

Sampling-based Lower Bounds for Counting Queries

Vibhav Gogate

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

Rina Dechter

Donald Bren School of Information and Computer Sciences
University of California, Irvine
Irvine, CA 92697, USA
Email: dechter@ics.uci.edu

Abstract

It is well known that the problem of computing relative approximations of weighted counting queries such as the probability of evidence in a Bayesian network, the partition function of a Markov network, and the number of solutions of a constraint satisfaction problem is NP-hard. In this paper, we settle therefore on an easier problem of computing high-confidence *lower bounds*. We propose to use importance sampling and Markov inequality for solving it. However, a straight-forward application of the Markov inequality often yields poor lower bounds. We therefore propose several new schemes for improving its performance in practice. Empirically, we show that our new schemes are quite powerful, often yielding substantially higher (better) lower bounds than all state-of-the-art schemes.

1 Introduction

Many inference problems in graphical models such as finding the probability of evidence in a Bayesian network, the partition function of a Markov network and the number of solutions of a constraint satisfaction problem are special cases of the following weighted counting problem: given a function $F : D \rightarrow \mathbb{R}^+$, find the sum of F over its domain D . Therefore, efficient algorithms for computing weighted counts are of paramount importance for a wide variety of applications that use graphical models, such as genetic linkage analysis [20, 1], car travel activity modeling [34, 25] functional verification [2, 16], target tracking [41], machine vision [19, 33], medical diagnosis [38, 42] and music parsing [43]. Unfortunately, the weighted counting problem is in $\#P$ and therefore there is no hope of designing efficient, general-purpose, polynomial time algorithms for it. Moreover, even approximations with relative error guarantees are NP-hard [11]. In this paper, we therefore settle on an easier problem of computing high confidence lower bounds, not insisting on having relative error guarantees.

Previous work on bounding approximations can be classified into two broad types: *deterministic approximations* (e.g., [18, 32, 3]) and *randomized approximations* (e.g., [10, 12]). A bounding algorithm is deterministic if it is always guaranteed to output a lower or an upper bound. On the other hand, a bounding algorithm is randomized if the approximation fails with a known probability $\delta \geq 0$. (For example, if $\delta = 0.1$, the bounds computed in roughly 1

out of 10 runs will be incorrect.) The work in this paper falls under the class of randomized approximations.

Existing randomized approximations [10, 12] use known inequalities such as the *Chebyshev and the Hoeffding inequalities* [30] for (lower and upper) bounding the weighted counts. These inequalities bound the deviation of the sample mean of N independent random variables from the actual mean. The idea which is in some sense similar to importance sampling [46, 21] is to express the counting problem as the problem of computing the mean (or the expected value) of independent random variables and then use the mean over the sampled random variables to bound the deviation from the true mean. The problem is that the number of samples required to guarantee high confidence bounds is inversely proportional to the true mean (the weighted counts). Therefore, if the weighted counts are arbitrarily small (e.g., $\leq 10^{-20}$), a large number of samples (approximately 10^{19}) are required to provide high confidence on the result.

We propose to alleviate this difficulty by using the *Markov inequality* instead of the Hoeffding inequality. Since the Markov inequality does not depend on N , even a single sample can yield a lower bound with some confidence. The only caveat is that unlike previous randomized approaches, our new schemes do not have any relative error guarantees. Recently, the Markov inequality was used to lower bound the number of solutions of a satisfiability (SAT) formula [29] showing good empirical results. We adapt and extend this approach as follows. We address one of the well known concerns in the statistics literature that the Markov inequality is quite weak and yields bad approximations. We argue that the Markov inequality is weak because it is based on a single sample and in fact good lower bounds can be obtained by extending it to multiple samples. Specifically, we propose several new schemes which utilize various statistics such as the average, the minimum, and the maximum, as well as the Martingale theory [7] to improve the Markov inequality. Our new schemes guarantee that as more samples are drawn, the lower bound will likely increase (improve).

We provide a thorough empirical evaluation demonstrating the potential of our new scheme. We compared against state-of-the-art deterministic approximations such as Variable elimination and Conditioning (VEC) [14] and bound propagation [32, 4] when computing the probability of evidence of a Bayesian network. For the task of lower bounding the number of models of a satisfiability formula, we compared against Relsat [45], which is a deterministic approximation and SampleCount [29], which is a randomized algorithm. Our results clearly show that our new randomized approximations based on the Markov inequality are far more scalable than deterministic approximations such as VEC, Relsat and bound propagation and in most cases yield far higher accuracy. Our schemes also yield higher lower bounds than SampleCount.

The research presented in this paper is based in part on [24].

The rest of this paper is organized as follows. In Section 2, we describe preliminaries and previous work. In Section 3, we present our basic lower bounding scheme and several enhancements. Experimental results are presented in Section 4 and we conclude in Section 5.

2 Notation, Background and Previous work

We denote variables by upper case letters (e.g., X, Y, \dots) and values of variables by lower case letters (e.g., x, y, \dots). Sets of variables are denoted by bold upper case letters (e.g., $\mathbf{X} = \{X_1, \dots, X_n\}$). We denote by $D(X_i)$ the set of possible values of X_i . $D(X_i)$ is also called the domain of X_i . $X_i = x_i$ or simply x_i when the variable is clear from the context, denotes an assignment of a value $x_i \in D(X_i)$ to X_i while $\mathbf{X} = \mathbf{x}$ (or simply \mathbf{x}) denotes a

sequence of assignments to all variables in \mathbf{X} , namely $\mathbf{x} = (X_1 = x_1, X_2 = x_2, \dots, X_n = x_n)$. $D(\mathbf{X})$ denotes the Cartesian product of the domains of all variables in \mathbf{X} , namely $D(\mathbf{X}) = D(X_1) \times \dots \times D(X_n)$. We denote the projection of an assignment \mathbf{x} to a set $\mathbf{S} \subseteq \mathbf{X}$ by $\mathbf{x}_{\mathbf{S}}$. Given an assignment \mathbf{y} and \mathbf{z} to the partition \mathbf{Y} and \mathbf{Z} of \mathbf{X} , $\mathbf{x} = (\mathbf{y}, \mathbf{z})$ denotes the composition of assignments to the two subsets.

$\sum_{\mathbf{x} \in D(\mathbf{X})}$ denotes the sum over all possible configurations of variables in \mathbf{X} , namely, $\sum_{\mathbf{x} \in D(\mathbf{X})} = \sum_{x_1 \in D(X_1)} \times \sum_{x_2 \in D(X_2)} \times \dots \times \sum_{x_n \in D(X_n)}$. For brevity, we will abuse notation and write $\sum_{x_i \in D(X_i)}$ as $\sum_{x_i \in X_i}$ and $\sum_{\mathbf{x} \in D(\mathbf{X})}$ as $\sum_{\mathbf{x} \in \mathbf{X}}$. The expected value $\text{Ex}_Q[X]$ of a random variable X with respect to a distribution Q is defined as: $\text{Ex}_Q[X] = \sum_{x \in X} xQ(x)$. The variance $\text{Var}_Q[X]$ of X is defined as: $\text{Var}_Q[X] = \sum_{x \in X} (x - \text{Ex}_Q[X])^2$. To simplify, we will write $\text{Ex}_Q[X]$ as $\text{Ex}[X]$ and $\text{Var}_Q[X]$ as $\text{Var}[X]$, when the identity of Q is clear from the context.

We denote (discrete) functions by upper case letters (e.g. F, H, C, I etc.), and the scope (set of arguments) of a function F by $V(F)$. Given an assignment \mathbf{y} to a superset \mathbf{Y} of $V(F)$, we will abuse notation and write $F(\mathbf{y}_{V(F)})$ as $F(\mathbf{y})$.

Definition 1. A **discrete graphical model** or a **Markov network** denoted by \mathcal{G} is a 3-tuple $\langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ where $\mathbf{X} = \{X_1, \dots, X_n\}$ is a finite set of variables, $\mathbf{D} = \{D(X_1), \dots, D(X_n)\}$ is a finite set of domains where $D(X_i)$ is the domain of variable X_i and $\mathbf{F} = \{F_1, \dots, F_m\}$ is a finite set of discrete-valued non-negative functions (also called potentials). The graphical model represents a joint distribution $P_{\mathcal{G}}$ over \mathbf{X} defined as:

$$P_{\mathcal{G}}(\mathbf{x}) = \frac{1}{Z} \prod_{i=1}^m F_i(\mathbf{x}) \quad (1)$$

where Z is a normalization constant, often called the partition function. It is given by:

$$Z = \sum_{\mathbf{x} \in \mathbf{X}} \prod_{i=1}^m F_i(\mathbf{x}) \quad (2)$$

The primary queries over Markov networks are computing the partition function and computing the marginal probability $P_{\mathcal{G}}(X_i = x_i)$.

We will refer to Z as **weighted counts**. The **weighted counting problem** is to compute the weighted counts.

Each graphical model is associated with a primal graph which depicts the dependencies between its variables.

Definition 2. The **primal graph** of a graphical model $\mathcal{G} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ is an undirected graph $G(\mathbf{X}, \mathbf{E})$ which has variables of \mathcal{G} as its vertices and an edge between two variables that appear in the scope of a function.

2.1 Bayesian and Constraint networks

Definition 3. A **Bayesian network** is a graphical model $\mathcal{B} = \langle \mathbf{X}, \mathbf{D}, \mathbf{G}, \mathbf{P} \rangle$ where $G = (\mathbf{X}, \mathbf{E})$ is a directed acyclic graph over the set of variables \mathbf{X} . Each function $P_i \in \mathbf{P}$ is a conditional probability table defined as $P_i(X_i | \mathbf{pa}_i)$, where $\mathbf{pa}_i = V(P_i) \setminus \{X_i\}$ is the set of parents of X_i in G .

The primal graph of a Bayesian network is also called the moral graph. When the entries of the CPTs are 0 and 1 only, they are called *deterministic or functional CPTs*. An evidence

$\mathbf{E} = \mathbf{e}$ is an instantiated subset of variables. A Bayesian network represents the following joint probability distribution:

$$P_{\mathcal{B}}(\mathbf{x}) = \prod_{i=1}^n P_i(\mathbf{x}_{\{X_i\}} | \mathbf{x}_{\mathbf{pa}_i}) \quad (3)$$

By definition, given a Bayesian network \mathcal{B} the probability of evidence $P_{\mathcal{B}}(\mathbf{e})$ is given by:

$$P_{\mathcal{B}}(\mathbf{e}) = \sum_{\mathbf{y} \in \mathbf{X} \setminus \mathbf{E}} \prod_{i=1}^n P_i((\mathbf{y}, \mathbf{e})_{\{X_i\}} | (\mathbf{y}, \mathbf{e})_{\mathbf{pa}_i}) \quad (4)$$

It is easy to see from Equations 2 and 4 that $P_{\mathcal{B}}(\mathbf{e})$ is equivalent to the weighted counts Z over an evidence instantiated Bayesian network. Another important query over a Bayesian network is computing the conditional marginal probability $P_{\mathcal{B}}(x_i | \mathbf{e})$ for a query variable $X_i \in \mathbf{X} \setminus \mathbf{E}$.

Definition 4. A **constraint network** is a graphical model $\mathcal{R} = \langle \mathbf{X}, \mathbf{D}, \mathbf{C} \rangle$ where $\mathbf{C} = \{C_1, \dots, C_m\}$ is a set of constraints. Each constraint C_i is a 0/1 function defined over its scope. Given an assignment \mathbf{x} , a constraint is said to be satisfied if $C_i(\mathbf{x}) = 1$. A constraint can also be expressed by a pair $\langle R_i, \mathbf{S}_i \rangle$ where R_i is a relation defined over the scope of C_i that contains all tuples for which $C_i(\mathbf{s}_i) = 1$. The primal graph of a constraint network is called the constraint graph.

A solution of a constraint network is an assignment \mathbf{x} to all variables that satisfies all the constraints. The primary query over a constraint network is to determine whether it has a solution and if it does to find one. Another important query is that of counting the number of solutions K of the constraint network, defined by:

$$K = \sum_{\mathbf{x} \in \mathbf{X}} \prod_{i=1}^m C_i(\mathbf{x}) \quad (5)$$

K is clearly identical to the weighted counts over a constraint network.

2.2 Previous work

Previous work by Dagum and Luby [12] and by Cheng [10] (these are the only two works that we are aware of) on randomized bounding algorithms for weighted counting has focused on providing relative-error guarantees. Their algorithms are based on a Monte Carlo simulation technique called importance sampling [36]. The main idea in importance sampling is to express the weighted counts as an expectation using an easy-to-sample distribution Q , which is called the proposal (or trial or importance) distribution. Then, the algorithm generate samples from Q and estimates the expectation (which equals the weighted counts) by a weighted average over the samples, where the weight of a sample \mathbf{x} is $\prod_{i=1}^m F_i(\mathbf{x})/Q(\mathbf{x})$. The weighted average is often called the sample mean.

Formally, given a proposal distribution Q such that $\prod_{i=1}^m F_i(\mathbf{x}) \geq 0 \Rightarrow Q(\mathbf{x}) \geq 0$, we can rewrite Equation 2 as follows:

$$Z = \sum_{\mathbf{x} \in \mathbf{X}} \frac{\prod_{i=1}^m F_i(\mathbf{x})}{Q(\mathbf{x})} Q(\mathbf{x}) = \text{Ex}_Q \left[\frac{\prod_{i=1}^m F_i(\mathbf{x})}{Q(\mathbf{x})} \right] \quad (6)$$

Given independent and identically distributed (i.i.d.) samples $(\mathbf{x}^1, \dots, \mathbf{x}^N)$ generated from Q , we can estimate Z by:

$$\widehat{Z}_N = \frac{1}{N} \sum_{k=1}^N \frac{\prod_{i=1}^m F_i(\mathbf{x}^k)}{Q(\mathbf{x}^k)} = \frac{1}{N} \sum_{k=1}^N w(\mathbf{x}^k) \quad (7)$$

where

$$w(\mathbf{x}) = \frac{\prod_{i=1}^m F_i(\mathbf{x})}{Q(\mathbf{x})}$$

is the weight of sample \mathbf{x} . It is easy to show that \widehat{Z}_N is unbiased, namely $\mathbf{E}_{\mathbf{x}}[\widehat{Z}_N] = Z$.

Dagum and Luby [12] provide an upper bound on the number of samples N required to guarantee that for any $\epsilon, \delta \geq 0$, the estimate \widehat{Z}_N approximates Z with relative error ϵ with probability at least $1 - \delta$. Formally,

$$\Pr[Z(1 - \epsilon) \leq \widehat{Z}_N \leq Z(1 + \epsilon)] \geq 1 - \delta \quad (8)$$

when N satisfies:

$$N \geq \frac{4}{Z\epsilon^2} \ln \frac{2}{\delta} \quad (9)$$

These bounds were later improved by Cheng [10] yielding:

$$N \geq \frac{1}{Z} \frac{1}{(1 + \epsilon) \ln(1 + \epsilon) - \epsilon} \ln \frac{2}{\delta} \quad (10)$$

In both these bounds (see Equations 9 and 10) N is inversely proportional to Z and therefore when Z is small, a large number of samples are required to achieve an acceptable confidence level $(1 - \delta) \geq 0.99$.

A bound on N is required because [12, 10] insist on approximating Z with a known relative error ϵ . If we relax this relative error requirement, we can use just one sample and the Markov inequality to obtain a high confidence lower bound on Z . Furthermore, we can improve the lower bound with more samples, as we demonstrate in the next section.

3 Markov Inequality based Lower Bounds

Proposition 1 (Markov Inequality). *For any random variable X and a real number $r \geq 1$, $\Pr(X \geq r\mathbb{E}[X]) \leq \frac{1}{r}$.*

The Markov inequality states that the probability that a random variable is r times its expected value is less than or equal to $1/r$.

We can apply the Markov inequality for lower bounding the weighted counts in a straightforward manner. We can consider the weight of each sample generated by importance sampling as a random variable. Because the expected value of the weight equals the weighted counts Z (see Equation 2), by Markov inequality, given a real number $r \geq 1$, the probability that the weight of a sample is greater than r times Z is less than $1/r$. Alternately, the weight of the sample divided by r is a lower bound on Z with probability greater than $1 - 1/r$. Formally, given a sample \mathbf{x} drawn independently from a proposal distribution Q , we have:

$$\Pr(w(\mathbf{x}) \geq r \times Z) \leq \frac{1}{r} \quad (11)$$

Rearranging Equation 11, we get:

$$\Pr\left(\frac{w(\mathbf{x})}{r} \leq Z\right) \geq 1 - \frac{1}{r} \quad (12)$$

Equation 12 can be used to probabilistically lower bound Z as shown in the following example.

Example 1. Let $r = 100$ and let \mathbf{x} be a sample generated using importance sampling. Then from Equation 12, $\frac{w(\mathbf{x})}{100}$ is a lower bound on Z with probability greater than $1 - (1/100) = 0.99$.

The lower bound based on the Markov inequality uses just one sample. In the following four subsections, we show how the lower bounds can be improved by utilizing multiple samples.

3.1 The Minimum scheme

This scheme due to Gomes et al. [29] uses the minimum over the sample weights to compute a lower bound on Z . Although this scheme was originally introduced in the context of lower bounding the number of solutions of a Boolean satisfiability (SAT) problem, we can easily modify it to compute a lower bound on the weighted counts as we show next.

Theorem 1 (minimum scheme). *Given N samples $(\mathbf{x}^1, \dots, \mathbf{x}^N)$ drawn independently from a proposal distribution Q such that $\mathbb{E}[w(\mathbf{x}^i)] = Z$ for $i = 1, \dots, N$ and a constant $0 \leq \alpha \leq 1$,*

$$\Pr\left[\min_{i=1}^N \left[\frac{w(\mathbf{x}^i)}{\beta}\right] \leq Z\right] \geq \alpha, \quad \text{where } \beta = \left(\frac{1}{1-\alpha}\right)^{\frac{1}{N}}$$

Proof. Consider an arbitrary sample \mathbf{x}^i . From the Markov inequality, we get:

$$\Pr\left[\frac{w(\mathbf{x}^i)}{\beta} \geq Z\right] \leq \frac{1}{\beta} \quad (13)$$

Since, the generated N samples are independent, the probability that the minimum over them is also a lower bound is given by:

$$\Pr\left[\min_{i=1}^N \left[\frac{w(\mathbf{x}^i)}{\beta}\right] \geq Z\right] \leq \frac{1}{\beta^N} \quad (14)$$

Rearranging Equation 14, we get:

$$\Pr\left[\min_{i=1}^N \left[\frac{w(\mathbf{x}^i)}{\beta}\right] \leq Z\right] \geq 1 - \frac{1}{\beta^N} \quad (15)$$

Substituting $\beta = \left(\frac{1}{1-\alpha}\right)^{\frac{1}{N}}$ in $1 - \frac{1}{\beta^N}$, we get:

$$\begin{aligned} 1 - \frac{1}{\beta^N} &= 1 - \frac{1}{\left(\left(\frac{1}{1-\alpha}\right)^{\frac{1}{N}}\right)^N} \\ &= 1 - \frac{1}{\frac{1}{1-\alpha}} \\ &= 1 - (1 - \alpha) \\ &= \alpha \end{aligned} \quad (16)$$

Algorithm 1: Minimum-scheme

Input: A graphical model $\mathcal{G} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, a proposal distribution Q , an integer N and a real number $0 \leq \alpha \leq 1$

Output: Lower Bound on Z that is correct with probability greater than α
 $minCount \leftarrow \infty$;

$$\beta = \left(\frac{1}{1-\alpha} \right)^{\frac{1}{N}};$$

for $i = 1$ **to** N **do**

 Generate a sample \mathbf{x}^i from Q ;
 IF $minCount \geq \frac{w(\mathbf{x}^i)}{\beta}$ **THEN** $minCount = \frac{w(\mathbf{x}^i)}{\beta}$;

Return $minCount$;

Therefore, from Equations 15 and 16, we have

$$\Pr \left[\min_{i=1}^k \left[\frac{w(\mathbf{x}^i)}{\beta} \right] \leq Z \right] \geq \alpha \quad (17)$$

□

Algorithm 1 describes the minimum scheme based on Theorem 1. The algorithm first calculates β based on the value of α and N . It then returns the minimum of $\frac{w(\mathbf{x}^i)}{\beta}$ ($minCount$ in Algorithm 1) over the N samples.

A good property of the minimum scheme is that with more samples the divisor, $\beta = \frac{1}{(1-\alpha)^{\frac{1}{N}}}$ decreases, thereby (possibly) increasing the lower bound. Its problem is that because it computes a minimum over the sample weights and thus unless the variance of the weights is very small, we expect the lower bound to decrease with an increase in the number of samples. Next, we present the average scheme, which avoids this problem.

3.2 The Average Scheme

An obvious scheme is to use the unbiased importance sampling estimator \widehat{Z}_N given in Equation 7. Because $\mathbb{E}_Q[\widehat{Z}_N] = Z$, from the Markov inequality $\frac{\widehat{Z}_N}{\beta}$ where $\beta = \frac{1}{1-\alpha}$ is a lower bound of Z with probability greater than α . Formally,

$$\Pr \left[\frac{\widehat{Z}_N}{\beta} \leq Z \right] \geq \alpha, \quad \text{where } \beta = \frac{1}{1-\alpha} \quad (18)$$

As more samples are drawn the average is likely to be get larger than the minimum value. This, in turn, increases the lower bound. However, unlike the minimum scheme in which the divisor β decreases with an increase in the sample size thereby increasing the lower bound, the divisor β in the average scheme remains constant. As a consequence, for example, if all the generated samples have the same weight (or almost the same weight), the lower bound due to the minimum scheme would be greater than the lower bound output by the average scheme. However, in practice the variance is typically never close to zero and therefore the average scheme is likely to be superior to the minimum scheme.

3.3 The Maximum scheme

We can even use the maximum instead of the average over the N i.i.d samples as shown in the following Lemma.

Lemma 1 (maximum scheme). *Given N samples $(\mathbf{x}^1, \dots, \mathbf{x}^N)$ drawn independently from a proposal distribution Q such that $\mathbb{E}[w(\mathbf{x}^i)] = Z$ for $i = 1, \dots, N$ and a constant $0 \leq \alpha \leq 1$,*

$$\Pr \left[\frac{\max_{i=1}^N w(\mathbf{x}^i)}{\beta} \leq Z \right] \geq \alpha, \text{ where } \beta = \frac{1}{1 - \alpha^{\frac{1}{N}}}$$

Proof. From Markov inequality, we have:

$$\Pr \left[\frac{w(\mathbf{x}^i)}{\beta} \leq Z \right] \geq 1 - \frac{1}{\beta} \quad (19)$$

Given a set of N independent events such that each event occurs with probability $\geq (1 - 1/\beta)$, the probability that all events occur is $\geq (1 - 1/\beta)^N$. In other words, given N independent samples such that the weight of each sample is a lower bound on Z with probability $\geq (1 - 1/\beta)$, the probability that all samples are a lower bound on Z is $\geq (1 - 1/\beta)^N$. Consequently,

$$\Pr \left[\frac{\max_{i=1}^N w(\mathbf{x}^i)}{\beta} \leq Z \right] \geq \left(\frac{1}{1 - \beta} \right)^N \quad (20)$$

Substituting the value of β in $\left(\frac{1}{1 - \beta} \right)^N$, we have:

$$\begin{aligned} \left(\frac{1}{1 - \beta} \right)^N &= \left(\frac{1}{1 - \frac{1}{1 - \alpha^{\frac{1}{N}}}} \right)^N \\ &= (1 - (1 - \alpha^{\frac{1}{N}}))^N \\ &= \alpha \end{aligned} \quad (21)$$

From Equations 20 and 21, we get:

$$\Pr \left[\frac{\max_{i=1}^N w(\mathbf{x}^i)}{\beta} \leq Z \right] \geq \alpha \quad (22)$$

□

The problem with the maximum scheme is that increasing the number of samples increases β and consequently the lower bound decreases. However, when only a few samples are available and the variance of the weights $w(\mathbf{x}^i)$ is large, the maximum value is likely to be larger than the sample average and obviously the minimum.

3.4 Using the Martingale Inequalities

Another approach to utilize the maximum over the N samples is to use the martingale inequalities.

Definition 5 (Martingale). A sequence of random variables X_1, \dots, X_N is a martingale with respect to another sequence Y_1, \dots, Y_N defined on a common probability space Ω iff $\mathbb{E}[X_i | Y_1, \dots, Y_{i-1}] = X_{i-1}$ for all i .

It is easy to see that given i.i.d. samples $(\mathbf{x}^1, \dots, \mathbf{x}^N)$ generated from Q , the sequence $\Lambda_1, \dots, \Lambda_N$, where $\Lambda_p = \prod_{i=1}^p \frac{w(\mathbf{x}^i)}{Z}$ forms a martingale as shown below:

$$\begin{aligned} \mathbb{E}[\Lambda_p | \mathbf{x}^1, \dots, \mathbf{x}^{p-1}] &= \mathbb{E}\left[\Lambda_{p-1} * \frac{w(\mathbf{x}^p)}{Z} \mid \mathbf{x}^1, \dots, \mathbf{x}^{p-1}\right] \\ &= \Lambda_{p-1} * \mathbb{E}\left[\frac{w(\mathbf{x}^p)}{Z} \mid \mathbf{x}^1, \dots, \mathbf{x}^{p-1}\right] \end{aligned}$$

Because $\mathbb{E}\left[\frac{w(\mathbf{x}^p)}{Z} \mid \mathbf{x}^1, \dots, \mathbf{x}^{p-1}\right] = 1$, we have $\mathbb{E}[\Lambda_p | \mathbf{x}^1, \dots, \mathbf{x}^{p-1}] = \Lambda_{p-1}$ as required. The expected value $\mathbb{E}[\Lambda_1] = 1$ and for such martingales which have a mean of 1, Breiman [7] provides the following extension of the Markov inequality:

$$\Pr(\max_{i=1}^N \Lambda_i \geq \beta) \leq \frac{1}{\beta} \quad (23)$$

and therefore,

$$\Pr\left(\left[\max_{i=1}^N \prod_{j=1}^i \frac{w(\mathbf{x}^j)}{Z}\right] \geq \beta\right) \leq \frac{1}{\beta} \quad (24)$$

From Inequality 24, we can prove that:

Theorem 2 (Random permutation scheme). *Given N samples $(\mathbf{x}^1, \dots, \mathbf{x}^N)$ drawn independently from a proposal distribution Q such that $\mathbb{E}[w(\mathbf{x}^i)] = Z$ for $i = 1, \dots, N$ and a constant $0 \leq \alpha \leq 1$,*

$$\Pr\left[\max_{i=1}^N \left(\frac{1}{\beta} \prod_{j=1}^i w(\mathbf{x}^j)\right)^{1/i} \leq Z\right] \geq \alpha, \text{ where } \beta = \frac{1}{1-\alpha}$$

Proof. From Inequality 24, we have:

$$\Pr\left(\left[\max_{i=1}^N \prod_{j=1}^i \frac{w(\mathbf{x}^j)}{Z}\right] \geq \beta\right) \leq \frac{1}{\beta} \quad (25)$$

Rearranging Inequality 25, we have:

$$\Pr\left[\max_{i=1}^N \left(\frac{1}{\beta} \prod_{j=1}^i w(\mathbf{x}^j)\right)^{1/i} \leq Z\right] \geq 1 - \frac{1}{\beta} = \alpha \quad (26)$$

□

Therefore, given N samples, the following quantity

$$\max_{i=1}^N \left(\frac{1}{\beta} \prod_{j=1}^i w(\mathbf{x}^j)\right)^{1/i} \text{ where } \beta = \frac{1}{1-\alpha}$$

is a lower bound on Z with a confidence greater than α . In general one could use any randomly selected permutation of the samples $(\mathbf{x}^1, \dots, \mathbf{x}^N)$ and apply inequality 24. We therefore call this scheme as the *random permutation scheme*.

Another related extension of Markov inequality for martingales deals with the order statistics of the samples. Let $\frac{w(\mathbf{x}^{(1)})}{Z} \leq \frac{w(\mathbf{x}^{(2)})}{Z} \leq \dots \leq \frac{w(\mathbf{x}^{(N)})}{Z}$ be the order statistics of the sample. Using martingale theory, Kaplan [31] proved that the random variable

$$\Theta^* = \max_{i=1}^N \prod_{j=1}^i \frac{w(\mathbf{x}^{(N-j+1)})}{Z \times \binom{N}{i}}$$

satisfies the inequality $\Pr(\Theta^* \geq k) \leq 1/k$. Therefore,

$$\Pr \left(\left[\max_{i=1}^N \prod_{j=1}^i \frac{w(\mathbf{x}^{(N-j+1)})}{Z \times \binom{N}{i}} \right] \geq \beta \right) \leq \frac{1}{\beta} \quad (27)$$

From Inequality 27, we can prove that:

Theorem 3 (Order Statistics scheme). *Given an order statistics of the weights $\frac{w(\mathbf{x}^{(1)})}{Z} \leq \frac{w(\mathbf{x}^{(2)})}{Z} \leq \dots \leq \frac{w(\mathbf{x}^{(N)})}{Z}$ of N samples $(\mathbf{x}^1, \dots, \mathbf{x}^N)$ drawn independently from a proposal distribution Q , such that $\mathbb{E}[w(\mathbf{x}^i)] = Z$ for $i = 1, \dots, N$ and a constant $0 \leq \alpha \leq 1$,*

$$\Pr \left[\max_{i=1}^N \left(\frac{1}{\beta} \prod_{j=1}^i \frac{w(\mathbf{x}^{(N-j+1)})}{\binom{N}{i}} \right)^{1/i} \leq Z \right] \geq \alpha, \text{ where } \beta = \frac{1}{1-\alpha}$$

Proof. From Inequality 27, we have:

$$\Pr \left(\left[\max_{i=1}^N \prod_{j=1}^i \frac{w(\mathbf{x}^{(N-j+1)})}{Z \times \binom{N}{i}} \right] \geq \beta \right) \leq \frac{1}{\beta} \quad (28)$$

Rearranging Inequality 28, we have:

$$\Pr \left[\max_{i=1}^N \left(\frac{1}{\beta} \prod_{j=1}^i \frac{w(\mathbf{x}^{(N-j+1)})}{\binom{N}{i}} \right)^{1/i} \leq Z \right] \geq 1 - \frac{1}{\beta} = \alpha \quad (29)$$

□

Thus, given N samples, the following quantity

$$\max_{i=1}^N \left(\frac{1}{\beta} \prod_{j=1}^i \frac{w(\mathbf{x}^{(N-j+1)})}{\binom{N}{i}} \right)^{1/i}, \text{ where } \beta = \frac{1}{1-\alpha}$$

is a lower bound on Z with probability greater than α . Because the lower bound is based on the order statistics, we call this scheme as the *order statistics* scheme.

To summarize, we have proposed five schemes that generalize the Markov inequality to multiple samples: (1) The minimum scheme, (2) The average scheme, (3) The maximum scheme, (4) The martingale random permutation scheme and (5) The martingale order statistics scheme. All these schemes can be used with any sampling scheme that outputs unbiased sample weights to yield a probabilistic lower bound on the weighted counts.

4 Empirical Evaluation

In this section, we compare the performance of the probabilistic lower bounding schemes presented in this paper with other deterministic schemes from literature. We also evaluate the relative performance of the five lower bounding schemes presented in Section 3. We conducted experiments on three weighted counting tasks: (a) Satisfiability model counting, (b) computing probability of evidence in a Bayesian network and (c) computing the partition function of a Markov network. Our experimental data clearly demonstrates that our new lower bounding schemes are more accurate, robust and scalable than all other deterministic approximations, yielding far better (higher) lower bounds on large, hard instances. We provide the details next.

4.1 The Algorithms Evaluated

We experimented with the following schemes. Hence forth, we call our new probabilistic lower bounding schemes as Markov-LB.

Markov-LB with SampleSearch and IJGP-sampling. As mentioned earlier, we can combine Markov-LB with any importance sampling scheme. Therefore, in order to compete and compare fairly with existing deterministic schemes, we combine Markov-LB with state-of-the-art importance sampling techniques such as IJGP-IS [25, 23] and IJGP-SampleSearch [27, 26, 22].

IJGP-IS uses the output of a generalized belief propagation scheme called Iterative Join Graph Propagation (IJGP) to construct a proposal distribution. It was shown that belief propagation schemes whether applied over the original graph or on clusters of nodes yield very good approximation to the true posterior than other available choices [17, 39, 55] and thus sampling from their output is an obvious choice (see [56, 25, 23] for more details).

IJGP [17, 37] is a generalized belief propagation scheme which is parametrized by a constant i , called the i -bound, yielding a class of algorithms IJGP(i) whose complexity is exponential in i , that trade-off accuracy and complexity. As i increases, accuracy generally increases. In our experiments, for every instance, we select the maximum i -bound that can be accommodated by 512 MB of space as follows. The space required by a message (or a function) is the product of the domain sizes of the variables in its scope. Given an i -bound, we can create a join graph whose cluster size is bounded by i as described in [17] and compute, in advance, the space required by IJGP by summing over the space required by the individual messages¹. We iterate from $i = 1$ until the space bound (of 512 MB) is surpassed. This ensures that IJGP terminates in a reasonable amount of time and requires bounded space.

On networks having substantial amount of determinism, we use IJGP-based SampleSearch (IJGP-SS) [27, 22] instead of IJGP-IS. It is known that on such networks pure importance sampling generates many useless zero weight samples which are eventually rejected. SampleSearch overcomes this *rejection problem* by explicitly searching for a non-zero weight sample, yielding a more efficient sampling scheme in such heavily deterministic databases. It was shown that SampleSearch is an importance sampling scheme which generates samples from a modification of the proposal distribution which is backtrack-free w.r.t. the constraints. Thus, in order to derive the weights of the samples generated by SampleSearch, all we need is to replace the proposal distribution with the backtrack-free distribution.

To reduce the variance of the weights, we combine both IJGP-IS and SampleSearch with w -cutset sampling [5](w -cutset sampling uses the Rao-Blackwell theorem [8, 35] to reduce

¹Note that we can do this without constructing the messages explicitly.

variance). The idea is to partition the variables \mathbf{X} into two sets \mathbf{K} and \mathbf{R} such that the treewidth of the graphical model restricted to \mathbf{R} is bounded by a constant w . The set \mathbf{K} is called the w -cutset. Then, we sample only the w -cutset variables and exactly compute the weighted counts of the graphical models given K . The latter can be computed efficiently using Bucket elimination [14] (since the treewidth is bounded).

Formally, given a graphical model $\mathcal{G} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, a w -cutset \mathbf{K} and a sample \mathbf{k} generated from a proposal distribution $Q(\mathbf{K})$, in w -cutset sampling, the weight of \mathbf{k} is given by:

$$w_{wc}(\mathbf{k}) = \frac{\sum_{\mathbf{r} \in \mathbf{R}} \prod_{j=1}^m F_j(\mathbf{r}, \mathbf{K} = \mathbf{k})}{Q(\mathbf{k})} \quad (30)$$

where $\mathbf{R} = \mathbf{X} \setminus \mathbf{K}$. Given a w -cutset \mathbf{K} , we can compute the sum in the numerator of Equation 30 in polynomial time (exponential in the constant w) using Bucket elimination [14].

It was demonstrated that the higher the w -bound [5], the lower the sampling variance. Here also, we select the maximum w such that the resulting bucket elimination algorithm uses less than 512 MB of space. We can choose the appropriate w by using a similar iterative scheme to the one described above for choosing the i -bound of IJGP.

Bound Propagation with Cut-set Conditioning. We also experimented with the state of the art any-time bounding scheme that combines sampling-based cut-set conditioning and bound propagation [32] and which is a part of Any-Time Bounds framework for bounding posterior marginals [3]. Given a subset of variables $\mathbf{C} \subset \mathbf{X} \setminus \mathbf{E}$, we can compute $P(\mathbf{e})$ exactly as follows:

$$P(\mathbf{e}) = \sum_{i=1}^k P(\mathbf{c}^i, \mathbf{e}) \quad (31)$$

The lower bound on $P(\mathbf{e})$ is obtained by computing $P(\mathbf{c}^i, \mathbf{e})$ for h high probability tuples of \mathbf{C} (selected through sampling) and bounding the remaining probability mass by computing a lower bound $P^L(\mathbf{c}_1, \dots, \mathbf{c}_q, \mathbf{e})$ on $P(\mathbf{c}_1, \dots, \mathbf{c}_q, \mathbf{e})$, $q \leq |\mathbf{C}|$, for a polynomial number of partially instantiated tuples of subset C , resulting in:

$$P(\mathbf{e}) \geq \sum_{i=1}^h P(\mathbf{c}^i, \mathbf{e}) + \sum_{i=1}^{k'} P_{BP}^L(\mathbf{c}_1^i, \dots, \mathbf{c}_q^i, \mathbf{e}) \quad (32)$$

where lower bound $P_{BP}^L(\mathbf{c}_1, \dots, \mathbf{c}_q, \mathbf{e})$ is obtained using bound propagation. Although bound propagation bounds marginal probabilities, it can be used to bound any joint probability $P(\mathbf{z})$ as follows:

$$P_{BP}^L(\mathbf{z}) = \prod_i P_{BP}^L(z_i | z_1, \dots, z_{i-1}) \quad (33)$$

where lower bound $P_{BP}^L(z_i | z_1, \dots, z_{i-1})$ is computed directly by bound propagation. We use here the same variant of bound propagation described in [4] that is used by the Any-Time Bounds framework. The lower bound obtained by Equation 32 can be improved by exploring a larger number of tuples h . After generating h tuples by sampling, we can stop the computation at any time after bounding $p \leq k'$ out of k' partially instantiated tuples and produce the result.

In our experiments we run the bound propagation with cut-set conditioning scheme until convergence or until a stipulated time bound has expired. Finally, we should note that the bound propagation with cut-set conditioning scheme provides deterministic lower and upper bounds on $P(\mathbf{e})$ while our Markov-LB scheme provides only a lower bound and it may fail with a probability $\delta \leq 0.01$.

Variable Elimination and Conditioning (VEC). When a problem having a high treewidth is encountered, variable or bucket elimination may be unsuitable, primarily because of its extensive memory demand. To alleviate the space complexity, we can use the w -cutset conditioning scheme [14]. Namely, we condition or instantiate enough variables or the w -cutset so that the remaining problem after removing the instantiated variables can be solved exactly using bucket elimination [14]. In our experiments we select the w -cutset in such a way that bucket elimination would require less than 1.5GB of space. Again, this is done to ensure that bucket elimination terminates in a reasonable amount of time and uses bounded space. Exact weighted counts can be computed by summing over the exact solution output by bucket elimination for all possible instantiations of the w -cutset. When VEC is terminated before completion, it outputs a partial sum yielding a lower bound on the weighted counts.

As pre-processing, the algorithm performs SAT-based variable domain pruning that often yields significant performance gains in practice. Here, we first convert all zero probabilities and constraints in the problem to a CNF formula F . Then, for each variable-value pair, we construct a new CNF formula F' by adding a clause corresponding to the pair to F and check using minisat [50] if F' is consistent or not. If F' is inconsistent then we delete the value from the domain of the variable. The implementation of this scheme is available publicly from our software website [15].

Markov-LB with SampleCount. SampleCount [29] is an algorithm for estimating the number of solutions of a Boolean Satisfiability problem. It is based on the ApproxCount algorithm of Wei and Selman [54]. ApproxCount is based on the formal result of [51], which states that if one can sample uniformly (or close to it) from the set of solutions of a SAT formula F , then one can exactly count (or approximate with a good estimate) the number of solutions of F . Consider a SAT formula F with S solutions. If we are able to sample solutions uniformly, then we can exactly compute the fraction of the number of solutions, denoted by γ that have a variable X set to *True* or 1 (and similarly to *False* or 0). If γ is greater than zero, we can set X to that particular value and simplify F to F' . The estimate of the number of solutions is now equal to the product of $\frac{1}{\gamma}$ and the number of solutions of F' . Then, we recursively repeat the process, leading to a series of multipliers, until all variables are assigned a value or until the conditioned formula is easy for exact model counters like Cachet [47]. To reduce the variance, [54] suggest to set the selected variable to a value that occurs more often in the given set of sampled solutions. In this scheme, the fraction for each variable branching is selected via a solution sampling method called SampleSat [53], which is an extension of the well-known local search SAT solver Walksat [48].

SampleCount [29] differs from ApproxCount in the following two ways: (a) SampleCount heuristically reduces the variance by branching on variables which are more balanced i.e. variables having multipliers $1/\gamma$ close to 2 and (b) At each branch point, SampleCount assigns a value to a variable by sampling it with probability 0.5 yielding an unbiased estimate of the solution counts. SampleCount is an importance sampling technique in which the weight of each sample equals $2^k \times s$, where k is the number of variables sampled and s is the model count of the SAT formula conditioned on the sampled assignment to the k sampled variables. Therefore, it can be easily combined with Markov-LB yielding the Markov-LB with SampleCount scheme.

In our experiments, we used an implementation of SampleCount available from the authors of gomes07. Following the recommendations made in [29], we use the following parameters for ApproxCount and SampleCount: (a) Number of samples for SampleSat = 20, (b) Number of variables remaining to be assigned a value before running Cachet = 100 and (c) local search

cutoff $\alpha = 100K$.

Relsat. Relsat [45] is an exact algorithm for counting the number of solutions of a SAT formula. When Relsat is stopped before termination, it yields a lower bound on the solution count. We used an implementation of Relsat available at <http://www.bayardo.org/resources.html>.

We experimented with four versions of Markov-LB (combined with SampleSearch, SampleCount and IJGP-Sampling): (a) Markov-LB as given in Algorithm 1, (b) Markov-LB with the average scheme, (c) Markov-LB with the martingale random permutation scheme and (d) Markov-LB with the martingale order statistics scheme. Note that the maximum scheme is subsumed by the Markov-LB with the martingale order statistics scheme. In all our experiments, we set $\alpha = 0.99$, namely there is better than 99% chance that our lower bounds are correct.

4.1.1 Evaluation Criteria

We evaluate the performance using the log relative error between the exact value of probability of evidence (or the solution counts for satisfiability problems) and the lower bound generated by the respective techniques. Formally, if Z is the actual probability of evidence (or solution counts) and \bar{Z} is the approximate probability of evidence (or solution counts), the log-relative error denoted by Δ is given by:

$$\Delta = \frac{\log(Z) - \log(\bar{Z})}{\log(Z)} \quad (34)$$

When the exact value of Z is not known, we use the highest lower bound reported by the schemes as a substitute for Z in Equation 34. We compute the log relative error instead of the usual relative error because when the probability of evidence is small ($\leq 10^{-10}$) or when the solution counts are large (e.g. $\geq 10^{10}$) the relative error between the exact and the approximate weighted counts will be arbitrarily close to 1 and we would need a large number of digits to determine the best performing scheme.

Notation in Tables The first column in each table (see for example Table 1) gives the name of the instance. The second column provides raw statistical information about the instance such as: (i) number of variables (n), (ii) average domain size (d), (iii) number of clauses (c) or number of evidence variables (e) and (iv) the upper bound on the treewidth of the instance computed using the min-fill algorithm (w). The third column provides the exact answer for the problem if available while the remaining columns display the output produced by the various schemes after the specified time-bound. The columns Min, Avg, Per and Ord give the log-relative-error Δ for the minimum, the average, the martingale random permutation and the martingale order statistics schemes respectively. For each instance, the log-relative error of the scheme yielding the best performance is highlighted in bold. The final column *Best LB* reports the best lower bound.

We organize our results in two parts. We first present results for networks which do not have determinism and compare Bound propagation and its improvements with IJGP-sampling based Markov-LB schemes. Then, we consider networks which have determinism and compare SampleSearch based Markov-LB with Variable elimination and Conditioning for probabilistic networks and with SampleCount for Boolean satisfiability problems.

Problem	$\langle n, d, e, w \rangle$	Exact	Markov-LB with IJGP-sampling				Bound propagation	Best LB
			Min	Avg	Per	Ord	Δ	
		P(e)	Δ	Δ	Δ	Δ	Δ	
Alarm								
BN_3	$\langle 100, 2, 36 \rangle$	2.8E-13	0.157	0.031	0.040	0.059	0.090	1.1E-13
BN_4	$\langle 100, 2, 51 \rangle$	3.6E-18	0.119	0.023	0.040	0.045	0.025	1.4E-18
BN_5	$\langle 125, 2, 55 \rangle$	1.8E-19	0.095	0.020	0.021	0.030	0.069	7.7E-20
BN_6	$\langle 125, 2, 71 \rangle$	4.3E-26	0.124	0.016	0.024	0.030	0.047	1.6E-26
BN_11	$\langle 125, 2, 46 \rangle$	8.0E-18	0.185	0.023	0.061	0.064	0.102	3.3E-18
CPCS								
CPCS-360-1	$\langle 360, 2, 20 \rangle$	1.3E-25	0.012	0.012	0.000	0.001	0.002	1.3E-25
CPCS-360-2	$\langle 360, 2, 30 \rangle$	7.6E-22	0.045	0.015	0.010	0.010	0.000	7.6E-22
CPCS-360-3	$\langle 360, 2, 40 \rangle$	1.2E-33	0.010	0.009	0.000	0.000	0.000	1.2E-33
CPCS-360-4	$\langle 360, 2, 50 \rangle$	3.4E-38	0.022	0.009	0.002	0.000	0.000	3.4E-38
CPCS-422-1	$\langle 422, 2, 20 \rangle$	7.2E-21	0.028	0.016	0.001	0.001	0.002	6.8E-21
CPCS-422-2	$\langle 422, 2, 30 \rangle$	2.7E-57	0.005	0.005	0.000	0.000	0.000	2.7E-57
CPCS-422-3	$\langle 422, 2, 40 \rangle$	6.9E-87	0.003	0.003	0.000	0.000	0.001	6.9E-87
CPCS-422-4	$\langle 422, 2, 50 \rangle$	1.4E-73	0.007	0.004	0.000	0.000	0.001	1.3E-73
Random								
BN_94	$\langle 53, 50, 6 \rangle$	4.0E-11	0.235	0.029	0.063	0.025	0.028	2.2E-11
BN_96	$\langle 54, 50, 5 \rangle$	2.1E-09	0.408	0.036	0.095	0.013	0.131	1.6E-09
BN_98	$\langle 57, 50, 6 \rangle$	1.9E-11	0.131	0.024	0.013	0.024	0.147	1.4E-11
BN_100	$\langle 58, 50, 8 \rangle$	1.6E-14	0.521	0.022	0.079	0.041	0.134	8.1E-15
BN_102	$\langle 76, 50, 15 \rangle$	1.5E-26	0.039	0.007	0.007	0.012	0.056	9.4E-27

Table 1: Table showing the log-relative error Δ of bound propagation and four versions of Markov-LB combined with IJGP-sampling for Bayesian networks having no determinism after 2 minutes of CPU time.

4.2 Results on networks having no determinism

Table 1 summarizes the results. We ran each algorithm for 2 minutes. We see that our new strategy of Markov-LB scales well with problem size and provides good quality high-confidence lower bounds on most problems. It clearly outperforms the bound propagation with cut-set conditioning scheme. We discuss the results in detail below.

Non-deterministic Alarm networks. The Alarm networks are one of the earliest Bayesian networks designed by medical experts for monitoring patients in intensive care. The evidence in these networks was set at random. These networks have between 100-125 binary nodes. We can see that Markov-LB with IJGP-sampling is slightly superior to the bound propagation based scheme accuracy-wise. Among the different versions of Markov-LB with IJGP-sampling, the average scheme performs better than the martingale schemes. The minimum scheme is the worst performing scheme.

The CPCS networks. The CPCS networks are derived from the Computer-based Patient Case Simulation system [42]. The nodes of CPCS networks correspond to diseases and findings and conditional probabilities describe their correlations. The CPCS360b and CPCS422b networks have 360 and 422 variables respectively. We report results on the two networks with 20,30,40 and 50 randomly selected evidence nodes. We see that the lower bounds reported by the bound propagation based scheme are slightly better than Markov-LB with IJGP-sampling on the CPCS360b networks. However, on the CPCS422b networks, Markov-LB with IJGP-sampling gives higher lower bounds. The martingale schemes (the random permutation and the order statistics) give higher lower bounds than the average scheme. Again, the minimum scheme is the weakest.

Random networks. The random networks are randomly generated graphs available from the UAI 2006 evaluation web site. The evidence nodes are generated at random. The networks have between 53 and 76 nodes and the maximum domain size is 50. We see that Markov-LB is better than the bound propagation based scheme on all random networks. The random permutation and the order statistics martingale schemes are slightly better than the average scheme on most instances.

4.3 Results on networks having determinism

In this subsection, we report on experiments for networks which have determinism. We experimented with five benchmark domains: (a) Latin square instances, (b) Langford instances, (c) FPGA routing instances, (d) Linkage instances and (e) Relational instances. The task of interest on the first three domains is counting solutions while the task of interest on the remaining domains is computing the probability of evidence.

4.3.1 Results on Satisfiability model counting

For model counting, we evaluate the lower bounding power of Markov-LB with SampleSearch and Markov-LB with SampleCount [29]. We ran both algorithms for 10 hours on each instance.

Results on the Latin Square instances Our first set of benchmark instances come from the normalized Latin squares domain. A Latin square of order s is an $s \times s$ table filled with s

Problem	$\langle n, k, c, w \rangle$	Exact	Markov-LB with SampleSearch				Markov-LB with SampleCount				REL SAT	Best
			Min	Avg	Per	Ord	Min	Avg	Per	Ord	Δ	
			Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
ls8-norm	$\langle 512, 2, 5584, 255 \rangle$	5.40E+11	0.387	0.012	0.068	0.095	0.310	0.027	0.090	0.090	0.344	3.88E+11
ls9-norm	$\langle 729, 2, 9009, 363 \rangle$	3.80E+17	0.347	0.021	0.055	0.070	0.294	0.030	0.097	0.074	0.579	1.59E+17
ls10-norm	$\langle 1000, 2, 13820, 676 \rangle$	7.60E+24	0.304	0.002	0.077	0.044	0.237	0.016	0.054	0.050	0.710	6.93E+24
ls11-norm	$\langle 1331, 2, 20350, 956 \rangle$	5.40E+33	0.287	0.023	0.102	0.026	0.227	0.036	0.094	0.034	0.783	7.37E+34
ls12-norm	$\langle 1728, 2, 28968, 1044 \rangle$		0.251	0.007	0.045	0.011	0.232	0.000	0.079	0.002	0.833	3.23E+43
ls13-norm	$\langle 2197, 2, 40079, 1558 \rangle$		0.250	0.005	0.080	0.000	0.194	0.015	0.087	0.044	0.870	1.26E+55
ls14-norm	$\langle 2744, 2, 54124, 1971 \rangle$		0.174	0.010	0.057	0.000	0.140	0.043	0.065	0.026	0.899	2.72E+67
ls15-norm	$\langle 3375, 2, 71580, 2523 \rangle$		0.189	0.015	0.080	0.000	0.130	0.053	0.077	0.062	0.923	4.84E+82
ls16-norm	$\langle 4096, 2, 92960, 2758 \rangle$		0.158	0.000	0.055	0.001	0.108	0.030	0.053	0.007	X	1.16E+97

Table 2: Table showing the log-relative error Δ of Relsat and four versions of Markov-LB combined with SampleSearch and SampleCount respectively for Latin Square instances after 10 hours of CPU time.

Problem	$\langle n, k, c, w \rangle$	Exact	Markov-LB with SampleSearch				Markov-LB with SampleCount				REL SAT	Best
			Min	Avg	Per	Ord	Min	Avg	Per	Ord	Δ	
			Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
lang12	$\langle 576, 2, 13584, 383 \rangle$	2.16E+05	0.464	0.051	0.128	0.171	0.455	0.067	0.103	0.175	0.000	2.16E+05
lang16	$\langle 1024, 2, 32320, 639 \rangle$	6.53E+08	0.475	0.008	0.106	0.131	0.378	0.019	0.097	0.023	0.365	7.68E+08
lang19	$\langle 1444, 2, 54226, 927 \rangle$	5.13E+11	0.405	0.041	0.109	0.095	0.420	0.156	0.219	0.200	0.636	1.70E+11
lang20	$\langle 1600, 2, 63280, 1023 \rangle$	5.27E+12	0.411	0.031	0.150	0.102	0.424	0.217	0.188	0.123	0.685	2.13E+12
lang23	$\langle 2116, 2, 96370, 1407 \rangle$	7.60E+15	0.389	0.058	0.119	0.100	0.418	0.215	0.284	0.211	X	9.15E+14
lang24	$\langle 2304, 2, 109536, 1535 \rangle$	9.37E+16	0.258	0.076	0.043	0.054	0.283	0.220	0.203	0.220	X	1.74E+16
lang27	$\langle 2916, 2, 156114, 1919 \rangle$		0.261	0.000	0.093	0.107	0.364	0.264	0.291	0.267	X	7.67E+19

Table 3: Table showing the log-relative error Δ of Relsat and four versions of Markov-LB combined with SampleSearch and SampleCount respectively for Langford instances after 10 hours of CPU time.

numbers from $\{1, \dots, s\}$ in such a way that each number occurs exactly once in each row and exactly once in each column. In a normalized Latin square the first row and column are fixed. The task here is to count the number of normalized Latin squares of a given order. The Latin squares were modeled as SAT formulas using the extended encoding given in [28]. The exact counts for these formulas are known up to order 11 [44].

Table 2 shows the results. The exact counts for Latin square instances are known only up to order 11. As pointed out earlier, when the exact results are not known, we use the highest lower bound reported by the schemes as a substitute for Z in Equation 34.

Among the different versions of Markov-LB with SampleSearch, we see that the average scheme performs better than the martingale order statistics scheme on 5 out of 8 instances while the martingale order statistics scheme is superior on the other 3 instances. The minimum scheme is the weakest scheme while the martingale random permutation scheme is between the minimum scheme and the average and martingale order statistics scheme.

Among the different versions of Markov-LB with SampleCount, we see very similar performance.

SampleSearch with Markov-LB generates better lower bounds than SampleCount with Markov-LB on 6 out of the 8 instances. The lower bounds output by Relsat are several orders of magnitude lower than those output by Markov-LB with SampleSearch and Markov-LB with SampleCount.

Problem	$\langle n, k, c, w \rangle$	Ex-act	Markov-LB with SampleSearch				Markov-LB with SampleCount				REL SAT	Best LB
			Min	Avg	Per	Ord	Min	Avg	Per	Ord	Δ	
			Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
9symml_gr_2pin_w6	$\langle 2604, 2, 36994, 413 \rangle$		0.192	0.000	0.075	0.006	0.087	0.073	0.076	0.075	0.491	2.76E+53
9symml_gr_rcs_w6	$\langle 1554, 2, 29119, 613 \rangle$		0.237	0.016	0.117	0.023	0.117	0.060	0.041	0.009	0.000	9.95E+84
alu2_gr_rcs_w8	$\langle 4080, 2, 83902, 1470 \rangle$		0.224	0.097	0.152	0.102	0.000	0.906	0.023	0.345	0.762	1.47E+235
apex7_gr_2pin_w5	$\langle 1983, 2, 15358, 188 \rangle$		0.158	0.003	0.073	0.000	0.064	0.023	0.047	0.036	0.547	2.71E+93
apex7_gr_rcs_w5	$\langle 1500, 2, 11695, 290 \rangle$		0.228	0.037	0.118	0.038	0.099	0.000	0.028	0.008	0.670	3.04E+139
c499_gr_2pin_w6	$\langle 2070, 2, 22470, 263 \rangle$		0.262	0.012	0.092	0.000	X	X	X	X	0.376	6.84E+54
c499_gr_rcs_w6	$\langle 1872, 2, 18870, 462 \rangle$		0.310	0.046	0.164	0.043	0.083	0.042	0.062	0.000	0.391	1.07E+88
c880_gr_rcs_w7	$\langle 4592, 2, 61745, 1024 \rangle$		0.223	0.110	0.142	0.110	0.000	0.000	0.000	0.003	0.845	1.37E+278
example2_gr_2pin_w6	$\langle 3603, 2, 41023, 350 \rangle$		0.112	0.000	0.026	0.000	0.005	0.005	0.005	0.005	0.756	2.78E+159
example2_gr_rcs_w6	$\langle 2664, 2, 27684, 476 \rangle$		0.176	0.050	0.079	0.054	0.056	0.005	0.000	0.005	0.722	1.47E+263
term1_gr_2pin_w4	$\langle 746, 2, 3964, 31 \rangle$		0.199	0.000	0.077	0.002	X	X	X	X	0.141	7.68E+39
term1_gr_rcs_w4	$\langle 808, 2, 3290, 57 \rangle$		0.252	0.000	0.090	0.017	X	X	X	X	0.175	4.97E+55
too_large_gr_rcs_w7	$\langle 3633, 2, 50373, 1069 \rangle$		0.156	0.026	0.073	0.000	X	X	X	X	0.608	7.73E+182
too_large_gr_rcs_w8	$\langle 4152, 2, 57495, 1330 \rangle$		0.147	0.000	0.038	0.020	X	X	X	X	0.750	8.36E+246
vda_gr_rcs_w9	$\langle 6498, 2, 130997, 2402 \rangle$		0.088	0.009	0.030	0.000	X	X	X	X	0.749	5.04E+300

Table 4: Table showing the log-relative error Δ of Relsat and four versions of Markov-LB combined with SampleSearch and SampleCount respectively for FPGA routing instances after 10 hours of CPU time.

Results on Langford instances Our second set of benchmark instances come from the Langford’s problem domain. The problem is parameterized by its (integer) size denoted by s . Given a set of s numbers $\{1, 2, \dots, s\}$, the problem is to produce a sequence of length $2s$ such that each $i \in \{1, 2, \dots, s\}$ appears twice in the sequence and the two occurrences of i are exactly i apart from each other. This problem is satisfiable only if n is 0 or 3 modulo 4. We encoded the Langford problem as a SAT formula using the channeling SAT encoding described in [52].

Table 3 shows the results. Among the different versions of Markov-LB with SampleSearch, we see again the superiority of the average scheme. The martingale order statistics and random permutation schemes are the second and the third best respectively. Among the different versions of SampleCount based Markov-LB, we see a similar trend where the average scheme performs better than other schemes on 6 out of the 7 instances.

Markov-LB with SampleSearch outperforms Markov-LB with SampleCount on 6 out of the 7 instances. The lower bounds output by Relsat are inferior by several orders of magnitude to the Markov-LB based lower bounds except on the lang12 instance which RELSAT solves exactly.

Results on FPGA routing instances Our final SAT domain is that of the FPGA routing instances. These instances are constructed by reducing FPGA (Field Programmable Gate Array) detailed routing problems into a satisfiability formula. The instances were generated by Gi-Joon Nam and were used in the SAT 2002 competition [49].

Table 4 shows the results for FPGA routing instances. We see a similar behavior to the Langford and Latin square instances in that the average and the martingale order statistics schemes are better than other schemes with the average scheme performing the best. Sample-Search based Markov-LB yields better lower bounds than SampleCount based Markov-LB on 11 out of the 17 instances. As in the other benchmarks, the lower bounds output by Relsat are inferior by several orders of magnitude.

Problem	$\langle n, k, c, w \rangle$	Exact	Markov-LB with SampleSearch				VEC	Best LB
			Min	Avg	Per	Ord		
			Δ	Δ	Δ	Δ		
BN_69.uai	$\langle 777, 7, 78, 47 \rangle$	5.28E-54	0.082	0.029	0.031	0.034	0.140	1.56E-55
BN_70.uai	$\langle 2315, 5, 159, 87 \rangle$	2.00E-71	0.275	0.035	0.101	0.046	0.147	6.24E-74
BN_71.uai	$\langle 1740, 6, 202, 70 \rangle$	5.12E-111	0.052	0.009	0.019	0.017	0.035	5.76E-112
BN_72.uai	$\langle 2155, 6, 252, 86 \rangle$	4.21E-150	0.021	0.002	0.004	0.007	0.023	2.38E-150
BN_73.uai	$\langle 2140, 5, 216, 101 \rangle$	2.26E-113	0.172	0.020	0.059	0.026	0.121	1.19E-115
BN_74.uai	$\langle 749, 6, 66, 45 \rangle$	3.75E-45	0.233	0.035	0.035	0.049	0.069	1.09E-46
BN_75.uai	$\langle 1820, 5, 155, 92 \rangle$	5.88E-91	0.077	0.005	0.024	0.019	0.067	1.98E-91
BN_76.uai	$\langle 2155, 7, 169, 64 \rangle$	4.93E-110	0.109	0.015	0.043	0.018	0.153	1.03E-111

Table 5: Table showing the log-relative error Δ of VEC and four versions of Markov-LB combined with SampleSearch for Linkage instances from the UAI 2006 evaluation after 3 hours of CPU time.

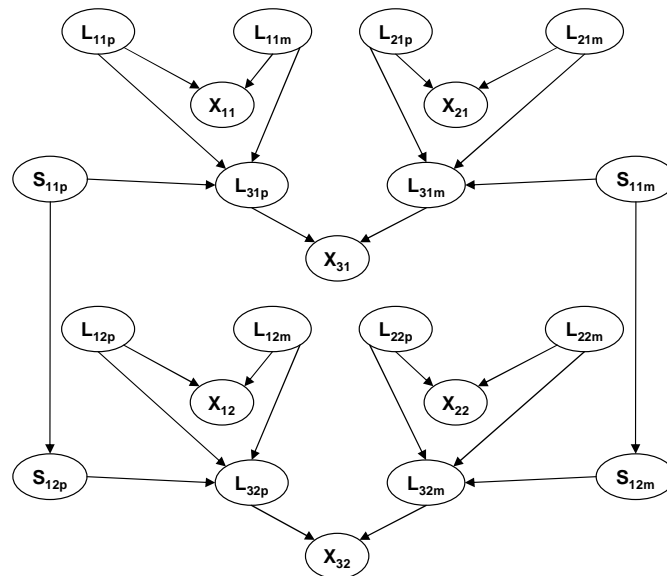


Figure 1: A fragment of a Bayesian network used in genetic linkage analysis.

4.3.2 Results on Linkage instances

The Linkage networks are generated by converting biological linkage analysis data into a Bayesian or Markov network. Linkage analysis is a statistical method for mapping genes onto a chromosome [40]. This is very useful in practice for identifying disease genes. The input is an ordered list of loci L_1, \dots, L_{k+1} with allele frequencies at each locus and a pedigree with some individuals typed at some loci. The goal of linkage analysis is to evaluate the likelihood of a candidate vector $[\theta_1, \dots, \theta_k]$ of recombination fractions for the input pedigree and locus order. The component θ_i is the candidate recombination fraction between the loci L_i and L_{i+1} .

The pedigree data can be represented as a Bayesian network with three types of random variables: genetic loci variables which represent the genotypes of the individuals in the pedigree (two genetic loci variables per individual per locus, one for the paternal allele and one for the maternal allele), phenotype variables, and selector variables which are auxiliary variables used to represent the gene flow in the pedigree. Figure 1 represents a fragment of a network that

Problem	$\langle n, k, c, w \rangle$	Exact	Markov-LB with SampleSearch				VEC	Best LB
			Min	Avg	Per	Ord		
			Δ	Δ	Δ	Δ	Δ	
pedigree18.uai	$\langle 1184, 1, 0, 26 \rangle$	7.18E-79	0.062	0.004	0.011	0.016	0.000	7.18E-79
pedigree1.uai	$\langle 334, 2, 0, 20 \rangle$	7.81E-15	0.034	0.020	0.020	0.020	0.000	7.81E-15
pedigree20.uai	$\langle 437, 2, 0, 25 \rangle$	2.34E-30	0.208	0.010	0.011	0.029	0.000	2.34E-30
pedigree23.uai	$\langle 402, 1, 0, 26 \rangle$	2.78E-39	0.093	0.007	0.016	0.019	0.000	2.78E-39
pedigree25.uai	$\langle 1289, 1, 0, 38 \rangle$	2.12E-119	0.006	0.022	0.019	0.019	0.024	1.69E-116
pedigree30.uai	$\langle 1289, 1, 0, 27 \rangle$	4.03E-88	0.014	0.039	0.039	0.035	0.042	1.85E-84
pedigree37.uai	$\langle 1032, 1, 0, 25 \rangle$	2.63E-117	0.031	0.005	0.005	0.006	0.000	2.63E-117
pedigree38.uai	$\langle 724, 1, 0, 18 \rangle$	5.64E-55	0.197	0.010	0.024	0.023	0.000	5.65E-55
pedigree39.uai	$\langle 1272, 1, 0, 29 \rangle$	6.32E-103	0.039	0.003	0.001	0.007	0.000	7.96E-103
pedigree42.uai	$\langle 448, 2, 0, 23 \rangle$	1.73E-31	0.024	0.009	0.007	0.010	0.000	1.73E-31
pedigree19.uai	$\langle 793, 2, 0, 23 \rangle$		0.158	0.018	0.000	0.031	0.011	3.67E-59
pedigree31.uai	$\langle 1183, 2, 0, 45 \rangle$		0.059	0.000	0.003	0.011	0.083	1.03E-70
pedigree34.uai	$\langle 1160, 1, 0, 59 \rangle$		0.211	0.006	0.000	0.012	0.174	4.34E-65
pedigree13.uai	$\langle 1077, 1, 0, 51 \rangle$		0.175	0.000	0.038	0.023	0.163	2.94E-32
pedigree40.uai	$\langle 1030, 2, 0, 49 \rangle$		0.126	0.000	0.036	0.008	0.025	4.26E-89
pedigree41.uai	$\langle 1062, 2, 0, 52 \rangle$		0.079	0.000	0.012	0.010	0.049	2.29E-77
pedigree44.uai	$\langle 811, 1, 0, 29 \rangle$		0.045	0.002	0.007	0.009	0.000	2.23E-64
pedigree51.uai	$\langle 1152, 1, 0, 51 \rangle$		0.150	0.003	0.027	0.000	0.139	1.01E-74
pedigree7.uai	$\langle 1068, 1, 0, 56 \rangle$		0.127	0.000	0.019	0.009	0.101	6.42E-66
pedigree9.uai	$\langle 1118, 2, 0, 41 \rangle$		0.072	0.000	0.009	0.009	0.028	1.41E-79

Table 6: Table showing the log-relative error Δ of VEC and four versions of Markov-LB combined with SampleSearch for Linkage instances from the UAI 2008 evaluation after 3 hours of CPU time.

describes parents-child interactions in a simple 2-loci analysis. The genetic loci variables of individual i at locus j are denoted by $L_{i,jp}$ and $L_{i,jm}$. Variables $X_{i,j}$, $S_{i,jp}$ and $S_{i,jm}$ denote the phenotype variable, the paternal selector variable and the maternal selector variable of individual i at locus j , respectively. The conditional probability tables that correspond to the selector variables are parameterized by the recombination ratio θ . The remaining tables contain only deterministic information. It can be shown that given the pedigree data, computing the likelihood of the recombination fractions is equivalent to computing the probability of evidence on the Bayesian network that model the problem (for more details consult [20]).

Table 5 show the results for linkage instances used in the UAI 2006 evaluation [6]. Here, we compare Markov-LB with SampleSearch with VEC. The bound propagation scheme [3] does not work on instances having determinism and therefore we do not report on it here. We clearly see that SampleSearch based Markov-LB yields higher lower bounds than VEC. Remember, however that the lower bounds output by VEC are correct (with probability 1) while the lower bounds output by Markov-LB are correct with probability ≥ 0.99 . We see that the average scheme is the best performing scheme. Martingale order statistics scheme is the second best while the Martingale random permutation scheme is the third best. The minimum scheme is the worst performing scheme.

Table 6 reports the results on Linkage instances encoded as Markov networks, used in the UAI 2008 evaluation [13]. VEC solves 10 instances exactly. On these instances, the lower bound output by SampleSearch based Markov-LB are quite accurate as evidenced by the small log relative error. On instances which VEC does not solve exactly, we clearly see that Markov-LB with SampleSearch yields higher lower bounds than VEC.

Comparing between different versions of Markov-LB, we see that the average scheme is

Problem	$\langle n, k, c, w \rangle$	Exact	Markov-LB with SampleSearch				VEC	Best LB
			Min	Avg	Per	Ord		
			Δ	Δ	Δ	Δ		
fs-01.uai	$\langle 10, 2, 7, 2 \rangle$	5.00E-01	0.000	0.000	0.000	0.000	0.000	5.00E-01
fs-04.uai	$\langle 262, 2, 226, 12 \rangle$	1.53E-05	0.116	0.116	0.116	0.116	0.000	1.53E-05
fs-07.uai	$\langle 1225, 2, 1120, 35 \rangle$	9.80E-17	0.028	0.004	0.014	0.016	0.079	1.78E-15
fs-10.uai	$\langle 3385, 2, 3175, 71 \rangle$	7.89E-31	0.071	0.064	0.064	0.065	X	9.57E-33
fs-13.uai	$\langle 7228, 2, 6877, 119 \rangle$	1.34E-51	0.077	0.077	0.077	0.077	X	1.69E-55
fs-16.uai	$\langle 13240, 2, 12712, 171 \rangle$	8.64E-78	0.085	0.019	0.048	0.025	X	3.04E-79
fs-19.uai	$\langle 21907, 2, 21166, 243 \rangle$	2.13E-109	0.051	0.050	0.050	0.050	X	8.40E-115
fs-22.uai	$\langle 33715, 2, 32725, 335 \rangle$	2.00E-146	0.053	0.006	0.022	0.009	X	2.51E-147
fs-25.uai	$\langle 49150, 2, 47875, 431 \rangle$	7.18E-189	0.050	0.005	0.026	0.004	X	1.57E-189
fs-28.uai	$\langle 68698, 2, 67102, 527 \rangle$	9.83E-237	0.231	0.017	0.023	0.011	X	4.53E-237
fs-29.uai	$\langle 76212, 2, 74501, 559 \rangle$	6.82E-254	0.259	0.101	0.201	0.027	X	9.44E-255
mastermind_03_08_03	$\langle 1220, 2, 48, 20 \rangle$	9.79E-08	0.283	0.039	0.034	0.096	0.000	9.79E-08
mastermind_03_08_04	$\langle 2288, 2, 64, 30 \rangle$	8.77E-09	0.562	0.045	0.145	0.131	0.000	8.77E-09
mastermind_03_08_05	$\langle 3692, 2, 80, 42 \rangle$	8.90E-11	0.432	0.041	0.021	0.095	0.000	1.44E-10
mastermind_04_08_03	$\langle 1418, 2, 48, 22 \rangle$	8.39E-08	0.297	0.041	0.072	0.082	0.000	8.39E-08
mastermind_04_08_04	$\langle 2616, 2, 64, 33 \rangle$	2.20E-08	0.640	0.026	0.155	0.103	0.034	1.38E-08
mastermind_05_08_03	$\langle 1616, 2, 48, 28 \rangle$	5.30E-07	0.625	0.062	0.188	0.185	0.000	5.30E-07
mastermind_06_08_03	$\langle 1814, 2, 48, 31 \rangle$	1.80E-08	0.510	0.058	0.193	0.175	0.000	1.80E-08
mastermind_10_08_03	$\langle 2606, 2, 48, 56 \rangle$	1.92E-07	0.839	0.058	0.297	0.162	0.058	7.90E-08

Table 7: Table showing the log-relative error Δ of VEC and four versions of Markov-LB combined with SampleSearch for relational instances after 3 hours of CPU time.

overall the best performing scheme. The Martingale order statistics scheme is the second best scheme while the Martingale random permutation scheme is the third best.

4.3.3 Results on Relational instances

The relational instances are generated by grounding the relational Bayesian networks using the Primula tool [9]. We experimented with ten Friends and Smoker networks and six mastermind networks from this domain which have between 262 to 76,212 variables.

In Table 7, we report the results on instances with 10 Friends and Smoker networks and 6 mastermind networks from this domain which have between 262 to 76,212 variables. On the 11 friends and smokers network, we can see that as the problems get larger the lower bounds output by Markov-LB with SampleSearch are higher than VEC. This clearly indicates that Markov-LB with SampleSearch is more scalable than VEC. VEC solves exactly six out of the eight mastermind instances while on the remaining two instances Markov-LB with SampleSearch yields higher lower bounds than VEC.

4.4 Summary of Experimental Results

Based on our large scale experimental evaluation, we see that applying Markov inequality and its generalizations to multiple samples generated by SampleSearch and IJGP-Sampling is more scalable than deterministic approaches such as Variable elimination and conditioning (VEC), Relsat and improved bound propagation. Among the different versions of Markov-LB, we find that the average and martingale order statistics schemes consistently yield higher lower bounds and therefore they should be preferred over the minimum scheme as well as the martingale random permutation scheme.

5 Conclusion

In this paper, we proposed a randomized approximation algorithm, *Markov-LB* for computing high confidence lower bounds on weighted counting tasks such as computing the probability of evidence in a Bayesian network, counting the number of solutions of a constraint network and computing the partition function of a Markov network. Markov-LB is based on importance sampling and the Markov inequality. Since a straight-forward application of the Markov inequality may lead to poor lower bounds, we proposed several improved measures such as the average scheme which utilizes the sample average and martingale schemes which utilize the maximum values from the sample weights. We showed that Markov-LB combined with state-of-the-art importance sampling approaches such as IJGP-sampling [23], SampleSearch [22], and SampleCount [29] is substantially superior to all other existing state-of-the-art deterministic algorithms such as bound propagation [4], Relsat [45] and variable elimination and conditioning.

References

- [1] David Allen and Adnan Darwiche. Rc-link: Genetic linkage analysis using Bayesian networks. *International Journal of Approximate Reasoning*, 48(2):499–525, 2008.
- [2] J Bergeron. *Writing Testbenches: Functional Verification of HDL Models*. Kluwer Academic Publishers, 2000.
- [3] Bozhena Bidyuk and Rina Dechter. An anytime scheme for bounding posterior beliefs. In *Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence*, pages 1095–1100, 2006.
- [4] Bozhena Bidyuk and Rina Dechter. Improving bound propagation. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI)*, pages 342–346, 2006.
- [5] Bozhena Bidyuk and Rina Dechter. Cutset sampling for Bayesian networks. *Journal of Artificial Intelligence Research*, 28:1–48, 2007.
- [6] Jeff Bilmes and Rina Dechter. Evaluation of Probabilistic Inference Systems of UAI’06. Available online at <http://ssli.ee.washington.edu/bilmes/uai06InferenceEvaluation/>, 2006.
- [7] Leo Breiman. *Probability*. Addison-Wesley, 1968.
- [8] George Casella and Christian P. Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.
- [9] M D Chavira, A Darwiche, and M Jaeger. Compiling relational Bayesian networks for exact inference. *International Journal of Approximate Reasoning*, 42(1-2):4–20, 2006.
- [10] Jian Cheng. Sampling algorithms for estimating the mean of bounded random variables. In *Computational Statistics*, pages 1–23. Volume 16(1), 2001.
- [11] Paul Dagum and Michael Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–153, 1993.

- [12] Paul Dagum and Michael Luby. An optimal approximation algorithm for Bayesian inference. *Artificial Intelligence*, 93:1–27, 1997.
- [13] Adnan Darwiche, Rina Dechter, Arthur Choi, Vibhav Gogate, and Lars Otten. Results from the Probabilistic Inference Evaluation of UAI’08. Available online at: <http://graphmod.ics.uci.edu/uai08/Evaluation/Report>, 2008.
- [14] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, 1999.
- [15] Rina Dechter, Vibhav Gogate, Lars Otten, Radu Marinescu, and Robert Mateescu. Graphical model algorithms at UC Irvine. website: <http://graphmod.ics.uci.edu/group/Software>, 2009.
- [16] Rina Dechter, Kalev Kask, Eyal Bin, and Roy Emek. Generating random solutions for constraint satisfaction problems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI)*, pages 15–21, 2002.
- [17] Rina Dechter, Kalev Kask, and Robert Mateescu. Iterative join graph propagation. In *Proceedings of the Eighteenth Conference in Uncertainty in Artificial Intelligence*, pages 128–136, 2002.
- [18] Rina Dechter and Irina Rish. Mini-buckets: A general scheme for bounded inference. *Journal of ACM*, 50(2):107–153, 2003.
- [19] Paul W. Fieguth, William C. Karl, and Alan S. Willsky. Efficient multiresolution counterparts to variational methods for surface reconstruction. *Computer Vision and Image Understanding*, 70(2):157–176, 1998.
- [20] Maáyan Fishelson and Dan Geiger. Optimizing exact genetic linkage computations. In *Proceedings of the seventh annual international conference on Research in computational molecular biology*, pages 114–121, 2003.
- [21] John Geweke. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica*, 57(6):1317–39, 1989.
- [22] V. Gogate and R. Dechter. SampleSearch: Importance sampling in presence of determinism. *Artificial Intelligence*, 175(2):694–729, February 2011.
- [23] Vibhav Gogate. *Sampling Algorithms for Probabilistic Graphical Models with Determinism*. PhD thesis, Computer Science, University of California, Irvine, USA, June 2009.
- [24] Vibhav Gogate, Bozhena Bidyuk, and Rina Dechter. Studies in lower bounding probability of evidence using the Markov inequality. In *Proceedings of the Twenty Third Conference on Uncertainty in Artificial Intelligence*, pages 141–148, 2007.
- [25] Vibhav Gogate and Rina Dechter. Approximate inference algorithms for hybrid Bayesian networks with discrete constraints. In *Proceedings of the Twenty First Annual Conference on Uncertainty in Artificial Intelligence*, pages 209–216, 2005.
- [26] Vibhav Gogate and Rina Dechter. Approximate counting by sampling the backtrack-free search space. In *Proceedings of Twenty Second Conference on Artificial Intelligence*, pages 198–203, 2007.

- [27] Vibhav Gogate and Rina Dechter. Samplesearch: A scheme that searches for consistent samples. *Proceedings of the Eleventh Conference on Artificial Intelligence and Statistics*, pages 147–154, 2007.
- [28] Carla Gomes and David Shmoys. Completing quasigroups or latin squares: A structured graph coloring problem. In *Proceedings of the Computational Symposium on Graph Coloring and Extensions*, 2002.
- [29] Carla P. Gomes, Jörg Hoffmann, Ashish Sabharwal, and Bart Selman. From sampling to model counting. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 2293–2299, 2007.
- [30] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.
- [31] Harold M Kaplan. A method of one-sided nonparametric inference for the mean of a nonnegative population. *The American Statistician*, 41(2):157–158, 1987.
- [32] Martijn Leisink and Bert Kappen. Bound propagation. *Journal of Artificial Intelligence Research*, 19:139–154, 2003.
- [33] Fei-Fei Li and Pietro Perona. A Bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 524–531, 2005.
- [34] Lin Liao, Donald J. Patterson, Dieter Fox, and Henry Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5-6):311–331, 2007.
- [35] J.S. Liu. *Monte-Carlo strategies in scientific computing*. Springer-Verlag, New York, 2001.
- [36] A. W. Marshall. The use of multi-stage sampling schemes in Monte Carlo computations. In *Symposium on Monte Carlo Methods*, pages 123–140, 1956.
- [37] R. Mateescu, K. Kask, V. Gogate, and R. Dechter. Join-graph propagation algorithms. *Journal of Artificial Intelligence Research*, 37:279–328, 2010.
- [38] Blackford Middleton, S Michael Shwe M., David Heckerman, D Harold Lehmann M., and Gregory Cooper. Probabilistic diagnosis using a reformulation of the internist-1/qmr knowledge base. *Medicine*, 30:241–255, 1991.
- [39] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 467–475, 1999.
- [40] Jurg Ott. *Analysis of Human Genetic Linkage*. The Johns Hopkins University Press, Baltimore, Maryland, 1999.
- [41] V. Pavlovic, J. M. Rehg, Tat-Jen Cham, and K. P. Murphy. A dynamic Bayesian network approach to figure tracking using learned dynamic models. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 94–101, 1999.

- [42] M. Pradhan, G. Provan, B. Middleton, and M. Henrion. Knowledge engineering for large belief networks. In *Proceedings of the 10th Canadian Conference on Artificial Intelligence*, pages 484–490, 1994.
- [43] C. Raphael. A hybrid graphical model for rhythmic parsing. *Artificial Intelligence*, 137(1-2):217–238, May 2002.
- [44] Terry Ritter. Latin squares: A literature survey. Available online at: <http://www.ciphersbyritter.com/RES/LATSQ.HTM>, 2003.
- [45] Jr. Roberto J. Bayardo and Joseph Daniel Pehoushek. Counting models using connected components. In *Proceedings of Seventeenth National Conference on Artificial Intelligence*, pages 157–162, 2000.
- [46] R. Y. Rubinstein. *Simulation and the Monte Carlo Method*. John Wiley & Sons Inc., 1981.
- [47] Tian Sang, Paul Beame, and Henry A. Kautz. Performing bayesian inference by weighted model counting. In *Proceedings, The Twentieth National Conference on Artificial Intelligence*, pages 475–482, 2005.
- [48] B. Selman, H. Kautz, and B. Cohen. Noise strategies for local search. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 337–343, 1994.
- [49] Laurent Simon, Daniel Le Berre, and E. Hirsch. The SAT 2002 competition. *Annals of Mathematics and Artificial Intelligence (AMAI)*, 43:307–342, 2005.
- [50] Niklas Sorensson and Niklas Een. Minisat v1.13-a SAT solver with conflict-clause minimization. In *SAT 2005 competition*, 2005.
- [51] L. G. Valiant. The complexity of enumeration and reliability problems. *Siam Journal of Computation*, 8(3):105–117, 1987.
- [52] Toby Walsh. Permutation problems and channelling constraints. In *Proceedings of the 8th International Conference on Logic Programming and Automated Reasoning (LPAR)*, pages 377–391, 2001.
- [53] Wei Wei, Jordan Erenrich, and Bart Selman. Towards efficient sampling: Exploiting random walk strategies. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pages 670–676, 2004.
- [54] Wei Wei and Bart Selman. A new approach to model counting. In *Proceedings of Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 324–339, 2005.
- [55] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51:2282–2312, 2004.
- [56] C. Yuan and M. J. Druzdzel. Importance sampling algorithms for Bayesian networks: Principles and performance. *Mathematical and Computer Modeling*, 43(9-10):1189–1207, 2006.