximately lifted algorithms.

In order to learn an accurate domain-reduced approximation of the original MLN, we specify a novel distance function that measures similarity based on the evidence presented to the MLN. This distance function helps cluster together objects having similar evidence-structure. The inherent symmetry in MLN representation makes it more likely that similar evidence structure translates to approximately similar marginal probabilities. Thus, we compute the marginal probability for a single element of the cluster and project the same results to all elements in the cluster, thereby significantly reducing the complexity of inference.

We evaluate our approach on several benchmark MLNs available on the Alchemy website (Kok et al., 2008). Also, in our experiments, we leverage a number of clustering algorithms from data-mining/machine learning literature implemented in Weka (Hall et al., 2009) to scale-up inference to very large domain-sizes. To show the generality of our approach, we experimented with two inference algorithms, a sampling based approach, namely, Gibbs sampling (Geman and Geman, 1984) and a variational inference approach, namely, belief propagation (Murphy et al., 1999; Singla and Domingos, 2008). Our results clearly illustrate that, using a fraction of the true groundings, we are able to approximate the marginal probabilities quite consistently on a wide variety of MLN structures with arbitrary evidence.

5.2.1 Input Specification

Recall that PTP and other lifted inference algorithms presented in the previous chapter assumed that the input MLNs were in a canonical form called the *normal form* (Jha et al., 2010) (Section 2.1.4). Normal forms are similar to other canonical forms typically used in lifted inference in algorithms such as FOVE (de Salvo Braz, 2007) and WFOMC (Van den Broeck et al., 2011). It turns out that the normal form restriction affects the performance of lifted inference when evidence is presented to the MLN. In fact, the evidence problem manifests itself during the process of normalizing the MLN. Specifically, given evidence, we perform a process referred to as *shattering* (de Salvo Braz, 2007; Van den Broeck and Darwiche, 2013) to ensure that we separate the formulas with known evidence atoms from the rest of the MLN. Unfortunately, shattering can increase the size of the MLN dramatically. This is illustrated by the following example.

Example 12. *Consider the MLN with one formula* $\neg$`Friends`$(x,y) \vee$ `Related`$(y,z) \vee$ `Likes`(z,x) *;* w*. Let us assume that* $\Delta_x = \Delta_y = \Delta_z = \{A, B, C\}$*. Assume that we obtain just two pieces of true evidence,* `Friends`(A,B) *and* `Likes`(C,B)*. The normal form for this MLN and evidence is given by the following formulas.*

$\neg$*Friends*$_1(A,B) \vee$ *Related*$_1(B,z) \vee$ *Likes*$_1(z,A)$; w

$\neg$*Friends*$_2(A,y_1) \vee$ *Related*$_2(y_1,z) \vee$ *Likes*$_1(z,A)$; w

$\neg$*Friends*$_3(B,B) \vee$ *Related*$_3(B,C) \vee$ *Likes*$_2(C,B)$; w

$\neg$*Friends*$_3(B,B) \vee$ *Related*$_4(B,z_1) \vee$ *Likes*$_2(z_1,B)$; w

$\neg$*Friends*$_4(x_1,B) \vee$ *Related*$_3(B,C) \vee$ *Likes*$_3(C,x_1)$; w

$\neg$*Friends*$_5(C,B) \vee$ *Related*$_4(B,z_1) \vee$ *Likes*$_4(z_1,C)$; w

$\neg$*Friends*$_6(B,y_1) \vee$ *Related*$_5(y_1,C) \vee$ *Likes*$_2(C,B)$; w

$\neg$*Friends*$_7(C,y_1) \vee$ *Related*$_6(y_1,z_1) \vee$ *Likes*$_4(z_1,C)$; w

$\neg$*Friends*$_6(B,y_1) \vee$ *Related*$_6(y_1,z_1) \vee$ *Likes*$_2(z_1,B)$; w

$\neg$*Friends*$_7(C,y_1) \vee$ *Related*$_5(y_1,C) \vee$ *Likes*$_5(C,C)$; w

where $\Delta_{x_1} = \{B, C\}$, $\Delta_{y_1} = \{A, C\}$ *and* $\Delta_{z_1} = \{A, B\}$

Thus as the evidence grows, each predicate is shattered into multiple predicates as shown in the above example. Therefore, evidence-instantiated MLNs become more and more propositional as the evidence increases and the power of lifted inference diminishes. To ensure scalability of inference in the presence of arbitrary evidence, here, we introduce a new canonical form which is a superset of normal MLNs called Σ-normal MLNs that separates evidence representation from the MLN representation. Specifically,

Definition 14. *$\mathcal{M}$ is a Σ-normal MLN iff $\mathcal{M}$ is a normal MLN in the absence of any evidence.*

The subtle distinction between normal and Σ-normal MLNs has the following important consequence. As the evidence presented to the MLN is modified, the structure of a normal MLN changes, i.e., new predicates and formulas are added to the MLN to *normalize* it, as illustrated in the Example 12. In contrast, the structure and size of a Σ-normal MLN is unaffected by any arbitrary evidence on the MLN. Thus, the MLN continues to be processed in lifted form even with the introduction of arbitrary evidence, thereby allowing us to scale up to much larger MLNs with evidence.

5.2.2 Problem Formulation

Let $\mathcal{M}$ denote a Σ-normal MLN with M predicates $\mathtt{R}_1$, $\mathtt{R}_2$, ..., $\mathtt{R}_M$, and N weighted formulas f_1, f_2, ..., f_N. Let $G_{\mathcal{M}}$ denote the propositional Markov network obtained by grounding all the formulas in $\mathcal{M}$. Let $\mathbf{E} = \{E_k\}_{k=1}^{S}$ be the set of *evidences*. Each $E_k \in \mathbf{E}$ represents a single ground atom that is known to be either `True` or `False`. Let $\mathbf{I}$ be a set of indexes of the form (i, j) such that $1 \leq i \leq M$, $1 \leq j \leq A_i$, where A_i is the arity of the i-th predicate. In other words, (i, j) is an index to the j-th argument of the i-th predicate in $\mathcal{M}$.

Let R be a binary relation on $\mathbf{I}$ such that (i, j) R (a, b) iff there exists a formula $f \in \mathcal{M}$ such that: (1) f contains atoms having predicate symbols indexed by i and a, and (2) a logical variable x of f appears as the j-th argument and as the b-th argument of atoms having predicate symbols indexed by i and a respectively. Clearly, R is symmetric and reflexive. Let R^+ be the transitive closure of R on $\mathbf{I}$. R^+ is an equivalence relation on $\mathbf{I}$. Let $\mathcal{I} = \{\mathcal{I}_1$ $\mathcal{I}_2$ ... $\mathcal{I}_P\}$ denote the set of equivalence classes of $\mathbf{I}$ due to the equivalence relation R^+. Let $\Delta_{\mathcal{I}_k}$ denote the domain (possible groundings) of an element of $\mathcal{I}_k$. Note that since we assume that the MLN is in Σ-normal form, all elements of $\Delta_{\mathcal{I}_k}$ have the same domain.

Example 13. *Let $\mathcal{M}$ contain exactly one formula $\mathtt{R}_1(x,y) \wedge \mathtt{R}_2(y,z) \Rightarrow \mathtt{R}_3(z,x)$. Let $\Delta_x = \Delta_y = \Delta_z = \{A,B\}$. $\mathcal{I} = \{\{(1,1), (3,2)\}, \{(1,2), (2,1)\}, \{(2,2), (3,1)\}\}$. $\Delta_{\mathcal{I}_1} = \{A,B\}$ and grounding $\mathcal{I}_1$ with A, yields the partially ground formula, $\mathtt{R}_1(A,y) \wedge \mathtt{R}_2(y,z) \Rightarrow \mathtt{R}_3(z,A)$.*

To reduce the total number of formulas in $G_{\mathcal{M}}$, we reduce the number of groundings in each $\mathcal{I}_k \in \mathcal{I}$ independently. Specifically, for each $\Delta_{\mathcal{I}_k}$, we learn a new domain, $\hat{\Delta}_{\mathcal{I}_k}$ and a surjective mapping $\zeta : \Delta_{\mathcal{I}_k} \rightarrow \hat{\Delta}_{\mathcal{I}_k}$, i.e., $\forall\ \mu \in \hat{\Delta}_{\mathcal{I}_k}$, $\exists\ C \in \Delta_{\mathcal{I}_k}$ such that $\zeta(C) = \mu$. We formulate this domain-reduction problem ($|\hat{\Delta}_{\mathcal{I}_k}| << |\Delta_{\mathcal{I}_k}|$) as a standard clustering problem below.

Definition 15. *Given a distance measure d between any two groundings of $\mathcal{I}_k \in \mathcal{I}$ and the number of clusters for $\mathcal{I}_k$ equal to r_k, we define the clustering problem as,*

$$\min_{\mathbf{C}_1 \ldots \mathbf{C}_P} \sum_{k=1}^{P} \sum_{j=1}^{r_k} \sum_{C_{kj} \in \mathbf{C}_{kj}} d(C_{kj}, \mu_{kj}) \tag{5.1}$$

where $\mathbf{C}_{kj}$ corresponds to all groundings of $\mathcal{I}_k$ that are placed in cluster j, μ_{kj} is the cluster-center of $\mathbf{C}_{kj}$, i.e., it represents the "average grounding" for that cluster, $\zeta^{-1}(\mu_{kj}) = \mathbf{C}_{kj}$.

Each cluster-center in some sense "compresses" the original domain, and we generate a new MLN $\hat{\mathcal{M}}$ from $\mathcal{M}$ by replacing each $\Delta_{\mathcal{I}_k}$ with $\hat{\Delta}_{\mathcal{I}_k} = \{\mu_{kj}\}_{j=1}^{r_k}$. Importantly, the formulation in Eq. (5.1) allows us control the inference-complexity in $\hat{\mathcal{M}}$ even when $G_{\mathcal{M}}$ is extremely large. For example, consider the MLN, $\mathtt{R}(x,\, y) \wedge \mathtt{S}(y,\, z) \Rightarrow \mathtt{T}(z,\, x)\ w$, even for $\Delta_x = \Delta_y = \Delta_z = \Delta_u = 100$, the number of formulas in $G_{\mathcal{M}}$ is already one million. Further, the state space is massive, i.e., exponential in the total number of ground atoms in the MLN. By clustering, we are essentially lifting this large space approximately and now any existing inference algorithm implicitly works in the lifted space. The complexity of this lifted space can be controlled by the number of specified clusters. Specifically, given $\mathcal{M}$ with M predicates where A is an upper bound on the arity of a predicate, if r is an upper bound on the number of clusters, i.e., $r = \max_k r_k$, then the approximately lifted state space has a complexity $O(\exp(Mr^A))$.

5.2.3 Evidence Approximation

Clearly, the ground atoms in $\hat{\mathcal{M}}$ are different from those in $\mathcal{M}$. Specifically, an atom in $\hat{\mathcal{M}}$ is ground with cluster-centers rather than constants from the original MLN. Thus, one ground atom in $\hat{\mathcal{M}}$ implicitly corresponds to multiple ground atoms in $\mathcal{M}$. This also means that in $\hat{\mathcal{M}}$, the original evidence $\mathbf{E}$ needs to be modified because it is specified on the ground atoms

of $\mathcal{M}$. Therefore, we approximate $\mathbf{E}$ with $\hat{\mathbf{E}}$ which specifies the evidence on atoms ground with cluster-centers instead of the original constants in $\mathcal{M}$ as follows.

Definition 16. *The* expansion *of the j-th ground atom corresponding to the i-th predicate ($\mathtt{R}_i(\mu_{i_1 j_1}, \ldots \mu_{i_{A_i} j_{A_i}})$) in $\hat{\mathcal{M}}$ is denoted by π_{ij} and consists of all distinct ground atoms of the form $\mathtt{R}_i(C_1, \ldots, C_{A_i})$ where $C_k \in \zeta^{-1}(\mu_{i_k j_k})$.*

Clearly, if we assert in $\hat{\mathbf{E}}$ that a ground atom in $\hat{\mathcal{M}}$ is `True` (or `False`), this implicitly asserts that every grounding in its expansion is `True` (or `False`). To best approximate $\mathbf{E}$ for $\hat{\mathcal{M}}$, we choose $\hat{\mathbf{E}}$ to minimize the following approximation error.

$$\min_{\hat{\mathbf{E}}} |\mathbf{E} \triangle \bar{\pi}(\hat{\mathbf{E}})| \tag{5.2}$$

where $\hat{\mathbf{E}}$ is a subset of the ground atoms in $\hat{\mathcal{M}}$ and each grounding is assigned a sign (positive/`True` or negative/`False`), $\bar{\pi}(\hat{\mathbf{E}})$ expands every grounding in $\hat{\mathbf{E}}$ and assigns each grounding in the expansion the same sign as its corresponding grounding in $\hat{\mathbf{E}}$. The $\triangle$ operator computes the symmetric difference between $\mathbf{E}$ and $\bar{\pi}(\hat{\mathbf{E}})$. (Note that a grounding with different signs is treated as distinct elements for our purpose). $\hat{\mathbf{E}}$ can be optimally chosen among all the ground atoms of $\hat{\mathcal{M}}$ as follows.

Proposition 2. *Eq.* (5.2) *is minimized if a) $\hat{E} \in \hat{\mathbf{E}}$ and $\hat{E}$ is positive implies that $n_+ \geq \frac{|\mathbf{S}|}{2}$ or b) $\hat{E} \in \hat{\mathbf{E}}$ and $\hat{E}$ is negative implies that $\left(n_- \geq \frac{|\mathbf{S}|}{2}\right)$, where $\mathbf{S}$ is the expansion of $\hat{E}$, n_+ is the number of positive evidences in $\mathbf{S} \cap \mathbf{E}$ and n_- the number if negative-evidences in $\mathbf{S} \cap \mathbf{E}$.*

Essentially, $\hat{E}$ that is obtained using Proposition 2 can be viewed as an approximation that makes $\mathbf{E}$ more symmetrical. Thus, it has the effect that minor variations in the true marginal probabilities are smoothed after evidence approximation. We show this visually with the following example.

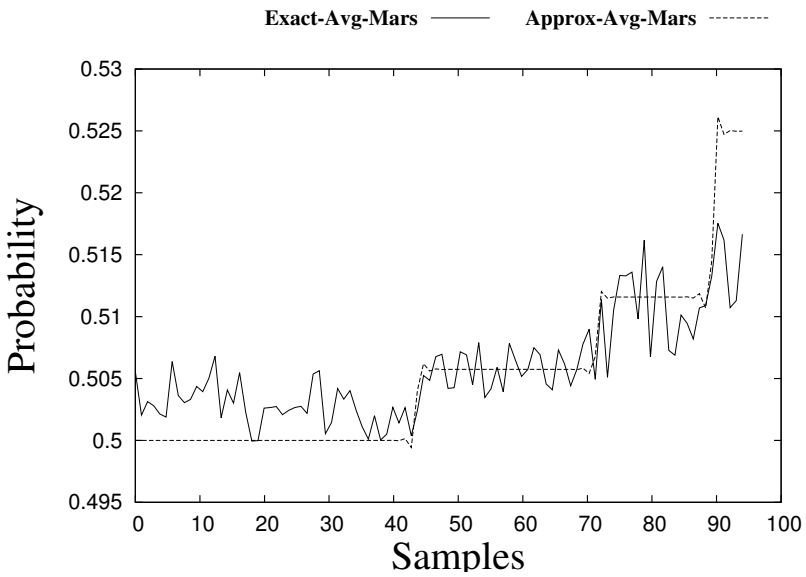


Figure 5.2. Illustrating evidence approximation for $\neg$`Studies`(x, y) $\vee$ `Teaches`(y, z) $\vee$ `Student` (z, x); 0.75. For varying combinations of samples on `Student` and `Studies`, the average true and approximate marginals (after evidence approximation) for all groundings of `Teaches` are plotted. As seen here, evidence approximation smooths out the variations in the true marginals by introducing additional symmetries in the evidence.

Example 14. *Let the MLN contain one formula* $\neg$***Studies****(x, y)* $\vee$ ***Teaches****(y, z)* $\vee$ ***Student****(z, x); 0.75. Let us assume that* ***Teaches*** *is the query atom and let* $|\Delta_x| = |\Delta_y| = |\Delta_z| = 3$*. Figure 5.2 plots the average of the true marginals and the average of the approximate marginals for* ***Teaches*** *for varying samples of positive and negative evidences on* ***Studies*** *and* ***Student****. As shown, the approximate evidence smooths out variations in the true marginals by introducing new symmetries in the evidence.*

5.2.4 Algorithm Specification

Algorithm 10 shows a schematic illustration of our algorithm to compute the marginal probabilities in an MLN given evidence. Algorithm 10 needs three other algorithms to be specified namely, the distance function, clustering algorithm and the inference algorithm. The amount of reduction applied to each domain is specified as the cluster-bound α.

Algorithm 10: Compute-Marginals

Input: MLN $\mathcal{M}$, Evidence $\mathbf{E}$, set of query predicates $\mathbf{Q}$, Distance function d, Clustering function $\mathcal{L}$, Inference algorithm $\mathcal{F}$, cluster-bound α

Output: Marginal probabilities $\mathcal{P}$ for each ground atom corresponding to a predicate in $\mathbf{Q}$

```
Compute the partition 𝓘 from 𝓜
𝓜̂ = 𝓜
for 𝓘_k ∈ 𝓘 do
    numclusters = α × Δ_{𝓘_k}
    (Δ̂_{𝓘_k}, ζ) = 𝓛(numclusters, d)
    Replace Δ_{𝓘_k} with Δ̂_{𝓘_k} in 𝓜̂
Construct Ê based on Proposition 2
𝒫̂ = 𝓕(𝓜̂, Ê, Q)
for Each R_k ∈ Q do
    for Each j, where j indexes the possible groundings of R_k in 𝓜̂ do
        for Each t, where t indexes the possible groundings of R_k in the expansion π_kj
        do
            𝒫(R_k, t) = 𝒫̂(R_k, j)
return 𝒫
```

Algorithm 10 starts by computing the partition $\mathcal{I}$ from the term dependencies in $\mathcal{M}$. Next, to each $\mathcal{I}_k \in \mathcal{I}$, the clustering algorithm $\mathcal{L}$ is applied which outputs the clustered domain $\Delta_{\mathcal{I}_k}$ as well as the mapping function ζ. $\Delta_{\mathcal{I}_k}$ is now replaced by its approximation in the new MLN $\hat{\mathcal{M}}$. Once all the domains are suitably reduced, the next step is to approximate the evidence based on the reduced domains. Using Proposition 2, for every grounding of every atom in $\hat{\mathcal{M}}$, we make a decision as to whether it is to be considered positive evidence, negative evidence or treated as a grounding whose truth value is unknown. This yields the approximate evidence set $\hat{\mathbf{E}}$. We then invoke the inference algorithm $\mathcal{F}$ to compute the marginals in $\hat{\mathcal{M}}$. Finally, we project the results obtained on $\hat{\mathcal{M}}$ back to the original domains. Specifically, if a grounding in $\hat{\mathcal{M}}$ has a marginal probability p, then each grounding in its expansion is assigned the same probability.

5.2.5 Evidence Based Distance Function

The distance function is a key parameter that affects the quality of the generated clusters in Eq. (5.1) and in turn the inference results computed in Algorithm 10. The advantage of our formulation is that it is quite easy to plug-in a new distance function and generate "new" inference algorithms targeted towards specific applications or datasets. Here, we develop a novel distance measure that uses the evidence presented to the MLN as features for clustering. We explain the intuition behind this distance function using the following two examples.

Example 15. *Consider an example MLN with one formula,* $\neg$`Smokes`$(x) \vee \neg$`Friends`$(x,y) \vee$ `Asthma`(y); 0.75. *Let* $\Delta_x = \Delta_y = \{A, B, C\}$. *Figure 5.3 shows several plots where, in each plot a different set of evidences on* `Smokes`(x) *and* `Asthma`(y) *is presented to the MLN and the resulting marginals for every ground atom of* `Friends`(x, y) *are plotted. As seen from the figure, several marginal probabilities turn out the be clustered together. We use the evidence information to design the distance function such that it helps learn these clusters.*

Example 16. *For the MLN in Figure 5.3, consider an example evidence instance where* `Smokes`(A) = `False` *and* `Smokes`(C) = `False`. *Using the evidence, we can assert that the formula* $\neg$`Smokes`$(A) \vee \neg$`Friends`$(A,y) \vee$ `Asthma`(y) *is satisfied in 3 groundings and the formula* $\neg$`Smokes`$(C) \vee \neg$`Friends`$(C,y) \vee$ `Asthma`(y) *is also satisfied in 3 groundings. The marginal probabilities for* `Friends`(A,y) *turn out to be the same as the marginals for* `Friends`(C,y), *equal to 0.5, while the marginal probabilities for* `Friends`(B,y) *is equal to 0.46. Thus ideally, the objects A and C should be clustered together since the evidence on them is "symmetrical" while the the object B should be in a separate cluster since its evidence structure is different. We formalize this example below.*

Let $\mathcal{M}_{C_{kj}}$ represent the MLN obtained after grounding $\mathcal{I}_k$ with the j-th constant in $\Delta_{\mathcal{I}_k}$. Clearly, in the general case, for any two distinct j_1, j_2, $\mathcal{M}_{C_{kj_1}}$ and $\mathcal{M}_{C_{kj_2}}$ are not

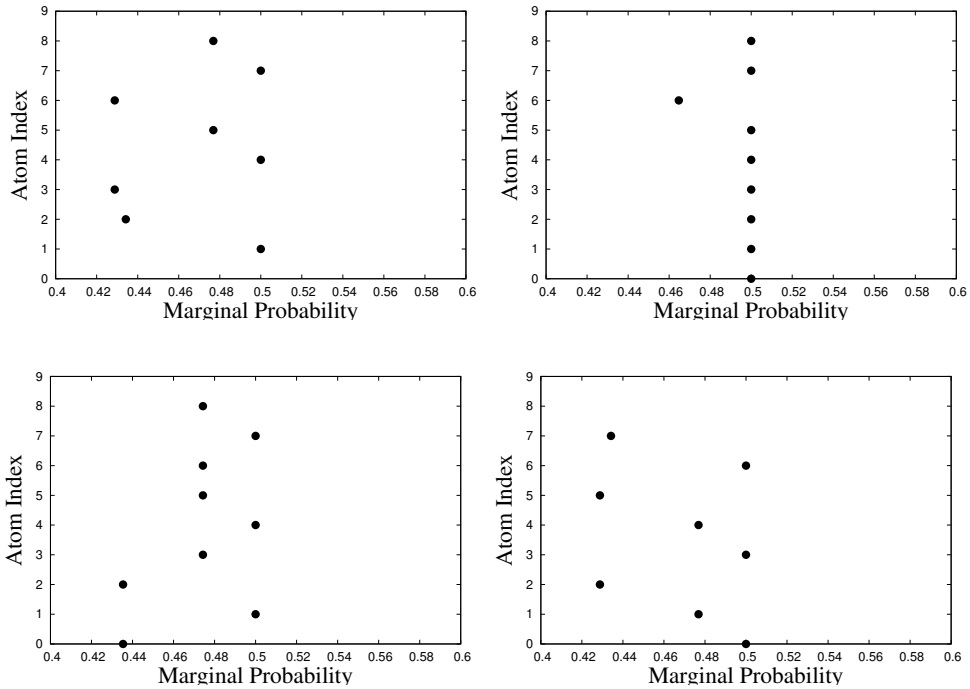


Figure 5.3. Illustrating clusters of marginal probabilities when given different evidence instances. For the MLN with 1 formula, $\neg$`Smokes`$(x) \vee \neg$`Friends`$(x,y) \vee$ `Asthma`(y); 0.75, where each logical variable has a domain-size equal to 3. The x-axis specifies the probabilities and the y-axis shows the index of a ground atom of `Friends` (0-9). In each figure, we input different evidences for `Smokes`(x) and `Asthma`(y), and plot the resulting marginal probabilities for every ground atom of `Friends`(x, y). Our distance function tries to cluster together atoms with similar marginals.

necessarily independent MLNs as there may be atoms in $\mathcal{M}_{C_{kj_1}}$ that are also present in $\mathcal{M}_{C_{kj_2}}$. However, in our distance function, we relax the constraints/dependencies between $\mathcal{M}_{C_{kj_1}}$, $\mathcal{M}_{C_{kj_2}}$ and assume these to be independent MLNs and compute the distance between these two MLNs. Specifically, we define a feature vector $\mathbf{U}_{C_{kj}} = c_{f_1}, \ldots c_{f_N}$, where c_{f_k} is the number of groundings in formula f_k of MLN $\mathcal{M}_{C_{kj}}$ satisfied due to the evidence $\mathbf{E}$. The distance is computed as $d(C_{kj_1}, C_{kj_2}) = ||\mathbf{U}_{C_{kj_1}} - \mathbf{U}_{C_{kj_2}}||$.

Even though the above distance function seems like an intuitive and reasonable heuristic, it turns out that computing the distance function efficiently is infeasible in the general case because computing the counts in the feature vector, $\mathbf{U}_{C_{kj}}$ is a hard problem when $\mathbf{E}$

is large. Specifically, the problem of computing the vector in $\mathbf{U}_{C_{kj}}$ for a Σ-normal MLN can be converted to sequence of queries in a database that contains evidence on ground atoms (details in the next section). The following theorem is a classical result in database theory (Papadimitriou and Yannakakis, 1999) that has also been revisited in (Domingos and Lowd, 2009) for MLNs,

Theorem 12. *Computing the number of satisfied groundings of a first-order clause in a database is #P-complete in the length of the clause.*

Thus by Theorem 12, computing $\mathbf{U}_{C_{kj}}$ exactly is hard. Therefore, we relax the joint-dependencies on the relations of the formula to keep the distance function computation feasible in the general case.

Bounded-join Queries on a Relational Database

We formalize the distance function using relational databases. From an implementation point of view, using databases implicitly gives us all the advanced query optimization and scalability features that are built into most modern relational databases. Also, relational databases tie in nicely with our input specification. Recall that we assume the MLN to be in Σ-normal form. In normal forms, every time the MLN is instantiated with new evidence, we need to pre-process the evidence and shatter the MLN that typically results in much larger new MLN with several new predicates. Thus specifying a normal MLN as database is harder since the MLN itself changes for every new instantiation of evidence. However, for Σ-normal MLNs, we simply store the evidence associated with each first-order predicate as the database and let the internal database query engine perform efficient joins using the stored evidence as needed since the MLN structure is always assumed to be fixed.

We now specify the schema of our database as follows. Each ground atom of the Σ-normal MLN is stored in the database along with its state. Specifically, the i-th predicate $\mathtt{R}_i$

is stored as a relational database table R_i with $A_i + 1$ columns (A_i is the arity), namely, id_1,, id_2 ... id_{A_i} and val. The first A_i columns correspond to a specific grounding and the val column specifies the state of the ground atom, i.e., whether that ground atom is evidence (`True` evidence ($val = 1$), `False` evidence ($val = 0$)) or its truth assignment is unknown ($val = -1$).

Given the above relational database, it turns out that computing the feature vectors in the distance function is simply a series of conjunctive queries to the relational database (we illustrate this in the example below) where each query is a hard problem (#P) as shown in Theorem 12. To scale up inference to arbitrarily large evidence sets and arbitrarily large formulas, we adopt the following approach. Instead of computing the exact number of groundings for a formula satisfied by the evidence, which involves an arbitrary number of joins over the relations in the formula, we approximate this with a vector of counts, where each count is computed on a subset of relations and the computation involves a bounded number of joins over these relations. We illustrate this in the following examples.

Example 17. *Let $\mathcal{M}$ contain one formula, $\neg$R(x, y) $\vee$ $\neg$S(y,z) $\vee$ T(z,x), where $\Delta_x = \{A, B, C\}$. To compute the count of satisfied groundings for $x = A$, we compute its inverse, i.e., the number of unsatisfied groundings for $x = A$. The satisfied count is simply the difference between the total number of groundings and the number of unsatisfied groundings. Since the total number of groundings $\Delta_y \times \Delta_z$ is a constant for all groundings of x, it does not affect the clustering and we simply ignore it. Computing the unsatisfied groundings for $x = A$ is in the class of conjunctive queries for databases and can be specified by the following relational algebra expression*

$$\sigma_{R.val=1 \wedge S.val=1 \wedge T.val=0}((\sigma_{R.id_1=A}(R) \bowtie_{R.id_2=S.id_1} S) \bowtie_{S.id_2=T.id_1 \wedge R.id_1=T.id_2} T) \tag{5.3}$$

where σ is the selection operator and $\bowtie$ is the join operator. Clearly, the above expression has two joins. However, if we impose a constraint that no joins are allowed during the

Algorithm 11: Build-Query

Input: Clausal formula f_t
Output: Relational-Algebra expression $\mathcal{Q}$

```
1 Q = ∅
2 for R_i ∈ f_t do
3     Rvalue = 1
4     if R_i is positive then
5         Rvalue = 0
6     if Q = ∅ then
7         Q = Q + σ_{R_i.val=Rvalue}(R_i)
8     else
9         Q = Q ⋈_θ σ_{R_i.val=Rvalue}(R_i)
```

computation of the feature vector, we approximate Eq. (5.3) *by implicitly assuming that each predicate in the formula is independent, i.e., we ignore the joins to obtain a vector of counts by counting the tuples returned by 3 separate queries,* $\sigma_{R.val=1 \wedge R.id_1=A}(R)$, $\sigma_{S.val=1}(S)$ *and* $\sigma_{T.val=0 \wedge T.id_2=A}(T)$. *An alternate distance function can be obtained if we only allow exactly one join in a query. In this case, we can get a better approximation of Eq.* (5.3) *by considering two queries,*

$$\sigma_{R.id_1=A \wedge R.val=1}(R) \bowtie_{R.id_2=S.id_1} \sigma_{S.val=1}(S)$$

$$\sigma_{S.val=1}(S) \bowtie_{S.id_2=T.id_1} \sigma_{T.val=0 \wedge T.id_2=A}(T)$$

Algorithm 12 generalizes the idea in the above example and computes the feature vectors for a specific $\mathcal{I}_k \in \mathcal{I}$. The algorithm generates multiple queries corresponding to each grounding of $\mathcal{I}_k$ such that the number of joins in each query is lesser than J. For this, we go over each formula f_t, and first check if f_t is *relevant* to $\mathcal{I}_k$, i.e., if f_t contains at least one atom corresponding to $\mathtt{R}_i$ such that for some p, $(i,\ p) \in \mathcal{I}_k$, then f_t is a relevant formula for clustering $\mathcal{I}_k$, otherwise, we ignore f_t. This is because, the features from f_t which are not relevant to $\mathcal{I}_k$ remains identical for every grounding of x and therefore never affects the clustering. For every relevant f_t, we first build the complete query which is a sequence of

Algorithm 12: Compute-Features

Input: $\mathcal{M}$ and its associated relational DB, join-bound J, $\mathcal{I}_k \in \mathcal{I}$

Output: Feature vector set $\{\mathbf{U}_{C_{kj}}\}_{j=1}^{\Delta_{\mathcal{I}_k}}$

1 $\mathbf{U} = \emptyset$
2 **for** $C_{kj} \in \Delta_{\mathcal{I}_k}$ **do**
3 $\mathbf{U}_{C_{kj}} = \emptyset$
4 **for** $f_t \in \mathbf{F}$ **do**
5 **if** f_t *is not relevant to* $\mathcal{I}_k$ **then**
6 continue
7 $\mathcal{Q}$ = Build-Query(f_t)
8 **while** $\mathcal{Q}$ *not empty* **do**
9 $\mathcal{Q}'$ = Select a sub-query containing up to the first J joins in $\mathcal{Q}$
10 **for** $R_i \in \mathcal{Q}'$ **do**
11 **if** $\exists\ p$ *such that* $(i, p) \in \mathcal{I}_k$ **then**
12 Wrap a select ($\sigma_{R_i.id_p = C_{kj}}$) around R_i
13 $\mathbf{U}_{C_{kj}}$.append(Count($\mathcal{Q}'$))
14 Let R_s be a table in $\mathcal{Q}'$ whose attribute participates in the θ-join after $\mathcal{Q}'$
15 **if** $R_s = \emptyset$ **then**
16 $\mathcal{Q} = (\mathcal{Q} - \mathcal{Q}')$
17 **else**
18 Relax the θ-join and include only those constraints involving R_s
19 $\mathcal{Q} = R_s \bowtie_\theta (\mathcal{Q} - \mathcal{Q}')$
20 $\mathbf{U}$.append($\mathbf{U}_{C_{kj}}$)
21 **return** $\mathbf{U}$

θ-joins on the tables corresponding to every atom in f_t. The query selects the the groundings of f_t that are not satisfied by the evidence. The θ in the join specifies variables shared among atoms in f_t. For example, in a formula $\neg\mathtt{R}(x) \vee \mathtt{S}(x)$, the θ-join is specified as $\sigma_{R.val=1}(R)$ $\bowtie_{R.id_1=S.id_1}$ $\sigma_{S.val=0}(S)$. Once we build the full query, we simply walk through the query executing no more than J joins at a time. For each atom which has a variable that corresponds to some element of $\mathcal{I}_k$, we ground the variable by enforcing the select condition in line 11 of the algorithm. We execute the partial query $\mathcal{Q}'$ with a maximum of J joins and store the result (count) in the feature vector. Next, we remove $\mathcal{Q}'$ from $\mathcal{Q}$ and relax the next θ- join

condition as follows. Among all the tables mentioned in $\mathcal{Q}'$, we select one table R_s, that participates in the next join operation in $\mathcal{Q} - \mathcal{Q}'$. We only retain the join conditions related to R_s in the next join in $\mathcal{Q} - \mathcal{Q}'$ and remove the rest of the conditions. We continue until we empty the original query $\mathcal{Q}$. Finally, we return the vector of counts accumulated across all queries for each grounding of $\mathcal{I}_k$.

5.2.6 Related Work

Several previous approaches have been suggested for improving the scalability of inference in MLNs. Most of these approaches can be termed as lifted inference algorithms since they either use rules that can be directly applied on the first-order structure or identify symmetries in the ground representation to perform efficient inference. Both exact (de Salvo Braz, 2007; Gogate and Domingos, 2011b; Van den Broeck et al., 2011; Bui et al., 2012) as well as approximate (Singla and Domingos, 2008; Kersting et al., 2009; Gogate et al., 2012; Niepert, 2012; Venugopal and Gogate, 2012; Bui et al., 2013) lifted algorithms have been developed that can greatly improve scalability. However, all these algorithms are efficient only when given the right MLN structure/evidence. Specifically, (Bui et al., 2012; Van den Broeck and Davis, 2012) show that efficient inference is possible only when presented with specific evidence-structures. More recently, (Van den Broeck and Darwiche, 2013) have proposed to counter the evidence-problem by adding more symmetries that make the MLN liftable. Specifically, they compute a low-rank boolean matrix factorization of the evidence matrix which implicitly induces a clustering whereas we explicitly cast it as a clustering problem thereby allowing us the flexibility to use a range of clustering algorithms and also better control of the inference-complexity. Further, (Van den Broeck and Darwiche, 2013) handles only binary evidence while our approach is much more general. Recently, (Beltagy and Mooney, 2014) have suggested another approach to handle the grounding problem in MLNs by making some modified assumptions about the world to suit natural language

understanding applications. Also, Singla et al. (Singla et al., 2014) developed a method where they approximated the messages in belief propagation. Finally, our approach of pre-processing the MLN is related to (Shavlik and Natarajan, 2009) which develops a systematic grounding procedure that can reduce the ground MLN size in many cases, and our approach of leveraging the query optimizations in relational databases for MLN inference is inspired by the Tuffy system (Niu et al., 2011).

5.2.7 Experiments

Setup

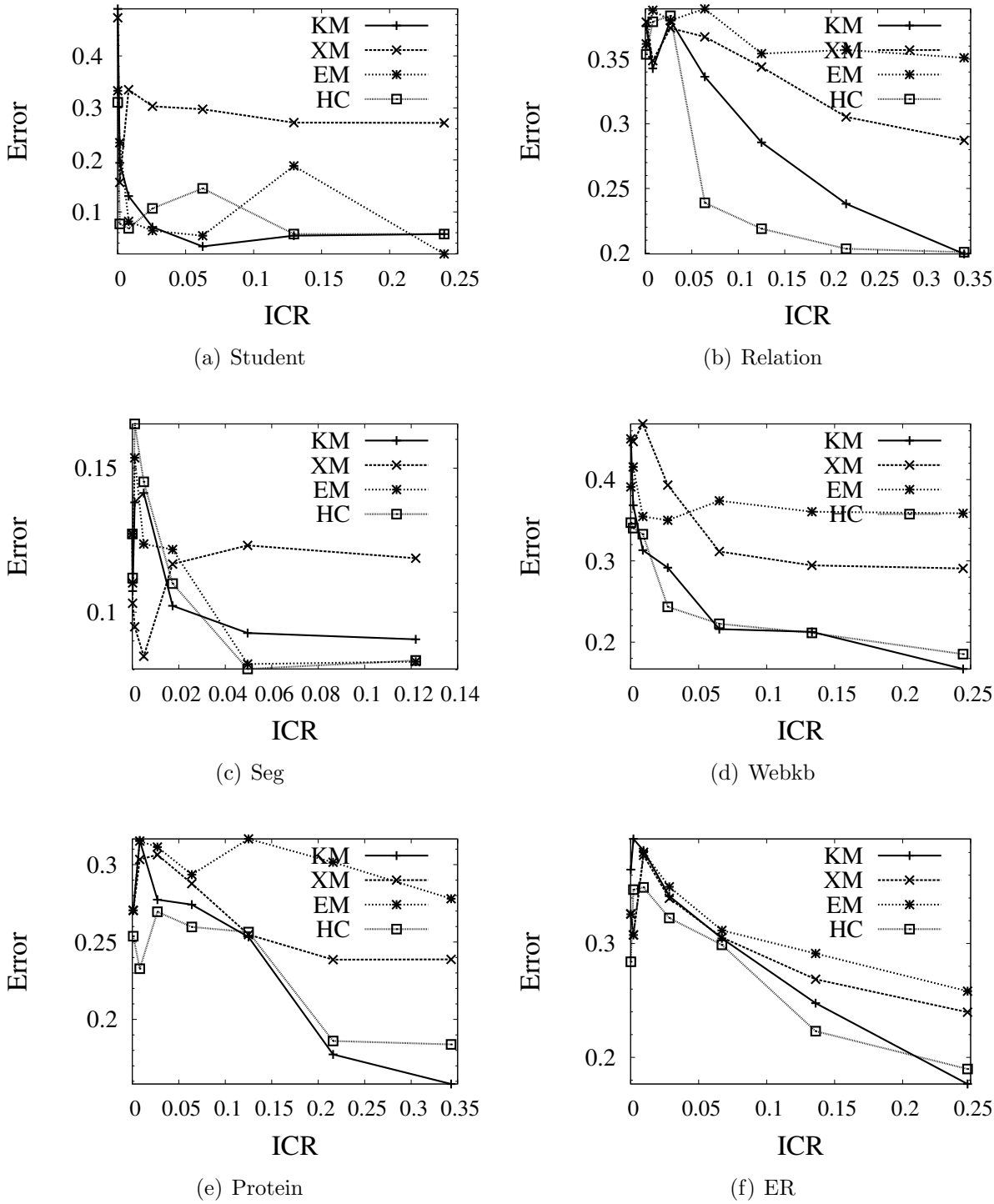
We evaluate our approach on 5 benchmark MLNs available in Alchemy (Kok et al., 2008), namely Entity Resolution (ER), Segmentation (Seg), Web Linkage analysis (WebKB), Hidden Markov Models (HMM) and Protein Interaction (Protein). Additionally, we added two new MLNs that have different structures called Student ($\texttt{Teaches}(i, c) \wedge \texttt{Prereq}(c, c1) \Rightarrow \texttt{Takes}(s, c1)$) and Relation ($\texttt{Related}(i, j) \wedge \texttt{Friends}(j, k) \Rightarrow \texttt{Loves}(k, i)$).

For our experiments, we implemented our system using MySQL on a quad-core Ubuntu machine with 6 GB RAM. To speed up query processing, we created n indexes for a table corresponding to a n-ary predicate, where the column corresponding to each argument of a predicate is indexed separately. For the distance function, we limit the number of joins (J) to 1. To show the generality of our approach, we evaluated it using two inference algorithms, namely, Gibbs sampling (Geman and Geman, 1984) and belief propagation (Singla and Domingos, 2008). We used the implementation of both these algorithms from Alchemy (Kok et al., 2008). For clustering, we experimented with four different algorithms available in Weka (Hall et al., 2009) namely, KMeans++ (KM), Expectation-Maximization (EM), Hierarchical clustering (HC) and XMeans (XM).

Approximation results on benchmarks

Figures 5.4 and 5.5 illustrate the accuracy of our approach on various benchmark MLNs. Figure 5.4 illustrates the results for approximately-lifted Gibbs sampling and Figure 5.5 illustrates the results for approximately-lifted belief propagation. The *x-axis* in both figures plots the inverse compression ratio ICR which is equal to $\frac{N_C}{N_G}$, where N_C is the total number of ground formulas in the compressed MLN and N_G is the total number of ground formulas in the original MLN. The *y-axis* shows the approximation error calculated as follows. $Err = \frac{\sum_{g \in G} D_{KL}(P_g || P'_g)}{|G|}$, where D_{KL} is the standard KL-Divergence distance measure, G refers to all ground atoms of a query predicate, P_g is the marginal distribution of ground atom g computed from the original MLN and P'_g is computed from the compressed MLN. Both P_g and P'_g are computed using Gibbs sampling for the results shown in Figure 5.4 and using belief propagation in Figure 5.5. In each MLN that we used for our experiments, we set 50% of arbitrary groundings as evidence, where 25% are `True` and 25% are `False`.

As seen from our results, as ICR increases, the approximation error reduces. At the same time, larger ICR increases the complexity of inference since N_C is larger. Thus, we can trade-off accuracy with computational complexity. Further, for MLN structures with more symmetries to exploit, the approximation error decreases far more rapidly as compared to asymmetric MLN structures. For instance, the Student MLN (Figures 5.4(a) and 5.5(a)) has much smaller approximation error than ER (Figures 5.4(f) and 5.5(f)) for comparable values of ICR. Overall, inference using belief propagation yielded more accurate results than Gibbs sampling and the clustering produced by KM and HC minimized the approximation error between the original and compressed MLN much more effectively. Note that determining the optimal number of clusters automatically for any given MLN is an interesting and challenging problem by itself. A possible extension of our work is to apply more advanced *non-parametric* techniques to systematically trade-off accuracy with number of clusters.

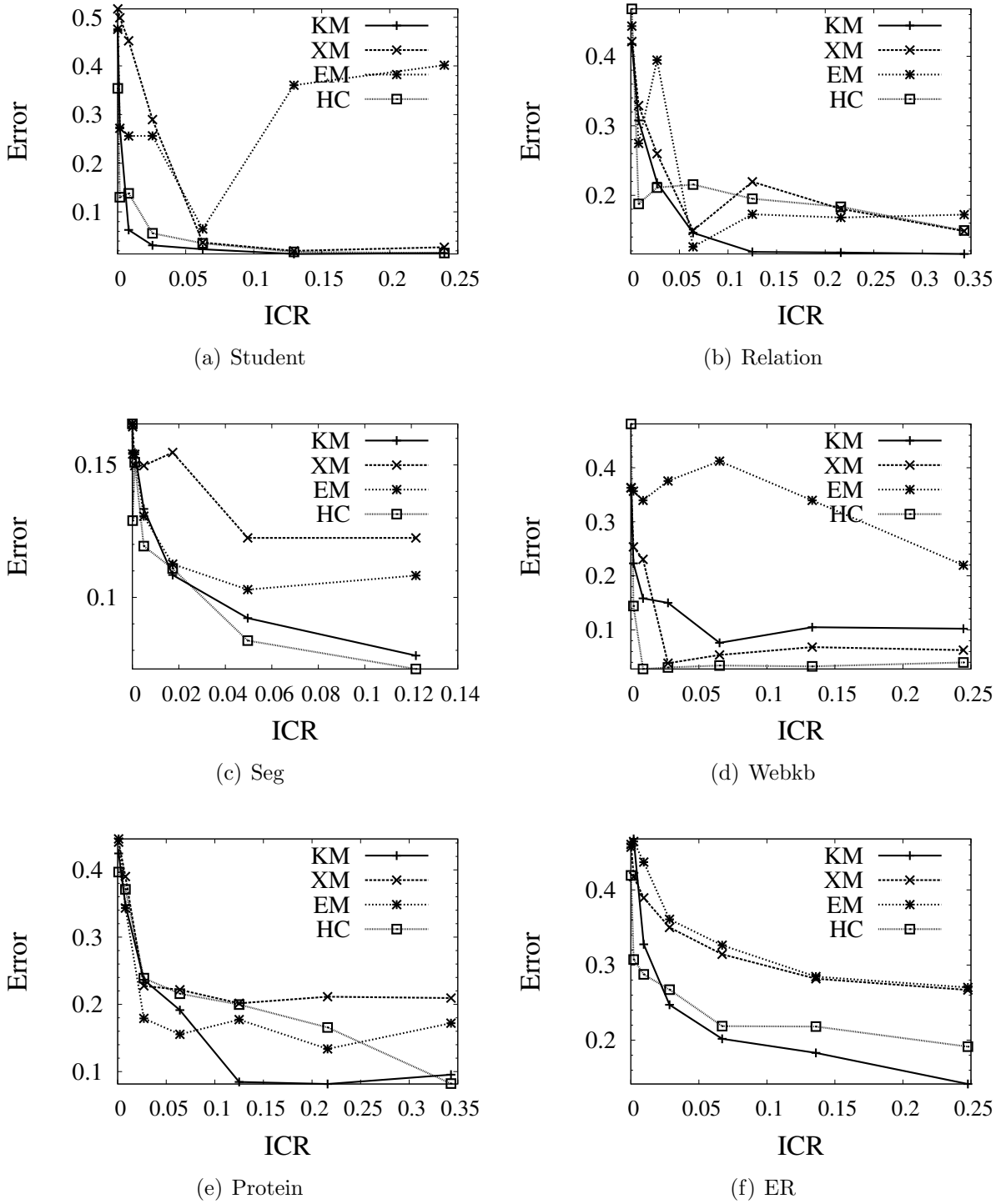


(a) Student (b) Relation

(c) Seg (d) Webkb

(e) Protein (f) ER

Figure 5.4. Approximation-error vs ICR. The y-axis shows the average KL-Divergence of the marginals computed on the clustered MLN from the marginals computed on the original MLN (smaller is better). The inference algorithm used is Gibbs sampling.



(a) Student

(b) Relation

(c) Seg

(d) Webkb

(e) Protein

(f) ER

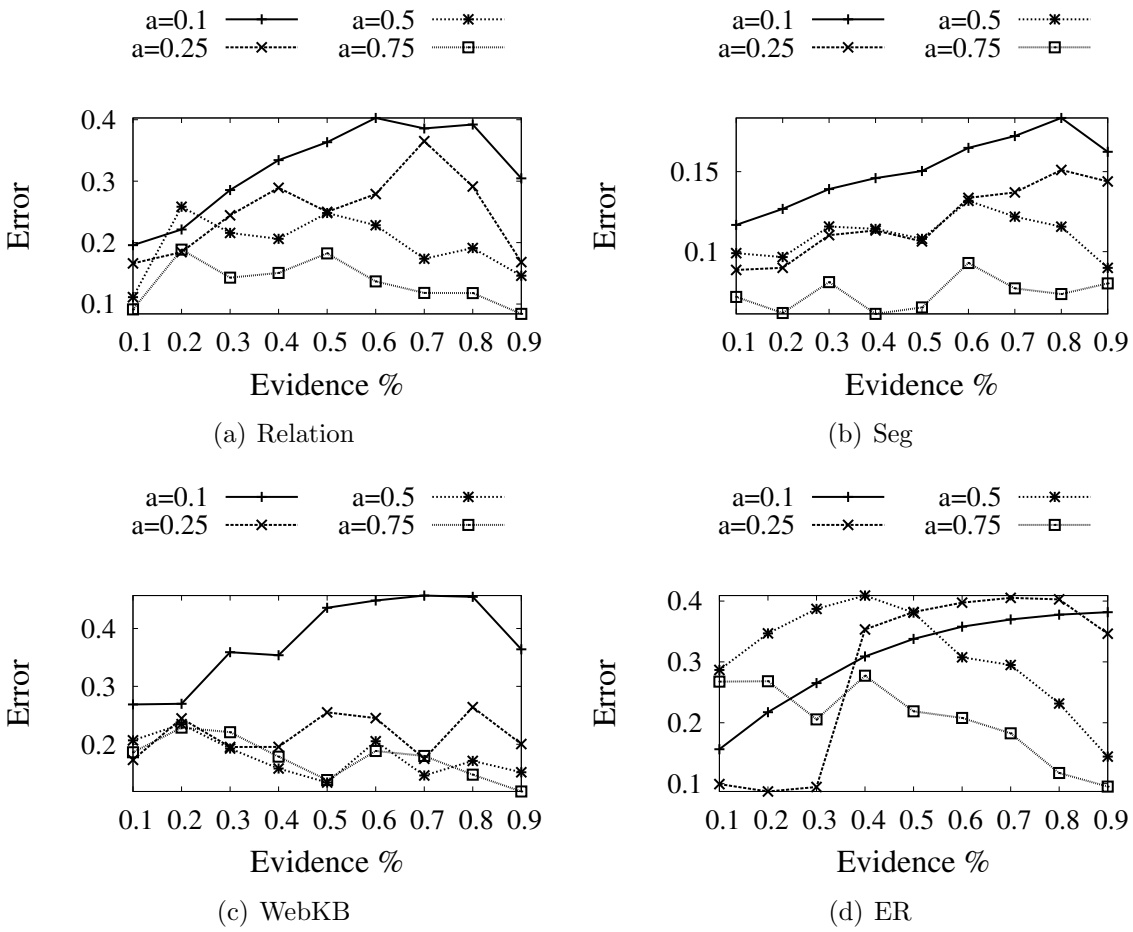
Figure 5.5. Approximation-error vs ICR. The y-axis shows the average KL-Divergence of the marginals computed on the clustered MLN from the marginals computed on the original MLN (smaller is better). The inference algorithm used is Belief Propagation.

Effect of Evidence

Figure 5.6 illustrates the error for different values of cluster-bounds (α) and varying amount of evidence. The results shown Figure 5.6 use K-Means++ for clustering and belief propagation for inference. As expected, using a larger value of α in most cases leads to lower errors due to a better approximation of the original MLN. Also, it can be seen that in most of the cases illustrated in Figure 5.6, for very small or very large amounts of evidence, the errors seem to go down. This is quite consistent with the effect that evidence has on MLNs. Evidence breaks symmetries in the MLN and thus if very few groundings or nearly all groundings are evidence, as there are more symmetries, the inference algorithms tend to give us better approximations (for all α values) than the cases shown in middle portion of the graphs where more random evidence makes inference more challenging.

Scalability

Figure 5.7 illustrates the scalability of our approach when handling large domain-sizes. For different domain-sizes, we show the time in seconds it takes to compute the compressed MLN. We used an α value of 0.25 for these experiments and introduced 50% random evidence with half of them `True` and the other half `False`. As expected, the time taken to compute the approximate MLN increases as the domain-size grows. However, it should be noted that none of the MLNs in Figure 5.7 could be processed by existing ground/lifted inference algorithms in Alchemy before running out of memory as the number of ground formulas is extremely large. For example, one instance of the Relation MLN in Figure 5.7 (a) has one billion groundings. Thus, without approximating the MLN, there is no feasible approach to inference in such large models. As shown by our results, we were able to complete processing the MLN in a reasonable amount of time even when the groundings reached a trillion as in Figure 5.7 (e). Also, the number of first-order formulas and their structure play a role in determining the complexity due to the distance function computation. Recall that we

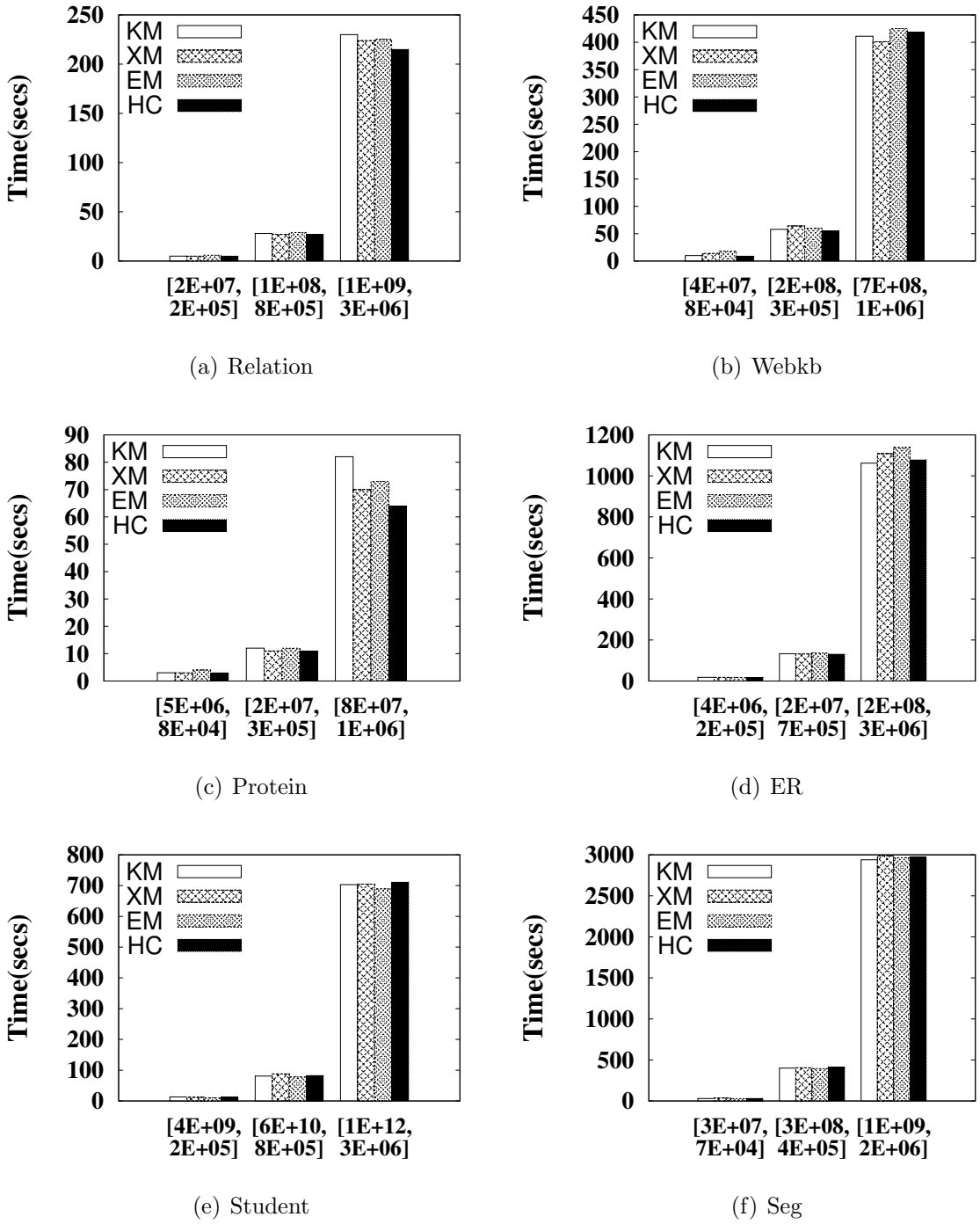


(a) Relation (b) Seg

(c) WebKB (d) ER

Figure 5.6. Illustrating the effect of evidence. The x-axis varies the amount of evidence on the atoms in the MLN. The y-axis plots the approximation error for varying cluster-bounds. The experiment is run using K-Means for clustering and belief propagation for inference.

compute a vector for every formula in the MLN. Therefore, a larger number of formulas mean more computations on the database. For instance, Figure 5.7 (e) has just one formula while (f) has 8 formulas which have more complex structure. Therefore, even though the number of ground formulas in (e) is a trillion while in (f) it is a billion, we took more time to process (f). Further, it can be seen that for each of the benchmarks, the third instance (the largest MLN) takes a visibly longer time when compared to the first two instances. This is expected because, when the size of the database grows really large as is the case for very large domain-sizes, it typically requires many more hard disk accesses for query processing



(a) Relation

(b) Webkb

(c) Protein

(d) ER

(e) Student

(f) Seg

Figure 5.7. Scalability experiments. The y-axis shows the time taken to form the approximate MLN and the x-axis shows $[N_f, N_a]$, where N_f is the number of ground formulas and N_a is the number of ground atoms.

which causes it to slow down. Finally, as seen in the results, the type of clustering has minimal impact on the time taken to process the MLN, i.e., nearly all clustering methods took approximately the same amount of time.

5.3 Application: Scalable Importance Sampling

As seen from the experiments in the previous section, in practice, the compressed MLN obtained though our evidence-based clustering approximates the distribution of the original MLN quite well. However, it is hard to analytically bound the quality of the approximations induced by the compression. That is, it may turn out that for some cases, the compressed MLN represents a distribution that is quite far off from the true distribution of the MLN. To address this problem, we present a new, more scalable importance sampling algorithm that uses approximate lifting for scalability but also provides guarantees.

In this section, we show how to significantly scale up the two main steps in importance sampling, namely, (a) constructing and sampling from an accurate proposal distribution and (b) computing the sample weight. Importantly, we carefully design each step, ensuring that we never incur the grounding or evidence problems. As a result, the computational complexity of our method is much smaller than existing lifted importance sampling approaches (Gogate et al., 2012). For step (a), we use the compressed MLN to design an informed proposal distribution. We then sample this proposal in an approximately lifted manner using Gibbs sampling. Each sample from our proposal corresponds to several approximately symmetric propositional samples. In step (b), we compute the importance weight of each lifted sample. However, computing the weight of a sample for MLNs is a computationally hard problem. In fact, for computing each sample weight, we need to enumerate over all possible ground formulas in the MLN, which is equivalent to the *grounding problem* that was discussed previously. Therefore, to ensure scalability of the sampler, we develop a novel,

tractable weighting scheme. Specifically, for each approximately lifted sample, we use a second sampler to sample a bounded number of ground formulas in the MLN. We then approximate the weight of the sample based on the sampled formulas. We show that our weighting method yields an importance sampler with asymptotic guarantees. Further, by combining the sampler with principles from exact lifting, we perform *Rao-Blackwellisation* (Casella and Robert, 1996) on our importance sampler thereby reducing variance in many cases.

5.3.1 Constructing and Sampling the Proposal Distribution

Recall that, in importance sampling (Geweke, 1989), we draw samples from a proposal distribution H that is easier to sample compared to sampling from the true distribution P. Each sample is then weighted with its *importance weight* to correct for the fact that it is drawn from the wrong distribution. To compute the marginal probabilities from the weighted samples, we use the following *Monte-Carlo* estimator.

$$P'(\bar{Q}) = \frac{\sum_{t=1}^{T} \mathbb{I}_{\bar{Q}}(\bar{\mathbf{s}}^{(t)}) w(\bar{\mathbf{s}}^{(t)})}{\sum_{t=1}^{T} w(\bar{\mathbf{s}}^{(t)})} \tag{5.4}$$

where $\bar{\mathbf{s}}^{(t)}$ is the t^{th} sample drawn from H, $\mathbb{I}_{\bar{Q}}(\bar{\mathbf{s}}^{(t)}) = 1$ iff the query atom Q is assigned $\bar{Q}$ in $\bar{\mathbf{s}}^{(t)}$ and 0 otherwise, $w(\bar{\mathbf{s}}^{(t)})$ is the importance weight of the sample given by $\frac{P(\bar{\mathbf{s}}^{(t)})}{H(\bar{\mathbf{s}}^{(t)})}$. Eq. (5.4) is called as a ratio estimate or a normalized estimate because *we only need to know each sample's importance weight up to a normalizing constant.* We utilize this key fact to design our sampler in a scalable manner.

Constructing a good proposal distribution that in some way mirrors the true distribution is arguably the most important task in importance sampling. Unfortunately, this is also a highly challenging problem since we do not really know what the true distribution looks like. However, recall that we in fact designed our compressed MLN (in the previous section) such that it tries to approximate the true distribution as closely as possible. We now leverage this compressed MLN and present a systematic approach to construct a highly accurate proposal distribution.

Formulas:
$\mathtt{R}(x) \vee \mathtt{S}(x, y)$, w
Domains:
Δ_x = $\{A_1, B_1, C_1, D_1\}$
Δ_y = $\{A_2, B_2, C_2, D_2\}$

(a)

Formulas:
$\mathtt{R}_1(\mu_1) \vee \mathtt{S}_1(\mu_1, \mu_3)$, w; $\mathtt{R}_2(\mu_2) \vee \mathtt{S}_2(\mu_2, \mu_3)$, w
$\mathtt{R}_1(\mu_1) \vee \mathtt{S}_3(\mu_1, \mu_4)$, w; $\mathtt{R}_2(\mu_2) \vee \mathtt{S}_4(\mu_2, \mu_4)$, w
Domains:
$g(\mu_1) = \{A_1, B_1\}$; $g(\mu_2) = \{C_1, D_1\}$
$g(\mu_3) = \{A_2, B_2\}$; and $g(\mu_4) = \{C_2, D_2\}$

(b)

Figure 5.8. (a) an example MLN $\mathcal{M}$ and (b) MLN $\hat{\mathcal{M}}$ obtained from $\mathcal{M}$ by grounding each logical variable in $\mathcal{M}$ by the cluster centers $\mu_1, \ldots, \mu_4$.

Given a MLN $\mathcal{M}$ in Σ-normal form, let $\mathbb{D}$ denote the set of all domains in $\mathcal{M}$ and let $\mathbf{D} \in \mathbb{D}$ be a set of objects that belong to the same domain. We compress $\mathcal{M}$ as described in the previous section. That is, we consider each $\mathbf{D} \in \mathbb{D}$ independently and learn a new domain $\mathbf{D}'$ where $|\mathbf{D}'| \ll \mathbf{D}$ and $\zeta : \mathbf{D} \to \mathbf{D}'$ is a surjective mapping, i.e., $\forall\ \mu \in \mathbf{D}'$, $\exists\ C \in \mathbf{D}$ such that $\zeta(C) = \mu$. In other words, each cluster of objects is replaced by its cluster center in the reduced domain.

Let $\hat{\mathcal{M}}$ be the network obtained by grounding $\mathcal{M}$ with its reduced domains (which corresponds to the cluster centers) and let $\mathcal{M}_G$ be the ground Markov network of $\mathcal{M}$ using the original domains. $\hat{\mathcal{M}}$ and $\mathcal{M}_G$ are related as follows. We can think of $\hat{\mathcal{M}}$ as an MLN, in which the logical variables are the cluster centers. If we set the domain of each logical variable corresponding to cluster center $\mu \in \mathbf{D}'$ to $g(\mu)$ where $g(\mu) = \{C \in \mathbf{D} | \zeta(C) = \mu\}$, then the ground Markov network of $\hat{\mathcal{M}}$ is $\mathcal{M}_G$. Figure 5.8 shows an example MLN $\mathcal{M}$ and its corresponding compressed MLN $\hat{\mathcal{M}}$. Notice that the Markov network obtained by grounding $\mathcal{M}$ is the same as the one obtained by grounding $\hat{\mathcal{M}}$.

In $\hat{\mathcal{M}}$ we assume that for each predicate, all its groundings are approximately symmetric to each other and obtain approximately lifted samples from $\hat{\mathcal{M}}$. The degree of this approximation can be controlled by the number of clusters used to compress $\mathcal{M}$. As we increase the number of clusters, each sample from $\hat{\mathcal{M}}$ becomes "more approximately lifted", i.e., it represents a greater number of approximately symmetric samples of $\mathcal{M}$.

Next, we describe how to generate samples from $\hat{\mathcal{M}}$. To keep the equations more readable, we assume that we only have positive evidence (i.e., an assertion that the ground atom is known to be true). Note that it is straightforward to extend the following equations to the general case in which we have both positive and negative evidences.

Let $\hat{\mathcal{M}}$ contain $\hat{K}$ predicates, for which we assume some ordering. Let $\mathbf{E}$ and $\mathbf{U}$ be two sets defined as follows. $E_i \in \mathbf{E}$ represents the number of (true) evidence ground atoms corresponding to the i-th predicate in $\hat{\mathcal{M}}$ and $U_i \in \mathbf{U}$ represents the number of ground atoms of the i-th predicate whose truth value is unknown.

Without loss of generality, let the j-th formula in $\hat{\mathcal{M}}$, denoted by f_j, contain the atoms $p_1, \ldots p_k$ where p_i is an instance of the p_i-th predicate and if $i \leq m$, it has a positive sign else it has a negative sign. The task is to now count the total number of satisfied groundings in f_j symbolically without actually going over the ground formulas. Unfortunately, this task is in $\#\mathcal{P}$. Therefore, we make the following approximation. Let $N(p_1, \ldots p_k)$ denote the number of satisfied groundings of f_j based on the assignments to all groundings of predicates indexed by $p_1, \ldots p_k$. Then, we will approximate $N(p_1, \ldots p_k)$ using $\sum_{i=1}^{k} N(p_i)$, thereby independently counting the number of satisfied groundings for each predicate. Clearly, our approximation overestimates the number of satisfied formulas because it ignores the joint dependencies between atoms in f. To compensate for this, we scale-down each count by a scaling factor (γ) which is the ratio of the actual number of ground formulas in f to the assumed number of ground formulas. Next, we define these counting equations formally.

Given the j-th formula f_j and a set of indexes $\mathbf{I}$, where $i \in \mathbf{I}$ corresponds to the i-th atom in f_j, let $\#G_{f_j}(\mathbf{I})$ denote the number of ground formulas in f_j if all the terms in all atoms specified by $\mathbf{I}$ are replaced by constants. For instance, in the example shown in Figure 5.8, let f be $\mathtt{R}_1(\mu_1) \vee \mathtt{S}_1(\mu_1, \mu_3)$, then, $\#G_f(\emptyset) = 4$, $\#G_f(\{1\}) = 2$ and $\#G_f(\{2\}) = 1$. We now count f_j's satisfied groundings symbolically as follows.

$$\mathcal{S}'_j = \gamma \sum_{i=1}^{m} E_{p_i} \#G_{f_j}(\{i\}) \tag{5.5}$$

where $\gamma = \frac{\#G_{f_j}(\emptyset)}{m\#G_{f_j}(\emptyset)} = \frac{1}{m}$ and $\mathcal{S}'_j$ is rounded to the nearest integer.

$$\mathcal{S}_j = \gamma \left(\sum_{i=1}^{m} \hat{S}_{p_i} \#G_{f_j}(\{i\}) + \sum_{i=m+1}^{k} (U_{p_i} - \hat{S}_{p_i}) \#G_{f_j}(\{i\}) \right) \tag{5.6}$$

where $\gamma = \frac{max(\#G_{f_j}(\emptyset) - \mathcal{S}'_j, 0)}{k\#G_{f_j}(\emptyset)}$, $\hat{S}_{p_i}$ is a *lifted symbol* representing the total number of true ground atoms (among the unknown atoms) of the p_i-th predicate and $\mathcal{S}_j$ is rounded to the nearest integer.

The symbolic (un-normalized) proposal probability is given by the following equation.

$$H(\hat{\mathbf{S}}, \mathbf{E}) = \exp \left(\sum_{j=1}^{C} w_j \mathcal{S}_j \right) \tag{5.7}$$

where C is the number of formulas in $\hat{\mathcal{M}}$ and w_j is the weight of the j-th formula.

Given the symbolic equation Eq. (5.7), we sample the set of lifted symbols, $\hat{\mathbf{S}}$, using Gibbs sampling. For this, we initialize all symbols to a random value. We then choose a random symbol $\hat{S}_i$ and substitute it in Eq. (5.7) for each value between 0 to U_i. This yields a conditional distribution on $\hat{S}_i$ given assignments to $\hat{\mathbf{S}}_{-i}$, where $\hat{\mathbf{S}}_{-i}$ refers to all symbols other than the i-th one. We then sample from this conditional distribution by taking into account that there are $\binom{U_i}{v}$ distinct assignments corresponding to the v-th value in the distribution, which corresponds to setting any v groundings of the i-th predicate to `True`. After the Markov chain has *mixed*, to reduce the dependency between successive Gibbs samples, we *thin* the samples and only use every p-th sample for estimation.

Note that during the process of sampling from the proposal, we only had to compute $\hat{\mathcal{M}}$, namely, we only ground the original MLN with the cluster-centers. Therefore, the representation is *lifted* because we do not ground $\hat{\mathcal{M}}$. This helps us scale up the sampling step to large domains-sizes (since we can control the number of clusters used to define $\hat{\mathcal{M}}$).

5.3.2 Computing the Importance Weight

In order to compute the marginal probabilities as in Eq. (5.4), given a sample, we need to compute (up to a normalization constant) the weight of that sample. It is easy to see

that each sample from our proposal (assignments on all symbols) is a *lifted sample*, i.e., it has multiple possible assignments in the state space corresponding to the original MLN. For instance, suppose in our running example in Figure 5.8, the symbol corresponding to $\mathtt{R}(\mu_1)$ has a value equal to 1, this corresponds to two different assignments in $\mathcal{M}$, either $\mathtt{R}(A_1)$ is true or $\mathtt{R}(B_1)$ is true. Formally, a sample from the proposal has $\prod_{i=1}^{\hat{K}} \binom{U_i}{\hat{S}_i}$ different assignments in the original distribution. Further, as we increase the amount of compression in the MLN, samples from the proposal become more lifted, i.e., $\hat{K}$ becomes smaller while U_i becomes larger. Thus the number of assignments of the original MLN that each lifted sample represents increases as we increase the amount of compression (or decrease the number of clusters). From the construction of our proposal, for any lifted sample, all its corresponding assignments in the state space of the original MLN are approximately symmetric. That is, we assume that each of these samples have approximately the same probability in the original MLN's distribution. Thus, to compute the (un-normalized) probability of a sample w.r.t $\mathcal{M}$, for any approximately lifted sample, $\hat{\mathbf{S}}^{(t)}$, we sample one of its corresponding propositional samples, say, $\bar{\mathbf{s}}^{(t)}$ uniformly at random and compute the importance weight (up to a multiplicative constant) using the following equation.

$$w(\hat{\mathbf{S}}^{(t)}, \mathbf{E}) = \frac{P(\bar{\mathbf{s}}^{(t)}, \mathbf{E})}{H(\hat{\mathbf{S}}^{(t)}, \mathbf{E})} \tag{5.8}$$

From the theory of importance sampling (cf. (Liu, 2001)), plugging-in the weights computed by Eq. (5.8) into Eq. (5.4) yields an asymptotically unbiased estimate of the query marginal probabilities, i.e., as $T \to \infty$, $\hat{P}(Q)$ almost surely approaches $P(Q)$.

However, in the case of MLNs, computing $w(\hat{\mathbf{S}}^{(t)}, \mathbf{E})$ turns out to be a computationally hard problem. Specifically, $P(\bar{\mathbf{s}}^{(t)}, \mathbf{E}) \propto \sum_f w_f N_f(\bar{\mathbf{s}}^{(t)}, \mathbf{E})$, where f is a formula of $\mathcal{M}$ with weight w_f and $N_f(\bar{\mathbf{s}}^{(t)}, \mathbf{E})$ refers to the number of groundings of f that are satisfied by the sample $(\bar{\mathbf{s}}^{(t)}, \mathbf{E})$. Thus, to compute $P(\bar{\mathbf{s}}^{(t)}, \mathbf{E})$, for every formula in $\mathcal{M}$, we need to go over each grounding of that formula and check if it is satisfied in $(\bar{\mathbf{s}}^{(t)}, \mathbf{E})$. The complexity of

this step is clearly equivalent to the grounding-problem in MLNs and therefore, exact weight computation is infeasible for large MLNs. To make this weight-computation step tractable, we develop a new weighting method that approximates the importance weight as follows. For each sample from the proposal, we use an additional sampler which samples a bounded number of groundings of a first-order formula in $\mathcal{M}$. Using only the sampled groundings of each formula, we approximate the importance weight of the sample drawn from the proposal distribution.

Formally, let $\mathbb{U}_i$ be a proposal distribution defined on the groundings of the i-th formula. Here, we define this distribution as a product of $|\mathbf{V}_i|$ uniform distributions where $\mathbf{V}_i = V_{i1} \ldots V_{ik}$ is the set of distinct variables in the i-th formula. Formally, $\mathbb{U}_i = \prod_{j=1}^{|\mathbf{V}_i|} \mathbb{U}_{ij}$, where $\mathbb{U}_{ij}$ is a uniform distribution over the domain of V_{ik}. A sample from $\mathbb{U}_i$ contains a grounding for every variable in the i-th formula. Using this, we can approximate the importance weight using the following equation.

$$\hat{w}(\bar{\mathbf{s}}^{(t)}, \mathbf{E}, \bar{\mathbf{u}}_i^{(t)}) = \frac{\exp\left(\sum_{i=1}^{M} \mathtt{w}_i \frac{N_i'(\bar{\mathbf{s}}^{(t)}, \mathbf{E}, \bar{\mathbf{u}}_i^{(t)})}{\beta \prod_{j=1}^{|\mathbf{V}_i|} \mathbb{U}_{ij}}\right)}{H(\hat{\mathbf{S}}^{(t)}, \mathbf{E})} \tag{5.9}$$

where M is the number of formulas in $\mathcal{M}$, $\mathtt{w}_i$ is the weight of the i-th formula, $\bar{\mathbf{u}}_i^{(t)}$ are β groundings of the i-th formula drawn from $\mathbb{U}_i$ and $N_i'(\bar{\mathbf{s}}^{(t)}, \mathbf{E}, \bar{\mathbf{u}}_i^{(t)})$ is the count of satisfied groundings in $\bar{\mathbf{u}}_i^{(t)}$ groundings of the i-th formula.

Theorem 13. *Using the importance weights shown in Eq.* (5.9) *in Eq.* (5.4) *yields an asymptotically unbiased estimate of the query marginals, i.e., as the number of samples, $T \to \infty$, the estimated marginal probabilities almost surely converge to the true marginal probabilities.*

Proof. For brevity, let $S^{(t)} = (\bar{\mathbf{s}}^{(t)}, \mathbf{E}, \bar{\mathbf{u}}_i^{(t)})$ and let $H^{(t)} = H(\hat{\mathbf{S}}^{(t)}, \mathbf{E})$ and assume that each formula has G groundings. The exact weight for each sample is equal to,

$$w(S^{(t)}) = \frac{\exp(\sum_{i=1}^{M} \mathtt{w}_i N_i(S^{(t)}))}{H^{(t)}} \tag{5.10}$$

where $N_i(S^{(t)}) = \sum_{g=1}^{G} N_{ig}(S^{(t)})$; $N_{ig}(S^{(t)}) = 1$ if the g-th grounding of the i-th formula is true in sample $S^{(t)}$ and $N_{ig}(S^{(t)}) = 0$ if the g-th grounding of the i-th formula is false in sample $S^{(t)}$

The normalized estimator for a query atom Q is given by,

$$P'(\bar{Q}) = \frac{\sum_{t=1}^{T} \mathbb{I}_{\bar{Q}}(\bar{\mathbf{s}}^{(t)}) w(\bar{\mathbf{s}}^{(t)})}{\sum_{t=1}^{T} w(\bar{\mathbf{s}}^{(t)})} \tag{5.11}$$

Let us now replace each exact weight $w(S^{(t)})$ by $\hat{w}(S^{(t)})$, where we approximate the summation $N_i(S^{(t)})$ with a sampling based estimate. We now compute the marginal probabilities using the following new estimator.

$$P''(\bar{Q}) = \frac{\sum_{t=1}^{T} \mathbb{I}_{\bar{Q}}(\bar{\mathbf{s}}^{(t)}) \hat{w}(\bar{\mathbf{s}}^{(t)})}{\sum_{t=1}^{T} \hat{w}(\bar{\mathbf{s}}^{(t)})} \tag{5.12}$$

As $T \to \infty$, the estimates for $N_i(S^{(t)})$ converge to their true values. Thus, $\hat{w}(S^{(t)})$ almost surely converges to $w(S^{(t)})$ and both the numerator and denominator in Eq. (5.12) converge to the values specified in Eq. (5.11). Therefore, as $T \to \infty$, $P''(\bar{Q}) \to P'(\bar{Q}) \to P(\bar{Q})$.

□

5.3.3 Rao-Blackwellisation

We now show how to perform Rao-Blackwellisation in our importance sampler to reduce its variance. Our main idea is to integrate principles from exact lifting when computing the sample weight. We illustrate this with the following example.

Example 18. *Consider an MLN with a single formula* $\neg$`R`*(x, y)* $\vee$ `S`*(y, z); w, where each variable has domain-size equal to d. Given a sample, the complexity of computing the sample weight, specifically the numerator in Eq.* (5.8) *is* $O(d^3)$*. However, for any specific value of y, say* $y = A$*, observe that the satisfied groundings in the MLN formula can be computed in closed form as,* $n_1 d + n_2 d - n_1 n_2$*, where* n_1 *is the number of false groundings of* `R`*(x, A) and* n_2 *is the number of true groundings in* `S`*(A, z). Therefore, we can reduce the computational complexity of computing the exact same sample weight to* $O(d^2)$ *time.*

To generalize the above example, consider an MLN which has no shared variables. That is, in each formula, each logical variable occurs exactly once in that formula. Sarkhel et al. (Sarkhel et al., 2014b) in their lifted algorithm show that in an MLN with no shared variables, the (un-normalized) probability of any sample can be computed efficiently since we can express the satisfied groundings for any formula in closed form as given below.

Let $A_1^+ \dots A_n^+$ be the positive-signed atoms and $A_1^- \dots A_n^-$ be the negative-signed atoms in f. Let $\bar{A}_i^+$ be the number of true groundings of A_i^+ in the sample and similarly, let $\bar{A}_i^-$ be the number of true groundings of A_i^-. Let d_i^+ be the total number of groundings of A_i^+ and d_i^- the number of groundings of $\bar{A}_i^-$. The total number of groundings of f that are satisfied by the sample can be computed using the following equation.

$$N_f = \prod_{i=1}^{n} d_i^+ \prod_{i=1}^{m} d_i^- - \prod_{i=1}^{n} (d_i^+ - \bar{A}_i^+) \prod_{i=1}^{m} \bar{A}_i^- \tag{5.13}$$

Using Eq. (5.13) to compute the satisfied groundings for each formula, we can express the probability of a sample in closed form for a non-shared MLN. We utilize this result and perform Rao-Blackwellisation in our importance sampler as follows. We partition the variables in each formula into two sets, $\mathbf{V_1}$ being the set of shared variables in the formula and $\mathbf{V_2}$ being the rest of the variables. Recall that when computing the sample weight, we sampled $\mathbf{V_1} \cup \mathbf{V_2}$ to approximate the numerator in Eq. (5.8). Instead, we now only sample $\mathbf{V_1}$ and conditioned on this sample, we compute Eq. (5.8) exactly.

Algorithm 13 illustrates our complete sampler. It assumes $\hat{\mathcal{M}}$ and ζ are provided as input. First, we construct the symbolic equation Eq. (5.7) that computes the weight of the proposal. In the outer sampler, we sample the symbols from Eq. (5.7) using Gibbs sampling. After the chain has mixed, for each sample from the outer sampler, for every formula in $\mathcal{M}$, we construct an inner sampler that uses Rao-Blackwellisation to approximate the sample weight. Specifically, for a formula f, we sample an assignment to each non-shared variable to create a partially ground formula, f' and compute the exact number of satisfied

Algorithm 13: Compute-Marginals

Input: $\hat{\mathcal{M}}$, ζ, Evidence $\mathbf{E}$, Query $\mathbf{Q}$, sampling threshold β, thinning parameter p, iterations T
Output: Marginal probabilities $\mathcal{P}$ for $\mathbf{Q}$
begin
  Construct the symbolic counting formula Eq. (5.7)
  `// Outer Sampler`
  for t = *1 to* T **do**
    Sample $\hat{\mathbf{S}}^{(t)}$ using Gibbs sampling on Eq. (5.7)
    After burn-in, for every p-th sample, generate $\bar{\mathbf{s}}^{(t)}$ from $\hat{\mathbf{S}}^{(t)}$
    for *each formula* f_i **do**
      `// Inner Sampler`
      for c = *1 to* β **do**
        `// Rao-Blackwellisation`
        f_i' = Partially ground formula created by sampling assignments to shared variables in f_i
        Compute the satisfied groundings in f_i'
      Compute the sample weight using Eq. (5.9)
    Update the marginal probability estimates using Eq. (5.12)

groundings in f'. Finally, we compute the sample weight as in Eq. (5.9) and update the normalized estimator in Eq. (5.12).

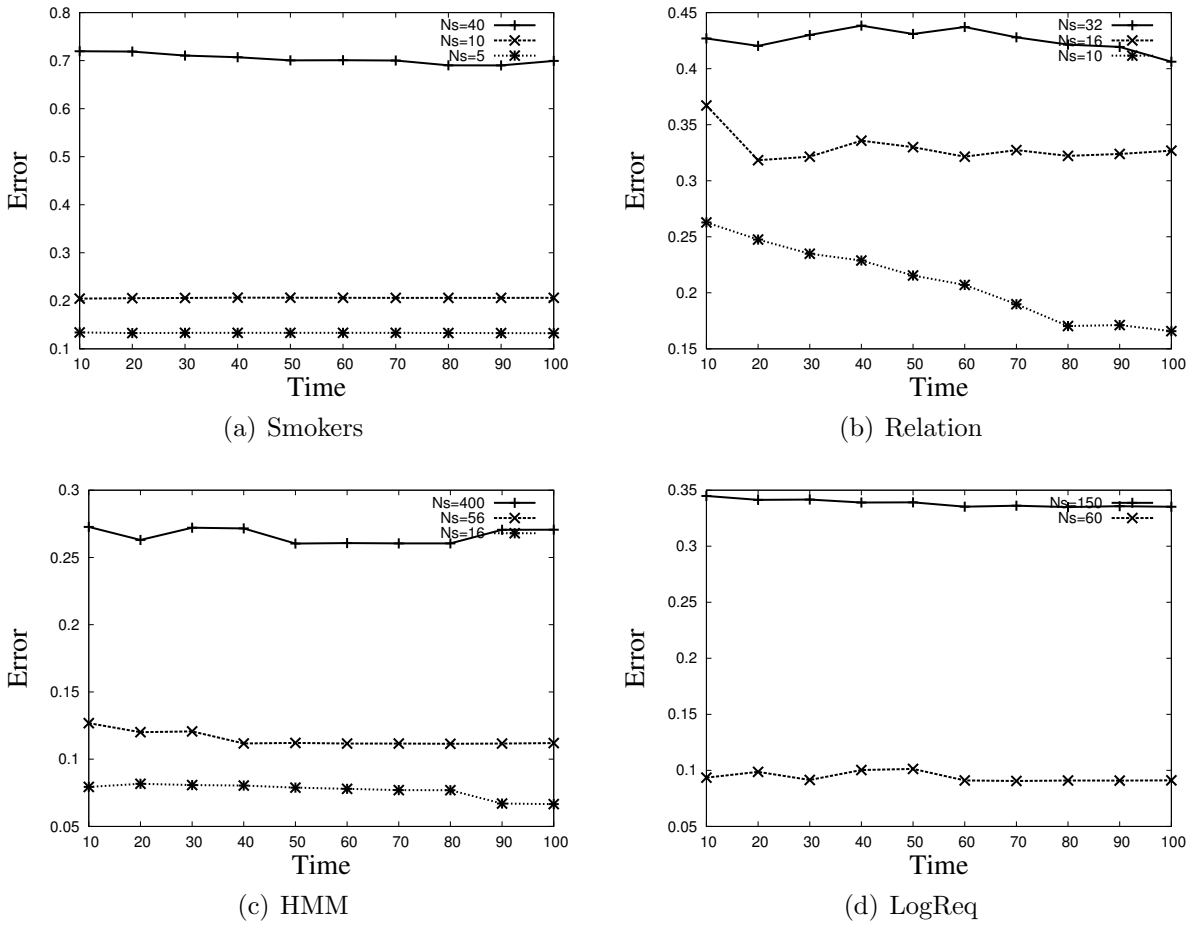
5.3.4 Experiments

We run two sets of experiments. First, to illustrate the trade-off between accuracy and complexity, we experiment with MLNs which can be solved exactly. Our test MLNs include Smokers and HMM (with few states) from the Alchemy website (Kok et al., 2008) and two additional MLNs, Relation ($\mathtt{R}(x, y) \Rightarrow \mathtt{S}(y, z)$), LogReq (randomly generated formulas with singletons). Next, to illustrate scalability, we use two Alchemy benchmarks that are far larger, namely Hypertext classification with 1 million ground formulas and Entity Resolution (ER) with 8 million ground formulas. For all MLNs, we randomly set 25% groundings as true and 25% as false. For clustering, we used the scheme in (Venugopal and Gogate, 2014a) with KMeans++ as the clustering method. For Gibbs sampling, we set the thinning parameter

to 5 and use a burn-in of 50 samples. We ran all experiments on a quad-core, 6GB RAM, Ubuntu laptop.

Figure 5.9 shows our results on the first set of experiments, where the y-axis plots the average KL-divergence between the true marginals for the query atoms and the marginals generated by our algorithm. The values are shown for varying values of $N_s = \frac{\#Groundings(\mathcal{M})}{\#Formulas(\hat{\mathcal{M}})}$. Intuitively, N_s indicates the amount by which $\mathcal{M}$ has been compressed to form the proposal. As illustrated in Figure 5.9, as N_s increases, the accuracy becomes lower in all cases because the proposal is a weaker approximation of the true distribution. However, at the same time, the computational complexity of the sampler decreases allowing us to trade-off accuracy with efficiency. Further, the MLN structure also determines the proposal accuracy. For example, LogReg that contains singletons yields an accurate estimate even for high values of N_s, while, for Relation, a smaller N_s yields such accuracy. This is because, singletons have symmetries (Gogate and Domingos, 2011b; Van den Broeck, 2011) which are in turn exploited by our clustering method when building the proposal distribution.

In the second set of experiments, we illustrate the scalability of our approach. Figure 5.10 shows our results. Here, we plot the sampling rate (number of samples generated in a minute), against N_s for different values of the sampling bound (b) which controls the percentage of total ground formulas that we visit to approximate the weight of each sample. Note that, for both the MLNs used here, we tried to compare the results with Alchemy but were unable to do so since Alchemy quickly runs out of memory while grounding the MLN. As expected, decreasing N_s, or increasing b increases complexity and therefore reduces the sampling rate. At the same time, we also expect the quality of the samples to be better because the proposal is of a better quality (see Figure 5.9) and also the weights are better approximations of the true weights. Thus, we can trade-off complexity with accuracy of our sampler. Importantly, these results show that by addressing the evidence/grounding problems, we can process large, arbitrarily structured MLNs/evidence which was not possible using existing approaches.



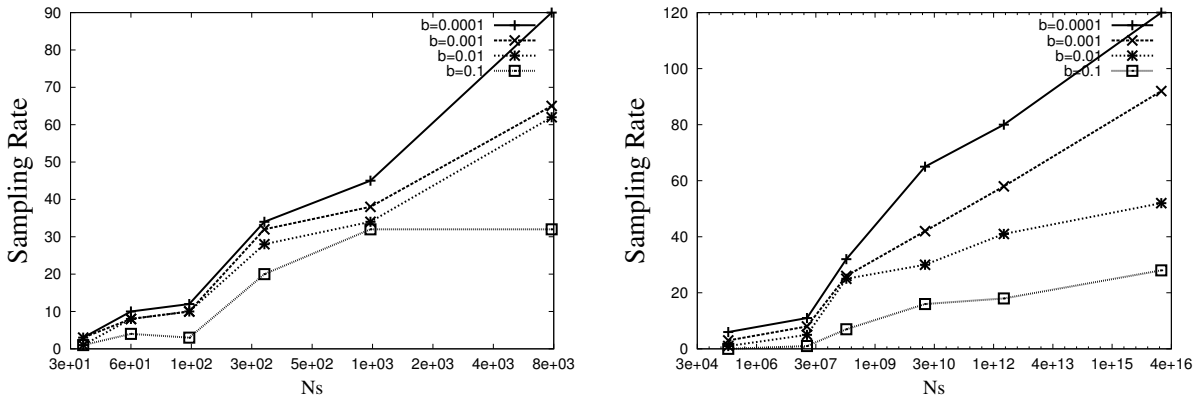
(a) Smokers (b) Relation

(c) HMM (d) LogReq

Figure 5.9. Tradeoff between computational efficiency and accuracy. The y-axis plots the average KL-divergence between the true marginals and the approximated ones for different values of N_s. Larger N_s implies weaker proposal, faster sampling. For this experiment, we set β (sampling bound) to 0.2. Note that changing β did not affect our results very significantly.

5.4 Summary

In this chapter, we presented techniques to scale up inference in MLNs that a) may not have explicit symmetries and/or b) have evidence that break symmetries. For this, we formulated a clustering problem to learn groups of approximately symmetrical objects and compressed the MLN based on these symmetries. To effectively learn these clusters, we defined a novel distance function that is sensitive to the evidence presented to the MLN and used it to replace groups of similar objects in the MLN by their cluster centers. Importantly, our



(a) Hyper (b) ER

Figure 5.10. Illustrating scalability. Sampling rate is plotted against N_s that controls quality of the proposal distribution. We show the results for different sampling bounds (b) that controls quality of weight approximation. (a) shows the results for Hypertext classification and (b) shows the results for ER. As seen in the plots, the sampling rate can be controlled by either introducing more approximations in the proposal (larger values of N_s) and/or by introducing approximations in the weighting method (larger values of b).

approach can be used as a pre-processing step by existing inference algorithms where the complexity of these algorithms can be controlled even for very large MLNs. However, utilizing approximate symmetries for lifted inference modifies the MLN distribution and it is complex to analytically bound this bias. To address this problem, we applied our approximate lifting idea to develop a new importance sampler that is not only scalable but also has asymptotic guarantees on its estimates. To scale up the sampling step of importance sampling, we used the compressed MLN to design the proposal distribution from which we obtained samples that are approximately lifted using Gibbs sampling. To scale up the weight estimation step, we developed a new weight estimation method that approximates the importance weight tractably. We showed that our new sampler yields asymptotically unbiased estimates and reduced variance of these estimates through Rao-Blackwellisation. Our experimental results on several benchmark MLNs clearly illustrated the high accuracy and scalability of our approximate approaches.

CHAPTER 6

EXPLOITING EFFICIENT COUNTING STRATEGIES FOR SCALABLE INFERENCE

Acknowledgements

I wish to acknowledge the contributions of Somdeb Sarkhel in the work presented here. Somdeb helped in the design of the MaxWalkSAT algorithm and running its experiments presented in this chapter.

6.1 Introduction

The previous two chapters presented various symmetry-exploiting approaches to scale up inference in MLNs. That is, we aim to perform inference over groups rather than over individual objects and the main challenge is to find these groups of objects that have symmetrical or exchangeable properties. Majority of the research in lifted inference has focused on detecting these symmetries efficiently, ideally, directly from first-order structure (cf. (Kimmig et al., 2014; Kersting, 2012)). In this chapter, we focus on an alternative, relatively unexplored direction that turns out to be equally important for scalability in MLN inference and augments research in lifted inference. Specifically, to compute the probability (up to a normalization constant) for any world ω of the MLN, we need to count the number of groundings of each formula in the MLN that are satisfied by the world ω. Most graphical model inference algorithms do not consider this to be a hard task. For instance, in the case of Markov networks, computing the un-normalized probability of an assignment to all random variables is simply the product of all potentials projected on the assignments, which in general is not considered computationally hard. However, for MLNs, ω can be extremely

large. For instance, a complete relational database corresponds to a single world of the MLN. Therefore, counting the satisfied groundings of each formula in ω is a non-trivial task (Papadimitriou and Yannakakis, 1999; Domingos and Lowd, 2009). However, even though this problem is a major bottleneck in several inference algorithms, it has not been considered in a sufficiently rigorous manner by existing MLN inference systems. For instance, we observed that existing state-of-the-art MLN systems such as Alchemy (Kok et al., 2006) and Tuffy (Niu et al., 2011) solve this counting problem either by storing the ground MLN or by using the following naive generate-and-test approach: generate a grounding and test whether it is true in ω. Both approaches are problematic since their space/time complexity is extremely large. Therefore, existing MLN inference systems fail to scale up when presented with large, real-world domains.

Solving the aforementioned counting problem efficiently is particularly important in the context of approximate inference algorithms such as Gibbs sampling (Geman and Geman, 1984; Venugopal and Gogate, 2012), MaxWalksat (Kautz et al., 1997) and MCSAT (Poon and Domingos, 2006). This is because, it turns out that in each of these algorithms, the counting problem manifests itself in different forms during every iteration of the algorithm and typically, these algorithms require several thousand iterations to converge. Thus, inference is extremely slow and non-scalable if we use naive solutions to solve the counting problem. In this chapter, we exploit advanced counting strategies to address this problem and show how these strategies can scale up inference algorithms remarkably in several cases by orders of magnitude.

Our main idea is to solve the aforementioned counting problem efficiently without grounding the full MLN and in most cases our approach is orders of magnitude better than the "generate-and-test" approach. Specifically, we encode each formula (f) as an instance of a Constraint Satisfaction Problem (CSP) ($\mathcal{C}$) such that the number of solutions to $\mathcal{C}$ can be directly used to count the satisfied groundings of f (a CSP is a special case of a Markov network). The main advantage of this encoding is its generality. That is, we can now leverage

several years of research advances in CSPs and graphical models and use virtually any exact/approximate inference algorithm along with its associated guarantees to efficiently solve the counting problem.

We demonstrate the power and generality of our approach by applying it to two widely used approximate inference algorithms: Gibbs sampling which is typically used for the marginal inference task and MaxWalkSAT which is used for the MAP inference task. We show that in both these algorithms, the main computational steps involve solving the encoded CSPs where the constraints change over time (dynamic CSP). To solve this dynamic CSP, we compile an exact junction tree for each CSP and then account for the changing constraints by modifying the junction tree messages efficiently. We evaluated both our algorithms on a wide variety of MLN benchmarks and compared them with algorithms implemented in two existing state-of-the-art MLN systems, Alchemy (Kok et al., 2006) and Tuffy (Niu et al., 2011). Our experiments clearly show that our new algorithms are several orders of magnitude more scalable than existing systems.

The rest of this chapter is organized as follows. We first describe our encoding from formulas to CSPs, then we describe the application of our technique to Gibbs sampling followed by its application to MaxWalkSAT.

6.2 Encoding the Counting Problem

An MLN represents a Markov network, defined as follows:

- We have one binary random variable in the Markov network for each possible ground atom.
- We have one propositional feature for each possible grounding of each first-order formula. The weight associated with the feature is the weight attached to the corresponding formula. A propositional feature having weight w represents the following function: $\phi(\mathbf{y}) = \exp(w)$ if $\mathbf{y}$ evaluates the feature to true and $\phi(\mathbf{y}) = 1$ otherwise.

The Markov network represents the following probability distribution:

$$\Pr(\omega) = \frac{1}{Z} \exp\left(\sum_i w_i N_{f_i}(\omega)\right) \quad (6.1)$$

$$= \frac{1}{Z} \prod_i \left[\phi_i(\omega_{S(\phi_i)})\right]^{N_{f_i}(\omega)} \quad (6.2)$$

where $\mathtt{w}_i$ is the weight of formula f_i, $N_{f_i}(\omega)$ is the number of groundings of f_i that evaluate to True given a world ω. The main computational bottleneck in several inference algorithms for MLNs, in particular Gibbs sampling and MaxWalksat, is computing $N_{f_i}(\omega)$. We call this counting problem the #`SATGround` problem. Next, we describe our approach for formulating #`SATGround` as a standard problem in constraint satisfaction. We then leverage advanced techniques in graphical model inference to solve this problem efficiently. As in the previous chapter, we require our input MLN to be in Σ-normal form (Section 5.2.1). That is, the MLN alone is in normal form and is not normalized/shattered with the evidence. This ensures that our approach is not affected by the evidence problem (Section 5.1).

6.2.1 CSP Formulation

A constraint network is a graphical model (Markov network) which contains a set of functions $\Phi = \{\phi_1, \ldots, \phi_m\}$, each defined over a subset of variables, denoted by $S(\phi)$. Each variable in $S(\phi)$ is a binary random variable and the range of all functions in Φ is $\{0, 1\}$. A 0/1 function ϕ, can also be thought of as a constraint or a relation in which all assignments $\mathbf{y}$ such that $\phi(\mathbf{y}) = 1$ are allowed while the ones having $\phi(\mathbf{y}) = 0$ are not allowed. A constraint satisfaction problem (CSP) is to find an assignment of values to all variables such that all constraints are satisfied (namely a solution). An important problem over constraint networks is computing the number of solutions to a CSP. This problem is equivalent to computing the partition function in a Markov network.

We first demonstrate our proposed CSP encoding for solving #SATGround with a simple example.[1] Consider an MLN with just one clause: $f = \forall x, \forall y, \forall z\ \mathtt{R}(x, y) \vee \mathtt{S}(y, z)$. We will focus on counting the number of false groundings of f given a world ω because it is easier to compute. Moreover, we can easily compute the number of true groundings $N_f(\omega)$ from it; $N_f(\omega)$ is equal to the number of all possible groundings (which is simply a product of the domain sizes) minus the number of false groundings.

Let us assume that the domain of each logical variable in f is $\{A, B\}$. Each triple (x, y, z) where each x, y and z can take values from the domain $\{A, B\}$ uniquely identifies each grounding of the formula. Consider a world ω shown in Figure 6.1(a). Let us associate two 0/1 functions $\phi_1(x, y)$ and $\phi_2(y, z)$ with $\mathtt{R}(x, y)$ and $\mathtt{S}(y, z)$ respectively. The 0/1 function has a value 1 iff the corresponding grounding is False in the world and 0 otherwise (see Figure 6.1(b)).

Given this set up, notice that if we take a product of the two functions $\phi_1(x, y)$ and $\phi_2(y, z)$, then the resulting function $\phi_3(x, y, z)$ will have a 1 associated with an entry (x, y, z) iff both $\mathtt{R}(x, y)$ and $\mathtt{S}(y, z)$ are False. Since the ground formula (x, y, z) evaluates to False iff both $\mathtt{R}(x, y)$ and $\mathtt{S}(y, z)$ are False, by extension $\phi_3(x, y, z) = 1$ implies that the ground formula (x, y, z) is False. Therefore, we can count the number of groundings of f that evaluate to False, by simply counting the number of ones in $\phi_3(x, y, z)$, which is the same as counting the number of solutions to the CSP having two functions $\phi_1(x, y)$ and $\phi_2(y, z)$.

Next, we will formalize this intuition and precisely define how to encode the #SATGround problem as a CSP solution counting problem.

Encoding *f-to-CSP*. Given a first-order clause f and a world ω, the corresponding CSP $\mathcal{C}$ has a variable for each (universally quantified) logical variable in f. The domain of each variable in $\mathcal{C}$ is the set of constants in the domain of the corresponding logical variable. For

[1]Note that #SATGround can also be reduced to the conjunctive query problem in relational databases and we can then use query optimization techniques (Vardi, 1982) to solve #SATGround.

$\mathtt{R}(A,A)$	1
$\mathtt{R}(A,B)$	0
$\mathtt{R}(B,A)$	0
$\mathtt{R}(B,B)$	1

$\mathtt{S}(A,A)$	0
$\mathtt{S}(A,B)$	1
$\mathtt{S}(B,A)$	0
$\mathtt{S}(B,B)$	1

(a) world ω

x	y	$\phi_1(x,y)$
A	A	0
A	B	1
B	A	1
B	B	0

y	z	$\phi_2(y,z)$
A	A	1
A	B	0
B	A	1
B	B	0

(b) Functions ϕ_1 and ϕ_2

x	y	z	$\phi_3(x,y,z)$
A	A	A	0
A	A	B	0
A	B	A	1
A	B	B	0

x	y	z	$\phi_3(x,y,z)$
B	A	A	1
B	A	B	0
B	B	A	0
B	B	B	0

(c) Function $\phi_3 = \phi_1 \times \phi_2$

Figure 6.1. (a) A possible world of an MLN having only one formula: $f = \forall x, \forall y, \forall z\ \mathtt{R}(x,y) \vee \mathtt{S}(y,z)$. The domain of each logical variable is $\{A, B\}$; (b) Functions ϕ_1 and ϕ_2 corresponding to $\mathtt{R}(x,y)$ and $\mathtt{S}(y,z)$ respectively; and (c) Function ϕ_3 which is equal to the product of the two functions given in (b). The number of 1s in ϕ_3 equals the number of groundings of f that evaluate to False.

each atom $\mathtt{R}(x_1, \ldots, x_u)$ in f, we have a relation ϕ in $\mathcal{C}$ defined as follows:

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

where $\overline{\mathbf{x}}_u = (x_1 = X_1, \ldots, x_u = X_u)$ denotes an assignment to the CSP 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.

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

Theorem 14. *Let f be a first-order clause, $x_1, \ldots, x_u$ be the (universally quantified) logical variables in f, ω be a world and let $\#Sol(\mathcal{C})$ denote the number of solutions of the CSP $\mathcal{C}$ obtained from (f, ω) using the f-to-CSP encoding. Then, $N_f(\omega) = \prod_{j=1}^{u} |\Delta_{x_j}| - \#Sol(\mathcal{C})$ where Δ_{x_j} is the set of constants in the domain of x_j.*

Proof. Let $\phi_1 \ldots \phi_K$ be the functions in $\mathcal{C}$. Let $\overline{\mathbf{x}}$ be a solution to $\mathcal{C}$. This means that $\prod_{j=1}^{K} \phi_j(\overline{\mathbf{x}}) = 1$ where $\phi_j(\overline{\mathbf{x}})$ is the projection of $\overline{\mathbf{x}}$ on ϕ_j. Since the variables in $\mathcal{C}$ correspond

Table 6.1. #SATGround complexities using various strategies. M is the number of formulas, $\mathbf{V}_i$ is the set of variables in the CSP encoded for the i^{th} formula, d is the domain-size of each variable and w_i^* is the treewidth of the CSP encoded for the i^{th} formula.

Algorithm	**Space Complexity**	**Time complexity**
Pre-Ground	$O(\sum_{i=1}^{M} d^{\lvert\mathbf{V}_i\rvert})$	$O(\sum_{i=1}^{M} d^{\lvert\mathbf{V}_i\rvert})$
Lazy-Ground	$O(1)$	$O(\sum_{i=1}^{M} d^{\lvert\mathbf{V}_i\rvert})$
And-OR Tree	$O(M)$	$O(\sum_{i=1}^{M} d^{w_i^* log(\lvert\mathbf{V}_i\rvert)+1})$
Junction Tree	$O(\sum_{i=1}^{M} d^{w_i^*})$	$O(\sum_{i=1}^{M} d^{w_i^*+1})$

to the universally quantified logical variables in f, we can directly obtain a ground formula $\bar{f}$ from $\overline{\mathbf{x}}$ by grounding each logical variable in f with the constant specified for it in $\overline{\mathbf{x}}$. Let $\bar{f}_1 \ldots \bar{f}_k$ be the ground atoms in $\bar{f}$. By our construction of $\mathcal{C}$,

$$\phi_j(\overline{\mathbf{x}}) = \begin{cases} 1 & \text{if } \bar{f}_j \text{ is a negative-signed literal and } \omega \text{ assigns 1 to } \bar{f}_j \\ 1 & \text{if } \bar{f}_j \text{ is a positive-signed literal and } \omega \text{ assigns 0 to } \bar{f}_j \\ 0 & \text{Otherwise} \end{cases}$$

Clearly, $\bar{f}$ is false in ω iff $\prod_{j=1}^{K} \phi_j(\overline{\mathbf{x}}) = 1$. Therefore, the total number of unsatisfied/false groundings of f due to assignment ω is equal to $\#Sol(\mathcal{C})$. Also, the total number of possible groundings of f is equal to $\prod_{j=1}^{u} |\Delta_{x_j}|$. Therefore, $N_f(\omega) = \prod_{j=1}^{u} |\Delta_{x_j}| - \#Sol(\mathcal{C})$.

□

6.2.2 Counting the Number of Solutions of the CSP

Since we have reduced the #SATGround problem to the CSP solution counting problem, it is clear that we can use any CSP/graphical model inference algorithm and leverage its advances and guarantees to efficiently compute the former. Table 6.1 shows the complexity bounds for various strategies and algorithms for solving the #SATGround problem.

Alchemy (Kok et al., 2006) uses the pre-ground strategy, namely it grounds all clauses that it is unable to lift. Thus, its worst case time and space complexity bounds are exponential in the maximum number of variables in the MLN formulas. The pre-ground strategy

is useful when there is large amount of evidence. In presence of evidence, one can use unit propagation to remove all clauses that are either True or False. This reduces the space complexity as well as the time complexity of subsequent counting problems.

An alternative strategy is to do lazy grounding, which reduces to solving the CSP using the generate and test approach. In this approach, we count the solutions by generating each tuple and testing whether it is a solution or not. Although this approach has constant space complexity, its worst case time complexity is the same as the pre-ground approach. Moreover, unlike the pre-ground approach, this approach is unable to take advantage of unit propagation and the worst-case results apply to all subsequent operations.

A better, more powerful approach is to use advanced search and elimination techniques such as AND/OR search (Dechter and Mateescu, 2007), recursive conditioning (Darwiche, 2001), junction tree propagation (Lauritzen and Spiegelhalter, 1988), as well as knowledge compilation techniques such as arithmetic circuits (Darwiche, 2003) and AND/OR multi-valued decision diagrams (Mateescu et al., 2008). Here, we focus on using the junction tree algorithm for computing the solution counts exactly. However, a key feature of our approach is that our formulation can easily leverage other advanced approximate inference methods from graphical models. To do this, we simply replace exact solution counting using junction trees with advanced approximate inference techniques such as SampleSearch (Gogate and Dechter, 2007a), Generalized BP (Yedidia et al., 2005), IJGP (Mateescu et al., 2010), WISH (Ermon et al., 2013), etc. Thus, our approach yields a novel way to apply inference techniques from graphical models to MLNs. That is, instead of using graphical model inference methods directly for solving inference problems in MLNs, here, we apply it to solve the counting problem in the CSPs encoded from MLN formulas. The distinction between the two is important because, the former is clearly not likely to be scalable since the Markov network underlying the MLN is always extremely large, however, the latter solves a simpler problem which is embedded within other inference algorithms, and using approximations here is likely to yield much more promising and scalable results.

6.2.3 Junction Trees for Solution Counting

We now briefly review the junction tree algorithm used to compute the number of solutions $\#Sol(\mathcal{C})$ of $\mathcal{C}$.

Definition 17. *Given the CSP $\mathcal{C}$, a* **junction tree** *is a tree $\mathcal{T}(\mathbf{V}, \mathbf{E})$ in which each vertex $V \in \mathbf{V}$ (also called a cluster) and edge $E \in \mathbf{E}$ are labeled with a subset of variables, denoted by $L(V)$ and $L(E)$ such that: (i) for every function ϕ defined in $\mathcal{C}$, there exists a vertex $L(V)$ such that $S(\phi) \subseteq L(V)$ and (ii) for every variable x in $\mathcal{C}$, the set of vertexes and edges in $\mathcal{T}$ that mention x form a connected sub-tree in $\mathcal{T}$ (called the running intersection property).*

Given a junction tree $\mathcal{T}$ of $\mathcal{C}$, we can compute the solution counts as well as the marginal probability of each CSP variable (the fraction of solutions that the variable participates in) by calibrating $\mathcal{T}$. We *calibrate* $\mathcal{T}$ by selecting a cluster as the root and performing sum-product message passing in two passes: from the leaves to the root (collect pass) and then from the root to the leaves (distribute pass). Formally, the message sent from cluster i to cluster j, denoted by $m_{i \to j}$, is given by

$$m_{i \to j}(\mathbf{y}) = \sum_{\overline{\mathbf{z}}} \prod_{\phi \in \Phi(V_i)} \phi(\overline{\mathbf{y}}, \overline{\mathbf{z}}) \prod_{k \in N(i) \setminus \{j\}} m_{k \to i}(\overline{\mathbf{y}}, \overline{\mathbf{z}}) \tag{6.3}$$

where $\Phi(V_i)$ is the set of functions assigned to vertex V_i, and $N(i)$ is the set of indexes of the neighbors of V_i in $\mathcal{T}$.

The number of solutions to $\mathcal{C}$ can be computed from any vertex V_k using the following equation:

$$\#Sol(\mathcal{C}) = \sum_{\overline{\mathbf{x}}} \prod_{\phi \in \Phi(V_k)} \phi(\overline{\mathbf{x}}) \prod_{j \in N(k)} m_{j \to k}(\overline{\mathbf{x}}) \tag{6.4}$$

The complexity of computing the solution counts using a junction tree is exponential in the maximum cluster size of the tree which equals treewidth plus 1. Next, we describe efficient implementations of two classical approximate inference algorithms, Gibbs sampling and MaxWalkSAT, using junction trees.

6.3 Application I: Gibbs Sampling

In Gibbs sampling, we start with a random world $\omega^{(0)}$ which is consistent with the state of the evidence atoms (atoms whose truth values are known). That is, if the given evidence assigns 0 (or 1) to an atom, $\omega^{(0)}$ assigns 0 (or 1) to that same atom. For all other atoms, the starting state is chosen randomly. Then, at each iteration $i > 0$, we compute the conditional distribution over a randomly chosen non-evidence ground atom $\mathtt{R}$ given $\omega^{(i-1)}_{-\mathtt{R}}$ where $\omega^{(i-1)}_{-\mathtt{R}}$ is the projection of $\omega^{(i-1)}$ on all ground atoms of the MLN except $\mathtt{R}$. Then, we sample a new value for $\mathtt{R}$, denoted by $\overline{\mathtt{R}}$, from this conditional distribution and set $\omega^{(i)} = (\omega^{(i-1)}_{-\mathtt{R}}, \overline{\mathtt{R}})$. Note that for brevity, we have abused notation and denoted the ground atom $\mathtt{R}(X_1, \ldots, X_r)$ as $\mathtt{R}$. Also, note that we never sample an evidence atom and therefore each subsequent world that is generated by our Gibbs sampler is consistent with the evidence and thus the sampler converges to the correct distribution.

The main computational bottleneck in Gibbs sampling is computing the conditional distribution over the ground atom $\omega^{(i)}$. It is given by:

$$\Pr(\mathtt{R} = j | \omega^{(i)}_{-\mathtt{R}}) \propto \sum_{f_k \in F(\mathtt{R})} \mathtt{w}_k N_{f_k}(\omega^{(i)}_{-\mathtt{R}}, \mathtt{R} = j) \tag{6.5}$$

where $j \in \{0, 1\}$ and $F(\mathtt{R})$ is the set of first-order formulas in the MLN that contain $\mathtt{R}$ (or more specifically the set of first order formulas that contain at least one ground atom that is in the Markov Blanket of at least one grounding of $\mathtt{R}$).

Given a formula f_k and a world ω, let $\mathcal{C}_k$ denote the constraint network obtained from (f_k, ω) using the *f-to-CSP* encoding. Let $\mathcal{T}_k$ denote the junction tree obtained from $\mathcal{C}_k$. If we calibrate the junction tree $\mathcal{T}_k$, then we can easily compute $N_{f_k}(\omega)$ from it (see Eq.6.4). However, technically, for Gibbs sampling, we need not calibrate the junction trees. That is, we can implement Gibbs sampling more efficiently using un-calibrated junction trees. Specifically, we designate a node in $\mathcal{T}_k$ to be the root and pass messages from the leaves to this designated root. The number of solutions to the CSP can be directly computed at the

root node. Thus, using un-calibrated junction trees is approximately twice as efficient as compared to using calibrated junction trees. However, calibration is needed when we require the correct distributions at each node (this is the case in the MaxWalkSAT algorithm that we present in the next section). Assuming that we compute $N_{f_k}(\omega)$ from $\mathcal{T}_k$, the main challenge is computing $N_{f_k}(\omega')$ where $\omega' = (\omega_{-\mathtt{R}}, \neg\omega_{\mathtt{R}})$. We describe how to compute it next.

Consider the two CSPs $\mathcal{C}_k$ and $\mathcal{C}'_k$ obtained from (f_k, ω) and (f_k, ω') respectively using the f-to-CSP encoding. Since f_k defines the scope of the functions in the CSP, both CSPs have functions defined over the same scope. Moreover, since ω and ω' differ only in a truth assignment to one ground atom, the corresponding functions in $\mathcal{C}_k$ and $\mathcal{C}'_k$ differ only in at most one entry. Thus, we can efficiently construct a calibrated junction tree $\mathcal{T}'_k$ from $\mathcal{T}_k$ by appropriately propagating only the changed entries. In most cases, the update operation is much more efficient because only a fraction of each message may change. For example, consider a simple junction tree with two clusters C_1 and C_2. Let C_1 contain $\phi_1(X, Y)$ and C_2 contain $\phi_2(Y, Z)$, where the message m_{12} is defined on Y. If $\phi_1(X, Y)$ changes exactly for one entry say $X = 0$ and $Y = 1$, then, out of the Δ_Y entries in m_{12}, exactly 1 entry corresponding to $Y = 1$ is modified. Multiplying m_{12} with $\phi_2(Y, Z)$ results in a change in Δ_Z out of $\Delta_Y\Delta_Z$ entries. In general, assuming that all functions that have changed are present in a cluster V in $\mathcal{T}_k$, we propagate the changed messages away from V towards the designated root stopping propagation if the old and new messages are identical to each other. We then compute the changed solution counts at the root.

6.4 Application II: MaxWalkSAT

The MAP task in MLNs is the task of finding the world with the maximum probability in the joint distribution represented by the MLN. It can be cast as an optimization problem

Algorithm 14: MaxWalkSAT($\mathcal{M}$, p, $maxFlips$)

Initialize: ω = a randomly generated possible world of $\mathcal{M}$.
$\omega_{Best} = \omega$
for $flip = 1$ *to* $maxFlips$ **do**
  if ω *satisfies all clauses in* $\mathcal{M}$ **then**
    return ω
  `// Clause selection step`
  f_k = a random unsatisfied ground clause in $\mathcal{M}$
  With probability p
  `// Random step`
    Flip a randomly selected ground atom of f_k
  Else
  `// Greedy step`
    Flip the ground atom of f_k that yields a world with the maximum weight
  if $\sum_{f_i} N_{f_i}(\omega) > \sum_{f_i} N_{f_i}(\omega_{Best})$ **then**
    $\omega_{Best} = \omega$
return ω_{Best}

where we find a possible world that maximizes the sum of weights of satisfied clauses as follows.

$$\max_{\omega} \sum_{f} N_f(\omega) \mathtt{w}_f \tag{6.6}$$

Any weighted satisfiability solver can be used to solve Eq. (6.6). However the most commonly used solver is MaxWalkSAT (Kautz et al., 1997). The latter is a weighted variant of WalkSAT, a local-search algorithm that was successfully applied for satisfiability testing (Selman et al., 1996).

Algorithm 14 illustrates the MaxWalkSAT algorithm. It takes as input an MLN $\mathcal{M}$ and a probability p. The algorithm begins by randomly generating a possible world. Then, at each iteration, it selects a false ground clause uniformly at random and flips the value assigned to one of its atoms as follows. With probability p, the atom (literal) to be flipped is selected uniformly at random and with probability $(1 - p)$, the atom which when flipped maximizes

the number of satisfied ground clauses is selected (greedy hill-climbing step which selects the best atom).

Unlike Gibbs sampling, for the MaxWalkSAT algorithm, we use calibrated junction trees since the clause selection step requires access to the probability distribution at each cluster. Assuming that we can perform this step (we describe how to do it later), the last two steps of flipping a random atom or the best atom in that clause can be accomplished by using a similar approach to what was described for Gibbs sampling. Specifically, we compute $N_{f_k}(\omega)$ from the calibrated junction tree $\mathcal{T}_k$ and to compute $N_{f_k}(\omega')$, where ω' is obtained from ω by flipping a single ground atom, we propagate the changed messages in $\mathcal{T}_k$. That is, assuming that all the changed functions are in cluster V in $\mathcal{T}_k$, we designate V as the root and distribute the changed parts of the messages away from it, stopping propagation beyond a cluster if the new message and the old message to the cluster are the same. Clearly, to find the best atom to flip in a clause with k atoms, we need to re-propagate the changed messages k times for k different worlds, where each world differs from the other worlds in exactly one potential entree. The challenging step in MaxWalkSAT is selecting a false ground clause uniformly at random. Next, we describe a procedure for accomplishing this.

Selecting a False Clause uniformly at random: We solve this problem using a two step procedure. In the first step, we select a first order formula f_i such that the probability of selecting f_i is proportional to the number of its false groundings. In the second step, we select a false ground clause of f_i uniformly at random.

To select a first-order formula, we first compute the number of false ground clauses for all first-order formulas (using the calibrated junction tree) and normalize them to yield a distribution over the formulas. We then sample a first-order formula from this distribution.

Let f_i be the sampled first-order formula. To select a false ground clause of f_i uniformly at random, we sample the calibrated junction tree of f_i using the junction tree solution sampling method described in (Dechter et al., 2002; Gogate, 2009). Sampling the junction

tree yields a solution to the corresponding CSP $\mathcal{C}_i$. Based on our encoding, each solution of $\mathcal{C}_i$ corresponds to a false clause of f_i.

Selecting a False Clause in Presence of Evidence: In presence of evidence, the solution sampling method described above cannot be used directly because the sampled ground clause may be *trivially false*. That is, suppose our solution sampler over $\mathcal{C}_i$ selects an unsatisfied ground clause yields a solution to the CSP where each ground atom corresponding to the solution is an evidence atom, none of the atoms can be flipped because their truth values are fixed. Thus, the solution sampling method must be modified so that it never selects a trivially false ground clause.

Before describing our method to solve this problem, we introduce some notation. Let $\mathcal{E}$, $\mathcal{E}_u$ and $\mathcal{E}_t$ denote the set of ground clauses, false ground clauses and trivially false ground clauses of f respectively. From the definition, it is obvious that $\mathcal{E}_t \subseteq \mathcal{E}_u \subseteq \mathcal{E}$ and any method that samples an element of the set $(\mathcal{E}_u \setminus \mathcal{E}_t)$ will not sample a trivially false clause. A simple way of ensuring this is to use rejection sampling to find such a clause. That is, whenever our solution sampler samples an element from $\mathcal{E}_t$, we reject that solution. However, this approach will be quite slow, especially when there is a large amount of evidence since it will lead to several rejected solutions.

A more clever approach to solve the above problem is to use two constraint networks. Formally, given a formula f and the constraint network $\mathcal{C}$ obtained from it, we define another constraint network (referred to as evidence network) $\mathcal{C}^\epsilon$ as follows. $\mathcal{C}^\epsilon$ is defined over same set of variables as $\mathcal{C}$. Corresponding to each function $\phi(\overline{\mathbf{x}}_u)$ in $\mathcal{C}$, we define a function $\phi^\epsilon(\overline{\mathbf{x}}_u)$ in $\mathcal{C}^\epsilon$ as follows: let $\mathtt{R}(x_1, \ldots, x_u)$ denote the atom corresponding to $\phi(\overline{\mathbf{x}}_u)$ in $\mathcal{C}$

$$\phi^\epsilon(\overline{\mathbf{x}}_u) = \begin{cases} \phi(\overline{\mathbf{x}}_u) & \text{if } \mathtt{R}(x_1, \ldots, x_u) \text{ is evidence} \\ 0 & \text{Otherwise} \end{cases}$$

where $\overline{\mathbf{x}}_u = (x_1 = X_1, \ldots, x_u = X_u)$. From construction of $\mathcal{C}^\epsilon$ it follows that $\#Sol(\mathcal{C}^\epsilon) = |\mathcal{E}_t|$. We can therefore select a first order formula from the MLN as follows. We select a formula f

with probability proportional to the total number of its non-trivial false groundings which is clearly equal to $\prod_{j=1}^{u} |\Delta_{x_j}| - \#Sol(\mathcal{C}^{\epsilon})$, where $x_1 \ldots x_u$ are the universally quantified logical variables in f. Once f is selected, selecting a non-trivially false ground clause of f is slightly more tricky. We do this as follows. We maintain two calibrated junction trees $\mathcal{T}^{\epsilon}$ and $\mathcal{T}$ constructed from the two constraint networks $\mathcal{C}^{\epsilon}$ and $\mathcal{C}$ respectively. We then dynamically obtain a distribution over non-trivially false ground clauses from $\mathcal{T}^{\epsilon}$ and $\mathcal{T}$ as given below.

Each entry in a message $m_{i\to j}$ in $\mathcal{T}$ corresponding to formula f can be mapped into a partially ground formula, where for every variable specified in the message, we ground those variables (in f) with values specified in the message entry. We refer to the set of all possible partially ground formulas that can be obtained from $m_{i\to j}$ as the projection of the ground formulas of f on $m_{i\to j}$.

Theorem 15. *For any message $m_{i\to j}$ in $\mathcal{T}$ corresponding to formula f, and its corresponding message $m^{\epsilon}_{i\to j}$ in $\mathcal{T}^{\epsilon}$, subtracting each entry in $m_{i\to j}$ with its corresponding entry in $m^{\epsilon}_{i\to j}$ yields a (un-normalized) distribution over non-trivially false ground formulas of f projected on $m_{i\to j}$.*

Proof. Let $x_1 \ldots x_u$ be the variables in $m_{i\to j}$ of $\mathcal{T}$. Since $\mathcal{T}$ is calibrated, $m_{i\to j}$ yields a (un-normalized) distribution over the elements of $\mathcal{E}_u$ projected on $m_{i\to j}$. That is, let the set of unsatisfied ground clauses corresponding to partially grounding the variables of f with $x_1 = \bar{x}_1 \ldots x_u = \bar{x}_u$ be the set $\bar{\mathcal{E}}_u$, then, the value corresponding to $x_1 = \bar{x}_1 \ldots x_u = \bar{x}_u$ in $m_{i\to j}$ is equal to $|\bar{\mathcal{E}}_u|$. Similarly, let the trivially unsatisfied ground clauses corresponding to grounding the variables of f with $x_1 = \bar{x}_1 \ldots x_u = \bar{x}_u$ be the set $\bar{\mathcal{E}}_t$, then, the value corresponding to $x_1 = \bar{x}_1 \ldots x_u = \bar{x}_u$ in $m^{\epsilon}_{i\to j}$ is equal to $|\bar{\mathcal{E}}_t|$.

For the distribution over $\mathcal{E}_u \setminus \mathcal{E}_t$, the value corresponding to $x_1 = \bar{x}_1 \ldots x_u = \bar{x}_u$ is equal to $|\bar{\mathcal{E}}_u \setminus \bar{\mathcal{E}}_t|$. However, by definition, $\bar{\mathcal{E}}_t \subseteq \bar{\mathcal{E}}_u$. Therefore, $|\bar{\mathcal{E}}_u \setminus \bar{\mathcal{E}}_t| = |\bar{\mathcal{E}}_u| - |\bar{\mathcal{E}}_t|$. Thus, $m_{i\to j} - m^{\epsilon}_{i\to j}$ yields the required distribution over $\mathcal{E}_u \setminus \mathcal{E}_t$ projected on $m_{i\to j}$. □

Using Theorem 15, we can compute a new junction tree $\mathcal{T}'_k$ which encodes the required distribution over non-trivially false ground clauses dynamically by subtracting the messages in $\mathcal{T}_k$ from the messages in $\mathcal{T}^\epsilon_k$. Note that, we can do this efficiently because the junction tree $\mathcal{T}^\epsilon_k$ needs to be calibrated only once as the evidence does not change. We then sample from $\mathcal{T}'_k$ using the solution sampling techniques detailed in (Dechter et al., 2002; Gogate, 2009) to obtain a non-trivially false ground clause uniformly at random.

6.5 Extensions

6.5.1 Existential Quantifiers

In this subsection, we describe how our approach, specifically the *f-to-CSP encoding*, can be extended to handle existential quantifiers. We consider two cases.

Case 1: If no universal quantifier is nested inside an existential one then each first-order formula is just a compact representation of a disjunction over the groundings of the existentially quantified variables. For example, if the domain is $\{A, B\}$, then the first-order formula $\forall x, \forall z\ \exists y\ \mathtt{R}(x, y) \vee \mathtt{S}(z, y)$ represents the clause $\forall x, \forall z\ \mathtt{R}(x, A) \vee \mathtt{R}(x, B) \vee \mathtt{S}(z, A) \vee \mathtt{S}(z, B)$. Given a world ω, we can encode this clause as a CSP having two variables x and z, and four unary constraints $\phi_1(x)$, $\phi_2(x)$, $\phi_3(z)$ and $\phi_4(z)$ corresponding to the truth assignments to $\mathtt{R}(x, A)$, $\mathtt{R}(x, B)$, $\mathtt{S}(z, A)$ and $\mathtt{S}(z, B)$ respectively in the world ω. In general, the variables in the CSP encoding are the universally quantified variables in the clause and we have a constraint over the universally quantified variables for each atom and each possible value assignment to the existentially quantified variables. Thus, an existentially quantified variable increases the number of constraints but does not factor in determining the complexity of the junction tree computations.

Case 2: If a universal quantifier is nested within an existential one, then the first-order clause corresponds to a disjunction of conjunctions of propositional clauses. This case is

much harder than the previous case and whether our approach can be extended to such cases is an open problem.

6.5.2 Lifted Inference

Our approach can be extended with several advances from research in lifted inference (cf. (Poole, 2003; de Salvo Braz, 2007; Gogate and Domingos, 2011b; Van den Broeck et al., 2011; Venugopal and Gogate, 2012; Bui et al., 2013; Sarkhel et al., 2014a)). For example, consider the lifted MaxWalkSAT algorithm described in (Sarkhel et al., 2014b). Sarkhel et al.'s algorithm works as follows. It first converts a normal MLN to a non-shared (normal) MLN by grounding all the shared terms in each formula. Then, it reduces the domain-size of all non-shared terms to 1 and computes the MAP value over this new MLN using MaxWalkSAT. Sarkhel et al.'s approach works because the MAP value for each atom in a non-shared MLN lies at the extreme: ground atoms in a non-shared MLN are either all true or all false in the MAP tuple. Thus, unlike propositional MaxWalkSAT where the search space is exponential in the number of ground atoms in the MLN, using Sarkhel et al.'s result, we can reduce this search space by orders of magnitude in several cases resulting in a much more accurate and scalable MAP solver. However, Sarkhel et al.'s approach uses a pre-ground strategy to solve #`SATGround` which is computationally inefficient(see Table 6.1). To combine Sarkhel et al.'s algorithm with our approach, we can reduce the domains of the CSP variables that correspond to non-shared logical variables of the MLN. Effectively, this removes the variable from all the junction tree computations, possibly decreasing its treewidth.

6.6 Experiments

6.6.1 Setup

We used 10 benchmark MLNs with varied structures and random evidence ($< 25\%$) to evaluate the performance of our Gibbs sampling and MaxWalkSAT algorithms. We compared

our system with two state-of-the-art MLN systems: Alchemy and Tuffy. Alchemy implements both Gibbs sampling as well as MaxWalkSAT whereas Tuffy only implements MaxWalkSAT. Of the 10 benchmarks, 5 were from Alchemy as follows.

(i) WebKB (webkb)

(ii) Entity Resolution (er)

(iii) Citation Segmentation (seg)

(iv) Protein Interaction (protein)

(v) Coreference Resolution (coref)

We added 5 synthetic benchmarks with varied structures as follows:

(i) student: $\neg\texttt{Student}(x, p) \vee \neg\texttt{Publish}(x, z) \vee\texttt{Cited}(z, u)$

(ii) relation: $\neg\texttt{Friends}(x, y) \vee \neg\texttt{Related}\ (y, z) \vee\texttt{Likes}\ (z, x)$

(iii) longchain: $\neg\texttt{R}_1(x_1,x_2) \vee \neg\texttt{R}_2(x_2,x_3) \ldots \texttt{R}_6(x_6,x_7)$

(iv) transitive1: $\neg\texttt{Likes}(x, y) \vee \neg\texttt{Likes}(y, z) \vee\texttt{Likes}(y, x)$

(v) transitive2: $\neg\texttt{Friends}\ (x, y) \vee \neg\texttt{Friends}(y, z) \vee \texttt{Friends}(z, x)$

.

Each synthetic benchmark was designed to illustrate the influence of MLN structure on scalability. Specifically, though similar-looking, student has a smaller treewidth (for its encoded CSP) compared to relation. Longchain illustrates an extremely large formula, but the encoded CSP has a small treewidth. Finally, though transitive1 and transitive2 appear similar, it turns out that in each step of Gibbs/MaxWalkSAT, very few messages of the junction tree underlying transitive1 need to be updated while for transitive2, nearly all messages need to be updated.

Tables 6.2 and 6.3 show our results for Gibbs sampling and MaxWalkSAT respectively. For our evaluation, we compute two metrics: the compilation time (*CTime*) in seconds and the sampling/flip rate (*SRate*/*FRate*). *CTime* is the time taken to initialize our junction trees. *SRate* is the number of samples generated in a second for Gibbs sampling and *FRate* is the number of flips per second in MaxWalkSAT.

6.6.2 Results for Gibbs Sampling

Both *CTime* as well as *SRate* depends upon the structure as well as the #groundings in the MLN. We can see from Table 6.2 that *CTime* was quite negligible for almost all the MLNs (at most 21 seconds). *SRate* depends upon the efficiency of updating the junction tree messages during Gibbs sampling. For example, in transitive1, we could generate 88,000 samples/second because each update operation is very efficient, while for transitive2 which has the same #groundings, we could generate only 73 samples/second. Similarly, the MLNs, student-100 and relation-500 have approximately the same #groundings, however, their *SRate*s are vastly different due to the treewidth of their encoded CSPs. Student has treewidth 1 whereas relation has treewidth 2, therefore, while we could collect more than 11,000 samples in a second for student-100, we could only collect about 275 samples for relation-500. On the other hand, for the same treewidth, #groundings affects *SRate*. For example, longchain-1000 is almost 10 million times larger than longchain-100. Therefore, *FRate* on longchain-1000 is just 10% of the *FRate* on longchain-1000. As seen from the results, none of these large benchmarks could be processed by Alchemy.

6.6.3 Results for MaxWalkSAT

Table 6.3 shows our results for MaxWalkSAT. *CTime* is very similar to the Gibbs sampler. Again, *FRate* depends upon the MLN structure and the number of groundings because both affect the efficiency of the junction tree operations. For instance, since student has lower

Table 6.2. Results on benchmarks for Gibbs sampling using our approach. *SRate* is the sampling rate (#samples/second) and *CTime* is the compilation time in seconds. X denotes that the system ran out of time or memory.

MLN	**#Groundings**	**Our System**		**Alchemy**
		CTime	*SRate*	
student-100	100 million	0	11397	X
student-500	10 billion	0	496	X
student-1000	100 billion	0	117	X
relation-100	1 million	0	7047	X
relation-500	100 million	1	274	X
relation-1000	1 billion	10	68	X
longchain-100	10^{14}	0	3235	X
longchain-500	10^{18}	0	126	X
longchain-1000	10^{21}	1	31	X
transitive1-100	1 million	0	88739	X
transitive1-500	100 million	0	24568	X
transitive1-1000	1 billion	0	8879	X
transitive2-100	1 million	1	73	X
transitive2-500	100 million	1	0.6	X
webkb	10 billion	1	183	X
seg	1 billion	0	32	X
er	1 billion	21	5	X
protein	100 million	1	116	X
coref	100 million	2	180	X

treewidth than relation, the *FRate* for student is much higher. Transitive2 has the lowest *FRate* because each update involves recomputing the junction tree messages from scratch. When these updates are efficient as in transitive1, *FRate* is several orders of magnitude higher. Both Alchemy and Tuffy did not work on most of the benchmarks except on three of the smallest-sized ones; our approach was slightly worse than Tuffy and Alchemy only on transitive2-100.

To summarize, our results clearly demonstrate the superior scalability of our approach over the pre-grounding approaches used by Tuffy and Alchemy.

6.7 Summary

For large MLNs a sub-step, which is typically a bottleneck, in several inference algorithms is "counting the true groundings of a first-order formula in a possible world". In this chapter, we

Table 6.3. Results on benchmarks for MaxWalkSAT. For each system, we show *CTime*;*FRate*, where *CTime* is the compilation time (in seconds) for our system or the grounding time in Alchemy/Tuffy. *FRate* is the flip rate (#Flips/second). "X" denotes that the system ran out of time or memory.

MLN	**#Groundings**	**Our System**		**Alchemy**		**Tuffy**	
		CTime	*FRate*	*CTime*	*FRate*	*CTime*	*FRate*
student-100	100 million	0	31629	X	X	X	X
student-500	10 billion	0	252	X	X	X	X
student-1000	100 billion	0	72	X	X	X	X
relation-100	1 million	0	2455	95	1000	75	300
relation-500	100 million	1	142	X	X	X	X
relation-1000	1 billion	13	36	X	X	X	X
longchain-100	10^{14}	0	928	X	X	X	X
longchain-500	10^{18}	0	50	X	X	X	X
longchain-1000	10^{21}	1	12	X	X	X	X
transitive1-100	1 million	0	32082	75	350	65	100
transitive1-500	100 million	0	1032	X	X	X	X
transitive1-1000	1 billion	0	284	X	X	X	X
transitive2-100	1 million	0	30	75	200	65	150
transitive2-500	100 million	1	0.2	X	X	X	X
webkb	10 billion	1	48	X	X	X	X
seg	1 billion	0	8	X	X	X	X
er	1 billion	26	1.2	X	X	X	X
protein	100 million	1	6.8	X	X	X	X
coref	100 million	3	6	X	X	X	X

proposed a novel approach to solve this counting problem efficiently by encoding it a CSP solution counting problem. The main strength of our approach is its generality, i.e., our encoding allows us to exploit numerous advances, guarantees, principles and techniques in the active research area of CSP and probabilistic graphical models. Further, to demonstrate the power of our approach, we developed a junction tree based exact CSP solution counting algorithm and applied it to two widely used MLN inference techniques, Gibbs sampling and MaxWalkSAT, both of which require an answer to a dynamic version of the counting problem at each iteration. Our experiments with these algorithms on a wide variety of MLN benchmarks with large domain-sizes clearly showed that our approach was orders of magnitude more scalable than existing state-of-the-art inference systems.

CHAPTER 7

JOINT INFERENCE FOR EXTRACTING BIOMEDICAL EVENTS

Acknowledgements

The work presented in this chapter is based on a joint project that we executed with the NLP research group at UT-Dallas. We deeply acknowledge Chen Chen and Dr. Vincent Ng's contribution to the work presented in this chapter. Specifically, they provided us with the NLP expertise and domain knowledge that was essential for this task. Further, Chen Chen and Dr. Ng developed the essential linguistic features and designed the SVM based pipeline event extractor used in this chapter. Chen Chen worked closely with me on the experiments and I acknowledge his contribution towards helping our systems reach state-of-the-art performance.

7.1 Introduction

Event extraction is the task of extracting and labeling all instances in a text document that correspond to a pre-defined event type. This task is quite challenging for a multitude of reasons: events are often nested, recursive and have several arguments; there is no clear distinction between arguments and events; etc. For instance, consider the BioNLP Genia event extraction shared task (Nédellec et al., 2013). In this task, participants are asked to extract instances of a pre-defined set of biomedical events from text. An event is identified by a keyword called the *trigger* and can have an arbitrary number of arguments that correspond to pre-defined argument types. The task is complicated by the fact that an event may serve as an argument of another event (nested events). An example of the task is shown in Figure 7.1.

As we can see, event $E13$ takes as arguments two events, $E14$ and $E12$, which in turn has $E11$ as one of its arguments.

A standard method that has been frequently employed to perform this shared task uses a *pipeline* architecture with three steps: (1) detect if a token is a trigger and assign a trigger type label to it; (2) for every detected trigger, determine all its arguments and assign types to each detected argument; and (3) combine the extracted triggers and arguments to obtain events. Though adopted by the top-performing systems such as the highest scoring system on the BioNLP'13 Genia shared task (Kim et al., 2013), this approach is problematic for at least two reasons. First, as is typical in pipeline architectures, errors may propagate from one stage to the next. Second, since each event/argument is identified and assigned a type independently of the others, it fails to capture the relationship between a trigger and its neighboring triggers, an argument and its neighboring arguments, etc.

More recently, researchers have investigated joint inference techniques for event extraction using Markov Logic Networks (MLNs) (e.g., (Poon and Domingos, 2007), (Poon and Vanderwende, 2010), (Riedel and McCallum, 2011a)), a statistical relational model that enables us to model the dependencies between different instances of a data sample. However, it is extremely challenging to make joint inference using MLNs work well in practice (Poon and Domingos, 2007). One reason is that it is generally difficult to model sophisticated linguistic features using MLNs. The difficulty stems from the fact that some of these features are extremely high dimensional (e.g., (Chen and Ng, 2012), (Huang and Riloff, 2012b), (Li et al., 2012), (Li et al., 2013), (Li et al., 2013)), and to reliably learn weights of formulas that encode such features, one would require an enormous number of data samples. Moreover, even the complexity of approximate inference on such models is quite high, often prohibitively so. For example, a trigram can be encoded as an MLN formula, $\texttt{Word}(w_1, p-1) \wedge \texttt{Word}(w_2, p) \wedge \texttt{Word}(w_3, p+1) \Rightarrow \texttt{Type}(p, T)$. For any given position (p), this formula has W^3 groundings, where W is the number of possible words, making it too large for learning/inference.

... demonstrated that HOIL-1L interacting protein (HOIP), a ubiquitin ligase that can catalyze the assembly of linear polyubiquitin chains, is recruited to DC40 in a TRAF2-dependent manner following engagement of CD40 ...

(a) Sentence fragment

ID	Event Type	Trigger	Arguments
$E11$	*Binding*	recruited	*Theme*={HOIL-1L interacting protein,CD40}
$E12$	*Regulation*	dependent	*Theme*=$E11$, *Cause*=TRAF2
$E13$	*+ve Regulation*	following	*Theme*=$E12$, *Cause*=$E14$
$E14$	*Binding*	engagement	*Theme*=CD40

(b) Events

Figure 7.1. Example of event extraction in the BioNLP Genia task. (b) shows all the events extracted from sentence (a). Note that successful extraction of $E13$ depends on $E12$ and $E14$.

Therefore, current MLN-based systems tend to include a highly simplified model ignoring powerful linguistic features. This is problematic because such features are essential for event extraction.

We describe two contributions in this chapter. First, we propose a novel model for biomedical event extraction based on MLNs that addresses the aforementioned limitations by leveraging the power of Support Vector Machines (SVMs) (Vapnik, 1995; Joachims, 1999) to handle high-dimensional features. Specifically, we (1) learn SVM models using rich linguistic features for trigger and argument detection and type labeling; (2) design an MLN composed of *soft formulas* (each of which encodes a soft constraint whose associated weight indicates how important it is to satisfy the constraint) and *hard formulas* (constraints that always need to be satisfied, thus having a weight of ∞) to capture the relational dependencies between triggers and arguments; and (3) encode the SVM output as *prior knowledge* in the MLN in the form of soft formulas, whose weights are computed using the confidence values generated by the SVMs. Note that in the ideal case, encoding (1) and (2) together in the MLN would yield a more powerful joint model. However, to do this, we would require more robust MLN learning software/tools that can handle sparse, high-dimensional features than those currently available (Kok et al., 2008). Our formulation though quite naturally allows SVMs and MLNs to complement each other's strengths and weaknesses: learning in a large

and sparse feature space is much easier with SVMs than with MLNs, whereas modeling relational dependencies is much easier with MLNs than with SVMs.

Our second contribution concerns making inference with this MLN feasible. Recall that inference involves detecting and assigning the type label to all the triggers and arguments. We show that existing Maximum-a-posteriori (MAP) inference methods, even the most advanced approximate ones (e.g., (Selman et al., 1996), (Marinescu and Dechter, 2009), (Sontag and Globerson, 2011)), are infeasible on our proposed MLN because of their high memory cost. Consequently, we identify decompositions of the MLN into disconnected components and solve each independently, thereby drastically reducing the memory requirements.

We evaluate our approach on the BioNLP 2009, 2011 and 2013 Genia shared task datasets. On the BioNLP'13 dataset, our model significantly outperforms state-of-the-art pipeline approaches and achieves the best F1 score to date. On the BioNLP'11 and BioNLP'09 datasets, our scores are slightly better and slightly worse respectively than the best reported results. However, they are significantly better than state-of-the-art MLN-based systems.

7.2 Background

7.2.1 Related Work

As a core task in information extraction, event extraction has received significant attention in the natural language processing (NLP) community. The development and evaluation of large-scale learning-based event extraction systems was propelled in part by the availability of annotated corpora produced as part of the Message Understanding Conferences (MUCs), the Automatic Content Extraction (ACE) evaluations, and the BioNLP shared tasks on event extraction. Previous work on event extraction can be broadly divided into two categories, one focusing on the development of features (henceforth *feature-based* approaches) and the other focusing on the development of models (henceforth *model-based* approaches).

Feature-based approaches. Early work on feature-based approaches has primarily focused on designing local sentence-level features such as token and syntactic features (Grishman et al., 2005; Ahn, 2006). Later, it was realized that local features were insufficient to reliably and accurately perform event extraction in complex domains and therefore several researchers proposed using *high-level features*. For instance, (Ji and Grishman, 2008) used global information from related documents; (Gupta and Ji, 2009) extracted implicit time information; (Patwardhan and Riloff, 2009) used broader sentential context; Liao and Grishman (Liao and Grishman, 2010, 2011) leveraged document-level cross-event information and topic-based features; and (Huang and Riloff, 2012b) explored discourse properties.

Model-based approaches. The model-based approaches developed to date have focused on modeling global properties and seldom use rich, high-dimensional features. To capture global event structure properties, (McClosky et al., 2011a) proposed a dependency parsing model. To extract event arguments, (Li et al., 2013) proposed an Integer Linear Programming (ILP) model to encode the relationship between event mentions. To overcome the error propagation problem associated with the pipeline architecture, several joint models have been proposed, including those that are based on MLNs (e.g., (Poon and Domingos, 2007), (Riedel et al., 2009), (Poon and Vanderwende, 2010)), structured perceptrons (e.g., (Li et al., 2013)), and dual decomposition with minimal domain adaptation (e.g., Riedel and McCallum (Riedel and McCallum, 2011a,b)).

In light of the high annotation cost required by supervised learning-based event extraction systems, several semi-supervised, unsupervised, and rule-based systems have been proposed. For instance, (Huang and Riloff, 2012a) proposed a bootstrapping method to extract event arguments using only a small amount of annotated data; (Lu and Roth, 2012) developed a novel unsupervised sequence labeling model; (Bui et al., 2013) implemented a rule-based approach to extract biomedical events; and (Ritter et al., 2012) used unsupervised learning to extract events from Twitter data.

Our work extends prior work by developing a rich framework that leverages sophisticated feature-based approaches as well as joint inference using MLNs. This combination gives us the best of both worlds because on one hand, it is challenging to model sophisticated linguistic features using MLNs while on the other hand, feature-based approaches employing sophisticated high-dimensional features suffer from error propagation as the model is generally not rich enough for joint inference.

7.2.2 The Genia Event Extraction Task

The BioNLP Shared Task (BioNLP-ST) series ((Kim et al., 2009), (Kim et al., 2011) and (Nédellec et al., 2013)) is designed to tackle the problem of extracting structured information from the biomedical literature. The Genia Event Extraction task is arguably the most important of all the tasks proposed in BioNLP-ST and is also the only task organized in all three events in the series.

The 2009 edition of the Genia task (Kim et al., 2009) was conducted on the Genia event corpus (Kim et al., 2008), which only contains abstracts of the articles that represent domain knowledge around NFκB proteins. The 2011 edition (Kim et al., 2011) augmented the dataset to include full text articles, resulting in two collections, the abstract collection and the full text collection. The 2013 edition (Kim et al., 2013) further augmented the dataset with recent full text articles but removed the abstract collection entirely.

The targeted event types have also changed slightly over the years. Both the 2009 and 2011 editions are concerned with nine fine-grained event sub-types that can be categorized into three main types, namely simple, binding and regulation events. These three main event types can be distinguished by the kinds of arguments they take. A simple event can take exactly one protein as its *Theme* argument. A binding event can take one or more proteins as its *Theme* arguments, and is therefore slightly more difficult to extract than a simple event. A regulation event takes exactly one protein or event as its *Theme* argument and optionally

one protein or event as its *Cause* argument. If a regulation event takes another event as its *Theme* or *Cause* argument, it will lead to a nested event. Regulation events are considered the most difficult-to-extract among the three event types owing in part to the presence of an optional *Cause* argument and their recursive structure. The 2013 edition introduced a new event type, protein-mod, and its three sub-types. Theoretically, a protein-mod event takes exactly one protein as its *Theme* argument and optionally one protein or event as its *Cause* argument. In practice, however, it rarely occurs: there are only six protein-mod events having *Cause* arguments in the training data for the 2013 edition. Consequently, our model makes the simplifying assumption that a protein-mod event can only take one *Theme* argument, meaning that we are effectively processing protein-mod events in the same way as simple events.

7.3 Pipeline Model

We implement a pipeline event extraction system using SVMs. This pipeline model serves two important functions: (1) providing a baseline for evaluation and (2) producing prior knowledge for the joint model.

Our pipeline model consists of two steps: trigger labeling and argument labeling. In the trigger labeling step, we determine whether a candidate trigger is a true trigger and label each true trigger with its trigger type. Then, in the argument labeling step, we identify the arguments for each true trigger discovered in the trigger labeling step and assign a role to each argument.

We recast each of the two steps as a classification task and employ $\mathbf{SVM}^{multiclass}$ (Tsochantaridis et al., 2004) to train the two classifiers. We describe each step in detail below.

Table 7.1. Features for trigger labeling and argument labeling.

(a) Features for trigger labeling

Token features	The basic token features (see Table 1(c)) computed from (1) the candidate trigger word and (2) the surrounding tokens in a window of two; character bigrams and trigrams of the candidate trigger word; word n-grams (n=1,2,3) of the candidate trigger word and its context words in a window of three; whether the candidate trigger word contains a digit; whether the candidate trigger word contains an upper case letter; whether the candidate trigger word contains a symbol.
Dependency features	The basic dependency path features (see Table 1(c)) computed using the shortest paths from the candidate trigger to (1) the nearest protein word, (2) the nearest protein word to its left, and (3) the nearest protein word to its right.
Other features	The distances from the candidate trigger word to (1) the nearest protein word, (2) the nearest protein word to its left, and (3) the nearest protein word to its right; the number of protein words in the sentence.

(b) Features for argument labeling

Token features	Word n-grams (n=1,2,3) of (1) the candidate trigger word and its context in a window of three and (2) the candidate argument word and its context in a window of three; the basic token features (see Table 1(c)) computed from (1) the candidate trigger word and (2) the candidate argument word; the trigger type of the candidate trigger word.
Dependency features	The basic dependency features (see Table 1(c)) computed using the shortest path from the candidate trigger word to the candidate argument word.
Other features	The distance between the candidate trigger word and the candidate argument word; the number of proteins between the candidate trigger word and the candidate argument word; the concatenation of the candidate trigger word and the candidate argument word; the concatenation of the candidate trigger type and the candidate argument word.

(c) Basic token and dependency features

Basic token features	Six features are computed given a token t, including: (a) the lexical string of t, (b) the lemma of t, (c) the stem of t obtained using the Porter stemmer (Porter, 1980), (d) the part-of-speech tag of t, (e) whether t appears as a true trigger in the training data, and (f) whether t is a protein name.
Basic dependency features	Six features are computed given a dependency path p, including: (a) the vertex walk in p, (b) the edge walk in p, (c) the n-grams (n=2,3,4) of the (stemmed) words associated with the vertices in p, (d) the n-grams (n=2,3,4) of the part-of-speech tags of the words associated with the vertices in p, (e) the n-grams (n=2,3,4) of the dependency types associated with the edges in p, and (f) the length of p.

7.3.1 Trigger Labeling

A preliminary study of the BioNLP'13 training data suggests that 98.7% of the true triggers' head words[1] are either verbs, nouns or adjectives. Therefore, we consider only those words whose part-of-speech tags belong to the above three categories as candidate triggers. To train the trigger classifier, we create one training instance for each candidate trigger in the training data. If the candidate trigger is not a trigger, the class label of the corresponding instance is *None*; otherwise, the label is the type of the trigger. Thus, the number of class labels equals the number of trigger types plus one. Each training instance is represented by the features described in Table 1(a). These features closely mirror those used in state-of-the-art trigger labeling systems such as (Miwa et al., 2010) and (Björne and Salakoski, 2013).

After training, we apply the resulting trigger classifier to classify the test instances, which are created in the same way as the training instances. If a test instance is predicted as *None* by the classifier, the corresponding candidate trigger is labeled as a non-trigger; otherwise, the corresponding candidate trigger is posited as a true trigger whose type is the class value assigned by the classifier.

7.3.2 Argument Labeling

The argument classifier is trained as follows. Each training instance corresponds to a candidate trigger and one of its candidate arguments.[2] A candidate argument for a candidate trigger *ct* is either a protein or a candidate trigger that appears in the same sentence as *ct*. If *ct* is not a true trigger, the label of the associated instance is set to *None*. On the other hand, if *ct* is a true trigger, we check whether the candidate argument in the associated instance

[1]Head words are found using Collins' (Collins, 1999) rules.

[2]Following the definition of the GENIA event extraction task, the protein names are provided as part of the input.

is indeed one of *ct*'s arguments. If so, the class label of the instance is the argument's role; otherwise, the class label is *None*. The features used for representing each training instance, which are modeled after those used in (Miwa et al., 2010) and (Björne and Salakoski, 2013), are shown in Table 1(b).

After training, we can apply the resulting classifier to classify the test instances, which are created in the same way as the training instances. If a test instance is assigned the class *None* by the classifier, the corresponding candidate argument is classified as not an argument of the trigger. Otherwise, the candidate argument is a true argument of the trigger whose role is the class value assigned by the classifier.

7.4 Joint Model

In this section, we describe our Markov logic model that encodes the relational dependencies in the shared task and uses the output of the pipeline model as prior knowledge (soft evidence). We begin by describing the structure of our Markov logic model, and then describe the parameter learning and inference algorithms for it.

7.4.1 MLN Structure

Figure 7.2 shows our proposed MLN for BioNLP event extraction, which we refer to as `BioMLN`. The MLN contains six predicates.

The *query predicates* in Figure 7.2(a) are those whose assignments are not given during inference and thus need to be predicted. Predicate `TriggerType`(*sid*, *tid*, *ttype*!) is true when the token located in sentence *sid* at position *tid* has type *ttype*. Δ_{ttype}, which denotes the set of constants (or objects) that the logical variable *ttype* can be instantiated to, includes all possible trigger types in the dataset plus *None* (which indicates that the token is not a trigger). The "!" symbol models commonsense knowledge that only one of the types in the domain Δ_{ttype} of *ttype* is true for every unique combination of *sid* and *tid*. Similarly,

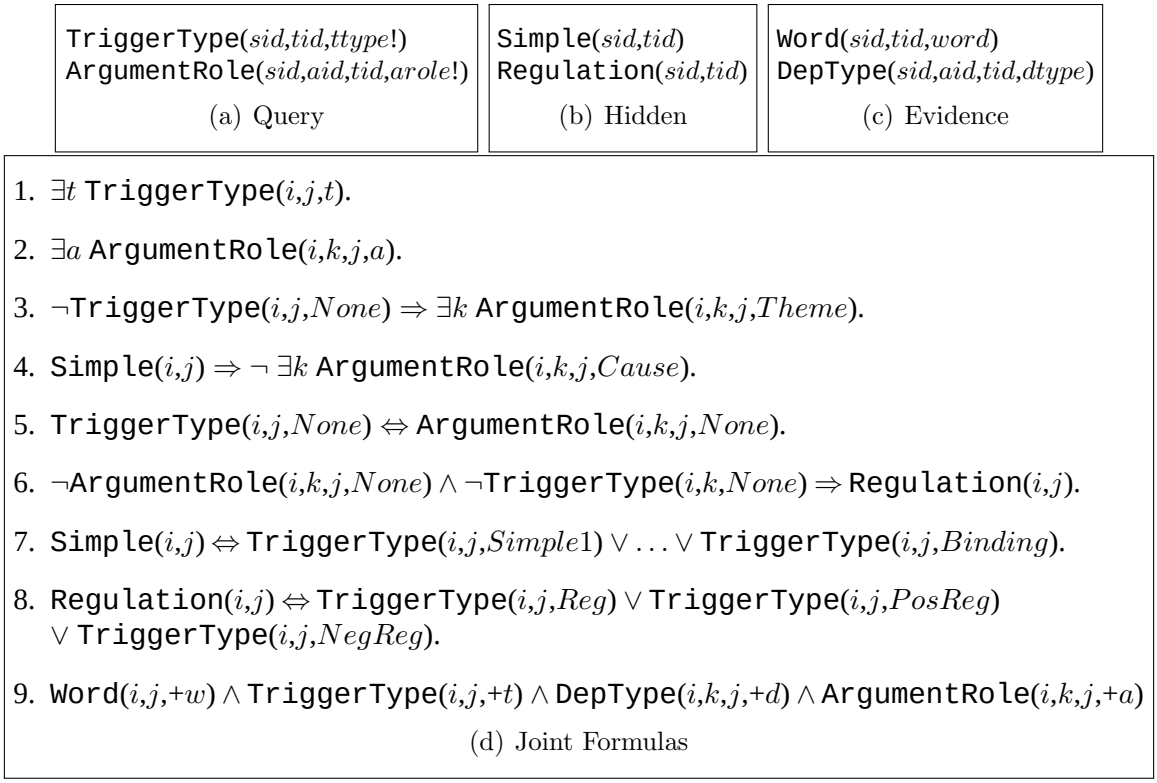


Figure 7.2. The BioMLN structure.

predicate ArgumentRole(sid, aid, tid, $arole$!) asserts that a token in sentence sid at position aid plays *exactly one* argument role, denoted by $arole$, with respect to the token at position tid. Δ_{arole} includes the two argument types, namely, $Theme$ and $Cause$ plus the additional $None$ that indicates that the token is not an argument.

The hidden predicates in Figure 7.2(b) are "clusters" of trigger types. Predicate Simple (sid ,tid) is true when the token in sentence sid at position tid corresponds to one of the *Simple* event trigger types (BioNLP'13 has 9 simple events, BioNLP'09/'11 have 5) or a binding event trigger type. Similarly, Regulation(sid, tid) asserts that the token in sentence sid at position tid corresponds to any of the three regulation event trigger types.

The *evidence* predicates in Figure 7.2(c) are those that are always assumed to be known during inference. We define two evidence predicates based on dependency structures. Word (sid, tid, $word$) is true when the word in sentence sid at position tid is equal to $word$.

`DepType`(*sid*, *aid*, *tid*, *dtype*) asserts that *dtype* is the dependency type in the dependency parse tree that connects the token at position *tid* to the token at position *aid* in sentence *sid*. If the word at *tid* and the word at *aid* are directly connected in the dependency tree, then *dtype* is the label of dependency edge with direction; otherwise *dtype* is $None$.

The MLN formulas, expressing commonsense, prior knowledge in the domain (Poon and Vanderwende, 2010; Riedel and McCallum, 2011a), are shown in Figure 7.2(d). All formulas, except Formula (9), are hard formulas, meaning that they have infinite weights. Note that during weight learning, we only learn the weights of soft formulas.

Formulas (1) and (2) along with the "!" constraint in the predicate definition ensure that the token types are mutually exclusive and exhaustive. Formula (3) asserts that every trigger should have an argument of type $Theme$, since a $Theme$ argument is mandatory for any event. Formula (4) models the constraint that a *Simple* or *Binding* trigger has no arguments of type $Cause$ since only regulation events have a $Cause$. Formula (5) asserts that non-triggers have no arguments and vice-versa. Formula (6) models the constraint that if a token is both an argument of t and a trigger by itself, then t must belong to one of the three regulation trigger types. This formula captures the recursive relationship between triggers. Formulas (7) and (8) connect the hidden predicates with the query predicates. Formula (9) is a soft formula encoding the relationship between triggers and arguments in a dependency parse tree. It joins a word and the dependency type label that connects the word token to the argument token in the dependency parse tree with the trigger types and argument types of the two tokens. The "+" symbol indicates that each grounding of Formula (9) may have a different weight.

7.4.2 Weight Learning

We can learn `BioMLN` from data either discriminatively or generatively. Since discriminative learning is much faster than generative learning, we use the former. In discriminative training, we maximize the conditional log-likelihood (CLL) of the query and the hidden variables

given an assignment to the evidence variables. In principle, we can use the standard gradient descent algorithm for maximizing the CLL. In each iteration of gradient descent, we update the weights using the following equation (cf. (Singla and Domingos, 2005) and (Domingos and Lowd, 2009)):

$$w_j^{t+1} = w_j^t - \alpha(\mathbb{E}_{\mathbf{w}}(n_j) - n_j) \tag{7.1}$$

where w_j^t represents the weight of the j^{th} formula in the t^{th} iteration, n_j is the number of groundings in which the j^{th} formula is satisfied in the training data, $\mathbb{E}_{\mathbf{w}}(n_j)$ is the expected number of groundings in which the j^{th} formula is satisfied given the current weight vector $\mathbf{w}$, and α is the learning rate.

As such, the update rule given in Equation (7.1) is likely to yield poor accuracy because the number of training examples of some types (e.g., *None*) far outnumber other types. To rectify this ill-conditioning problem (Singla and Domingos, 2005; Lowd and Domingos, 2007a), we divide the gradient with the number of true groundings in the data, namely, we compute the gradient using $\frac{(\mathbb{E}_{\mathbf{w}}(n_j)-n_j)}{n_j}$.

Another key issue with using Equation (7.1) is that computing $\mathbb{E}_{\mathbf{w}}(n_j)$ requires performing inference over the MLN. This step is intractable, #P-complete in the worst case. To circumvent this problem and for fast, scalable training, we instead propose to use the voted perceptron algorithm (Collins, 2002; Singla and Domingos, 2005). This algorithm approximates $\mathbb{E}_{\mathbf{w}}(n_j)$ by counting the number of satisfied groundings of each formula in the MAP assignment. Computing the MAP assignment is much easier (although still NP-hard in the worst case) than computing $\mathbb{E}_{\mathbf{w}}(n_j)$, and as a result the voted perceptron algorithm is more scalable than the standard gradient descent algorithm. In addition, it converges much faster.

7.4.3 Testing

In the testing phase, we combine `BioMLN` with the output of the pipeline model (see Section 7.3) to obtain a new MLN, which we refer to as `BioMLN`$^+$. For every candidate trigger,

the SVM trigger classifier outputs a vector of signed confidence values (which is proportional to the distance from the separating hyperplane) of dimension Δ_{ttype} with one entry for each trigger type. Similarly, for every candidate argument, the SVM argument classifier outputs a vector of signed confidence values of dimension Δ_{arole} with one entry for each argument role. In BioMLN$^+$, we model the SVM output as soft evidence, using two soft unit clauses, TriggerType$(i,+j,+t)$ and ArgumentRole$(i,+k,+j,+a)$. We use the confidence values to determine the weights of these clauses. Intuitively, higher (smaller) the confidence, higher (smaller) the weight.

Specifically, the weights of the soft unit clauses are set as follows. If the SVM trigger classifier determines that the trigger in sentence i at position j belongs to type t with confidence $C_{i,j}$, then we attach a weight of $\frac{C_{i,j}}{\alpha n_i}$ to the clause TriggerType(i,j,t). Here, n_i denotes the number of trigger candidates in sentence i. Similarly, if the SVM argument classifier determines that the token at position k in sentence i belongs to the argument role a with respect to the token at position j, with confidence $C'_{i,k,j}$, then we attach a weight of $\frac{C'_{i,k,j}}{\beta \sum_{j=1}^{n_i} m_{ij}}$ to the clause ArgumentRole$(i,\ k,\ j,a)$. Here, m_{ij} denotes the number of argument candidates for the j^{th} trigger candidate in sentence i. α and β act as *scale parameters* for the confidence values ensuring that the weights don't get too large (or too small).

7.4.4 Inference

As we need to perform MAP inference, both at training time and at test time, in this subsection we will describe how to do it efficiently by exploiting unique properties of our proposed BioMLN.

Naively, we can perform MAP inference by grounding BioMLN to a Markov network and then reducing the Markov network by removing from it all (grounded propositional) formulas that are inconsistent with the evidence. On the reduced Markov network, we can then

compute the MAP solution using standard MAP solvers such as MaxWalkSAT (a state-of-the-art local search based MAP solver) (Selman et al., 1996) and Gurobi[3] (a state-of-the-art, parallelized ILP solver).

The problem with the above approach is that grounding the MLN is infeasible in practice; even the reduced Markov network is just too large. For example, assuming a total of $|\Delta_{sid}|$ sentences and a maximum of N tokens in a sentence, Formula (3) alone has $O(|\Delta_{sid}|N^3)$ groundings. Concretely, at training time, assuming 1000 sentences with 10 tokens per sentence, Formula (3) itself yields one million groundings. Clearly, this approach is not scalable. It turns out, however, that the (ground) Markov network can be decomposed into several disconnected components, each of which can be solved independently. This greatly reduces the memory requirement of the inference step. Specifically, for every grounding of *sid*, we get a set of nodes in the Markov network that are disconnected from the rest of the Markov network and therefore independent of the rest of the network. Formally,

Proposition 3. *For any world ω of the* `BioMLN`*,*

$$P_{\mathcal{M}}(\omega) = P_{\mathcal{M}_i}(\omega_i) P_{\mathcal{M} \setminus \mathcal{M}_i}(\omega \setminus \omega_i) \tag{7.2}$$

where ω_i is the world ω projected on the groundings of sentence i and $\mathcal{M}_i$ is `BioMLN` *grounded only using sentence i.*

Using Equation (7.2), it is easy to see that the MLN $\mathcal{M}$ can be decomposed into $|\Delta_{sid}|$ disjoint MLNs, $\{\mathcal{M}_k\}_{k=1}^{|\Delta_{sid}|}$. The MAP assignment to $\mathcal{M}$ can be computed using, $\bigcup_{i=1}^{|\Delta_{sid}|} \left(\arg\max_{\omega_i} P_{\mathcal{M}_i}(\omega_i) \right)$. This result ensures that to approximate the expected counts $\mathbb{E}_{\mathbf{w}}(n_j)$, it is sufficient to keep exactly *one sentence's groundings in memory.* Specifically, $\mathbb{E}_{\mathbf{w}}(n_j)$ can be written as $\sum_{k=1}^{|\Delta_{sid}|} \mathbb{E}_{\mathbf{w}}(n_j^k)$, where $\mathbb{E}_{\mathbf{w}}(n_j^k)$ indicates the expected number of satisfied groundings of the j^{th} formula in the k^{th} sentence. Since the MAP computation is decomposable, we can estimate $\mathbb{E}_{\mathbf{w}}(n_j^k)$ using MAP inference on just the k^{th} sentence.

[3]http://www.gurobi.com/

Table 7.2. Statistics on the BioNLP datasets, which consist of annotated papers/abstracts from PubMed. (x, y, z): x in training, y in development and z in test. #TT indicates the total number of trigger types. The total number of argument types is 2.

Dataset	**#Papers**	**#Abstracts**	**#TT**	**#Events**
BioNLP'13	(10,10,14)	(0,0,0)	13	(2817,3199,3348)
BioNLP'11	(5,5,4)	(800,150,260)	9	(10310,4690,5301)
BioNLP'09	(0,0,0)	(800,150,260)	9	(8597,1809,3182)

7.5 Evaluation

7.5.1 Experimental Setup

We evaluate our system on the BioNLP'13 (Kim et al., 2013), '11 (Kim et al., 2011) and '09 (Kim et al., 2009) Genia datasets for the main event extraction shared task. Note that this task is the most important one for Genia and therefore has the most active participation. Statistics on the datasets are shown in Table 7.2. All our evaluations use the online tool provided by the shared task organizers. We report scores obtained using the *approximate span, recursive* evaluation.

To generate features, we employ the supporting resources provided by the organizers. Specifically, sentence split and tokenization are done using the GENIA tools, while part-of-speech information is provided by the BLLIP parser that uses the self-trained biomedical model (McClosky, 2010). Also, we create dependency features from the parse trees provided by two dependency parsers, the Enju parser (Miyao and Tsujii, 2008) and the aforementioned BLLIP parser that uses the self-trained biomedical model, which results in two sets of dependency features.

For MAP inference, we use Gurobi, a parallelized ILP solver. After inference, a post-processing step is required to generate biomedical events from the extracted triggers and arguments. Specifically, for binding events, we employ a learning-based method similar to (Björne and Salakoski, 2011), while for the other events, we employ a rule-based approach

Table 7.3. Recall (Rec.), Precision (Prec.) and F1 score on the BioNLP'13 test data.

System	Rec.	Prec.	F1
Our System	48.95	59.24	**53.61**
EVEX (Hakala et al., 2013)	45.44	58.03	50.97
TEES-2.1 (Björne and Salakoski, 2013)	46.17	56.32	50.74
BIOSEM (Bui et al., 2013)	42.47	62.83	50.68
NCBI (Liu et al., 2013)	40.53	61.72	48.93
DLUTNLP (Li et al., 2013)	40.81	57.00	47.56

similar to (Björne et al., 2009). Both the SVM baseline system and the combined MLN+SVM system employ the same post-processing strategy.

During weight learning, in order to combat the problem of different initializations yielding radically different parameter estimates, we start at several different initialization points and average the weights obtained after 100 iterations of gradient descent. However, we noticed that if we simply choose random initialization points, the variance of the weights was quite high and some initialization points were much worse than others. To counter this, we use the following method to systematically initialize the weights. Let n_i be the number of satisfied groundings of formula f_i in the training data and m_i be the total number of possible groundings of f_i. We use a threshold γ to determine whether we wish to make the initial weight positive or negative. If $\frac{n_i}{m_i} \leq \gamma$, then we choose the initial weight uniformly at random from the range $[-0.1, 0]$. Otherwise, we chose it from the range $[0, 0.1]$. These steps ensure that the weights generated from different initialization points have smaller variance. Also, in the testing phase, we set the scale parameters for the soft evidence as $\alpha = \beta = \max_{c \in \mathbf{C}} |c|$, where $\mathbf{C}$ is the set of SVM confidence values.

7.5.2 Results on the BioNLP'13 Dataset

Among the three datasets, the BioNLP'13 dataset is most "realistic" one because it is the only one that contains full papers and no abstracts. As a result, it is also the most challenging dataset among the three. Table 7.3 shows the results of our system along with the results of other top systems published in the official evaluation of BioNLP'13. Our system achieves the

	SVM			MLN+SVM		
Type	**Rec.**	**Prec.**	**F1**	**Rec.**	**Prec.**	**F1**
Simple	64.47	87.89	74.38	73.11	78.99	75.94
Protein-Mod	66.49	79.87	72.57	72.25	69.70	70.95
Binding	39.04	50.00	43.84	48.05	43.84	45.85
Regulation	23.51	56.21	33.15	36.47	50.86	42.48
Overall	37.90	67.88	**48.64**	48.95	59.24	**53.61**

(a) Test

	SVM			MLN+SVM		
Type	**Rec.**	**Prec.**	**F1**	**Rec.**	**Prec.**	**F1**
Simple	55.79	81.63	66.28	63.21	75.10	68.64
Protein-Mod	64.47	87.89	74.38	71.14	85.63	77.72
Binding	31.90	48.77	38.57	47.99	50.00	48.97
Regulation	20.13	52.46	29.10	28.57	43.41	34.46
Overall	34.42	66.14	**45.28**	43.50	57.45	**49.51**

(b) Development

Figure 7.3. Comparison of the combined model (MLN+SVM) with the pipeline model on the BioNLP'13 test and development data.

best F1-score (an improvement of 2.64 points over the top-performing system) and has a much higher recall (mainly because our system detects more regulation events which outnumber other event types in the dataset) and a slightly higher precision than the winning system. Of the top five teams, NCBI is the only other joint inference system, which adopts joint pattern matching to predict triggers and arguments at the same time. These results illustrate the challenge in using joint inference effectively. NCBI performed much worse than the SVM-based pipeline systems, EVEX and TEES2.1. It was also worse than BIOSEM, a rule-based system that uses considerable domain expertise. Nevertheless, it was better than DLUTNLP, another SVM-based system.

Figure 7.3 compares our baseline pipeline model with our combined model. We can clearly see that the combined model has a significantly better F1 score than the pipeline model on most event types. The regulation events are considered the most complex events to detect because they have a recursive structure. At the same time, this structure yields a large number of joint dependencies. The advantage of using a rich model such as MLNs

Table 7.4. Results on the BioNLP'11 test data.

System	Rec.	Prec.	F1
Our System	53.42	63.61	**58.07**
Miwa12 (Miwa et al., 2012)	53.35	63.48	57.98
Riedel11 (Riedel et al., 2011)	–	–	56
UTurku (Björne and Salakoski, 2011)	49.56	57.65	53.30
MSR-NLP (Quirk et al., 2011)	48.64	54.71	51.50

can be clearly seen in this case; the combined model yields a 10 point and 6 point increase in F1-score on the test data and development data respectively compared to the pipeline model.

7.5.3 Results on the BioNLP'11 Dataset

Table 7.4 shows the results on the BioNLP'11 dataset. We can see that our system is marginally better than Miwa12, which is a pipeline-based system. It is also more than two points better than Riedel11, a state-of-the-art structured prediction-based joint inference system. Reidel11 incorporates the Stanford predictions (McClosky et al., 2011b) as features in the model. On the two hardest, most complex tasks, detecting regulation events (which have recursive structures and more joint dependencies than other event types) and detecting binding events (which may have multiple arguments), our system performs better than both Miwa12 and Riedel11.

Specifically, our system's F1 score for regulation events is 46.84, while those of Miwa12 and Riedel11 are 45.46 and 44.94 respectively. Our system's F1 score for the binding event is 58.79, while those of Miwa12 and Riedel11 are 56.64 and 48.49 respectively. These results clearly demonstrate the effectiveness of enforcing joint dependencies along with high-dimensional features.

7.5.4 Results on the BioNLP'09 Dataset

Table 7.5 shows the results on the BioNLP'09 dataset. Our system has a marginally lower score (by 0.11 points) than Miwa12, which is the best performing system on this dataset.

Table 7.5. Results on the BioNLP'09 test data. "–" indicates that the corresponding values are not known.

System	**Rec.**	**Prec.**	**F1**
Miwa12 (Miwa et al., 2012)	52.67	65.19	58.27
Our System	53.96	63.08	**58.16**
Riedel11 (Riedel et al., 2011)	–	–	57.4
Miwa10 (Miwa et al., 2010)	50.13	64.16	56.28
Bjorne (Björne et al., 2009)	46.73	58.48	51.95
PoonMLN (Poon&Vanderwende,2010)	43.7	58.6	50.0
RiedelMLN (Riedel et al., 2009)	36.9	55.6	44.4

Specifically, our system achieves a higher recall but a lower precision than Miwa12. However, note that Miwa12 used co-reference features while we are able to achieve similar accuracy without the use of co-reference data. The F1 score of Miwa10, which does not use co-reference features, is nearly 2 points lower than that of our system. Our system also has a higher F1 score than Reidel11, which is the best joint inference-based system for this task.

On the regulation events, our system (47.55) outperforms both Miwa12 (45.99) and Riedel11 (46.9), while on the binding event, our system (59.88) is marginally worse than Miwa12 (59.91) and significantly better than Riedel11 (52.6). As mentioned earlier, these are the hardest events to extract. Also, existing MLN-based joint inference systems such as RiedelMLN and PoonMLN do not achieve state-of-the-art results because they do not leverage complex, high-dimensional features.

7.6 Summary

Markov logic networks (MLNs) are a powerful representation that can compactly encode rich relational structures and ambiguities (uncertainty). As a result, they are an ideal representation for complex NLP tasks that require joint inference, such as event extraction. Unfortunately, the superior representational power greatly complicates inference and learning over MLN models. Even the most advanced methods for inference and learning in MLNs (Gogate and Domingos, 2011b) are unable to handle complex, high-dimensional

features, and therefore existing MLN systems primarily use low-dimensional features. This limitation severely affects the accuracy of MLN-based NLP systems, and as a result, in some cases their performance is inferior to pipeline methods that do not employ joint inference.

In this chapter, we presented a general approach for exploiting the power of high-dimensional linguistic features in MLNs. Our approach involves reliably processing and learning high-dimensional features using SVMs and encoding their output as low-dimensional features in MLNs. We showed that we could achieve scalable learning and inference in our proposed MLN model by exploiting MLN structure. Our results on the BioNLP shared tasks from '13, '11, and '09 clearly show that our proposed combination is extremely effective, achieving the best or second best published score on all three datasets.

CHAPTER 8

FUTURE WORK

Digital data is being generated at an incredibly fast rate. Utilizing this large-scale data (or *big data*) effectively is a key challenge facing researchers and practitioners in Machine Learning (ML) and Artificial Intelligence (AI). Since real-world data is quite often relational and noisy, in principle, the field of statistical relational learning is well-suited to leverage big data for the next generation of AI/ML applications. In this dissertation, we have focused upon several fundamental challenges and ideas that aim to push statistical relational models towards processing large-scale data. Next, we discus some promising ways in which we can extend the ideas presented in this dissertation.

8.1 Advancing Approximate Lifting

Leveraging approximate symmetries (presented in Chapter 5) seems to be the right approach for scalable inference (or learning) in large, real-world domains. There are several possible extensions to this framework that would be extremely useful. For instance, (i) defining advanced heuristics (e.g., graph-theory based heuristics) for detecting approximate symmetries in lifted inference, (ii) approximation guarantees, (iii) connecting approximate lifting with variational inference approximations (Jordan et al., 1999) in graphical models and utilizing theory/algorithms from this area and (iv) improving the power of approximately lifted models by leveraging kernel-based methods (Vapnik, 1995; Shawe-Taylor and Cristianini, 2004) and ensemble methods in machine learning such as boosting (Schapire, 2003) and bootstrapping.

8.2 Large Scale Weight Learning

Weight learning assumes that the formulas of the MLN are encoded from background knowledge and we need to learn the weights of these formulas from data. The standard approach is to learn weights by maximizing the likelihood of the data, and this is typically done using gradient based methods. However, in MLNs, it turns out that computing the gradient each time involves solving an inference problem. Therefore, inference is the most important substep in weight learning. Algorithms such as contrastive divergence (Hinton, 2002) use Gibbs sampling while voted perceptron (Collins, 2002; Singla and Domingos, 2005) uses MAP inference for approximating the gradient during learning. However, the current state-of-the-art weight learning algorithms for MLNs are far from scalable. Most of them use background knowledge and ad-hoc domain-specific heuristics to perform weight learning in real-world domains (Singla and Domingos, 2005; Lowd and Domingos, 2007a). Therefore, it takes a lot of expertise to apply MLNs to completely new applications as compared to standard Machine learning algorithms such as SVMs. The inference techniques presented in this dissertation can be utilized to address this problem and perform weight learning more effectively on large scale data. Particularly, the novel encoding of formulas as Markov networks presented in Chapter 6 can yield potentially exciting future research in weight-learning. For instance, we can utilize this encoding to develop new types of gradient approximations in MLNs where we leverage techniques with complexity and/or approximation guarantees from graphical models such as Bethe approximations (Yedidia et al., 2005), variational inference (Wainwright and Jordan, 2008), generalized BP (Yedidia et al., 2000; Mateescu et al., 2010), SampleSearch (Gogate and Dechter, 2007a) and WISH (Ermon et al., 2013, 2014). Importantly, this can spawn a completely new family of weight learning methods that, unlike existing approaches can scale up to large scale domains and successfully leverage the power of MLNs in practical applications.

8.3 Learning Feasible Structures

Structure learning is a much harder problem than weight learning. Here, the task is to learn both the formulas and its associated weights from data. One possible approach that is to learn structures where exact (lifted) inference is tractable (Domingos and Webb, 2012). However, the main problem with this approach is that the expressiveness of learned structures would be fairly limited. For instance, with our current knowledge of lifted inference, the types of structures where exact inference is tractable is quite small. Also, as mentioned several times in this dissertation, with evidence, the subset of tractable structures further shrinks, which makes it even more difficult to represent sufficiently complex distributions. Further, the notion of *domain liftability* (complexity of exact inference is polynomial in domain-size) is still too expensive when we consider very large datasets (e.g., the domain of users on facebook). In fact, as we have shown in this dissertation, even approximate inference is computationally infeasible for real-world, large-scale domains. An interesting future research direction is to learn structures which though not tractable for exact lifting are feasible for approximate inference even when presented with large domains (e.g., web-scale data). In other words, inference on learned structures should have *domain independent* complexity guarantees. This can yield much more expressive structures since we are not restricting structures to be exactly solvable and at the same time, from a practical standpoint, the learned structures will admit feasible inference. One idea to achieve define this feasibility is to use our results which connect (approximate) inference complexity with treewidth of formulas (cf. Chapter 6). Using this, we can impose a new bias on structure learning such that we only learn bounded treewidth formulas (Note that the Markov network underlying the full MLN can still have large treewidths due to which we can express more complex distributions) which in turn makes approximate inference computationally feasible even over very large datasets.

8.4 Systems Engineering

Advances in parallel/distributed computing and data management have been successfully incorporated in several systems. For instance, GraphLab (Low et al., 2010) uses parallel architectures to scale up inference in graphical models. Tuffy (Niu et al., 2011) uses advanced database management and optimization strategies to store and process MLNs efficiently. However, most of these systems inherently work in the propositional space, i.e., on the ground Markov network. The challenge is to successfully leverage these system-level advances with the symmetry-exploiting as well as the advanced counting strategies described in this dissertation. Doing so is likely to yield the required scalability for big data applications.

8.5 Joint Inference Applications

A natural application for MLNs is in *structured prediction* tasks (e.g., collective classification, semantic parsing, sequence labeling, information extraction, etc.), namely tasks that are much harder than classification based tasks. For many of these tasks, considering relational dependencies in the data, i.e., performing joint inference is vital. However, currently traditional Machine Learning models such as SVMs are routinely applied to these problems since they tend to work "out-of-the-box" even though these models are fundamentally not expressive enough for these tasks. Consequently, we need to apply richer models to solve these structured prediction problems more effectively. However, as seen in this dissertation (Chapter 7), applying rich models to real-world problems is a non-trivial task because of the complexity of these models. Adapting and applying lifted techniques for joint inference in other challenging problems in natural language understanding such as web information extraction as well as in other domains where joint inference is likely to be useful such as Bioinformatics, cyber-security and computer vision is yet another possible research direction.

8.6 Lifted Inference in High-level Languages

Probabilistic programs such as Church (Goodman et al., 2008) and Infer.net (Minka et al., 2014) represent probability distributions using high-level programming languages. Probabilistic programs have several attractive features. First, they are extremely powerful representations (Turing complete languages) and thus can represent complex distributions (e.g., distributions induced by recursive programs). Further, being high-level programs, they are also in human-readable format. This makes it easy for non-experts to compose machine learning algorithms through probabilistic programs since the complexity of underlying inference/learning is abstracted from the user. However, as is the case with statistical relational models, inference and learning in these languages is highly challenging and in many ways even harder. Therefore, incorporating advances from lifted inference into probabilistic programming languages is a topic of great interest. One possible direction is to develop an interface layer to automatically "compile" programs into a Markov logic (or equivalent) specification using which we can leverage a vast amount of research from the SRL community.

CHAPTER 9

CONCLUSION

The past couple of decades has seen a surge in Artificial Intelligence (AI) and Machine learning (ML) applications aided by the development of tools and techniques such as decision trees, support vector machines and ensemble-based learning. Most of these techniques though primarily solve classification problems where the desired output is simple. With an unprecedented growth in data and computational power, application designers are envisioning far more sophisticated tasks that go well beyond classification. Such tasks require much richer models and algorithms.

In this dissertation, we focused on one such rich model called Markov Logic Networks (MLNs) that combine first-order logic with probabilistic graphical models, and developed advanced algorithms for probabilistic inference in them. We utilized the logical and statistical structure in MLNs to scale up inference to much harder and larger problems than what was possible using existing systems. By leveraging advanced techniques from sampling theory, machine learning, CSPs, databases, SAT and discrete optimization, we developed (i) inference techniques for models with logical constraints, (ii) symmetry-exploiting *lifted* inference methods and (iii) fast-counting based scalable inference methods. Further, we were able to show the utility of our approaches on Biomedical event extraction, a challenging real-world application in natural language understanding, on which we were able to obtain state-of-the-art results.

In conclusion, the synergy between large-scale data, rich modeling languages, powerful computing architectures and scalable algorithms is likely to play a key role in the next generation of AI/ML applications. We hope that the concepts, ideas and algorithms from this dissertation and its extensions is a step forward in the design of such systems.

REFERENCES

Ahmadi, B., K. Kersting, M. Mladenov, and S. Natarajan (2013). Exploiting Symmetries for Scaling Loopy Belief Propagation and Relational Training. *Machine Learning 92*(1), 91–132.

Ahn, D. (2006). The Stages of Event Extraction. In *Proceedings of the Workshop on Annotating and Reasoning About Time and Events*, pp. 1–8.

Arnborg, S., D. G. Corneil, and A. Proskurowski (1987, April). Complexity of Finding Embeddings in a k-tree. *SIAM Journal of Algebraic Discrete Methods 8*(2), 277–284.

Beltagy, I. and R. J. Mooney (2014). Efficient Markov Logic Inference for Natural Language Semantics. In *Proceedings of the Fourth International Workshop on Statistical Relational AI at AAAI (StarAI-2014)*, pp. 9–14.

Bidyuk, B. and R. Dechter (2007). Cutset Sampling for Bayesian Networks. *Journal of Artificial Intelligence Research 28*, 1–48.

Björne, J., J. Heimonen, F. Ginter, A. Airola, T. Pahikkala, and T. Salakoski (2009). Extracting Complex Biological Events with Rich Graph-Based Feature Sets. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pp. 10–18.

Björne, J. and T. Salakoski (2011). Generalizing Biomedical Event Extraction. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pp. 183–191.

Björne, J. and T. Salakoski (2013). TEES 2.1: Automated Annotation Scheme Learning in the BioNLP 2013 Shared Task. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pp. 16–25.

Broecheler, M., L. Mihalkova, and L. Getoor (2010). Probabilistic Similarity Logic. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pp. 73–82.

Bui, H., T. Huynh, and R. de Salvo Braz (2012). Exact Lifted Inference with Distinct Soft Evidence on Every Object. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pp. 1875–1881. AAAI Press.

Bui, H., T. Huynh, and S. Riedel (2013). Automorphism Groups of Graphical Models and Lifted Variational Inference. In *Proceedings of the Twenty-Nineth Conference on Uncertainty in Artificial Intelligence*, pp. 132–141. AUAI Press.

Carnap, R. (1950). *Logical Foundations of Probability*. Chicago: University of Chicago Press.

Casella, G. and C. P. Robert (1996, March). Rao-Blackwellisation of Sampling Schemes. *Biometrika 83*(1), 81–94.

Chen, C. and V. Ng (2012). Joint Modeling for Chinese Event Extraction with Rich Linguistic Features. In *Proceedings of the 24th International Conference on Computational Linguistics*, pp. 529–544.

Cheng, J. and M. J. Druzdzel (2000). AIS-BN: An Adaptive Importance Sampling Algorithm for Evidential Reasoning in Large Bayesian Networks. *Journal of Artificial Intelligence Research 13*, 155–188.

Chib, S. (1995). Marginal Likelihood from the Gibbs Output. *Journal of the American Statistical Association 90*, 1313–1321.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. Ph. D. thesis, University of Pennsylvania, Philadelphia, PA.

Collins, M. (2002). Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, Philadelphia, PA, pp. 1–8. ACL.

Darwiche, A. (2001, February). Recursive Conditioning. *Artificial Intelligence 126*, 5–41.

Darwiche, A. (2003). A Differential Approach to Inference in Bayesian Networks. *Journal of the ACM 50*, 280–305.

Darwiche, A. (2009). *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Davis, M. and H. Putnam (1960). A Computing Procedure for Quantification Theory. *Journal of the Association of Computing Machinery 7*(3), 201–215.

De Raedt, L., A. Kimmig, and H. Toivonen (2007). ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pp. 2462–2467.

de Salvo Braz, R. (2007). *Lifted First-Order Probabilistic Inference*. Ph. D. thesis, University of Illinois, Urbana-Champaign, IL.

Dechter, R. (1999). Bucket elimination: A Unifying Framework for Reasoning. *Artificial Intelligence 113*, 41–85.

Dechter, R. (2003). *Constraint Processing*. Morgan Kaufmann.

Dechter, R., K. Kask, E. Bin, and R. Emek (2002). Generating Random Solutions for Constraint Satisfaction Problems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pp. 15–21.

Dechter, R. and R. Mateescu (2007). AND/OR Search Spaces for Graphical Models. *Artificial Intelligence 171*(2-3), 73–106.

Diaconis, P. and D. Freedman (1980). Finite Exchangeable Sequences. *The Annals of Probability 8*(4), 745–764.

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

Domingos, P. and W. Webb (2012). A Tractable First-Order Probabilistic Logic. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pp. 1902–2909.

Eén, N. and N. Sörensson (2003). An Extensible SAT-Solver. In *SAT Competition 2003*, Volume 2919 of *Lecture Notes in Computer Science*, pp. 502–518. Springer.

Ermon, S., C. P. Gomes, A. Sabharwal, and B. Selman (2013). Taming the Curse of Dimensionality: Discrete Integration by Hashing and Optimization. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 334–342.

Ermon, S., C. P. Gomes, A. Sabharwal, and B. Selman (2014). Low-density Parity Constraints for Hashing-Based Discrete Integration. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 271–279.

Fishelson, M. and D. Geiger (2004). Optimizing Exact Genetic Linkage Computations. *Journal of Computational Biology 11*(2/3), 263–275.

Gaifman, H. (1964). Concerning Measures on Boolean Algebras. *Pacific Journal of Mathematics 14(1)*, 61–73.

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

Geman, S. and D. Geman (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 6*, 721–741.

Genesereth, M. R. and E. Kao (2013). *Introduction to Logic, Second Edition*. Morgan & Claypool Publishers.

Getoor, L. and B. Taskar (Eds.) (2007). *Introduction to Statistical Relational Learning*. MIT Press.

Geweke, J. (1989). Bayesian Inference in Econometric Models using Monte Carlo Integration. *Econometrica 57*(6), 1317–39.

Gilks, W. R., S. Richardson, and D. J. Spiegelhalter (1996). *Markov Chain Monte Carlo in Practice*. London, UK: Chapman and Hall.

Gogate, V. (2009). *Sampling Algorithms for Probabilistic Graphical Models with Determinism*. Ph. D. thesis, University of California, Irvine.

Gogate, V., B. Bidyuk, and R. Dechter (2007). Studies in Lower Bounding Probabilities of Evidence using the Markov Inequality. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pp. 141–148.

Gogate, V. and R. Dechter (2007a). Approximate Counting by Sampling the Backtrack-free Search Space. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence*, pp. 198–203. AAAI Press.

Gogate, V. and R. Dechter (2007b). SampleSearch: A Scheme that Searches for Consistent Samples. In *Proceedings of the Eleventh Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 147–154.

Gogate, V. and R. Dechter (2011). SampleSearch: Importance sampling in Presence of Determinism. *Artificial Intelligence 175*(2), 694–729.

Gogate, V. and P. Domingos (2011a). Approximation by Quantization. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pp. 247–255. AUAI Press.

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

Gogate, V., A. Jha, and D. Venugopal (2012). Advances in Lifted Importance Sampling. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI Press.

Gomes, C. P., J. Hoffmann, A. Sabharwal, and B. Selman (2007). From Sampling to Model Counting. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pp. 2293–2299.

Gonzalez, J., Y. Low, A. Gretton, and C. Guestrin (2011). Parallel Gibbs Sampling: From Colored Fields to Thin Junction Trees. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 324–332.

Goodman, N. D., V. K. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum (2008). Church: A Language for Generative Models. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pp. 220–229.

Gries, O. (2011). Gibbs Sampling with Deterministic Dependencies. In *Proceedings of the 5th international conference on Multi-Disciplinary Trends in Artificial Intelligence*, pp. 418–427.

Grishman, R., D. Westbrook, and A. Meyers (2005). NYU's English ACE 2005 system description. In *Proceedings of the ACE 2005 Evaluation Workshop*. Washington.

Gupta, P. and H. Ji (2009). Predicting unknown Time Arguments based on Cross-Event Propagation. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pp. 369–372.

Hajishirzi, H. and E. Amir (2008). Sampling First Order Logical Particles. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pp. 248–255.

Hakala, K., S. Van Landeghem, T. Salakoski, Y. Van de Peer, and F. Ginter (2013). EVEX in ST'13: Application of a large-scale text mining resource to event extraction and network construction. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pp. 26–34.

Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl. 11*(1), 10–18.

Hamze, F. and N. de Freitas (2004). From Fields to Trees. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*, pp. 243–250.

Hastings, W. K. (1970, April). Monte Carlo Sampling Methods Using Markov Chains and their Applications. *Biometrika 57*(1), 97–109.

Hinton, G. E. (2002). Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation 14*(8), 1771–1800.

Huang, R. and E. Riloff (2012a). Bootstrapped Training of Event Extraction Classifiers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 286–295.

Huang, R. and E. Riloff (2012b). Modeling Textual Cohesion for Event Extraction. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 1664–1670.

Huynh, T. N. and R. J. Mooney (2009). Max-Margin Weight Learning for Markov Logic Networks. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD*, pp. 564–579.

Hwang, C. and M. A. S. (1979). *Multiple objective decision making, methods and applications: a state-of-the-art survey*. Springer-Verlag.

Jaeger, M. (1997). Relational Bayesian Networks. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pp. 266–273.

Jaeger, M. (2015, 3). Lower Complexity Bounds for Lifted Inference. *Theory and Practice of Logic Programming 15*, 246–263.

Jensen, C. S., U. Kjaerulff, and A. Kong (1993). Blocking Gibbs Sampling in Very Large Probabilistic Expert Systems. *International Journal of Human Computer Studies. Special Issue on Real-World Applications of Uncertain Reasoning 42*, 647–666.

Jerrum, M., L. Valiant, and V. Vazirani (1986). Random Generation of Combinatorial Structures from a Uniform. *Theoretical Computer Science 43*, 169–188.

Jha, A., V. Gogate, A. Meliou, and D. Suciu (2010). Lifted Inference from the Other Side: The tractable Features. In *Proceedings of the Twenty-Fourth Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 973–981.

Ji, H. and R. Grishman (2008). Refining Event Extraction through Cross-Document Inference. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 254–262.

Joachims, T. (1999). Making Large-Scale SVM Learning Practical. In B. Schlkopf, C. Burges, and A. Smola (Eds.), *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, USA.

Jordan, M. I., Z. Ghahramani, T. Jaakkola, and L. K. Saul (1999). An Introduction to Variational Methods for Graphical Models. *Machine Learning 37*(2), 183–233.

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

Kautz, H., B. Selman, and M. Shah (1997). ReferralWeb: Combining Social Networks and Collaborative Filtering. *Communications of the ACM 40*(3), 63–66.

Kersting, K. (2012). Lifted Probabilistic Inference. In *Proceedings of 20th European Conference on Artificial Intelligence (ECAI)*, pp. 33–38.

Kersting, K., B. Ahmadi, and S. Natarajan (2009). Counting Belief Propagation. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 277–284. AUAI Press.

Kim, J.-D., T. Ohta, S. Pyysalo, Y. Kano, and J. Tsujii (2009). Overview of BioNLP'09 Shared Task on Event Extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pp. 1–9.

Kim, J.-D., T. Ohta, and J. Tsujii (2008). Corpus Annotation for Mining Biomedical Events from Literature. *BMC bioinformatics 9*(1), 10.

Kim, J.-D., S. Pyysalo, T. Ohta, R. Bossy, N. Nguyen, and J. Tsujii (2011). Overview of BioNLP Shared Task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pp. 1–6.

Kim, J.-D., Y. Wang, T. Takagi, and A. Yonezawa (2011). Overview of Genia Event Task in BioNLP Shared Task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pp. 7–15.

Kim, J.-D., Y. Wang, and Y. Yasunori (2013). The Genia Event Extraction Shared Task, 2013 Edition - Overview. In *Proceedings of the BioNLP Shared Task 2013 Workshop*.

Kimmig, A., L. Mihalkova, and L. Getoor (2014). Lifted Graphical Models: A Survey. *Machine Learning 99*(1), 1–45.

Kindermann, R. and J. L. Snell (1980). *Markov Random Fields and Their Applications*. AMS.

Kok, S., M. Sumner, M. Richardson, P. Singla, H. Poon, and P. Domingos (2006). The Alchemy System for Statistical Relational AI. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA. http://alchemy.cs.washington.edu.

Kok, S., M. Sumner, M. Richardson, P. Singla, H. Poon, D. Lowd, J. Wang, and P. Domingos (2008). The Alchemy System for Statistical Relational AI. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA. http://alchemy.cs.washington.edu.

Kokolakis, G. and P. Nanopoulos (2001). Bayesian Multivariate Micro-Aggregation under the Hellinger Distance Criterion. *Research in official statistics 4*, 117–125.

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

Lauritzen, S. L. and D. J. Spiegelhalter (1988). 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.

Li, L., Y. Wang, and D. Huang (2013). Improving Feature-Based Biomedical Event Extraction System by Integrating Argument Information. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pp. 111–115.

Li, P., G. Zhou, Q. Zhu, and L. Hou (2012). Employing Compositional Semantics and Discourse Consistency in Chinese Event Extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1006–1016.

Li, P., Q. Zhu, and G. Zhou (2013). Argument Inference from Relevant Event Mentions in Chinese Argument Extraction. In *Proceedings of the Fifty First Annual Meeting of the Association for Computational Linguistics*, pp. 1477–1487.

Li, Q., H. Ji, and L. Huang (2013). Joint Event Extraction via Structured Prediction with Global Features. In *Proceedings of the Fifty First Annual Meeting of the Association for Computational Linguistics*, pp. 73–82.

Liang, P., M. I. Jordan, and D. Klein (2010). Type-Based MCMC. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, pp. 573–581.

Liao, S. and R. Grishman (2010). Using Document Level Cross-Event Inference to Improve Event Extraction. In *Proceedings of the Forty Eigth Annual Meeting of the Association for Computational Linguistics*, pp. 789–797.

Liao, S. and R. Grishman (2011). Acquiring Topic Features to Improve Event Extraction: in Pre-selected and Balanced Collections. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pp. 9–16.

Liu, H., K. Verspoor, D. C. Comeau, A. MacKinlay, and W. J. Wilbur (2013). Generalizing an Approximate Subgraph Matching-based System to Extract Events in Molecular Biology and Cancer Genetics. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pp. 76–84.

Liu, J. S. (2001). *Monte Carlo Strategies in Scientific Computing*. Springer Publishing Company, Incorporated.

Liu, J. S., W. H. Wong, and A. Kong (1994). Covariance Structure of the Gibbs Sampler with Applications to the Comparison of Estimators and Augmentation Schemes. *Biometrika 81*, 27–40.

Low, Y., J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein (2010). GraphLab: A New Framework For Parallel Machine Learning. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pp. 340–349. AUAI Press.

Lowd, D. and P. Domingos (2007a). Efficient Weight Learning for Markov Logic Networks. In *Principles of Knowledge Discovery in Databases*, Warsaw, Poland, pp. 200–211. Springer.

Lowd, D. and P. Domingos (2007b). Recursive Random Fields. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, Hyderabad, India, pp. 950–955. AAAI Press.

Lu, W. and D. Roth (2012). Automatic Event Extraction with Structured Preference Modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pp. 835–844.

Marinescu, R. and R. Dechter (2009). AND/OR Branch-and-Bound Search for Combinatorial Optimization in Graphical Models. *Artificial Intelligence 173*(16-17), 1457–1491.

Marler, R. T. and J. S. Arora (2004, April). Survey of Multi-Objective Optimization Methods for Engineering. *Structural and Multidisciplinary Optimization 26*(6), 369–395.

Mateescu, R., R. Dechter, and R. Marinescu (2008). AND/OR Multi-Valued Decision Diagrams (AOMDDs) for Graphical Models. *Journal of Artificial Intelligence Research 33*, 465–519.

Mateescu, R., K. Kask, V. Gogate, and R. Dechter (2010). Iterative Join Graph Propagation algorithms. *Journal of Artificial Intelligence Research 37*, 279–328.

McCallum, A. and B. Wellner (2004). Conditional Models of Identity Uncertainty with Application to Noun Coreference. In *Proceedings of the Eighteenth Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 905–912.

McClosky, D. (2010). *Any Domain Parsing: Automatic Domain Adaptation for Natural Language Parsing*. Ph. D. thesis, Ph.D. thesis, Brown University, Providence, RI.

McClosky, D., M. Surdeanu, and C. Manning (2011a). Event Extraction as Dependency Parsing. In *Proceedings of the Association for Computational Linguistics: Human Language Technologies*, pp. 1626–1635.

McClosky, D., M. Surdeanu, and C. Manning (2011b). Event Extraction as Dependency Parsing for BioNLP 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pp. 41–45.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953). Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics 21*, 1087–1092.

Mihalkova, L. and R. Mooney (2007). Bottom-Up Learning of Markov Logic Network Structure. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, Corvallis, OR, pp. 625–632. IMLS.

Milch, B., B. Marthi, S. J. Russell, D. Sontag, D. L. Ong, and A. Kolobov (2005). BLOG: Probabilistic Models with Unknown Objects. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1352–1359.

Milch, B. and S. J. Russell (2006). General-Purpose MCMC Inference over Relational Structures. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pp. 349–358.

Milch, B., L. S. Zettlemoyer, K. Kersting, M. Haimes, and L. P. Kaelbling (2008). Lifted Probabilistic Inference with Counting Formulas. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pp. 1062–1068.

Minka, T., J. Winn, J. Guiver, S. Webster, Y. Zaykov, B. Yangel, A. Spengler, and J. Bronskill (2014). Infer.NET 2.6. Microsoft Research Cambridge. http://research.microsoft.com/infernet.

Miwa, M., S. Pyysalo, T. Hara, and J. Tsujii (2010). Evaluating Dependency Representation for Event Extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 779–787.

Miwa, M., R. Sætre, J.-D. Kim, and J. Tsujii (2010). Event Extraction with Complex Event Classification using Rich Features. *Journal of Bioinformatics and Computational Biology 8*(01), 131–146.

Miwa, M., P. Thompson, and S. Ananiadou (2012). Boosting Automatic Event Extraction from the Literature using Domain Adaptation and Coreference Resolution. *Bioinformatics 28*(13), 1759–1765.

Miyao, Y. and J. Tsujii (2008). Feature Forest Models for Probabilistic HPSG Parsing. *Computational Linguistics 34*(1), 35–80.

Murphy, K. P., Y. Weiss, and M. I. Jordan (1999). Loopy Belief Propagation for Approximate Inference: An Empirical Study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 467–475.

Natarajan, S., T. Khot, K. Kersting, B. Gutmann, and J. Shavlik (2012). Gradient-based Boosting for Statistical Relational Learning: The Relational Dependency Network Case. *Machine Learning 86*(1), 25–56.

Neal, R. (2000). Slice Sampling. *Annals of Statistics 31*, 705–767.

Neal, R. (2008). The Harmonic Mean of the Likelihood: Worst monte carlo method ever.

Neal, R. M. (1993). Probabilistic Inference Using Markov chain Monte Carlo Methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, Toronto, Canada.

Nédellec, C., R. Bossy, J.-D. Kim, J.-J. Kim, T. Ohta, S. Pyysalo, and P. Zweigenbaum (2013). Overview of BioNLP Shared Task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pp. 1–7.

Newton, M. and A. Raftery (1994). Approximate Bayesian Inference with the Weighted Likelihood Bootstrap. *Journal of the Royal Statistical Society 56*, 3–48.

Niepert, M. (2012). Markov Chains on Orbits of Permutation Groups. In *Proceedings of the Twenty-Eigth Conference on Uncertainty in Artificial Intelligence*, pp. 624–633. AUAI Press.

Niepert, M. and G. Van den Broeck (2014). Tractability through Exchangeability: A New Perspective on Efficient Probabilistic Inference. In *Proceedings of the Twenty-Eigth AAAI Conference on Artificial Intelligence*, pp. 2467–2475.

Niu, F., C. Ré, A. Doan, and J. W. Shavlik (2011). Tuffy: Scaling up Statistical Inference in Markov Logic Networks using an RDBMS. *PVLDB 4*(6), 373–384.

Ortiz, L. E. and L. P. Kaelbling (2000). Adaptive Importance Sampling for Estimation in Structured Domains. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pp. 446–454.

Papadimitriou, C. H. and M. Yannakakis (1999). On the Complexity of Database Queries. *Journal of Computer and System Sciences 58*(3), 407 – 427.

Paskin, M. A. (2003). Sample Propagation. In *Advances in Neural Information Processing Systems*, pp. 425–432.

Patwardhan, S. and E. Riloff (2009). A Unified Model of Phrasal and Sentential Evidence for Information Extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 151–160.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufmann.

Pfeffer, A. (2001). IBAL: A Probabilistic Rational Programming Language. In B. Nebel (Ed.), *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pp. 733–740. Morgan Kaufmann.

Poole, D. (2003). First-Order Probabilistic Inference. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pp. 985–991.

Poon, H. and P. Domingos (2006). Sound and Efficient Inference with Probabilistic and Deterministic Dependencies. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pp. 458–463. AAAI Press.

Poon, H. and P. Domingos (2007). Joint Inference in Information Extraction. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence*, Vancouver, Canada, pp. 913–918. AAAI Press.

Poon, H., P. Domingos, and M. Sumner (2008). A General Method for Reducing the Complexity of Relational Inference and its Application to MCMC. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pp. 1075–1080.

Poon, H. and L. Vanderwende (2010). Joint Inference for Knowledge Extraction from Biomedical Literature. In *HLT-NAACL*, pp. 813–821.

Porter, M. F. (1980). An Algorithm for Suffix Stripping. *Program 14*(3), 130–137.

Quirk, C., P. Choudhury, M. Gamon, and L. Vanderwende (2011). MSR-NLP Entry in BioNLP Shared Task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pp. 155–163.

Raghavan, S. and R. J. Mooney (2011, September). Abductive Plan Recognition by Extending Bayesian Logic Programs. In *Proceedings of the European Conference on Machine Learning/Principles and Practice of Knowledge Discovery in Databases*, Volume 2, pp. 629–644.

Richardson, M. and P. Domingos (2006). Markov Logic Networks. *Machine Learning 62*, 107–136.

Riedel, S., H.-W. Chun, T. Takagi, and J. Tsujii (2009). A Markov Logic Approach to Bio-Molecular Event Extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pp. 41–49.

Riedel, S. and A. McCallum (2011a). Fast and Robust Joint Models for Biomedical Event Extraction. In *EMNLP*, pp. 1–12.

Riedel, S. and A. McCallum (2011b). Robust Biomedical Event Extraction with Dual Decomposition and Minimal Domain Adaptation. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pp. 46–50.

Riedel, S., D. McClosky, M. Surdeanu, A. McCallum, and C. D. Manning (2011). Model Combination for Event Extraction in BioNLP 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pp. 51–55.

Ritter, A., Mausam, O. Etzioni, and S. Clark (2012). Open Domain Event Extraction from Twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1104–1112.

Roberts, G. and J. Rosenthal (2007). Coupling and Ergodicity of Adaptive MCMC. *Journal of Applied Probability 44(2)*, 458–477.

Roberts, G. O. and J. S. Rosenthal (2009). Examples of Adaptive MCMC. *Journal of Computational and Graphical Statistics 18*(2), 349–367.

Roth, D. (1996). On the Hardness of Approximate Reasoning. *Artificial Intelligence 82*, 273–302.

Roth, D. and W. Yih (2005). Integer Linear Programming Inference for Conditional Random Fields. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, pp. 736–743.

Rubinstein, R. Y. (1981). *Simulation and the Monte Carlo Method*. John Wiley & Sons Inc.

Russell, S. J. and P. Norvig (2003). *Artificial Intelligence: A Modern Approach* (2 ed.). Pearson Education.

Sang, T., P. Beame, and H. Kautz (2005). Solving Bayesian networks by weighted model counting. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pp. 475–482.

Sarkhel, S., D. Venugopal, P. Singla, and V. Gogate (2014a). An Integer Polynomial Programming Based Framework for Lifted MAP Inference. In *Proceedings of the Twenty-Eigth Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 3302–3310.

Sarkhel, S., D. Venugopal, P. Singla, and V. Gogate (2014b). Lifted MAP Inference for Markov Logic Networks. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS*, pp. 859–867.

Schapire, R. (2003). The Boosting Approach to Machine Learning: An Overview. In *Nonlinear Estimation and Classification*, Volume 171 of *Lecture Notes in Statistics*, pp. 149–171. Springer New York.

Scharstein, D. and R. Szeliski (2002, April). A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal on Computer Vision 47*(1-3), 7–42.

Selman, B., H. Kautz, and B. Cohen (1996). Local Search Strategies for Satisfiability Testing. In D. S. Johnson and M. A. Trick (Eds.), *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, pp. 521–532. Washington, DC: American Mathematical Society.

Shavlik, J. W. and S. Natarajan (2009). 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*, pp. 1951–1956.

Shawe-Taylor, J. and N. Cristianini (2004). *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press.

Shwe, M., B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, and G. Cooper (1991). Probabilistic Diagnosis using a Reformulation of the INTERNIST-1/QMR Knowledge Base. I. The Probabilistic Model and Inference Algorithms. *Methods of Information in Medicine 30*(4), 241–55.

Singla, P. and P. Domingos (2005). Discriminative Training of Markov Logic Networks. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, Pittsburgh, PA, pp. 868–873. AAAI Press.

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

Singla, P. and P. Domingos (2008). Lifted First-Order Belief Propagation. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pp. 1094–1099. AAAI Press.

Singla, P., A. Nath, and P. Domingos (2014). Approximate Lifting Techniques for Belief Propagation. In *Proceedings of the Twenty-Eigth AAAI Conference on Artificial Intelligence*, pp. 2497–2504.

Sontag, D. and A. Globerson (2011). *Introduction to Dual Decomposition for Inference*, pp. 219–254. MIT Press.

Taghipour, N., D. Fierens, G. Van den Broeck, J. Davis, and H. Blockeel (2013). Completeness Results for Lifted Variable Elimination. In *Proceedings of the Sixteenth Conference on Artificial Intelligence and Statistics*, pp. 572–580.

Tsochantaridis, I., T. Hofmann, T. Joachims, and Y. Altun (2004). Support Vector Machine Learning for Interdependent and Structured Output Spaces. In *Proceedings of the 21st International Conference on Machine Learning*, pp. 104–112.

Van den Broeck, G. (2011). On the Completeness of First-Order Knowledge Compilation for Lifted Probabilistic Inference. In *Proceedings of the Twenty-Fifth Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 1386–1394.

Van den Broeck, G. (2013). *Lifted Inference and Learning in Statistical Relational Models*. Ph. D. thesis, KU Leuven.

Van den Broeck, G. and A. Darwiche (2013). 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)*, pp. 2868–2876.

Van den Broeck, G. and J. Davis (2012). Conditioning in First-Order Knowledge Compilation and Lifted Probabilistic Inference. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pp. 1961–1967. AAAI Press.

Van den Broeck, G., N. Taghipour, W. Meert, J. Davis, and L. De Raedt (2011). Lifted Probabilistic Inference by First-Order Knowledge Compilation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 2178–2185.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. New York, NY: Springer.

Vardi, M. Y. (1982). The Complexity of Relational Query Languages (Extended Abstract). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pp. 137–146. ACM.

Venugopal, D., C. Chen, V. Gogate, and V. Ng (2014). 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)*, pp. 831–843. ACL.

Venugopal, D. and V. Gogate (2012). On Lifting the Gibbs Sampling Algorithm. In *Proceedings of the Twenty-Sixth Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 1664–1672.

Venugopal, D. and V. Gogate (2013a). Dynamic Blocking and Collapsing for Gibbs Sampling. In *Proceedings of the Twenty-Nineth Conference on Uncertainty in Artificial Intelligence*.

Venugopal, D. and V. Gogate (2013b). GiSS: Combining Gibbs Sampling and SampleSearch for Inference in Mixed Probabilistic and Deterministic Graphical Models. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pp. 897–904.

Venugopal, D. and V. Gogate (2014a). Evidence-Based Clustering for Scalable Inference in Markov Logic. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD*, pp. 258–273.

Venugopal, D. and V. Gogate (2014b). Scaling-up Importance Sampling for Markov Logic Networks. In *Proceedings of the Twenty-Eigth Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 2978–2986.

Venugopal, D., S. Sarkhel, and V. Gogate (2015). 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*, pp. 3606–3612.

Wainwright, M. J., T. S. Jaakkola, and A. S. Willsky (2003). Tree-reweighted Belief Propagation Algorithms and Approximate ML Estimation by Pseudo-Moment Matching. In *In Nineth Workshop on Artificial Intelligence and Statistics*.

Wainwright, M. J. and M. I. Jordan (2008). Graphical Models, Exponential Families, and Variational Inference. *Found. Trends Mach. Learn. 1*(1-2), 1–305.

Wei, W., J. Erenrich, and B. Selman (2004). Towards Efficient Sampling: Exploiting Random Walk Strategies. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pp. 670–676.

Wemmenhove, B., J. M. Mooij, W. Wiegerinck, M. A. R. Leisink, H. J. Kappen, and J. P. Neijt (2007). Inference in the Promedas Medical Expert System. In *AIME*, pp. 456–460. Springer.

Wemmenhove, B., J. M. Mooij, W.Wiegerinck, M. A. R. Leisink, H. J. Kappen, and J. P. Neijt (2007). Inference in the Promedas Medical Expert System. In *Eleventh Conference on Artificial Intelligence in Medicine*, pp. 456–460.

Yedidia, J. S., W. T. Freeman, and Y. Weiss (2000). Generalized Belief Propagation. In *Proceedings of the Fourteenth Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 689–695.

Yedidia, J. S., W. T. Freeman, and Y. Weiss (2005). Constructing Free-Energy Approximations and Generalized Belief Propagation Algorithms. *IEEE Transactions on Information Theory 51*(7), 2282–2312.

Yu, H. and R. A. van Engelen (2012). Measuring the Hardness of Stochastic Sampling on Bayesian Networks with Deterministic Causalities: the k-Test. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*.

Zhang, N. and D. Poole (1994). A Simple Approach to Bayesian Network Computations. In *Proceedings of the Tenth Biennial Canadian Artificial Intelligence Conference*, pp. 171–178.

VITA

Deepak Venugopal was born in Bangalore, India. He obtained his bachelor's degree in information science from B.M.S College of Engineering in 2001 and his master's degree in computer science from The University of Texas at Dallas in 2004. After this, he worked as an R&D engineer in the software industry before starting his Ph.D. in computer science at The University of Texas at Dallas. During his Ph.D., he primarily worked in the fields of Artificial Intelligence and Machine Learning, and developed several advanced algorithms for probabilistic reasoning.