GiSS: Combining Gibbs Sampling and SampleSearch for Inference in Mixed Probabilistic and Deterministic Graphical Models

Deepak Venugopal
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75080, USA
dxv021000@utdallas.edu

Vibhav Gogate
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75080, USA
vgogate@hlt.utdallas.edu

Abstract

Mixed probabilistic and deterministic graphical models are ubiquitous in real-world applications. Unfortunately, Gibbs sampling, a popular MCMC technique, does not converge to the correct answers in presence of determinism and therefore cannot be used for inference in such models. In this paper, we propose to remedy this problem by combining Gibbs sampling with SampleSearch, an advanced importance sampling technique which leverages complete SAT/CSP solvers to generate high quality samples from hard deterministic spaces. We call the resulting algorithm, GiSS. Unlike Gibbs sampling which yields unweighted samples, GiSS yields weighted samples. Computing these weights exactly can be computationally expensive and therefore we propose several approximations. We show that our approximate weighting schemes yield consistent estimates and demonstrate experimentally that GiSS is competitive in terms of accuracy with state-of-the-art algorithms such as SampleSearch, MC-SAT and Belief propagation.

Introduction

Deterministic dependencies (or hard constraints) are ubiquitous in real-world probabilistic graphical models (PGMs). For example, PGMs used in linkage analysis (Fishelson and Geiger 2004), medical diagnosis (Shwe et al. 1991) and entity resolution (McCallum and Wellner 2004) often contain both probabilistic and deterministic dependencies. Unfortunately, popular approximate inference methods such as Gibbs sampling (Geman and Geman 1984) and Belief propagation (Murphy, Weiss, and Jordan 1999; Yedidia, Freeman, and Weiss 2005) often perform poorly in the presence of determinism. Improving the performance of approximate inference methods in the presence of determinism is therefore quite important and can have a significant impact. In this paper, we focus on Gibbs sampling, arguably the most popular MCMC inference technique, and seek to improve its accuracy in the presence of determinism.

Gibbs sampling performs poorly in the presence of determinism because determinism fractures the support (assignments that have non-zero probability) of the probability space into disconnected clusters (Gilks, Richardson, and Spiegelhalter 1996). Since Gibbs sampling makes local moves, it will get trapped in one cluster and its Markov chain will converge to the marginal probability distribution over that cluster, yielding incorrect answers.

Many solutions have been proposed in the past to address the aforementioned problem. Notable examples are Blocking (Jensen, Kjaerulff, and Kong 1993), Rao-Blackwellisation (Casella and Robert 1996) and the recently proposed scheme by (Gries 2011). However, none of them are scalable to large PGMs (see the related work section for a discussion). In this paper, we propose a hybrid algorithm, which we call GiSS to solve this problem. GiSS combines Gibbs sampling with SampleSearch (Gogate and Dechter 2007; 2011), a state-of-the-art importance sampling algorithm that leverages CSP/SAT solvers to generate high quality samples from hard deterministic spaces.

The key idea in GiSS is straight-forward: 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 while Gibbs sampling 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 $\#\mathcal{P}$. 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.

We conducted a detailed experimental study comparing the accuracy of GiSS with three state-of-the-art techniques from literature: SampleSearch, MC-SAT and Belief propagation. For evaluation, we used PGMs with determinism from several benchmark domains. Our experiments show that GiSS is competitive with or better than the other algorithms on all the instances, clearly demonstrating its value.

Copyright © 2013, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Background and Notation

We begin with a brief review of PGMs and sampling algorithms (for details, see (Darwiche 2009; Koller and Friedman 2009; Liu 2001)). 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.

Definition 1. A PGM or a Markov network (Koller and Friedman 2009), denoted by $\mathcal{M}$, is a pair $\langle \mathbf{X}, \Phi \rangle$ where $\mathbf{X}$ is a set of discrete random variables and Φ is a set of non-negative real valued functions. Each function $\phi \in \Phi$ is defined over a subset of variables called its scope, denoted by $S(\phi)$. A PGM represents the following joint probability distribution

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

where $\overline{\mathbf{x}}_{S(\phi)}$ is the projection of $\overline{\mathbf{x}}$ on $S(\phi)$ and Z is the normalization constant or the partition function given by

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

We say that a function ϕ is **deterministic** if at least one of its entries is zero (namely, $\exists \overline{\mathbf{x}}\ s.t.\ \phi(\overline{\mathbf{x}}) = 0$). For brevity, henceforth, we will abuse notation and write $\phi(\overline{\mathbf{x}}_{S(\phi)})$ as $\phi(\overline{\mathbf{x}})$. The two key inference tasks in PGMs are computing the partition function and computing the one-variable marginals, namely the marginal probability distribution $P(X)$ of each variable $X \in \mathbf{X}$. Both tasks are $\#\mathcal{P}$-complete (Roth 1996).

Importance Sampling

Importance Sampling (Geweke 1989) is a standard Monte Carlo estimation technique that can be used to approximately compute the partition function as well as all one-variable marginals. The main idea is to use a probability distribution Q, called the proposal distribution, and reformulate the inference problem as computing the expected value of a random variable. Then, we draw independent samples from Q and approximate the expected value by the sample mean. Formally, we can express Z as

$$Z = \sum_{\overline{x}} \frac{\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}) Q(\overline{\mathbf{x}})}{Q(\overline{\mathbf{x}})} = \mathbb{E}_Q \left[\frac{\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}})}{Q(\overline{\mathbf{x}})} \right] \tag{2}$$

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:

$$\widehat{Z} = \frac{1}{T} \sum_{t=1}^{T} \left[\frac{\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}^{(t)})}{Q(\overline{\mathbf{x}}^{(t)})} \right] = \frac{1}{T} \sum_{t=1}^{T} w(\overline{\mathbf{x}}^{(t)}) \tag{3}$$

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.

In this paper, we will make the standard assumption that Q is a Bayesian network; since it is easy to generate independent samples from a Bayesian network using logic sampling (Pearl 1988). 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 for all i.

Importance sampling can also be used to approximate one-variable marginals using the following estimate:

$$\widehat{P}_T(\overline{x}) = \frac{\sum_{t=1}^{T} \delta_{\overline{x}}(\overline{\mathbf{x}}^{(t)}) w(\overline{\mathbf{x}}^{(t)})}{\sum_{t=1}^{T} w(\overline{\mathbf{x}}^{(t)})} \tag{4}$$

where $\delta_{\overline{x}}(\overline{\mathbf{x}}) = 1$ iff $\overline{\mathbf{x}}$ contains the assignment $\overline{x}$ and 0 otherwise. 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*. We will leverage this property extensively throughout the paper.

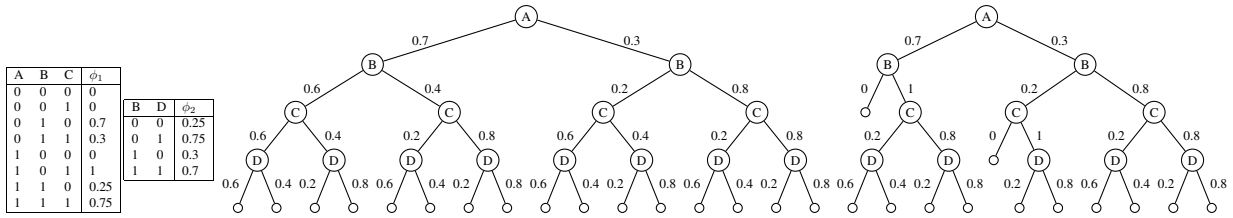
SampleSearch

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 $\widehat{Z}$ will equal zero and $\widehat{P}_T(\mathbf{x})$ will equal $0/0$.

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 Fig. 1. 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 $\#\mathcal{P}$-complete (Roth 1996; Yu and van Engelen 2011) and thus there is no hope of solving it using other methods.

SampleSearch (Gogate and Dechter 2007; 2011) solves the rejection problem by interleaving backtracking search with sampling constructing the backtrack-free distribution

[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) A PGM (b) Proposal Distribution as a probability tree (c) Backtrack-free probability tree of (b)

Figure 1: (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.

on the fly. It is based on the observation that although constructing the backtrack-free distribution is a $\#\mathcal{P}$-complete problem, the problem of generating samples from it is a much easier $\mathcal{NP}$-complete problem. 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 2007) 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 $w(\overline{\mathbf{x}}^{(t)}) = \frac{\prod_{\phi\in\Phi}\phi(\overline{\mathbf{x}}^{(t)})}{Q_{BF}(\overline{\mathbf{x}}^{(t)})}$ and use Equations (3) and (4) to estimate the partition function and one-variable marginals respectively.

In practice, we can speed-up SampleSearch by using advanced backtracking 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.

Gibbs Sampling

Gibbs sampling (Geman and Geman 1984) is one of the most widely used MCMC algorithms to date. The algorithm begins with a random assignment $\overline{\mathbf{x}}^{(0)}$ to all variables. Then, for $t = 1,\ldots,T$, it performs the following steps (each step is called a Gibbs iteration). Let $(X_1,\ldots,X_n)$ be an arbitrary ordering of variables in $\mathcal{M}$. Then, for $i = 1$ to n, it generates a new value $\overline{x}_i^{(t)}$ for X_i by sampling a value from the distribution $P(X_i|\overline{\mathbf{x}}_{-i}^{(t)})$ where $\overline{\mathbf{x}}_{-i}^{(t)} = (\overline{x}_1^t,\ldots,\overline{x}_{i-1}^t,\overline{x}_{i+1}^{(t-1)},\ldots,\overline{x}_n^{(t-1)})$. After T samples are generated, all one-variable marginals can be estimated using the following quantity

$$\widetilde{P}_T(\overline{x}_i) = \frac{1}{T}\sum_{t=1}^{T} P(\overline{x}_i|\overline{\mathbf{x}}_{-i}^{(t)}) \quad (5)$$

In the limit of infinite samples, $\widetilde{P}_T(\overline{x}_i)$ will converge to the $P(\overline{x}_i)$ if the underlying Markov chain is ergodic, i.e., 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 has deterministic dependencies, ergodicity breaks down and the estimate given in Eq. (5) no longer converges to $P(\overline{x}_i)$.

The GiSS Algorithm

In this section, we describe our main contribution, 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 1 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.

Computing the Sample Weights

Unlike Gibbs sampling in which every sample has the same weight, each sample generated by GiSS needs to be

Algorithm 1: 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 $i = 1$ *to* T **do**
  Use SampleSearch to generate a sample $\overline{\mathbf{x}}_d^{(i)}$ 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**
    $\overline{\mathbf{x}}_p^{(j)} =$ Gibbs-iteration($\mathcal{M}|\overline{\mathbf{x}}_d^{(i)}$);
    Compute the sample weight $w(\overline{\mathbf{x}}_d^{(i)}, \overline{\mathbf{x}}_p^{(j)})$;
    for *each value* x *in the domain of* $X \in \mathbf{X}$ **do**
      $\widehat{P}(\overline{x}) = \widehat{P}(\overline{x}) + w(\overline{\mathbf{x}}_d^{(i)}, \overline{\mathbf{x}}_p^{(j)}) \delta_{\overline{x}}(\overline{\mathbf{x}}_d^{(i)}, \overline{\mathbf{x}}_p^{(j)})$;
Normalize the estimates and **return** $\{P(\overline{x})\}$;

weighted appropriately in order to guarantee convergence to the correct answer. It is relatively straight-forward to derive such a weighting scheme. Intuitively, since we perform importance sampling only over $\mathbf{X}_d$, the weight of the full sample $(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p)$ should be equal to the (importance) weight of $\overline{\mathbf{x}}_d$. Namely,

$$w(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p) = w(\overline{\mathbf{x}}_d) = \frac{P(\overline{\mathbf{x}}_d)}{Q_{BF}(\overline{\mathbf{x}}_d)} = \frac{1}{Z} \frac{\sum_{\overline{\mathbf{x}}_p} \prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p)}{Q_{BF}(\overline{\mathbf{x}}_d)} \propto \frac{Z(\mathcal{M}|\overline{\mathbf{x}}_d)}{Q_{BF}(\overline{\mathbf{x}}_d)} \quad (6)$$

where $Z(\mathcal{M}|\overline{\mathbf{x}}_d)$ is the partition function of $\mathcal{M}|\overline{\mathbf{x}}_d$. Using the law of large numbers, it is straight-forward to show that:

Theorem 1. *Assuming that the Gibbs sampling algorithm generates samples from* $\mathcal{M}|\overline{\mathbf{x}}_d^{(i)}$ *and the weight* $w(\overline{\mathbf{x}}_d^{(i)}, \overline{\mathbf{x}}_p^{(j)})$ *of each sample equals* $\frac{Z(\mathcal{M}|\overline{\mathbf{x}}_d^{(i)})}{Q_{BF}(\overline{\mathbf{x}}_d^{(i)})}$*, 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. (6) requires computing the partition function of $\mathcal{M}|\overline{\mathbf{x}}_d$. Since, the latter is intractable in general, we consider three sampling-based 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

$$\widehat{Z}_h(\mathcal{M}|\overline{\mathbf{x}}_d) = \frac{2^{|\mathbf{X}_p|} \times U}{\sum_{j=1}^{U} \frac{1}{\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p^{(j)})}} \quad (7)$$

Substituting the harmonic mean estimate of $Z(\mathcal{M}|\overline{\mathbf{x}}_d)$ given in Eq. (7) in Eq. (6), we get

$$\widehat{w}_h(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p) = \frac{2^{|\mathbf{X}_p|} \times U}{\sum_{j=1}^{U} \frac{1}{\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p^{(j)})}} Q_{BF}(\overline{\mathbf{x}}_d) \quad (8)$$

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, Valiant, and Vazirani 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, Weiss, and Jordan 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:

$$\widehat{Z}_c(\mathcal{M}|\overline{\mathbf{x}}_d) = \frac{\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p^*)}{\widehat{P}(\overline{\mathbf{x}}_p^*)} \quad (9)$$

Substituting the product estimate of $Z(\mathcal{M}|\overline{\mathbf{x}}_d)$ given in Eq. (9) in Eq. (6), we get

$$\widehat{w}_c(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p) = \frac{\prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}_d, \overline{\mathbf{x}}_p^*)}{\widehat{P}(\overline{\mathbf{x}}_p^*) Q_{BF}(\overline{\mathbf{x}}_d)} \quad (10)$$

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 $Z(\mathcal{M}|\overline{\mathbf{x}}_d)$,

$$Z(\mathcal{M}|\overline{\mathbf{x}}_d) = \prod_{i=1}^{k+1} \frac{Z\left((\mathcal{M}|\overline{\mathbf{x}}_d)^{\beta_i}\right)}{Z\left((\mathcal{M}|\overline{\mathbf{x}}_d)^{\beta_{i-1}}\right)} \quad (11)$$

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. (11) using the following quantity (Neal 1993):

$$\frac{1}{U}\sum_{j=1}^{U}\prod_{\phi\in\Phi}(\phi(\overline{\mathbf{x}}^{(j)}))^{\beta_i-\beta_{i-1}} \tag{12}$$

Substituting the ratio estimates obtained using Eq. (12) in Eq. (11), we obtain an estimate for $Z(\mathcal{M}|\overline{\mathbf{x}}_d)$. Substituting this estimate in Eq. (6), we get

$$\widehat{w}_a(\overline{\mathbf{x}}_d,\overline{\mathbf{x}}_p)=\frac{1}{U}\frac{\prod_{i=1}^{k+1}\sum_{j=1}^{U}\prod_{\phi\in\Phi}(\phi(\overline{\mathbf{x}}^{(j)}))^{\beta_i-\beta_{i-1}}}{Q_{BF}(\overline{\mathbf{x}}_d)} \tag{13}$$

We can show that all three weighting schemes are correct in the sense that they yield consistent estimates (the proof is straight-forward and omitted for lack of space).

Theorem 2. *Assuming that the Gibbs sampling algorithm generates samples from $\mathcal{M}|\overline{\mathbf{x}}_d^{(i)}$ and the weight $w(\overline{\mathbf{x}}_d^{(i)},\overline{\mathbf{x}}_p^{(j)})$ of each sample is given by either Eq.* (8)*, Eq.* (10) *or Eq.* (13)*, the estimate output by Algorithm* `GiSS` *converges almost surely to $P(\overline{x})$ as $T\rightarrow\infty$.*

Related Work

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, Kjaerulff, and Kong 1993) and Rao-Blackwellization (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, Erenrich, and Selman 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 is therefore more widely applicable.

Experiments

We compared `GiSS` with four approximate inference algorithms from literature: MC-SAT (Poon and Domingos 2006), Belief Propagation (BP) (Murphy, Weiss, and Jordan 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 (see (Gogate and Dechter 2011), Algorithm 3 for details). 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. Formally, the proposal distribution used by `GiSS` is $Q(\mathbf{X}_d)$ while the one used by SampleSearch is $Q(\mathbf{X}_d)Q(\mathbf{X}_p|\mathbf{X}_d)$. 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. BP and Gibbs sampling serve as baselines.

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 used in our study are available freely from the UAI 2008 repository (http://graphmod.ics.uci.edu/uai08/). 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

Fig. 2 shows 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, Beame, and Kautz 2005). Almost 75% of the functions in these PGMs are deterministic. Fig. 2(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 determinstic. Fig. 2(d)-(f) 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). These models (e.g., Markov logic networks (Domingos and Lowd 2009)) combine first-order logic and probability and because of their compactness have

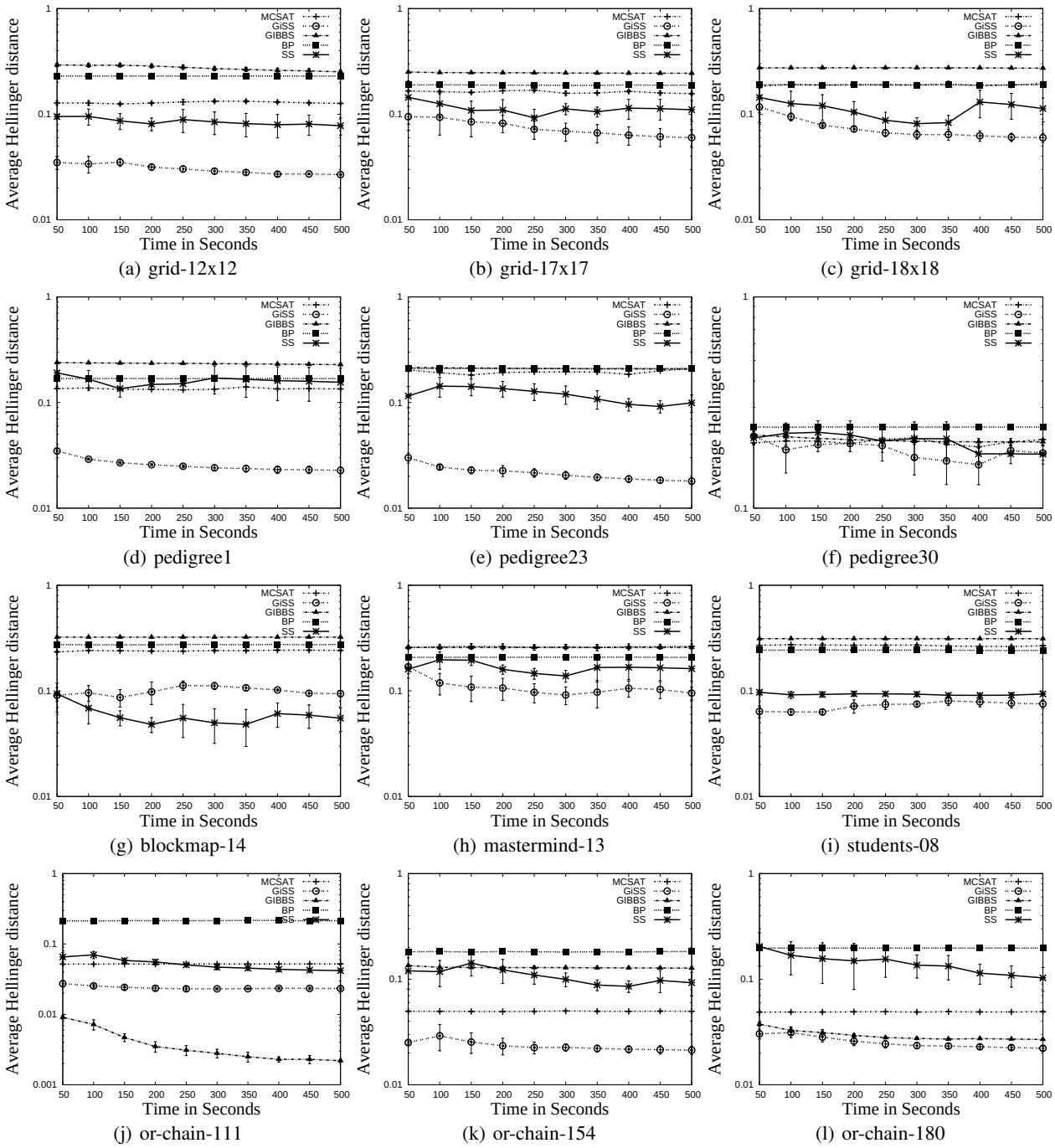
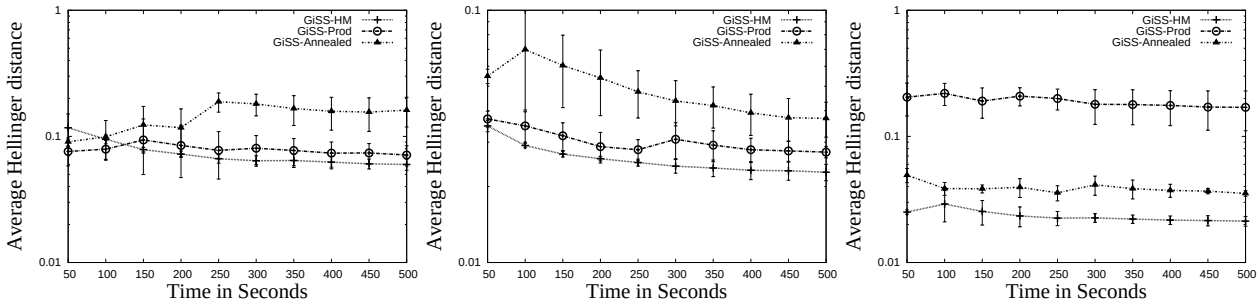


Figure 2: Average Hellinger distance between the exact and the approximate one-variable marginals plotted as a function of time along with error bars (indicating the standard deviation taken over 5 runs) for GiSS, MC-SAT, BP, SampleSearch (SS) and Gibbs sampling. (a)-(c):Grids; (d)-(f):Linkage; (g)-(i):Relational; (j)-(l):Promedas.

quickly emerged as the knowledge representation tool of choice. 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 Fig. 2(g)-(i), 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



(a) grid-18x18 (b) pedigree1 (c) or-chain-154

Figure 3: 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).

et al. 2008). 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 Fig. 2(j)-(l), 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). 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

Fig. 3 shows the impact of varying time on the accuracy of the three weighting schemes. For these experiments, 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. Its weights are however less accurate than the other two. Our results suggest that *larger sample size is more critical to improving the accuracy than high quality weights*.

Discussion and Summary

GiSS is based on the premise that whenever possible, it is better to use Gibbs sampling 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 over a fraction of the variables in the PGM, namely the deterministic variables. Through GiSS, we hope to achieve the best of both worlds, leveraging the power of an importance sampling technique, specifically SampleSearch to sample from hard deterministic spaces while retaining the desirable properties of Gibbs sampling over non-deterministic spaces.

GiSS, along with the associated weighting schemes can also be viewed as a general scheme for combining Gibbs sampling with importance sampling. All we have to do is partition the variables into two subsets, $\mathbf{X}_d$ and $\mathbf{X}_p$, perform importance sampling over $\mathbf{X}_d$ and Gibbs sampling over $\mathbf{X}_p$ given $\mathbf{X}_d$. The question that needs to be addressed however is how to the partition the variables and we plan to address it in future work.

To summarize, in this paper, we introduced GiSS, a hybrid algorithm that combines Gibbs sampling with SampleSearch (importance sampling). The main virtue of this new algorithm 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.

Several avenues remain for future work: combining GiSS with Blocking and Rao-Blackwellised Gibbs sampling; developing new approximate weighting schemes; developing formula-based (Gogate and Domingos 2010) version of GiSS; lifting GiSS; extending our algorithm to sample first-order dynamic systems (Hajishirzi and Amir 2008); etc.

Acknowledgements This research was partly funded by the ARO MURI grant W911NF-08-1-0242. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO or the U.S. Government.

References

Casella, G., and Robert, C. P. 1996. Rao-Blackwellisation of Sampling Schemes. *Biometrika* 83(1):81–94.

Chib, S. 1995. Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association.* 90:1313–1321.

Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

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

Eén, N., and Sörensson, N. 2003. An Extensible SAT-solver. In *SAT Competition 2003*, volume 2919 of *Lecture Notes in Computer Science*, 502–518. Springer.

Fishelson, M., and Geiger, D. 2004. Optimizing Exact Genetic Linkage Computations. *Journal of Computational Biology* 11(2/3):263–275.

Geman, S., and Geman, D. 1984. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6:721–741.

Getoor, L., and Taskar, B., 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.; Richardson, S.; and Spiegelhalter, D. J., eds. 1996. *Markov Chain Monte Carlo in Practice*. London, UK: Chapman and Hall.

Gogate, V., and Dechter, R. 2007. SampleSearch: A scheme that Searches for Consistent Samples. *Proceedings of the 11th Conference on Artificial Intelligence and Statistics (AISTATS)* 147–154.

Gogate, V., and Dechter, R. 2011. SampleSearch: Importance sampling in presence of determinism. *Artificial Intelligence* 175(2):694–729.

Gogate, V., and Domingos, P. 2010. Formula-Based Probabilistic Inference. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, 210–219.

Gries, O. 2011. Gibbs sampling with deterministic dependencies. In *Proceedings of the 5th international conference on Multi-Disciplinary Trends in Artificial Intelligence*, 418–427.

Hajishirzi, H., and Amir, E. 2008. Sampling first order logical particles. In *UAI*, 248–255.

Jensen, C. S.; Kjaerulff, U.; and Kong, A. 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.; Valiant, L.; and Vazirani, V. 1986. Random generation of combinatorial structures from a uniform. *Theoretical Computer Science.* 43:169–188.

Kok, S.; Sumner, M.; Richardson, M.; Singla, P.; Poon, H.; and Domingos, P. 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.

Kokolakis, G., and Nanopoulos, P. 2001. Bayesian multivariate micro-aggregation under the Hellinger distance criterion. *Research in official statistics.* 4:117–125.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Liu, J. S. 2001. *Monte Carlo Strategies in Scientific Computing*. Springer Publishing Company, Incorporated.

McCallum, A., and Wellner, B. 2004. Conditional models of identity uncertainty with application to noun coreference. In *Proceedings of the 18th Annual Conference on Neural Information Processing Systems (NIPS)*.

Murphy, K. P.; Weiss, Y.; and Jordan, M. I. 1999. Loopy Belief Propagation for Approximate Inference: An Empirical Study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 467–475.

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.

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. Available online at http://radfordneal.wordpress.com/2008/08/17/the-harmonic-mean-of-the-likelihood-worst-monte-carlo-method-ever/.

Newton, M., and Raftery, A. 1994. Approximate Bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society.* 56:3–48.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufmann.

Poon, H., and Domingos, P. 2006. Sound and Efficient Inference with Probabilistic and Deterministic Dependencies. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Boston, MA: AAAI Press. This volume.

Roth, D. 1996. On the Hardness of Approximate Reasoning. *Artificial Intelligence* 82:273–302.

Sang, T.; Beame, P.; and Kautz, H. 2005. Solving Bayesian networks by weighted model counting. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, 475–482.

Shwe, M.; Middleton, B.; Heckerman, D.; Henrion, M.; Horvitz, E.; Lehmann, H.; and Cooper, G. 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.

Wei, W.; Erenrich, J.; and Selman, B. 2004. Towards efficient sampling: exploiting random walk strategies. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 670–676. AAAI Press.

Wemmenhove, B.; Mooij, J.; Wiegerinck, W.; Leisink, M.; and Kappen, H. J. 2008. Inference in the promedas medical expert system. In *Proceedings of the 11th Conference on Artificial Intelligence in Medicine (AIME 2007)*. Springer.

Yedidia, J. S.; Freeman, W. T.; and Weiss, Y. 2005. Constructing free-energy approximations and generalized Belief propagation algorithms. *IEEE Transactions on Information Theory* 51(7):2282–2312.

Yu, H., and van Engelen, R. 2011. 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*, 786–795.