SampleSearch: Importance Sampling in presence of Determinism

Vibhav Gogate*,[a], Rina Dechter[a]

[a]*Donald Bren School of Information and Computer Sciences,*
University of California, Irvine,
Irvine, CA 92697, USA

Abstract

The paper focuses on developing effective importance sampling algorithms for mixed probabilistic and deterministic graphical models. The use of importance sampling in such graphical models is problematic because it generates many useless zero weight samples which are rejected yielding an inefficient sampling process. To address this *rejection problem*, we propose the *SampleSearch* scheme that augments sampling with systematic constraint-based backtracking search. We characterize the bias introduced by the combination of search with sampling, and derive a weighting scheme which yields an unbiased estimate of the desired statistics (e.g. probability of evidence). When computing the weights exactly is too complex, we propose an approximation which has a weaker guarantee of asymptotic unbiasedness. We present results of an extensive empirical evaluation demonstrating that SampleSearch outperforms other schemes in presence of significant amount of determinism.

1. Introduction

The paper investigates importance sampling algorithms for answering *weighted counting and marginal queries* over mixed probabilistic and deterministic networks (Dechter and Larkin, 2001; Larkin and Dechter, 2003; Dechter and Mateescu, 2004; Mateescu and Dechter, 2009). The mixed networks framework treats probabilistic graphical models such as Bayesian and Markov networks (Pearl, 1988), and deterministic graphical models such as constraint networks (Dechter, 2003) as a single graphical model. Weighted counts express the probability of evidence of a Bayesian network, the partition function of a Markov network and the number of solutions of a constraint network. Marginals seek the marginal distribution of each variable, also called as belief updating or posterior estimation in a Bayesian or Markov network.

It is straightforward to design importance sampling algorithms (Marshall, 1956; Rubinstein, 1981; Geweke, 1989) for approximately answering counting and marginal queries because both are variants of *summation problems* for which importance sampling was designed. Weighted counts is the sum of a function over some domain while a marginal is a ratio between two sums. The main idea is to transform a summation into

*Corresponding author
Email addresses: vgogate@gmail.com (Vibhav Gogate), dechter@ics.uci.edu (Rina Dechter)

an expectation using a special distribution called the proposal (or importance or trial) distribution from which it would be easy to sample. Importance sampling then generates samples from the proposal distribution and approximates the expectation (also called the true average or the true mean) by a weighted average over the samples (also called the sample average or the sample mean). The sample mean can be shown to be an unbiased estimate of the original summation, and therefore importance sampling yields an unbiased estimate of the weighted counts. For marginals, importance sampling has to compute a ratio of two unbiased estimates yielding an asymptotically unbiased estimate only.

In presence of hard constraints or zero probabilities, however, importance sampling may suffer from the *rejection problem*. The rejection problem occurs when the proposal distribution does not faithfully capture the constraints in the mixed network. Consequently, many samples generated from the proposal distribution may have zero weight and would not contribute to the sample mean. In extreme cases, the probability of generating a rejected sample can be arbitrarily close to one yielding completely wrong estimates of both weighted counts and marginals in practice.

In this paper, we propose a sampling scheme called *SampleSearch* to remedy the rejection problem. SampleSearch combines systematic backtracking search with Monte Carlo sampling. In this scheme, when a sample is supposed to be rejected, the algorithm continues instead with randomized backtracking search until a sample with non-zero weight is found. This problem of generating a non-zero weight sample is equivalent to the problem of finding a solution to a satisfiability (SAT) or a constraint satisfaction problem (CSP). SAT and CSPs are NP-Complete problems and therefore the idea of generating just one sample by solving an NP-Complete problem may seem inefficient. However, recently SAT/CSP solvers have achieved unprecedented success and are able to solve some large industrial problems having as many as a million variables within a few seconds[1]. Therefore, solving a constant number of NP-complete problems to approximate a #P-complete problem such as weighted counting is no longer unreasonable.

We show that SampleSearch generates samples from a modification of the proposal distribution which is *backtrack-free*. The backtrack-free distribution can be obtained by removing all partial assignments which lead to a zero weight sample. Namely, the backtrack-free distribution is zero whenever the target distribution from which we wish to sample is zero. We propose two schemes to compute the backtrack-free probability of the generated samples which is required for computing the sample weights. The first is a computationally intensive method which involves invoking a CSP or a SAT solver $O(n \times d)$ times where n is the number of variables and d is the maximum domain size. The second scheme approximates the backtrack-free probability by consulting information gathered during SampleSearch's operation. This latter scheme has several desirable properties: (i) it runs in linear time, (ii) it yields an asymptotically unbiased estimate and (iii) it can provide upper and lower bounds on the exact backtrack-free probability.

Finally, we present empirical evaluation demonstrating the power of SampleSearch.

[1]See results of SAT competitions available at http://www.satcompetition.org/.

We implemented SampleSearch on top of IJGP-wc-IS (Gogate and Dechter, 2005), a powerful importance sampling technique which uses a generalized belief propagation algorithm (Yedidia, Freeman, and Weiss, 2004) called Iterative Join Graph propagation (IJGP) (Dechter, Kask, and Mateescu, 2002) to construct a proposal distribution and w-cutset (Rao-Blackwellised) sampling (Bidyuk and Dechter, 2007) to reduce the variance. The search was implemented using the *minisat SAT solver* (Sorensson and Een, 2005). We conducted experiments on three tasks: (a) counting models of a SAT formula (b) computing the probability of evidence in a Bayesian network and the partition function of a Markov network, and (c) computing posterior marginals in Bayesian and Markov networks.

For model counting, we compared against three approximate algorithms: ApproxCount (Wei, Erenrich, and Selman, 2004), SampleCount (Gomes, Hoffmann, Sabharwal, and Selman, 2007) and Relsat (Roberto J. Bayardo and Pehoushek, 2000) as well as with IJGP-wc-IS, our vanilla importance sampling scheme on three classes of benchmark instances. Our experiments show that on most instances, given the same time bound SampleSearch yields solution counts which are closer to the true counts by a few orders of magnitude compared with the other schemes. It is clearly better than IJGP-wc-IS which failed on all benchmark SAT instances and was unable to generate a single non-zero weight sample in ten hours of CPU time.

For the problem of computing the probability of evidence in a Bayesian network, we compared SampleSearch with Variable Elimination and Conditioning (VEC) (Dechter, 1999), an advanced generalized belief propagation scheme called Edge Deletion Belief Propagation (EDBP) (Choi and Darwiche, 2006) as well as with IJGP-wc-IS on linkage analysis (Fishelson and Geiger, 2003) and relational (Chavira, Darwiche, and Jaeger, 2006) benchmarks. Our experiments show that on most instances the estimates output by SampleSearch are more accurate than those output by EDBP and IJGP-wc-IS. VEC solved some instances exactly, however on the remaining instances it was substantially inferior.

For the posterior marginal task, we experimented with linkage analysis benchmarks, with partially deterministic grid benchmarks, with relational benchmarks and with logistics planning benchmarks. Here, we compared the accuracy of SampleSearch against three other schemes: the two generalized belief propagation schemes of Iterative Join Graph Propagation (Dechter et al., 2002) and Edge Deletion Belief Propagation (Choi and Darwiche, 2006) and an adaptive importance sampling scheme called Evidence Prepropagated Importance Sampling (EPIS) (Yuan and Druzdzel, 2006). Again, we found that except for the grid instances, SampleSearch consistently yields estimates having smaller error than the other schemes.

Based on this large scale experimental evaluation, we conclude that SampleSearch consistently yields very good approximations. In particular, on large instances which have a substantial amount of determinism, SampleSearch yields an order of magnitude improvement over state-of-the-art schemes.

The rest of the paper is organized as follows. In Section 2, we present notation and preliminaries on graphical models and importance sampling. In Section 3, we present the rejection problem and show how to overcome it using the backtrack-free distribution. Section 4 describes the SampleSearch scheme and various improvements. In Section 5,

we present experimental results and we conclude in Section 6. The paper is based on earlier conference papers (Gogate and Dechter, 2007a,b).

2. Preliminaries and Background

We denote variables by upper case letters (e.g. $X, Y, \ldots$) and values of variables by lower case letters (e.g. $x, y, \ldots$). Sets of variables are denoted by bold upper case letters, (e.g. $\mathbf{X} = \{X_1, \ldots, X_n\}$) while sets of values are denoted by bold lower case letters (e.g. $\mathbf{x} = \{x_1, \ldots, x_n\}$). $X = x$ denotes an assignment of value to a variable while $\mathbf{X} = \mathbf{x}$ denotes an assignment of values to all variables in the set. We denote by $\mathbf{D}_i$ the set of possible values of X_i (also called as the domain of X_i). We denote the projection of an assignment $\mathbf{x}$ to a set $\mathbf{S} \subseteq \mathbf{X}$ by $\mathbf{x_S}$.

$\sum_{\mathbf{x}\in\mathbf{X}}$ denotes the sum over the possible values of variables in $\mathbf{X}$, namely, $\sum_{x_1\in X_1} \times \sum_{x_2\in X_2} \times \ldots \times \sum_{x_n\in X_n}$. The expected value $\mathbb{E}_Q[X]$ of a random variable X with respect to a distribution Q is defined as: $\mathbb{E}_Q[X] = \sum_{x\in X} xQ(x)$. The variance $V_Q[X]$ of X is defined as: $V_Q[X] = \sum_{x\in X}(x - \mathbb{E}_Q[X])^2$.

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

2.1. Bayesian, Constraint and Markov Networks

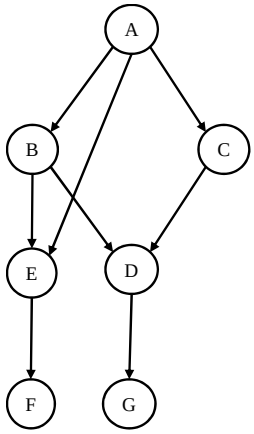
Definition 1 (**Graphical Models**). *A discrete graphical model $\mathcal{G}$ is a 3-tuple $\langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}\rangle$ where $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ is a finite set of variables, $\boldsymbol{D} = \{\boldsymbol{D}_1, \ldots, \boldsymbol{D}_n\}$ is a finite set of domains where $\boldsymbol{D}_i$ is the domain of variable X_i and $\boldsymbol{F} = \{F_1, \ldots, F_m\}$ is a finite set of discrete-valued functions. Each function F_i is defined over a subset $\boldsymbol{S}_i \subseteq \boldsymbol{X}$ of variables. The graphical model represents a product of all of its functions.*

Each graphical model is associated with a primal graph which captures the dependencies present in the model.

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

Definition 3 (**Bayesian or Belief Networks**). *A Bayesian network is a graphical model $\mathcal{B} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{G}, \boldsymbol{P}\rangle$ where $G = (\boldsymbol{X}, \boldsymbol{E})$ is a directed acyclic graph over the set of variables $\boldsymbol{X}$. The functions $P = \{P_1, \ldots, P_n\}$ are conditional probability tables $P_i = P(X_i|\boldsymbol{pa}_i)$, where $\boldsymbol{pa}_i = scope(P_i) \setminus \{X_i\}$ 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 $\boldsymbol{E} = \boldsymbol{e}$ is an instantiated subset of variables.*

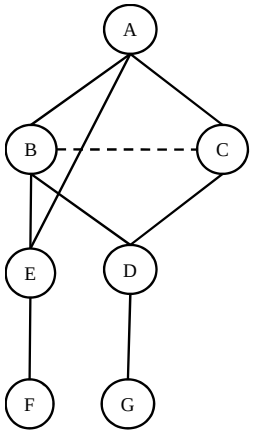
A Bayesian network represents the joint probability distribution given by $P_{\mathcal{B}}(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|\mathbf{pa}_i)$ and therefore can be used to answer any query defined over the joint distribution. In this paper, we consider two queries: (a) computing the probability of evidence $P(\mathbf{E} = \mathbf{e})$ and (b) computing the posterior marginal distribution $P(X_i|\mathbf{E} = \mathbf{e})$ for each variable $X_i \in \mathbf{X}$.



(a)

D	B	C	P(D|B,C)
0	0	0	0.2
1	0	0	0.8
0	0	1	0.7
1	0	1	0.3
0	1	0	0.1
1	1	0	0.9
0	1	1	0.6
1	1	1	0.4

(b)



(c)

Figure 1: (a) An example Bayesian network, (b) An example CPT $P(D|B,C)$ and (C) Moral graph of the Bayesian network shown in (a).

Example 1. *Figure 1 (a) shows an example Bayesian network over seven variables* $\{A, B, C, D, E, F, G\}$. *The network depicts structural relationships between different variables. The conditional probability tables (CPTs) associated with variables* A, B, C, D, E, F *and* G *are* $P(A)$, $P(B|A)$, $P(C|A)$, $P(D|B,C)$, $P(E|A,B)$, $P(F|E)$ *and* $P(G|D)$ *respectively. An example CPT for* $P(D|B,C)$ *is given in Figure 1(b). Figure 1(c) shows the network's moral graph which coincides with the primal graph. The Bayesian network represents the joint distribution:*

$$P(A,B,C,D,E,F,G) = P(A)P(B|A)P(C|A)P(D|B,C)P(E|A,B)P(F|E)P(G|D)$$

Definition 4 (Markov Networks). *A Markov network is a graphical model* $\mathcal{T} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{H} \rangle$ *where* $\boldsymbol{H} = \{H_1, \ldots, H_m\}$ *is a set of potential functions where each potential* H_i *is a non-negative real-valued function defined over subset* $\boldsymbol{S}_i$ *of variables. The Markov network represents a joint distribution over the variables* $\boldsymbol{X}$ *given by:*

$$P(\boldsymbol{X}) = \frac{1}{Z}\prod_{i=1}^{m} H_i(\boldsymbol{S}_i) \quad \textit{where} \quad Z = \sum_{\boldsymbol{x}\in\boldsymbol{X}}\prod_{i=1}^{m} H_i(\boldsymbol{x})$$

where the normalizing constant Z *is often referred to as the partition function.*

The primary queries over Markov networks are computing the posterior distribution (marginals) over all variables $X_i \in \mathbf{X}$ and finding the partition function.

Definition 5 (Constraint Networks). *A constraint network is a graphical model* $\mathcal{R} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{C} \rangle$ *where* $\boldsymbol{C} = \{C_1, \ldots, C_m\}$ *is a set of constraints. Each constraint* C_i *is a 0/1 function defined over a subset of variables* $\boldsymbol{S}_i$, *called its scope. Given an assignment* $\boldsymbol{S}_i = \boldsymbol{s}_i$, *a constraint is satisfied if* $C_i(\boldsymbol{s}_i) = 1$. *A constraint can also be expressed by a pair* $\langle R_i, \boldsymbol{S}_i \rangle$ *where* R_i *is a relation defined over the variables* $\boldsymbol{S}_i$ *and contains all tuples* $\boldsymbol{S}_i = \boldsymbol{s}_i$ *for which* $C_i(\boldsymbol{s}_i) = 1$. *The primal graph of a constraint network is called the constraint graph.*

The primary query over a constraint network is to decide whether it has a solution i.e. to find an assignment $\mathbf{X} = \mathbf{x}$ to all variables such that all constraints are satisfied or to prove that no such assignment exists. Another important query is that of counting the number of solutions of the constraint network.

Propositional Satisfiability

A special case of a constraint network is a *propositional satisfiability problem* (SAT). A propositional or Boolean formula F is an expression defined over variables having binary domains: $\{False, True\}$ or $\{0, 1\}$. Every Boolean formula can be converted into an equivalent formula in conjunctive normal form (CNF). A CNF formula F is a conjunction of *clauses* $Cl_1, \ldots, Cl_t$ (denoted as a set $\{Cl_1, \ldots, Cl_t\}$) where a clause is a disjunction (denoted by $\vee$) of *literals* (literals are variables or their negations). For example, $Cl = (P \vee \neg Q \vee \neg R)$ is a clause over three variables P, Q and R, and P, $\neg Q$ and $\neg R$ are literals. A clause is said to be satisfied if one of its literals is assigned the value $True$ or 1. A solution or a model of a formula F is an assignment of values to all variables such that all clauses are satisfied. Common queries in SAT are *satisfiability* i.e. finding a model or proving that none exists, and *model counting* i.e. counting the number of models or solutions.

2.2. Mixed Networks

Throughout the paper, we will use the framework of mixed networks defined in (Dechter and Mateescu, 2004; Mateescu and Dechter, 2009). Mixed networks represent all the deterministic information explicitly in the form of constraints facilitating the use of constraint processing techniques developed over the past three decades for efficient probabilistic inference. This framework includes Bayesian, Markov and constraint networks as a special case. Therefore, many inference tasks become equivalent when we consider a mixed network view allowing a unifying treatment of all these problems within a single framework. For example, problems such as computing the probability of evidence in a Bayesian network, the partition function in a Markov network and counting solutions of a constraint network can be expressed as weighted counting over mixed networks.

Definition 6 (**Mixed Network**). *(Dechter and Mateescu, 2004; Mateescu and Dechter, 2009) A mixed network is a four-tuple* $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}, \boldsymbol{C} \rangle$ *where* $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ *is a set of random variables,* $\boldsymbol{D} = \{\boldsymbol{D}_1, \ldots, \boldsymbol{D}_n\}$ *is a set of domains where* $\boldsymbol{D}_i$ *is the domain of* X_i*,* $\boldsymbol{F} = \{F_1, , \ldots, F_m\}$ *is a set of non-negative real valued functions where each* F_i *is defined over a subset of variables* $\boldsymbol{S}_i \subseteq \boldsymbol{X}$ *(its scope) and* $\boldsymbol{C} = \{C_1, \ldots, C_p\}$ *is a set of constraints (or* $0/1$ *functions). A mixed network represents a joint distribution over* $\boldsymbol{X}$ *given by:*

$$P_{\mathcal{M}}(\mathbf{x}) = \begin{cases} \frac{1}{Z} \prod_{i=1}^{m} F_i(\mathbf{x}) & \text{if } \boldsymbol{x} \in sol(\boldsymbol{C}) \\ 0 & \text{otherwise} \end{cases}$$

where $sol(\boldsymbol{C})$ *is the set of solutions of* $\boldsymbol{C}$ *and* $Z = \sum_{\boldsymbol{x} \in sol(\boldsymbol{C})} \prod_{i=1}^{m} F_i(\mathbf{x})$ *is the normalizing constant.*

The ***primal graph*** *of a mixed network has variables as its vertices and an edge between any two variables than appear in the scope of a function* $F \in \boldsymbol{F}$ *or a constraint* $C \in \boldsymbol{C}$*.*

We can define several queries over the mixed network. In this paper, however we will focus on the following two queries:

Definition 7 (**The Weighted Counting Task**). *Given a mixed network* $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}, \boldsymbol{C} \rangle$, *the weighted counting task is to compute the normalization constant given by:*

$$Z = \sum_{\mathbf{x} \in Sol(\mathbf{C})} \prod_{i=1}^{m} F_i(\mathbf{x}) \tag{1}$$

where $sol(\boldsymbol{C})$ *is the set of solutions of the constraint portion* $\boldsymbol{C}$ *of* $\mathcal{M}$. *Equivalently, if we represent the constraints in* $\boldsymbol{C}$ *as* $0/1$ *functions, we can rewrite* Z *as:*

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

We will refer to Z as **weighted counts**.

Definition 8 (**Marginal task**). *Given a mixed network* $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}, \boldsymbol{C} \rangle$, *the marginal task is to compute the marginal distribution at each variable. Namely, for each variable* X_i *and* $x_i \in \boldsymbol{D}_i$, *compute:*

$$P(x_i) = \sum_{\mathbf{x} \in \mathbf{X}} \delta_{x_i}(\mathbf{x}) P_{\mathcal{M}}(\mathbf{x}), \text{ where } \delta_{x_i}(\mathbf{x}) = \begin{cases} 1 & \text{if } X_i \text{ is assigned the value } x_i \text{ in } \boldsymbol{x} \\ 0 & \text{otherwise} \end{cases}$$

To be able to use the constraint portion of the mixed network more effectively, for the remainder of the paper, we require that *all zero probabilities in the mixed network are also represented as constraints.* It is easy to define such a network as we show below.

Definition 9 (**Modified Mixed network**). *Given a mixed network* $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}, \boldsymbol{C} \rangle$, *a modified mixed network is a four-tuple* $\mathcal{M}' = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}, \boldsymbol{C}' \rangle$ *where* $\boldsymbol{C}' = \boldsymbol{C} \cup \{H_i\}_{i=1}^{m}$ *where*

$$H_i(\mathbf{S}_i = \mathbf{s}_i) = \begin{cases} 0 & \text{if } F_i(\boldsymbol{s}_i) = 0 \\ 1 & \text{Otherwise} \end{cases} \tag{3}$$

H_i *can also be expressed as a relation. The set of constraints* $\boldsymbol{C}'$ *is called the* ***flat constraint network*** *of the probability distribution* $P_{\mathcal{M}}$.

Clearly, the modified mixed network M' and the original mixed network M are equivalent in that $P_{\mathcal{M}'} = P_{\mathcal{M}}$.

It is easy to see that the weighted counts over a mixed network specialize to (a) the probability of evidence in a Bayesian network, (b) the partition function in a Markov network and (c) the number of solutions of a constraint network. The marginal task expresses the task of computing posterior marginals in a Bayesian or Markov network.

2.3. Importance Sampling for approximating the weighted counts and marginals

Importance sampling (Marshall, 1956; Geweke, 1989) is a general Monte Carlo simulation technique which can be used for estimating various statistics of a given target distribution. Since it is often hard to sample from the target distribution, the main idea is to generate samples from another easy-to-simulate distribution Q called the proposal (or trial or importance) distribution and then estimate various statistics over the target distribution by a weighted sum over the samples. The weight of a sample is the ratio between the probability of generating the sample from the target distribution and its probability based on the proposal distribution. In this subsection, we describe how the weighted counts and posterior marginals can be approximated via importance sampling. We first describe how to generate samples from Q followed by some preliminaries on statistical estimation theory.

We assume throughout the paper that the proposal distribution is specified in the product form along a variable ordering $o = (X_1, \ldots, X_n)$ as:

$$Q(\mathbf{X}) = \prod_{i=1}^{n} Q_i(X_i|X_1, \ldots, X_{i-1}).$$

Q is therefore specified as a Bayesian network with CPTs $Q = \{Q_1, \ldots, Q_n\}$ along the ordering o. We can generate a full sample from this product form specification as follows. For $i = 1$ to n, sample $X_i = x_i$ from the conditional distribution $Q(X_i|X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$ and set $X_i = x_i$. This is often referred to as an *ordered Monte Carlo sampler* or logic sampling (Pearl, 1988).

Thus, when we say that Q is easy to sample from, we assume that Q can be expressed in a product form and can be specified in polynomial space, namely,

$$Q(\mathbf{X}) = \prod_{i=1}^{n} Q_i(X_i|X_1, \ldots, X_{i-1}) = \prod_{i=1}^{n} Q_i(X_i|\mathbf{Y_i}) \tag{4}$$

where $\mathbf{Y}_i \subseteq \{X_1, \ldots, X_{i-1}\}$. The size of the set $\mathbf{Y}_i$ is assumed to be bounded by a constant.

Definition 10 (**An estimator**). *An estimator is a function of data (or samples) that produces an estimate for an unknown parameter or statistics of the distribution that produced the data (or samples).*

Definition 11 (**Unbiased and Asymptotically Unbiased Estimator**). *Given a probability distribution Q and a statistics θ of Q, an estimator $\widehat{\theta_N}$ which is based on N random samples drawn from Q, is an unbiased estimator of θ if $\mathbb{E}_Q[\widehat{\theta_N}] = \theta$. Similarly, an estimator $\widetilde{\theta_N}$ which is based on N random samples drawn from Q, is an asymptotically unbiased estimator of θ if $\lim_{N\to\infty} \mathbb{E}_Q[\widetilde{\theta_N}] = \theta$. Clearly, all unbiased estimators are asymptotically unbiased.*

Note that we will denote an unbiased estimator of a statistics θ by $\widehat{\theta}$, an asymptotically unbiased estimator by $\widetilde{\theta}$ and an arbitrary estimator by $\overline{\theta}$.

The notion of unbiasedness and asymptotic unbiasedness is important because it helps to characterize the performance of an estimator which we explain briefly below (for more details see (Rubinstein, 1981)). The mean-squared error of an estimator $\overline{\theta}$ is given by:

$$\begin{aligned} MSE(\overline{\theta}) &= \mathbb{E}_Q[(\overline{\theta} - \theta)^2] && (5) \\ &= \mathbb{E}_Q[\overline{\theta}^2] - 2\mathbb{E}_Q[\overline{\theta}]\theta + \theta^2 && (6) \\ &= \left[\mathbb{E}_Q[\overline{\theta}^2] - \mathbb{E}_Q[\overline{\theta}]^2\right] + \left[\mathbb{E}_Q[\overline{\theta}]^2 - 2\mathbb{E}_Q[\overline{\theta}]\theta + \theta^2\right] && (7) \end{aligned}$$

The bias of $\overline{\theta}$ is given by:

$$B_Q[\overline{\theta}] = \mathbb{E}_Q[\overline{\theta}] - \theta$$

The variance of $\overline{\theta}$ is given by:

$$V_Q[\overline{\theta}] = \mathbb{E}_Q[\overline{\theta}^2] - \mathbb{E}_Q[\overline{\theta}]^2$$

From the definitions of bias, variance and mean-squared error, we get:

$$MSE(\overline{\theta}) = V_Q[\overline{\theta}] + \left[B_Q[\overline{\theta}]\right]^2 \tag{8}$$

In other words, the mean squared error of an estimator is equal to bias squared plus variance (Rubinstein, 1981). For an unbiased estimator, the bias is zero and therefore one can reduce its mean squared error by reducing its variance. In case of an asymptotically unbiased estimator, the bias goes to zero as the number of samples tend to infinity. However, for a finite sample size it may have a non-zero bias. Although in principle an unbiased estimator seems to be better than an asymptotically unbiased estimator, the latter may have lower MSE than the former because it may have lower variance.

2.3.1. Estimating weighted counts

Consider the expression for weighted counts (see Definition 7).

$$Z = \sum_{\mathbf{x}\in\mathbf{X}} \prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x}) \tag{9}$$

If we have a proposal distribution $Q(\mathbf{X})$ such that $\prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x}) > 0 \rightarrow Q(\mathbf{x}) > 0$, we can rewrite Equation 9 as follows:

$$Z = \sum_{\mathbf{x}\in\mathbf{X}} \frac{\prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})}{Q(\mathbf{x})} Q(\mathbf{x}) = \mathbb{E}_Q\left[\frac{\prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})}{Q(\mathbf{x})}\right] \tag{10}$$

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

$$\widehat{Z}_N = \frac{1}{N} \sum_{k=1}^{N} \frac{\prod_{i=1}^{m} F_i(\mathbf{x}^k) \prod_{j=1}^{p} C_j(\mathbf{x}^k)}{Q(\mathbf{x}^k)} = \frac{1}{N} \sum_{k=1}^{N} w(\mathbf{x}^k) \tag{11}$$

where

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

is the weight of sample $\mathbf{x}^k$. By definition, the variance of the weights is given by:

$$V_Q[w(\mathbf{x})] = \sum_{\mathbf{x}\in\mathbf{X}} \left(w(\mathbf{x}) - Z\right)^2 Q(\mathbf{x}) \tag{12}$$

We can estimate the variance of $\widehat{Z}_N$ by (see for example (Rubinstein, 1981)):

$$\widehat{V_Q}[\widehat{Z}_N] = \frac{1}{N(N-1)} \sum_{k=1}^{N} \left(w(\mathbf{x}^k) - \widehat{Z}_N\right)^2 \tag{13}$$

and it can be shown that $\widehat{V_Q}[\widehat{Z}_N]$ is an unbiased estimator of $V_Q[\widehat{Z}_N]$, namely,

$$\mathbb{E}_Q[\widehat{V_Q}[\widehat{Z}_N]] = V_Q[\widehat{Z}_N]$$

We can show that (Rubinstein, 1981):

1. $\mathbb{E}_Q[\widehat{Z}_N] = Z$ i.e. $\widehat{Z}_N$ is *unbiased*.
2. $\lim_{N\to\infty} \widehat{Z}_N = Z$, with probability 1 (follows from the central limit theorem).
3. $\mathbb{E}_Q\left[\widehat{V_Q}[\widehat{Z}_N]\right] = V_Q[\widehat{Z}_N] = V_Q[w(\mathbf{x})]/N$

Therefore, $V_Q[\widehat{Z}_N]$ can be reduced by either increasing the number of samples N or by reducing the variance of the weights. It is easy to see that if $Q \propto \prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})$, then for any sample $\mathbf{x}$, we have $w(\mathbf{x}) = Z$ yielding an optimal (zero variance) estimator. However, making $Q \propto \prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})$ is NP-hard and therefore in order to have a small MSE in practice, it is recommended that Q must be as "close" as possible to the function it tries to approximate which in our case is $\prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})$ (Rubinstein, 1981; Liu, 2001).

2.3.2. Estimating the marginals

The marginal problem is defined as:

$$P(x_i) = \sum_{\mathbf{x}\in\mathbf{X}} \delta_{x_i}(\mathbf{x}) P_{\mathcal{M}}(\mathbf{x}) \tag{14}$$

where $P_{\mathcal{M}}$ is defined by:

$$P_{\mathcal{M}}(\mathbf{x}) = \frac{1}{Z} \prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x}) \tag{15}$$

Given a proposal distribution $Q(\mathbf{x})$ satisfying $P_{\mathcal{M}}(\mathbf{x}) > 0 \rightarrow Q(\mathbf{x}) > 0$, we can rewrite Equation 14 as follows:

$$P(x_i) = \sum_{\mathbf{x}\in\mathbf{X}} \frac{\delta_{x_i}(\mathbf{x}) P_{\mathcal{M}}(\mathbf{x})}{Q(\mathbf{x})} Q(\mathbf{x}) = \mathbb{E}_Q\left[\frac{\delta_{x_i}(\mathbf{x}) P_{\mathcal{M}}(\mathbf{x})}{Q(\mathbf{x})}\right] \tag{16}$$

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

$$\widehat{P_N}(x_i) = \frac{1}{N}\sum_{k=1}^{N}\frac{\delta_{x_i}(\mathbf{x}^k)P_{\mathcal{M}}(\mathbf{x}^k)}{Q(\mathbf{x}^k)} = \frac{1}{N}\sum_{k=1}^{N}\frac{\delta_{x_i}(\mathbf{x}^k)\prod_{i=1}^{m}F_i(\mathbf{x}^k)\prod_{j=1}^{p}C_j(\mathbf{x}^k)}{ZQ(\mathbf{x}^k)} \tag{17}$$

Unfortunately, Equation 17, while an unbiased estimator of $P(x_i)$ cannot be evaluated because Z is not known. We can sacrifice unbiasedness and estimate $P(x_i)$ by properly weighted samples (Liu, 2001).

Definition 12 (**A Properly weighted sample**). *A set of weighted samples $\{\boldsymbol{x}^k, w(\boldsymbol{x}^k)\}_{k=1}^N$ drawn from a distribution G are said to be properly weighted with respect to a distribution P if for any discrete function H,*

$$\mathbb{E}_G[H(\mathbf{x}^k)w(\mathbf{x}^k)] = c\mathbb{E}_P[H(\mathbf{x})]$$

where c is a normalization constant common to all samples.

Given the set of weighted samples, we can estimate $\mathbb{E}_P[H(\mathbf{x})]$ as:

$$\widetilde{\mathbb{E}}_P[H(\mathbf{x})] = \frac{\sum_{k=1}^{N}H(\mathbf{x}^k)w(\mathbf{x}^k)}{\sum_{k=1}^{N}w(\mathbf{x}^k)}$$

Substituting Equation 15 in Equation 16, we have:

$$P(x_i) = \frac{1}{Z}\mathbb{E}_Q\left[\frac{\delta_{x_i}(\mathbf{x})\prod_{i=1}^{m}F_i(\mathbf{x})\prod_{j=1}^{p}C_j(\mathbf{x})}{Q(\mathbf{x})}\right] \tag{18}$$

It is easy to prove that (Liu, 2001):

Proposition 1. *Given $w(\boldsymbol{x}) = \frac{\delta_{x_i}(\boldsymbol{x})\prod_{i=1}^{m}F_i(\boldsymbol{x})\prod_{j=1}^{p}C_j(\boldsymbol{x})}{Q(\boldsymbol{x})}$, the set of weighted samples $\{\boldsymbol{x}^k, w(\boldsymbol{x}^k)\}_{k=1}^N$ are properly weighted with respect to $P_{\mathcal{M}}$.*

Therefore, we can estimate $P(x_i)$ by:

$$\widetilde{P_N}(x_i) = \frac{\sum_{k=1}^{N}w(\mathbf{x}^k)\delta_{x_i}(\mathbf{x}^k)}{\sum_{k=1}^{N}w(\mathbf{x}^k)} \tag{19}$$

It is easy to prove that $\lim_{N\to\infty}\mathbb{E}[\widetilde{P_N}(x_i)] = P(x_i)$ i.e. it is *asymptotically unbiased*. Therefore, by weak law of large numbers the sample average $\widetilde{P_N}(x_i)$ converges almost surely to $P(x_i)$ as $N \to \infty$. Namely,

$$\lim_{N\to\infty}\widetilde{P_N}(x_i) = P(x_i) \quad \text{, with probability 1 (from the weak law of large numbers)}$$

Also it was shown in (Liu, 2001) that in order to have small estimation error, the proposal distribution Q should be as close as possible to the target distribution $P_{\mathcal{M}}$.

3. Eliminating the Rejection Problem using the Backtrack-free distribution

In this section, we describe the rejection problem and show that the problem can be mitigated by modifying the proposal distribution. Given a mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$, a proposal distribution Q defined over $\mathbf{X}$ suffers from the rejection problem if the probability of generating a sample from Q that violates the constraints of $P_{\mathcal{M}}$ expressed in $\mathbf{C}$ is relatively high. When a sample $\mathbf{x}$ violates some constraints in $\mathbf{C}$, its weight $w(\mathbf{x})$ is zero and it is effectively rejected from the sample average. In an extreme case, if the probability of generating a rejected sample is arbitrarily close to one, then even after generating a large number of samples, the estimate of the weighted counts (given by Equation 11) would be zero and the estimate of the marginals (given by Equation 19) would be ill-defined. Clearly, if Q properly encodes all the zeros in $\mathcal{M}$, then we would have no rejection.

Definition 13 (**Zero Equivalence**). *A distribution P is zero equivalent to a distribution P', iff their flat constraint networks (see Definition 9) are equivalent. Namely, they have the same set of consistent solutions.*

Clearly then, given a mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$ representing $P_{\mathcal{M}}$ and given a proposal distribution $Q = \{Q_1, \ldots, Q_n\}$ which is zero equivalent to $P_{\mathcal{M}}$, every sample $\mathbf{x}$ generated from Q satisfies $P_{\mathcal{M}}(\mathbf{x}) > 0$ and no sample generated from Q would be rejected.

Because Q is expressed in a product form: $Q(\mathbf{X}) = \prod_{i=1}^{n} Q_i(X_i|\ X_1,\ \ldots,\ X_{i-1})$ along $o = (X_1, \ldots, X_n)$, we can make Q zero equivalent to $P_{\mathcal{M}}$ by modifying its components $Q_i(X_i|X_1, \ldots, X_{i-1})$ along o. To accomplish that, we have to make the set $Q = \{Q_1, \ldots, Q_n\}$ backtrack-free along o relative to the constraints in $\mathbf{C}$. The following definitions formalize this notion.

Definition 14 (**consistent and globally consistent partial sample**). *Given a set of constraints $\boldsymbol{C}$ defined over $\boldsymbol{X} = \{X_1, \ldots, X_n\}$, a partial sample $(x_1, \ldots, x_i)$ is consistent if it does not violate any constraint in $\boldsymbol{C}$. A partial sample $(x_1, \ldots, x_i)$ is globally consistent if it can be extended to a solution of $\boldsymbol{C}$ (i.e. it can be extended to a full assignment to all n variables that satisfies all constraints in $\boldsymbol{C}$).*

Note that a consistent partial sample may not be globally consistent.

Definition 15 (**Backtrack-free distribution of Q w.r.t. $\mathbf{C}$**). *Given a mixed network $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}, \boldsymbol{C} \rangle$ and a proposal distribution $Q = \{Q_1, \ldots, Q_n\}$ representing $Q(\boldsymbol{X}) = \prod_{i=1}^{n} Q_i(X_i|\ X_1,\ \ldots,\ X_{i-1})$ along an ordering o, the backtrack-free distribution $Q^F = \{Q_1^F, \ldots, Q_n^F\}$ of Q along o w.r.t. $\boldsymbol{C}$ where $Q^F(\boldsymbol{X}) = \prod_{i=1}^{n} Q_i^F(X_i|X_1, \ldots, X_{i-1})$ is defined by:*

$$Q_i^F(x_i|x_1, \ldots, x_{i-1}) \begin{cases} = \alpha Q_i(x_i|x_1, \ldots, x_{i-1}) & \text{if } (x_1, \ldots, x_i) \text{ is globally consistent w.r.t } \boldsymbol{C} \\ = 0 & \text{otherwise.} \end{cases}$$

where α is a normalization constant.

Let $\mathbf{x}_{i-1} = (x_1, \ldots, x_{i-1})$ and define the set $\mathbf{B}_i^{\mathbf{x}_{i-1}} = \{x_i' \in \mathbf{D}_i | (x_1, \ldots, x_{i-1}, x_i')$ is not globally consistent w.r.t. $\mathbf{C}$ $\}$. Then, α can be expressed by:

$$\alpha = \frac{1}{1 - \sum_{x_i' \in \mathbf{B}_i^{\mathbf{x}_{i-1}}} Q_i(x_i'|x_1, \ldots, x_{i-1})}$$

Algorithm 1: Sampling from the Backtrack-free distribution

Input: A mixed network $\mathcal{M} = \langle \mathbf{X,D,F,C} \rangle$, a proposal distribution Q along an ordering o and an oracle

Output: A full sample $(x_1, \ldots, x_n)$ from the backtrack free distribution Q^F of Q

```
x = φ;
for i=1 to n do
    Q_i^F(X_i|x) = Q_i(X_i|x);
    for each value x_i ∈ D_i do
        y = x ∪ x_i;
        if oracle says that y is not globally consistent w.r.t C then
            Q_i^F(x_i|x) = 0 ;
    Normalize Q_i^F(X_i|x) and generate a sample X_i = x_i from it;
    x = x ∪ x_i;
return x
```

We borrow the term *backtrack-free* from the constraint satisfaction literature (Freuder, 1982; Dechter, 2003). An order o is said to be backtrack-free w.r.t. a set of constraints $\mathbf{C}$ if it guarantees that no inconsistent partial assignment would be generated along o (i.e. every sample generated would not be rejected). By definition, a proposal distribution $Q = \{Q_1, \ldots, Q_n\}$ is backtrack-free along o w.r.t. its flat constraint network (see Definition 9). The modification of the proposal distribution defined in Definition 15 takes a proposal distribution that is backtrack-free relative to itself and modifies its components to yield a distribution that is backtrack-free relative to $P_{\mathcal{M}}$.

Given a mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$ and a proposal distribution $Q = \{Q_1, \ldots, Q_n\}$ along o, we now show how to generate samples from the backtrack-free distribution $Q^F = \{Q_1^F, \ldots, Q_n^F\}$ of Q w.r.t. $\mathbf{C}$. Algorithm 1 assumes that we have an oracle which takes a partial assignment $(x_1, \ldots, x_i)$ and a constraint satisfaction problem $\langle \mathbf{X}, \mathbf{D}, \mathbf{C} \rangle$ as input and answers "yes" if the assignment is globally consistent and "no" otherwise. Given a partial assignment $(x_1, \ldots, x_{i-1})$, the algorithm constructs $Q_i^F(X_i|x_1, \ldots, x_{i-1})$ and samples a value for X_i as follows. $Q_i^F(X_i|x_1, \ldots, x_{i-1})$ is initialized to $Q_i(X_i|x_1, \ldots, x_{i-1})$. Then, for each assignment $(x_1, \ldots, x_{i-1}, x_i)$ extending to $X_i = x_i$, it checks whether $(x_1, \ldots, x_{i-1}, x_i)$ is globally consistent relative to $\mathbf{C}$ using the oracle. If not, it sets $Q_i^F(x_i|x_1, \ldots, x_{i-1})$ to zero, normalizes $Q_i^F(x_i|x_1, \ldots, x_{i-1})$ and generates a sample from it. Repeating this process along the order $(X_1, \ldots, X_n)$ yields a single sample from Q^F. Note that for each sample, the oracle should be invoked a maximum of $O(n \times d)$ times where n is the number of variables and d is the maximum domain size.

Given samples $(\mathbf{x}^1, \ldots, \mathbf{x}^N)$ generated from Q^F, we can estimate Z (defined in Equation 2) by replacing Q by Q^F in Equation 11. We get:

$$\widehat{Z}_N = \frac{1}{N} \sum_{k=1}^{N} \frac{\prod_{i=1}^{m} F_i(\mathbf{x}^k) \prod_{j=1}^{p} C_j(\mathbf{x}^k)}{Q^F(\mathbf{x}^k)} = \frac{1}{N} \sum_{k=1}^{N} w^F(\mathbf{x}^k) \tag{20}$$

where

$$w^F(\mathbf{x}) = \frac{\prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})}{Q^F(\mathbf{x})} \tag{21}$$

is the backtrack-free weight of the sample.

Similarly, we can estimate the posterior marginals by replacing the weight $w(\mathbf{x})$ in Equation 19 with the backtrack-free weight $w^F(\mathbf{x})$.

$$\widetilde{P_N}(x_i) = \frac{\sum_{k=1}^{N} w^F(\mathbf{x}^k)\delta_{x_i}(\mathbf{x}^k)}{\sum_{k=1}^{N} w^F(\mathbf{x}^k)} \tag{22}$$

Clearly, $\widehat{Z}_N$ defined in Equation 20 is an unbiased estimate of Z while $\widetilde{P}_N(x_i)$ defined in Equation 22 is an asymptotically unbiased estimate of the posterior marginals $P(x_i)$.

In practice, one could use any constraint solver as a substitute for the oracle in Algorithm 1. However, generating samples using an exact solver would be inefficient in many cases. Next, we present the SampleSearch scheme which integrates backtracking search with sampling. In essence, we integrate more naturally sampling with a specific oracle that is based on systematic backtracking search, hopefully, generating a more efficient scheme.

4. The SampleSearch Scheme

In a nutshell, SampleSearch incorporates systematic backtracking search into the ordered Monte Carlo sampler so that all full samples are solutions of the constraint portion of the mixed network but it does not insist on backtrack-freeness of the search process. We will sketch our ideas using the most basic form of systematic search: chronological backtracking, emphasizing that the scheme can work with any advanced systematic search scheme. In our empirical work, we will indeed use advanced search schemes such as minisat (Sorensson and Een, 2005).

Given a mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$ and a proposal distribution $Q(\mathbf{X})$, the ordered Monte Carlo sampler samples variables along the order $o = (X_1, \ldots, X_n)$ from Q and rejects a partial sample $(x_1, \ldots, x_i)$ if it violates any constraints in $\mathbf{C}$. Upon rejecting a sample, the sampler starts sampling anew from the first variable (X_1) in the ordering. Instead, when there is a dead-end at $(x_1, \ldots, x_{i-1}, x_i)$ SampleSearch modifies the conditional probability as $Q_i(X_i = x_i | x_1, \ldots, x_{i-1}) = 0$ to reflect that $(x_1, \ldots, x_i)$ is not consistent, normalizes the distribution $Q_i(X_i | x_1, \ldots, x_{i-1})$ and re-samples X_i from the normalized distribution. The newly sampled value may be consistent in which case the algorithm proceeds to variable X_{i+1} or it may be inconsistent. If we repeat the process we may reach a point where $Q_i(X_i | x_1, \ldots, x_{i-1})$ is 0 for all values of X_i. In this case, $(x_1, \ldots, x_{i-1})$ is inconsistent and therefore the algorithm revises the distribution at X_{i-1} by setting $Q_{i-1}(X_{i-1} = x_{i-1} | x_1, \ldots, x_{i-2}) = 0$, normalizes Q_{i-1} and re-samples a new value for X_{i-1} and so on. SampleSearch repeats this process until a consistent full sample that satisfies all constraints in $\mathbf{C}$ is generated. By construction, this process always yields a consistent full sample.

The pseudo-code for SampleSearch is given in Algorithm 2. It can be viewed as a depth first backtracking search (DFS) over the state space of consistent partial assignments searching for a solution to a constraint satisfaction problem $\langle \mathbf{X}, \mathbf{D}, \mathbf{C} \rangle$, whose

Algorithm 2: SampleSearch

Input: A mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$, the proposal distribution $Q(\mathbf{X}) = \prod_{i=1}^{n} Q_i(X_i|X_1, \ldots, X_{i-1})$ along an ordering $o = (X_1, \ldots, X_n)$

Output: A consistent full sample $\mathbf{x} = (x_1, \ldots, x_n)$

SET i=1, $D'_i = D_i$ (copy domains), $Q'_1(X_1) = Q_1(X_1)$ (copy distribution), $\mathbf{x} = \emptyset$;
while $1 \leq i \leq n$ **do**
    `// Forward phase`
    if D'_i *is not empty* **then**
        Sample $X_i = x_i$ from Q'_i and remove it from D'_i;
        if $(x_1, \ldots, x_i)$ *violates any constraint in* $\boldsymbol{C}$ **then**
            SET $Q'_i(X_i = x_i|x_1, \ldots, x_{i-1}) = 0$ and normalize Q'_i;
            Goto step 3.;
        $\mathbf{x} = \mathbf{x} \cup x_i$, $i = i + 1$, $D'_i = D_i$, $Q'_i(X_i|x_1, \ldots, x_{i-1}) = Q_i(X_i|x_1, \ldots, x_{i-1})$;
    `// Backward phase`
    else
        $\mathbf{x} = \mathbf{x} \backslash x_{i-1}$.;
        SET $Q'_{i-1}(X_{i-1} = x_{i-1}|x_1, \ldots, x_{i-2}) = 0$ and normalize $Q'_{i-1}(X_{i-1}|x_1, \ldots, x_{i-2})$;
        SET $i = i - 1$;
if $i = 0$ **then**
    return inconsistent;
else
    return $\mathbf{x}$;

value ordering is stochastically guided by Q. The updated distribution that guides the search is Q'. In the forward phase, variables are sampled in sequence and a current partial sample (or assignment) is extended by sampling a value x_i for the next variable X_i using the current distribution Q'_i. If for all values $x_i \in \mathbf{D}_i$, $Q'_i(x_i|x_1, \ldots, x_{i-1}) = 0$, then SampleSearch backtracks to the previous variable X_{i-1} (backward phase) and updates the distribution Q'_{i-1} by setting $Q'_{i-1}(x_{i-1}|x_1, \ldots, x_{i-2}) = 0$ and normalizing Q'_{i-1} and continues.

4.1. The Sampling Distribution of SampleSearch

Let $I = \prod_{i=1}^{n} I_i(X_i|X_1, \ldots, X_{i-1})$ be the sampling distribution of SampleSearch along the ordering $o = (X_1, \ldots, X_n)$. We will show that:

Theorem 1 (**Main Result**). *Given a mixed network $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}, \boldsymbol{C} \rangle$ and a proposal distribution Q, the sampling distribution I of SampleSearch coincides with the backtrack-free probability distribution Q^F of Q w.r.t. $\boldsymbol{C}$, i.e. $\forall\ i\ Q_i^F = I_i$.*

To prove this theorem, we need the following proposition:

Proposition 2. *Given a mixed network $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}, \boldsymbol{C} \rangle$, a proposal distribution $Q = \{Q_1, \ldots, Q_n\}$ and a partial assignment $(x_1, \ldots, x_{i-1})$ which is globally consistent w.r.t. $\boldsymbol{C}$, SampleSearch samples values without replacement from the domain $\boldsymbol{D}_i$ of X_i until a globally consistent extension $(x_1, \ldots, x_{i-1}, x_i)$ is generated.*

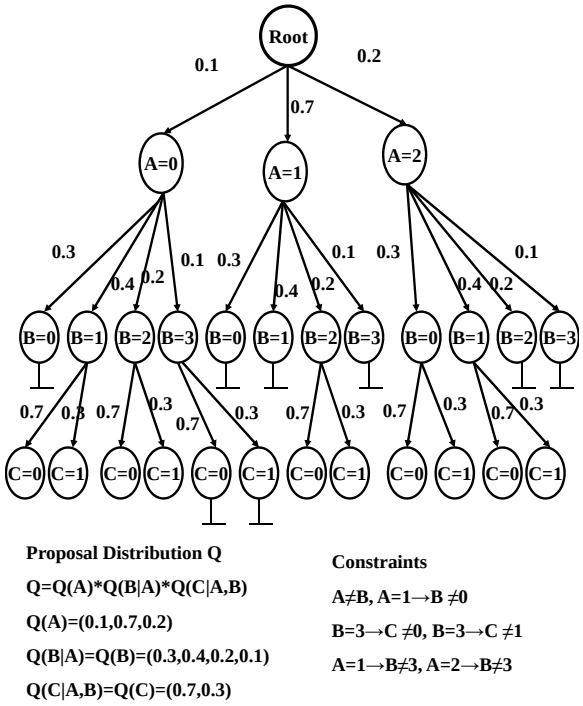


Figure 2: A full OR search tree given a set of constraints and a proposal distribution.

Proof. Consider a globally inconsistent extension $(x_1, \ldots, x_{i-1}, x'_i)$ of $(x_1, \ldots, x_{i-1})$. Let $Q'_i(X_i|x_1, \ldots, x_{i-1})$ be the most recently updated proposal distribution. Because SampleSearch is systematic, if $(x_1, \ldots, x'_i)$ is sampled then SampleSearch would eventually detect its inconsistency by not being able to extend it to a solution. At this point, it will set $Q'_i(x'_i|x_1, \ldots, x_{i-1}) = 0$ either in step 6 or step 11 and normalize Q'_i. In other words, x'_i is sampled just once yielding sampling without replacement from $Q'_i(X_i|x_1, \ldots, x_{i-1})$. On the other hand, again because of its systematic nature, if a globally consistent extension $(x_1, \ldots, x_i)$ is sampled, SampleSearch will always extend it to a full sample that is consistent. □

We can use Proposition 2 to derive $I_i(x_i|x_1, \ldots, x_{i-1})$, the probability of sampling a globally consistent extension $(x_1, \ldots, x_{i-1}, x_i)$ to a globally consistent assignment $(x_1, \ldots, x_{i-1})$ from $Q_i(X_i|x_1, \ldots, x_{i-1})$ as illustrated in the next example (Example 2).

Example 2. *Consider the complete search tree corresponding to the proposal distribution and to the constraints given in Figure 2. The inconsistent partial assignments are grounded in the figure. Each arc is labeled with the probability of generating the child node from Q given an assignment from the root node to its parent. Consider the full assignment* $(A = 0, B = 2, C = 0)$*. Based on Proposition 2, the five different ways in which this assignment could be generated by SampleSearch (called as DFS-traces) are shown in Figure 3. In the following, we show how to compute the probability* $I_B(B = 2|A = 0)$ *i.e. the probability of sampling* $B = 2$ *given* $A = 0$*. Given* $A = 0$*, the events that could lead to sampling* $B = 2$ *are shown in Figure 3, (a)* $\langle B = 2\rangle|A = 0$ *(b)* $\langle B = 0, B = 2\rangle|A = 0$ *(c)* $\langle B = 3, B = 0\rangle|A = 0$ *(d)*

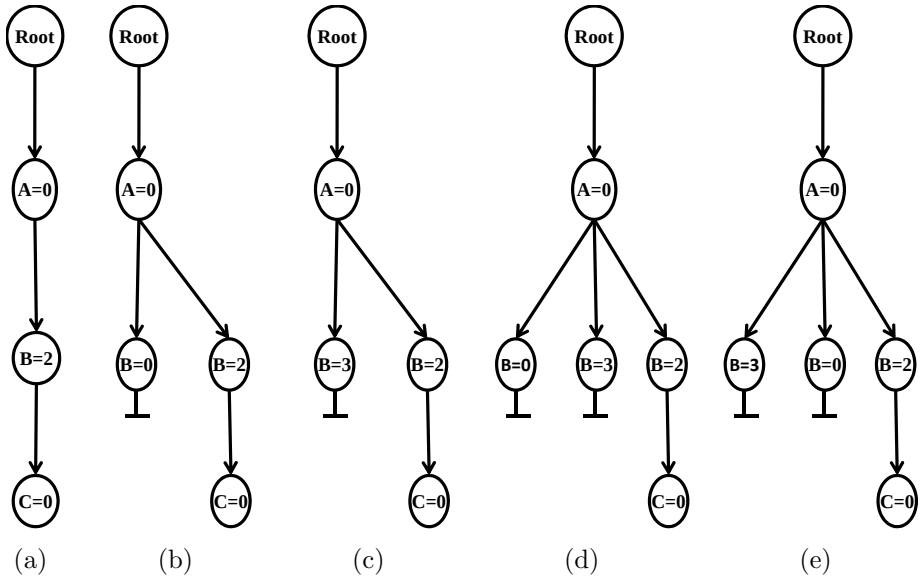


Figure 3: Five possible traces of SampleSearch which lead to the sample ($A = 0$, $B = 2$, $C = 0$). The children of each node are specified from left to right in the order in which they are generated.

$\langle B = 0, B = 3, B = 2\rangle | A = 0$ *and (e)* $\langle B = 3, B = 0, B = 2\rangle | A = 0$. *The notation* $\langle B = 3, B = 0, B = 2\rangle | A = 0$ *means that given* $A = 0$*, the states were sampled in the order from left to right* $(B = 3, B = 0, B = 2)$. *Clearly, the probability of* $I_B(B = 2|A = 0)$ *is the sum over the probability of these events. Let us now compute the probability of the event* $\langle B = 3, B = 0, B = 2\rangle | A = 0$. *The probability of sampling* $B = 3|A = 0$ *from* $Q(B|A = 0) = (0.3, 0.4, 0.2, 0.1)$ *is 0.1. The assignment* $(A = 0, B = 3)$ *is inconsistent and therefore the distribution* $Q(B|A = 0)$ *is changed by SampleSearch to* $Q'(B|A = 0) = (0.3/0.9, 0.4/0.9, 0.2/0.9, 0) = (3/9, 4/9, 2/9, 0)$. *Subsequently, the probability of sampling* $B = 0$ *from* Q' *is 3/9. However, the assignment* $(A = 0, B = 0)$ *is also globally inconsistent and therefore the distribution is changed to* $Q''(B|A = 0) \propto (0, 4/9, 2/9, 0) = (0, 2/3, 1/3, 0)$. *Next, the probability of sampling* $B = 2$ *from* Q'' *is* $1/3$. *Therefore, the probability of the event* $\langle B = 3, B = 0, B = 2\rangle | A = 0$ *is* $0.1 \times (3/9) \times (1/3) = 1/90$. *By calculating the probabilities of the remaining events using the approach described above and taking the sum, one can verify that the probability of sampling* $B = 2$ *given* $A = 0$ *i.e.* $I_B(B = 2|A = 0) = 1/3$.

We will now show that:

Proposition 3. *Given a mixed network* $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}, \boldsymbol{C}\rangle$*, an initial proposal distribution* $Q = \{Q_1, \ldots, Q_n\}$ *and a partial assignment* $(x_1, \ldots, x_{i-1}, x_i)$ *which is globally consistent w.r.t.* $\boldsymbol{C}$*, the probability* $I_i(x_i|x_1, \ldots, x_{i-1})$ *of sampling* x_i *given* $(x_1, \ldots, x_{i-1})$ *using SampleSearch is proportional to* $Q_i(x_i|x_1, \ldots, x_{i-1})$*, i.e.* $I_i(x_i|x_1, \ldots, x_{i-1}) \propto Q_i(x_i|x_1, \ldots, x_{i-1})$.

Proof. The proof is obtained by deriving a general expression for $I_i(x_i|x_1, \ldots, x_{i-1})$, summing the probabilities of all events that can lead to this desired partial sample. Consider a globally consistent partial assignment $\mathbf{x}_{i-1} = (x_1, \ldots, x_{i-1})$. Let us assume that the

domain of the next variable X_i given $\mathbf{x}_{i-1}$, denoted by $\mathbf{D}_i^{\mathbf{x}_{i-1}}$ is partitioned into $\mathbf{D}_i^{\mathbf{x}_{i-1}} = \mathbf{R}_i^{\mathbf{x}_{i-1}} \cup \mathbf{B}_i^{\mathbf{x}_{i-1}}$ where $\mathbf{R}_i^{\mathbf{x}_{i-1}} = \{x_i \in \mathbf{D}_i^{\mathbf{x}_{i-1}} | (x_1, \ldots, x_{i-1}, x_i)$ *is globally consistent*$\}$ and $\mathbf{B}_i^{\mathbf{x}_{i-1}} = \mathbf{D}_i^{\mathbf{x}_{i-1}} \setminus \mathbf{R}_i^{\mathbf{x}_{i-1}}$.

We introduce some notation. Let $\mathbf{B}_i^{\mathbf{x}_{i-1}} = \{x_{i,1}, \ldots, x_{i,q}\}$. Let $j = 1, \ldots, 2^q$ index the sequence of all subsets of $\mathbf{B}_i^{\mathbf{x}_{i-1}}$ with $\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}$ denoting the j-th element of this sequence. Let $\pi(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}})$ denote the sequence of all permutations of $\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}$ with $\pi_k(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}})$ denoting the k-th element of this sequence. Finally, let $Pr(\pi_k(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}), x_i | \mathbf{x}_{i-1})$ be the probability of generating x_i and $\pi_k(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}})$ given $\mathbf{x}_{i-1}$.

The probability of sampling $x_i \in \mathbf{R}_i^{\mathbf{x}_{i-1}}$ given $\mathbf{x}_{i-1}$ is obtained by summing over all the events that generate $X_i = x_i$ given $\mathbf{x}_{i-1}$:

$$I_i(x_i|\mathbf{x}_{i-1}) = \sum_{j=1}^{2^q} \sum_{k=1}^{|\pi(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}})|} Pr(\pi_k(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}), x_i | \mathbf{x}_{i-1}) \tag{23}$$

where, $Pr(\pi_k(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}), x_i | \mathbf{x}_{i-1})$ is given by:

$$Pr(\pi_k(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}), x_i | \mathbf{x}_{i-1}) = Pr(\pi_k(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}) | \mathbf{x}_{i-1}) Pr(x_i | \pi_k(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}), \mathbf{x}_{i-1}) \tag{24}$$

Substituting Equation 24 in Equation 23, we get:

$$I_i(x_i|\mathbf{x}_{i-1}) = \sum_{j=1}^{2^q} \sum_{k=1}^{|\pi(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}})|} Pr(\pi_k(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}) | \mathbf{x}_{i-1}) Pr(x_i | \pi_k(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}), \mathbf{x}_{i-1}) \tag{25}$$

where $Pr(x_i | \pi_k(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}), \mathbf{x}_{i-1})$ is the probability with which the value x_i is sampled given that $(\pi_k(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}), \mathbf{x}_{i-1})$ is proved inconsistent. Because, we sample without replacement (see Proposition 2) from Q_i, this probability is given by:

$$Pr(x_i | \pi_k(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}), \mathbf{x}_{i-1}) = \frac{Q_i(x_i|\mathbf{x}_{i-1})}{1 - \sum_{x'_i \in \mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}} Q_i(x'_i|\mathbf{x}_{i-1})} \tag{26}$$

From Equations 25 and 26, we get:

$$I_i(x_i|\mathbf{x}_{i-1}) = \sum_{j=1}^{2^q} \sum_{k=1}^{|\pi(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}})|} \frac{Q_i(x_i|\mathbf{x}_{i-1})}{1 - \sum_{x'_i \in \mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}} Q_i(x'_i|\mathbf{x}_{i-1})} Pr(\pi_k(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}) | \mathbf{x}_{i-1}) \tag{27}$$

$Q_i(x_i|\mathbf{x}_{i-1})$ does not depend on the indices j and k in Equation 27 and therefore we can rewrite Equation 27 as:

$$I_i(x_i|\mathbf{x}_{i-1}) = Q_i(x_i|\mathbf{x}_{i-1}) \left(\sum_{j=1}^{2^q} \sum_{k=1}^{|\pi(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}})|} \frac{Pr(\pi_k(\mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}) | \mathbf{x}_{i-1})}{1 - \sum_{x'_i \in \mathbf{B}_{i,j}^{\mathbf{x}_{i-1}}} Q_i(x'_i|\mathbf{x}_{i-1})} \right) \tag{28}$$

The term enclosed in brackets in Equation 28 does not depend on x_i and therefore it follows that if $(x_1, \ldots, x_{i-1}, x_i)$ is globally consistent:

$$I_i(x_i|\mathbf{x}_{i-1}) \propto Q_i(x_i|\mathbf{x}_{i-1}) \tag{29}$$

which is what we wanted to prove. □

We now have the necessary components to prove Theorem 1:

Proof of Theorem 1. From Proposition 2, $I_i(x_i|\mathbf{x}_{i-1})$ equals zero iff x_i is not globally consistent and from Proposition 3, for all other values, $I_i(x_i|\mathbf{x}_{i-1}) \propto Q_i(x_i|\mathbf{x}_{i-1})$. Therefore, the normalization constant equals $1 - \sum_{x'_i \in \mathbf{B}_i^{\mathbf{x}_{i-1}}} Q_i(x'_i|\mathbf{x}_{i-1})$. Consequently,

$$I_i(x_i|\mathbf{x}_{i-1}) = \frac{Q_i(x_i|\mathbf{x}_{i-1})}{1 - \sum_{x'_i \in \mathbf{B}_i^{\mathbf{x}_{i-1}}} Q_i(x'_i|\mathbf{x}_{i-1})} \tag{30}$$

The right hand side of Equation 30 is by definition equal to $Q_i^F(x_i|\mathbf{x}_{i-1})$ (see Definition 15). □

4.2. Computing $Q^F(\mathbf{x})$

Once we have the sample, we still need to compute the weights for estimating the marginals and the weighted counts, which in turn requires computing $Q_i^F(x_i|\mathbf{x}_{i-1})$. From Definition 15, we see that to compute the components $Q_i^F(x_i|\mathbf{x}_{i-1})$ for a sample $\mathbf{x} = (x_1, \ldots, x_n)$, we have to determine all values $x'_i \in \mathbf{D}_i$ which cannot be extended to a solution. One way to accomplish that, as described in Algorithm 1 is to use an oracle. The oracle should be invoked a maximum of $n \times (d-1)$ times where n is the number of variables and d is the maximum domain size. Methods such as adaptive consistency (Dechter, 2003) or an exact CSP solver can be used as oracles. But then, what have we gained by SampleSearch, if ultimately, we need to use the oracle almost the same number of times as the sampling method presented in Algorithm 1. Next, we will show how to approximate the backtrack-free probabilities on the fly while still maintaining some desirable guarantees.

4.2.1. Approximating $Q^F(\mathbf{x})$

During the process of generating the sample $\mathbf{x}$, SampleSearch may have discovered one or more values in the set $\mathbf{B}_i^{\mathbf{x}_{i-1}}$ and therefore we can build an approximation of $Q_i^F(x_i|\mathbf{x}_{i-1})$ as follows. Let $\mathbf{A}_i^{\mathbf{x}_{i-1}} \subseteq \mathbf{B}_i^{\mathbf{x}_{i-1}}$ be the set of values in the domain of X_i that were proved to be inconsistent given $\mathbf{x}_{i-1}$ while generating a sample $\mathbf{x}$. We use the set $\mathbf{A}_i^{\mathbf{x}_{i-1}}$ to compute an approximation $T_i^F(x_i|\mathbf{x}_{i-1})$ of $Q_i^F(x_i|\mathbf{x}_{i-1})$ as follows:

$$T_i^F(x_i|\mathbf{x}_{i-1}) = \frac{Q_i(x_i|\mathbf{x}_{i-1})}{1 - \sum_{x'_i \in \mathbf{A}_i^{\mathbf{x}_{i-1}}} Q_i(x'_i|\mathbf{x}_{i-1})} \tag{31}$$

Finally we compute $T^F(\mathbf{x}) = \prod_{i=1}^n T_i^F(x_i|\mathbf{x}_{i-1})$. However, $T^F(\mathbf{x})$ does not guarantee asymptotic unbiasedness when replacing $Q^F(\mathbf{x})$ for computing the weight $w^F(\mathbf{x})$ in Equation 21.

To remedy the situation, we can store each sample $(x_1, \ldots, x_n)$ and all its partial assignments $(x_1, \ldots, x_{i-1}, x'_i)$ that were proved inconsistent during each trace of an independent execution of SampleSearch called DFS-traces (for example, Figure 3 shows the five DFS-traces that could generate the sample $(A = 0, B = 2, C = 0)$). After executing SampleSearch N times generating N samples, we can use all the stored DFS-traces to compute an approximation of $Q^F(\mathbf{x})$ as illustrated in the following example.

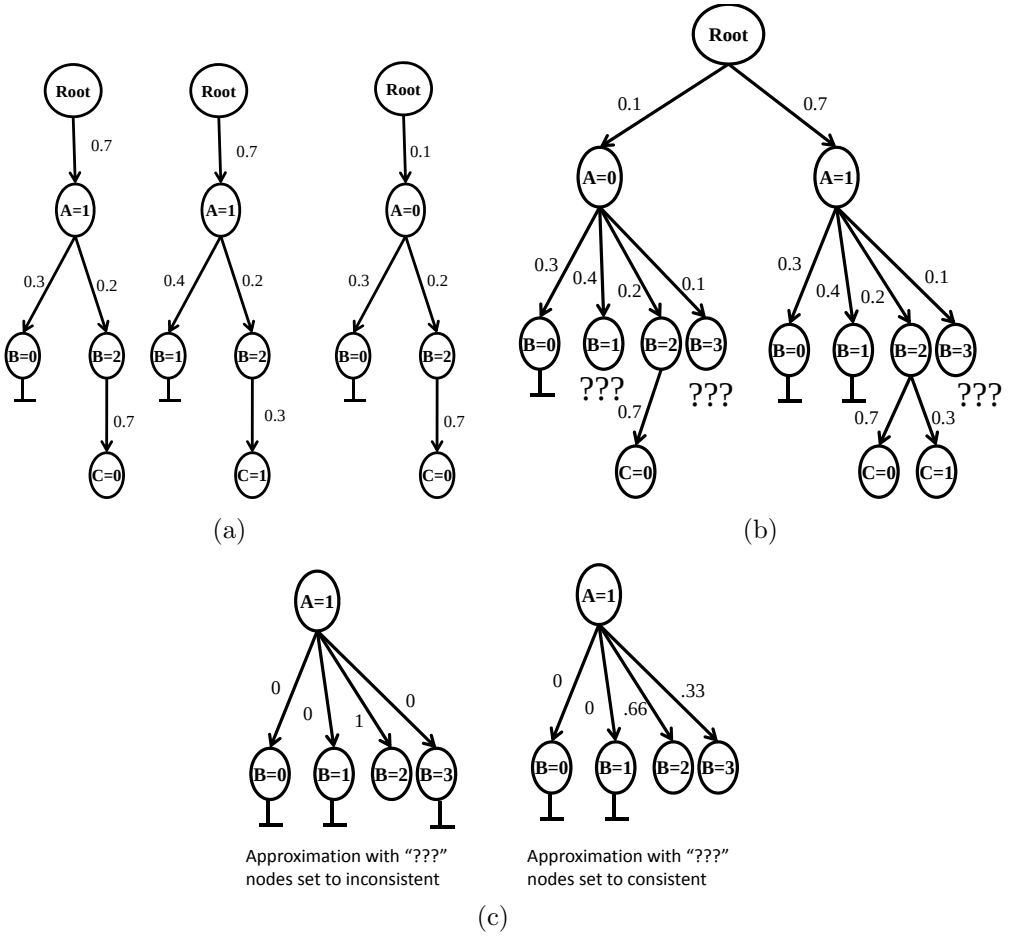


Figure 4: (a) Three DFS-traces (b) Combined information from the three DFS-traces given in (a) and (c) Two possible approximations of $I(B|A=1)$

Example 3. *Consider the three traces given in Figure 4 (a). We can combine the information from the three traces as shown in Figure 4(b). Consider the assignment* $(A = 1, B = 2)$*. The backtrack-free probability of generating* $B = 2$ *given* $A = 1$ *requires the knowledge of all the values of* B *which are inconsistent. Based on the combined traces, we know that* $B = 0$ *and* $B = 1$ *are inconsistent (given* $A = 1$*) but we do not know whether* $B = 3$ *is consistent or not because it is not explored (indicated by "???" in Figure 4(b)). Setting the unexplored nodes to either inconsistent or consistent gives us the two different approximations shown in Figure 4(c).*

Generalizing Example 3, we consider two bounding approximations denoted by U_N^F and L_N^F respectively which are based on setting each unexplored node in the combined N traces to consistent or inconsistent respectively. As we will show, these approximations can be used to bound the sample mean $\widehat{Z}_N$ from above and below[2].

Definition 16 (Upper and Lower Approximations of Q^F by U_N^F and L_N^F). *Given a mixed network* $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}, \boldsymbol{C} \rangle$*, an initial proposal distribution* $Q = \{Q_1, \ldots, Q_n\}$*, a combined sample tree generated from* N *independent runs of SampleSearch and a partial sample* $\boldsymbol{x}_{i-1} = (x_1, \ldots, x_{i-1})$ *generated in one of the* N *independent runs, we define two sets:*

- $\mathbf{A}_{N,i}^{\boldsymbol{x}_{i-1}} \subseteq \boldsymbol{B}_i^{\boldsymbol{x}_{i-1}} = \{x_i \in \boldsymbol{D}_i^{\boldsymbol{x}_{i-1}} |\ (x_1, \ldots, x_{i-1}, x_i)$ *was proved to be inconsistent during the* N *independent runs of SampleSearch* $\}$*.*
- $\mathbf{C}_{N,i}^{\boldsymbol{x}_{i-1}} \subseteq \mathbf{D}_i^{\boldsymbol{x}_{i-1}} = \{x_i \in \boldsymbol{D}_i^{\boldsymbol{x}_{i-1}} |\ (x_1, \ldots, x_{i-1}, x_i)$ *was not explored during the* N *independent runs of SampleSearch* $\}$*.*

We can set all the nodes in $\mathbf{C}_{N,i}^{\boldsymbol{x}_{i-1}}$ *(i.e. the nodes which are not explored) either to consistent or inconsistent yielding:*

$$U_N^F(\boldsymbol{x}) = \prod_{i=1}^{n} U_{N,i}^F(x_i|\mathbf{x}_{i-1}) \quad \textit{where}$$

$$U_{N,i}^F(x_i|\mathbf{x}_{i-1}) = \frac{Q_i(x_i|\mathbf{x}_{i-1})}{1 - \sum_{x_i' \in \mathbf{A}_{N,i}^{\boldsymbol{x}_{i-1}}} Q_i(x_i'|\mathbf{x}_{i-1})} \quad (32)$$

$$L_N^F(\boldsymbol{x}) = \prod_{i=1}^{n} L_{N,i}^F(x_i|\mathbf{x}_{i-1}) \quad \textit{where}$$

$$L_{N,i}^F(x_i|\mathbf{x}_{i-1}) = \frac{Q_i(x_i|\mathbf{x}_{i-1})}{1 - \sum_{x_i' \in \mathbf{A}_{N,i}^{\boldsymbol{x}_{i-1}} \cup \mathbf{C}_{N,i}^{\boldsymbol{x}_{i-1}}} Q_i(x_i'|\mathbf{x}_{i-1})} \quad (33)$$

[2]Note that it is easy to envision other approximations in which we designate some unexplored nodes as consistent while others as inconsistent based on the domain knowledge or via some other Monte Carlo estimate. We consider the two extreme options because they usually work well in practice and bound the sample mean from above and below.

It is clear that as N grows, the sample tree grows and therefore more inconsistencies will be discovered and as $N \to \infty$, all inconsistencies will be discovered making the respective sets approach $\mathbf{A}_{N,i}^{\mathbf{x}_{i-1}} = \mathbf{B}_i^{\mathbf{x}_{i-1}}$ and $\mathbf{C}_{N,i}^{\mathbf{x}_{i-1}} = \phi$. Clearly then,

Proposition 4. $\lim_{N\to\infty} U_N^F(\mathbf{x}) = \lim_{N\to\infty} L_N^F(\mathbf{x}) = Q^F(\mathbf{x})$

As before, given a set of i.i.d. samples $(\mathbf{x}^1 = (x_1^1, \ldots, x_n^1), \ldots, \mathbf{x}^N = (x_1^N, \ldots, x_n^N))$ generated by *SampleSearch*, we can estimate the weighted counts Z using the two statistics $U_N^F(\mathbf{x})$ and $L_N^F(\mathbf{x})$ by:

$$\widetilde{Z}_N^U = \frac{1}{N}\sum_{k=1}^{N} \frac{\prod_{i=1}^m F_i(\mathbf{x}^k)\prod_{j=1}^p C_j(\mathbf{x}^k)}{U_N^F(\mathbf{x}^k)} = \frac{1}{N}\sum_{k=1}^{N} w_N^U(\mathbf{x}^k) \tag{34}$$

where

$$w_N^U(\mathbf{x}^k) = \frac{\prod_{i=1}^m F_i(\mathbf{x}^k)\prod_{j=1}^p C_j(\mathbf{x}^k)}{U_N^F(\mathbf{x}^k)}$$

is the weight of the sample based on the combined sample tree using the upper approximation U_N^F.

$$\widetilde{Z}_N^L = \frac{1}{N}\sum_{k=1}^{N} \frac{\prod_{i=1}^m F_i(\mathbf{x}^k)\prod_{j=1}^p C_j(\mathbf{x}^k)}{L_N^F(\mathbf{x}^k)} = \frac{1}{N}\sum_{k=1}^{N} w_N^L(\mathbf{x}^k) \tag{35}$$

where

$$w_N^L(\mathbf{x}^k) = \frac{\prod_{i=1}^m F_i(\mathbf{x}^k)\prod_{j=1}^p C_j(\mathbf{x}^k)}{L_N^F(\mathbf{x}^k)}$$

is the weight of the sample based on combined sample tree using the lower approximation L_N^F.

Similarly, for marginals, we can develop the statistics.

$$\widetilde{P}_N^U(x_i) = \frac{\sum_{k=1}^N w_N^U(\mathbf{x}^k)\delta_{x_i}(\mathbf{x}^k)}{\sum_{k=1}^N w_N^U(\mathbf{x}^k)} \tag{36}$$

and

$$\widetilde{P}_N^L(x_i) = \frac{\sum_{k=1}^N w_N^L(\mathbf{x}^k)\delta_{x_i}(\mathbf{x}^k)}{\sum_{k=1}^N w_N^L(\mathbf{x}^k)} \tag{37}$$

Next, in the following three theorems, we state some interesting properties of $\widetilde{Z}_N^L$, $\widetilde{Z}_N^U$, $\widetilde{P}_N^L(x_i)$ and $\widetilde{P}_N^U(x_i)$. The proofs are provided in the appendix.

Theorem 2. $\widetilde{Z}_N^L \leq \widehat{Z}_N \leq \widetilde{Z}_N^U$.

Theorem 3. *The estimates $\widetilde{Z}_N^U$ and $\widetilde{Z}_N^L$ of Z given in Equations 34 and 35 respectively are asymptotically unbiased. Similarly, the estimates $\widetilde{P}_N^U(x_i)$ and $\widetilde{P}_N^L(x_i)$ of $P(x_i)$ given in Equations 36 and 37 respectively are asymptotically unbiased.*

Theorem 4. *Given N samples output by SampleSearch for a mixed network $\mathcal{M} = \langle X, D, F, C\rangle$, the space and time complexity of computing $\widetilde{Z}_N^L$, $\widetilde{Z}_N^U$, $\widetilde{P}_N^L(x_i)$ and $\widetilde{P}_N^U(x_i)$ given in Equations 35, 34, 37 and 36 is $O(N \times d \times n)$.*

In summary, we presented two approximations for the backtrack-free probability Q^F which are used to bound the sample mean $\widehat{Z}_N$. We proved that the two approximations yield an asymptotically unbiased estimate of the weighted counts and marginals. They will also enable trading bias with variance as we discuss next.

4.2.2. Bias-Variance Tradeoff

As pointed in Section 2, the mean squared error of an estimator can be reduced by either controling the bias or by increasing the number of samples. The estimators $\widetilde{Z}_N^U$ and $\widetilde{Z}_N^L$ have more bias than the unbiased estimator $\widehat{Z}_N^F$ (which has a bias of zero but requires invoking an exact CSP solver $O(n \times d)$ times). However, given a fixed time bound, we expect that the estimators $\widetilde{Z}_N^U$ and $\widetilde{Z}_N^L$ will allow larger sample size than $\widehat{Z}_N^F$. Moreover, $\widetilde{Z}_N^U$ and $\widetilde{Z}_N^L$ bound $\widehat{Z}_N^F$ from above and below and therefore the absolute distance $|\widetilde{Z}_N^U - \widetilde{Z}_N^L|$ can be used to estimate their bias. If $|\widetilde{Z}_N^U - \widetilde{Z}_N^L|$ is small enough, then we can expect $\widetilde{Z}_N^U$ and $\widetilde{Z}_N^L$ to perform better than $\widehat{Z}_N^F$ because they can be based on a larger sample size.

4.3. Incorporating Advanced Search Techniques in SampleSearch

Theorem 1 is applicable to any search procedure that is systematic i.e. once the search procedure encounters an assignment $(x_1, \ldots, x_i)$, it will either prove that the assignment is inconsistent or return with a full consistent sample extending $(x_1, \ldots, x_i)$. Therefore, we can use any advanced systematic search technique (Dechter, 2003) instead of naive backtracking and easily show that:

Proposition 5. *Given a mixed network $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}, \boldsymbol{C} \rangle$ and an initial proposal distribution $Q = \{Q_1, \ldots, Q_n\}$, SampleSearch augmented with any systematic advanced search technique generates independent and identically distributed samples from the backtrack-free probability distribution Q^F of Q w.r.t. $\boldsymbol{C}$.*

While advanced search techniques would not change the sampling distribution of SampleSearch, in practice, they can have a significant impact on its time complexity and the quality of the upper and lower approximations. In particular, since SAT solvers developed over the last decade are quite efficient, we can represent the constraints in the mixed network using a CNF formula[3] and use minisat (Sorensson and Een, 2005) as our SAT solver. However, we have to make minisat (or any other state-of-the-art SAT solver e.g. RSAT (Pipatsrisawat and Darwiche, 2007)) *systematic* via the following changes (the changes can be implemented with minimal effort):

- **Turn off random restarts and far backtracks.** The use of restarts and far backtracks makes a SAT solver non-systematic and therefore they cannot be used.
- **Change variable and value Ordering.** We change the variable ordering to respect the structure of the input proposal distribution Q, namely given $Q(\mathbf{X}) = \prod_{i=1}^{n} Q_i \ (X_i |\ X_1 \ , \ldots \ , X_{i-1})$, we order variables as $o = (X_1, \ldots, X_n)$. Also, at

[3]It is easy to convert any (relational) constraint network to a CNF formula. In our implementation, we use the direct encoding described in (Walsh, 2000).

each decision point, variable X_i is assigned a value x_i by sampling it from $Q_i(X_i |x_1 , \ldots , x_{i-1})$.

5. Empirical Evaluation

We conducted empirical evaluation on three tasks: (a) counting models of SAT formula, (b) computing probability of evidence and partition function in Bayesian and Markov networks respectively, and (c) computing posterior marginals in a Bayesian and Markov network.

The results are organized as follows. In the next subsection, we present the implementation details of SampleSearch. Section 5.2 describes other techniques that we compared with. In Section 5.3, we describe the results for the weighted counting task while in Section 5.4, we focus on the posterior marginals task.

5.1. SampleSearch with Iterative Join Graph Propagation and w-cutset sampling (IJGP-wc-SS)

In our experiments, we show how SampleSearch (SS) operates on top of an advanced importance sampling algorithm IJGP-wc-IS presented earlier (Gogate and Dechter, 2005); referred to as IJGP-wc-SS. IJGP-wc-IS uses a generalized belief propagation scheme called Iterative Join Graph Propagation (IJGP) to construct a proposal distribution and the w-cutset sampling framework (Bidyuk and Dechter, 2007) to reduce the variance of the weights by sampling only over a subset of variables. Below, we outline the details of IJGP-wc-IS followed by those of IJGP-wc-SS.

- *The Proposal distribution*: The performance of importance sampling is highly dependent on how close the proposal distribution is to the posterior distribution (Rubinstein, 1981; Cheng and Druzdzel, 2000). In IJGP-wc-IS, we obtain $Q = \{Q_1, \ldots, Q_n\}$ from the output of Iterative Join Graph Propagation (IJGP) (Dechter et al., 2002) which was shown to yield good performance in earlier studies (Yuan and Druzdzel, 2006; Gogate and Dechter, 2005). IJGP is a generalized belief propagation (Yedidia et al., 2004) technique for approximating the posterior distribution in graphical models (for more details see (Dechter et al., 2002)). It runs the same message passing as *join tree propagation* (Kask, Dechter, Larrosa, and Dechter, 2005) over the clusters of a *join graph* rather than a *join tree*, iteratively. A join graph is a decomposition of functions of the mixed network into a graph of clusters that satisfies all the properties required of a valid join tree decomposition except the tree requirement. The time and space complexity of IJGP can be controlled by its i-bound parameter which bounds the cluster size of its join graph. IJGP is exponential in its i-bound and its accuracy generally increases with the i-bound. 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 (Dechter et al., 2002) and compute, in advance, the space required by IJGP by summing over the space required by the

individual messages[4]. 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.

- *w-cutset sampling:* As mentioned in Section 2.3, the mean squared error of importance sampling can be reduced by reducing the variance of the weights. To reduce the variance of the weights, we combine importance sampling with w-cutset sampling (Bidyuk and Dechter, 2007). The main idea in w-cutset sampling is to partition the variables $\mathbf{X}$ into two sets $\mathbf{K}$ and $\mathbf{R}$ such that the treewidth of the mixed network restricted to $\mathbf{R}$ is bounded by a constant w. The set $\mathbf{K}$ is called the w-cutset. Because we can efficiently compute marginals and weighted counts over the mixed network restricted to $\mathbf{R}$ given $\mathbf{K} = \mathbf{k}$ by using exact inference techniques such as bucket elimination (Dechter, 1999), we can only sample the variables in $\mathbf{K}$ using a proposal distribution $Q(\mathbf{K})$ and perform exact inference over $\mathbf{R}$ given $\mathbf{K}$. From the Rao-Blackwell theorem (Casella and Robert, 1996; Liu, 2001), it is easy to show that sampling from the subspace $\mathbf{K}$ reduces the variance.

  Formally, given a mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$, 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:

$$w_{wc}(\mathbf{k}) = \frac{\sum_{\mathbf{r} \in \mathbf{R}} \prod_{j=1}^{m} F_j(\mathbf{r}, \mathbf{K} = \mathbf{k}) \prod_{a=1}^{p} C_a(\mathbf{r}, \mathbf{K} = \mathbf{k})}{Q(\mathbf{k})} \tag{38}$$

  where $\mathbf{R} = \mathbf{X} \setminus \mathbf{K}$. Given a w-cutset $\mathbf{K}$, we can compute the sum in the numerator of Equation 38 in polynomial time (exponential in the constant w) using bucket elimination (Dechter, 1999).

  It was demonstrated that the higher the w-bound (Bidyuk and Dechter, 2007), 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.

- *Variable Ordering:* We use the min-fill ordering for constructing the join graph for IJGP because it yields better performance than other ordering heuristics such as the min-degree and topological ordering. Sampling is performed in reverse min-fill ordering.

The implementation details of IJGP-wc-SS are given in Algorithm 3. The algorithm takes as input a mixed network and integer i, w and N which specify the i-bound for IJGP, w for creating a w-cutset and the number of samples N respectively[5]. In Steps 1-2, the algorithm creates a join graph along the min fill ordering and runs IJGP. Then, in Step 3, it computes a w-cutset $\mathbf{K}$ for the mixed network. Then the algorithm creates

[4]Note that we can do this without constructing the messages explicitly.

[5]This is done after we determine the i-bound and the w for the w-cutset.

Algorithm 3: Implementation details of IJGP-wc-SS (SampleSearch with IJGP based proposal and w-cutset sampling)

Input: A mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$, integers i, N and w.
Output: A set of N samples globally consistent w.r.t. $\mathbf{C}$

1 Create a min-fill ordering $o = (X_1, \ldots, X_n)$;
2 Create a join-graph JG with i-bound i along o using the join-graph structuring algorithm given in (Dechter et al., 2002) and run IJGP on JG;
3 Create a w-cutset $\mathbf{K} \subseteq \mathbf{X}$ using the greedy scheme described in (Bidyuk and Dechter, 2004, 2007). Let $\mathbf{K} = \{K_1, \ldots, K_t\}$;
4 Create a proposal distribution $Q(\mathbf{K}) = \prod_{i=1}^{t} Q_i(K_i|K_1, \ldots, K_{i-1})$ from the messages and functions in JG using the the following heuristic scheme (Gogate and Dechter, 2005). First, we find a cluster A in JG that mentions K_i and has the largest number of variables common with the previous variables $\{K_1, \ldots, K_{i-1}\}$. Then, we construct $Q_i(K_i|K_1, \ldots, K_{i-1})$ by marginalizing out all variables not mentioned in $K_1, \ldots, K_i$ from the marginal over the variables of A;
5 **for** *i=1 to N do* **do**
6 Apply minisat based SampleSearch on $\mathcal{M}$ with proposal distribution $Q(\mathbf{K})$ to get a sample $\mathbf{k}^i$.;
7 Store the DFS-trace of the sample $\mathbf{k}^i$ in a combined sample tree.
8 Output the required statistics (marginals or weighted counts) based on the combined sample tree;

a proposal distribution over the w-cutset $\mathbf{K}$, $Q(\mathbf{K}) = \prod_{i=1}^{t} Q_i(K_i|K_1, \ldots, K_{i-1})$ from the output of IJGP using a heuristic scheme outlined in Step 4. Finally, in Steps 5-8 the algorithm executes minisat based SampleSearch on the mixed network to generate the required N samples and outputs the required statistics.

Henceforth, we will refer to the estimates of IJGP-wc-SS generated using the upper and lower approximations of the backtrack-free probability given by Equations 34 and 35 as IJGP-wc-SS/UB and IJGP-wc-SS/LB respectively. Note that IJGP-wc-SS/UB and IJGP-wc-SS/LB bound the sample mean $\widehat{Z}_N$ from above and below respectively and not the true mean or the (exact) weighted counts Z.

5.2. Alternative schemes

In addition to IJGP-wc-SS and IJGP-wc-IS, we experimented with the following schemes.

1. Iterative Join Graph Propagation (IJGP)

In our experiments, we used an anytime version of IJGP (Dechter et al., 2002) in which we start with an i-bound of 1, run IJGP until convergence or 10 iterations whichever is earlier. Then we increase the i-bound by one and reconstruct the join graph. We do this until one the following conditions is met: (a) i equals the treewidth in which case IJGP yields exact marginals or (b) the 2 GB space limit is reached or (c) the prescribed time-bound is reached.

2. ApproxCount and SampleCount (Wei and Selman, 2005) introduced an approximate solution counting scheme called ApproxCount. ApproxCount is based on the formal result of (Valiant, 1987) 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 (Sang, Beame, and Kautz, 2005). To reduce the variance, (Wei and Selman, 2005) 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 (Wei et al., 2004), which is an extension of the well-known local search SAT solver Walksat (Selman, Kautz, and Cohen, 1994). We experimented with an anytime version of ApproxCount in which we report the cumulative average accumulated over several runs.

SampleCount (Gomes et al., 2007) 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. We experimented with an anytime version of SampleCount in which we report the unbiased cumulative averages over several runs[6].

In our experiments, we used an implementation of ApproxCount and SampleCount available from the respective authors (Wei et al., 2004; Gomes et al., 2007). Following the recommendations made in (Gomes et al., 2007), 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 $\alpha = 100K$.

3. Evidence Pre-propagated Importance sampling (EPIS) is an adaptive importance sampling algorithm for computing marginals in Bayesian networks (Yuan and Druzdzel, 2006). The algorithm uses loopy belief propagation (Pearl, 1988; Murphy, Weiss, and Jordan, 1999) to construct the proposal distribution. In our experiments, we used the anytime implementation of EPIS submitted to the UAI 2008 evaluation (Darwiche, Dechter, Choi, Gogate, and Otten, 2008).

4. Edge Deletion Belief Propagation (EDBP)(Choi and Darwiche, 2006) is an approximation algorithm for computing posterior marginals and for computing probability of evidence. EDBP solves exactly a simplified version of the original problem, obtained

[6]In the original paper, SampleCount (Gomes et al., 2007) was investigated for lower bounding solution counts. Here, we evaluate the unbiased solution counts computed by the algorithm.

Problem Type	IJGP-wc-SS IJGP-wc-IS	IJGP	EDBP	EPIS-BN	VEC	SampleCount ApproxCount Relsat
Bayesian networks $P(e)$	$\surd$		$\surd$		$\surd$	
Markov Networks Z	$\surd$		$\surd$		$\surd$	
Bayesian networks Mar	$\surd$	$\surd$	$\surd$	$\surd$		
Markov networks Mar	$\surd$	$\surd$	$\surd$			
Model counting	$\surd$					$\surd$

Z: partition function, P(e): probability of evidence and Mar: posterior marginals.

Table 1: Query types handled by various solvers.

by deleting some of the edges from the primal graph. Deleted edges are selected based on two criteria : quality of approximation and complexity of computation (tree-width reduction) which is parameterized by an integer k, called the k-bound. Subsequently, information loss from the lost dependencies is compensated for by using several heuristic techniques. The implementation of this scheme is available from the authors (Choi and Darwiche, 2006).

5. Variable Elimination + 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 (Dechter, 1999). 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 (Dechter, 1999). 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 by converting all zero probabilities and constraints in the problem to a CNF formula and performing singleton-consistency enforcement i.e. pruning all variable-value pairs which are inconsistent (detected using minisat (Sorensson and Een, 2005)). The implementation of this scheme is available publicly from our software website (Dechter, Gogate, Otten, Marinescu, and Mateescu, 2009).

6. Relsat (Roberto J. Bayardo and Pehoushek, 2000) is an exact algorithm for counting solutions of a satisfiability problem. When Relsat is stopped before completion, it yields a lower bound on the number of solutions. The implementation of Relsat is available publicly from the authors web site (Roberto J. Bayardo and Pehoushek, 2000).

The benchmarks and the solvers for the different task types are shown in Figure 5. Table 1 summarizes different query types that can be handled by the various solvers. A '$\surd$' indicates that the algorithm is able to approximately estimate the query while a lack of $\surd$ indicates otherwise.

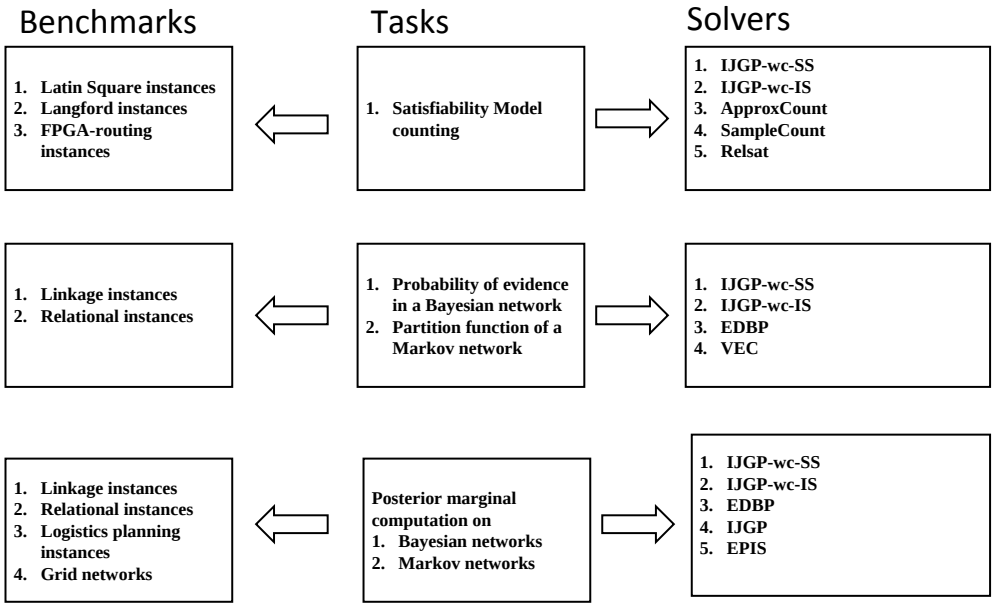


Figure 5: Chart showing the scope of our experimental study

5.3. Results for Weighted Counts

Notation in Tables

The first column in each table (see Table 2 for example) gives the name of the instance. The second column provides various statistical information about the instance such as the number of variables n, the average domain size d, the number of clauses or constraints c, the number of evidence variables e and the treewidth of the instance w (computed using the min-fill heuristic). The fourth column provides the exact answer for the problem instance if available while the remaining columns display the results for the various solvers when terminated at the specified time-bound. The solver(s) giving the best results is highlighted in each row. A "*" next to the output of a solver indicates that it solved the problem instance exactly (before the time-bound expired) followed by the number of seconds it took to solve the instance enclosed in brackets. An "X" indicates that no solution was output by the solver.

5.3.1. Satisfiability instances

For the task of counting solutions (or models) of a satisfiability formula, we evaluate the algorithms on formulas from three domains: (a) normalized Latin square problems, (b) Langford problems, (c) FPGA-Routing instances. We ran each algorithm for 10 hours on each instance.

Results on instances for which exact solution counts are known

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 numbers from $\{1, \ldots, s\}$ 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

Problem	$\langle n, k, c, w \rangle$		Exact	Sample Count	Approx Count	REL SAT	IJGP-wc-SS/LB	IJGP-wc-SS/UB	IJGP-wc-IS
ls8-norm	$\langle 512, 2, 5584, 255 \rangle$	Z	5.40E11	**5.15E+11**	3.52E+11	2.44E+08	5.91E+11	5.91E+11	X
		M		16514	17740		236510	236510	0
ls9-norm	$\langle 729, 2, 9009, 363 \rangle$	Z	3.80E17	4.49E+17	1.26E+17	1.78E+08	**3.44E+17**	3.44E+17	X
		M		7762	8475		138572	138572	0
ls10-norm	$\langle 1000, 2, 13820, 676 \rangle$	Z	7.60E24	**7.28E+24**	1.17E+24	1.36E+08	6.74E+24	6.74E+24	X
		M		3854	4313		95567	95567	0
ls11-norm	$\langle 1331, 2, 20350, 956 \rangle$	Z	5.40E33	2.08E+34	4.91E+31	1.09E+08	**3.87E+33**	3.87E+33	X
		M		2002	2289		66795	66795	0

Table 2: Table showing the solution counts Z and the number of consistent samples M (only for the sampling based solvers) output by IJGP-wc-SS, IJGP-wc-IS, ApproxCount, SampleCount and Relsat after 10 hours of CPU time for 4 Latin Square instances for which the exact solution counts are known.

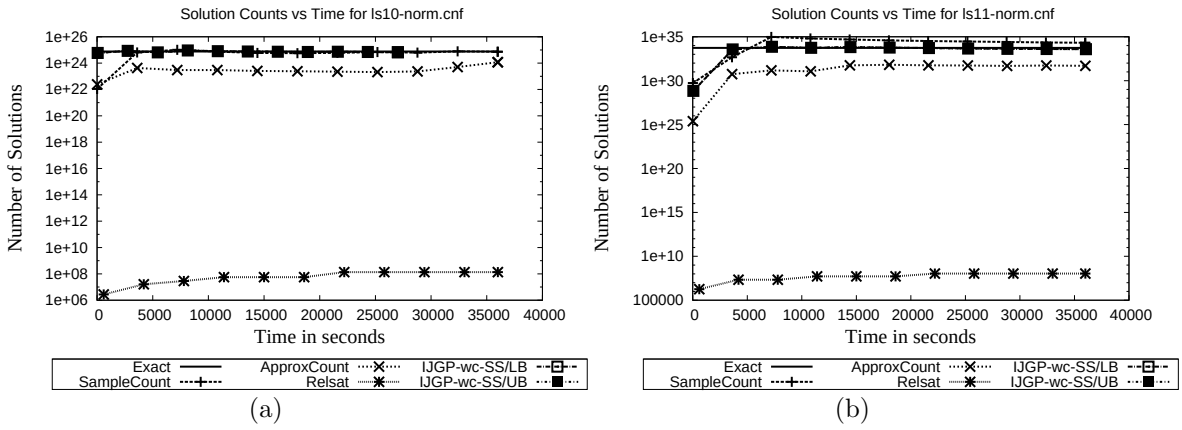


Figure 6: Time versus solution counts for two sample Latin square instances. IJGP-wc-IS is not plotted in the figures because it fails on all the instances.

modeled as SAT formulas using the extended encoding given in (Gomes and Shmoys, 2002). The exact counts for these formulas are known up to order 11 (Ritter, 2003).

Table 2 shows the results for latin square instances up to order 11 for which exact solution counts are known. ApproxCount and Relsat underestimate the counts by several orders of magnitude. On the other hand, IJGP-wc-SS/UB, IJGP-wc-SS/LB and SampleCount yield very good estimates close to the true counts. The counts output by IJGP-wc-SS/UB and IJGP-wc-SS/LB are the same for all instances indicating that the sample mean is accurately estimated by the upper and lower approximations of the backtrack-free distribution (see the discussion on bias versus variance in Section 4.2.2). IJGP-wc-IS fails on all instances and is unable to generate a single consistent sample in ten hours. IJGP-wc-SS generates far more solution samples as compared with SampleCount and ApproxCount. In Figure 6 (a) and (b), we show how the estimates output by various solvers change with time for the two largest instances. Here, we can clearly see the superior convergence of IJGP-wc-SS/LB, IJGP-wc-SS/UB and SampleCount over other approaches.

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 $\{1, 2, \ldots, s\}$, the problem is to produce a sequence of length $2s$ such that each $i \in \{1, 2, \ldots, s\}$ appears twice in the sequence and the two occurrences of i are exactly i apart

Problem	$\langle n, k, c, w \rangle$		Exact	Sample Count	Approx Count	REL SAT	IJGP-wc-SS/LB	IJGP-wc-SS/UB	IJGP-wc-IS
lang12	$\langle 576, 2, 13584, 383 \rangle$	Z	2.16E+5	1.93E+05	2.95E+04	**2.16E+05**$^{*(297s)}$	2.16E+05	2.16E+05	X
		M		2720	4668		999991	999991	0
lang16	$\langle 1024, 2, 32320, 639 \rangle$	Z	6.53E+08	5.97E+08	8.22E+06	6.28E+06	**6.51E+08**	6.99E+08	X
		M		328	641		14971	14971	0
lang19	$\langle 1444, 2, 54226, 927 \rangle$	Z	5.13E+11	9.73E+10	6.87E+08	8.52E+05	**6.38E+11**	7.31E+11	X
		M		146	232		3431	3431	0
lang20	$\langle 1600, 2, 63280, 1023 \rangle$	Z	5.27E+12	1.13E+11	3.99E+09	8.55E+04	2.83E+12	**3.45E+12**	X
		M		120	180		2961	2961	0
lang23	$\langle 2116, 2, 96370, 1407 \rangle$	Z	7.60E+15	7.53E+14	3.70E+12	X	4.17E+15	**4.19E+15**	X
		M		38	54		1111	1111	0
lang24	$\langle 2304, 2, 109536, 1535 \rangle$	Z	9.37E+16	1.17E+13	4.15E+11	X	8.74E+15	**1.40E+16**	X
		M		25	33		271	271	0

Table 3: Table showing the solution counts Z and the number of consistent samples M (only for the sampling based solvers) output by IJGP-wc-SS, IJGP-wc-IS , ApproxCount, SampleCount and Relsat after 10 hours of CPU time for Langford's problem instances. A "*" next to the output of a solver indicates that it solved the problem exactly (before the time-bound of 10 hours expired) followed by the number of seconds it took to solve the instance exactly.

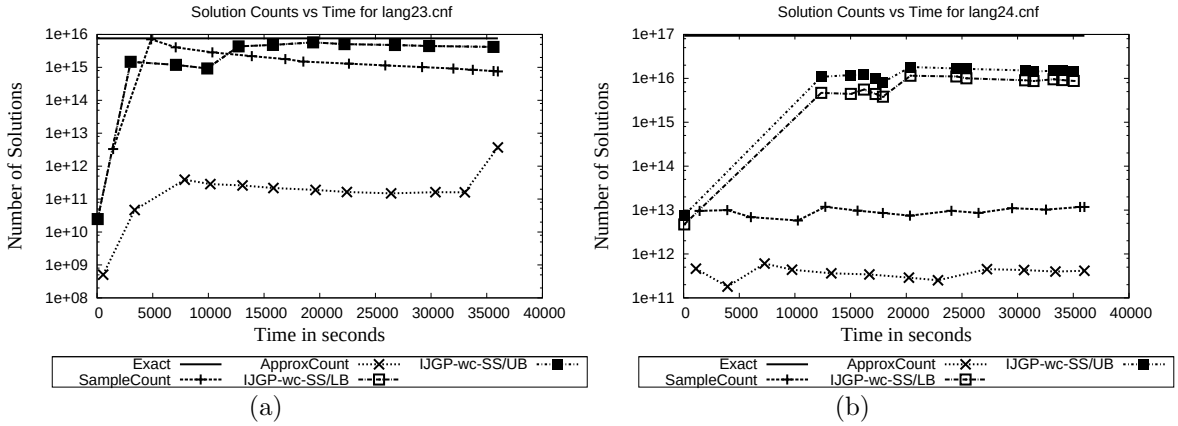


Figure 7: Time versus solution counts for two sample Langford instances. IJGP-wc-IS and Relsat are not plotted in the figures because they fail on the given instances.

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 (Walsh, 2001).

Table 3 presents the results. ApproxCount and Relsat severely under estimate the true counts except on the instance of size 12 (lang12 in Table 3) which Relsat solves exactly in about 5 minutes. SampleCount is inferior to IJGP-wc-SS/UB and IJGP-wc-SS/LB by several orders of magnitude. IJGP-wc-SS/UB is slightly better than IJGP-wc-SS/LB. Unlike the Latin square instances, the solution counts output by IJGP-wc-SS/LB and IJGP-wc-SS/UB are different for large problems but the difference is small. IJGP-wc-IS fails on all instances because it does not perform search. Again, we see that IJGP-wc-SS generates far more consistent samples as compared with SampleCount and ApproxCount. In Figure 7 (a) and (b), we show how the the estimates output by various solvers change with time for the two largest instances. Here, we clearly see the superior anytime performance of IJGP-wc-SS/LB and IJGP-wc-SS/UB.

Problem	$\langle n, k, c, w \rangle$		Exact	Sample Count	REL SAT	IJGP-wc-SS/LB	IJGP-wc-IS
ls12-norm	$\langle 1728, 2, 28968, 1044 \rangle$	Z		**2.23E+43**	1.26E+08	1.25E+43	X
		M		1064		13275	0
ls13-norm	$\langle 2197, 2, 40079, 1558 \rangle$	Z		3.20E+54	9.32E+07	**1.15E+55**	X
		M		566		6723	0
ls14-norm	$\langle 2744, 2, 54124, 1971 \rangle$	Z		5.08E+65	7.1E+07	**1.24E+70**	X
		M		299		3464	0
ls15-norm	$\langle 3375, 2, 71580, 2523 \rangle$	Z		3.12E+79	2.06E+07	**2.03E+83**	X
		M		144		1935	0
ls16-norm	$\langle 4096, 2, 92960, 2758 \rangle$	Z		7.68E+95	X	**2.08E+98**	X
		M		58		1530	0

Table 4: Table showing the lower bounds on solution counts Z and the number of consistent samples M (only for the sampling-based solvers) output by IJGP-wc-SS/LB, IJGP-wc-IS, SampleCount and Relsat after 10 hours of CPU time for 5 Latin Square instances for which the exact solution counts are not known. The entries for IJGP-wc-IS, SampleCount and IJGP-wc-SS/LB contain the lower bounds computed by combining their respective sample weights with the Markov inequality based Average and Martingale schemes given in (Gogate et al., 2007).

Results on instances for which exact solution counts are not known

When exact results are not available evaluating the capability of SampleSearch or any other approximation algorithm is problematic because the quality of the approximation (namely how close the approximation is to the exact) cannot be assessed. To allow some comparison on such hard instances we evaluate the power of the various sampling schemes for yielding good lower-bound approximations whose quality can be compared (the higher the better) even when exact solution is not available. Specifically, when the exact solution counts are not known, we compare the lower bounds obtained by combining IJGP-wc-SS/LB, IJGP-wc-IS and SampleCount with the Markov inequality based martingale and average schemes described in our previous work (Gogate et al., 2007). These lower bounding schemes (Gomes et al., 2007; Gogate et al., 2007) take as input: (a) a set of unbiased sample weights or a lower bound on the unbiased sample weights and (b) a real number $0 < \alpha < 1$, and output a lower bound on the weighted counts Z (or solution counts in case of a SAT formula) that is correct with probability greater than α. In our experiments, we set $\alpha = 0.99$ which means that the lower bounds are correct with probability greater than 0.99.

We will show that the samples derived from SampleSearch (IJGP-wc-SS/LB) give rise to superior lower bounds compared with other sampling-based schemes. Since the quality of lower bounds can be compared even when we dont have an exact solution, comparing lower-bounds facilitate a comparative evaluation even on instances for which exact weighted count sare not available[7].

IJGP-wc-SS/UB cannot be used to lower bound Z because it outputs upper bounds on the unbiased sample weights. Likewise, ApproxCount cannot be used to lower bound Z because it is not unbiased. Finally, note that Relsat always yields a lower bound on the solution counts with probability one. *When we compare lower bounds, the higher*

[7]We still cannot evaluate the quality of the marginals when the exact solution is not known because the Markov inequality based schemes (Gomes et al., 2007; Gogate et al., 2007) cannot lower bound marginal probabilities.

Problem	$\langle n, k, c, w \rangle$		Exact	SampleCount	Relsat	IJGP-wc-SS/LB	IJGP-wc-IS
9symml_gr_2pin_w6	$\langle 2604, 2, 36994, 413 \rangle$	Z		3.36E+51	3.41E+32	**3.06E+53**	X
		M		3		6241	0
9symml_gr_rcs_w6	$\langle 1554, 2, 29119, 613 \rangle$	Z		**8.49E+84**	3.36E+72	2.80E+82	X
		M		374		16911	0
alu2_gr_rcs_w8	$\langle 4080, 2, 83902, 1470 \rangle$	Z		1.21E+206	1.88E+56	**1.69E+235**	X
		M		8		841	0
apex7_gr_2pin_w5	$\langle 1983, 2, 15358, 188 \rangle$	Z		5.83E+93	4.83E+49	**2.33E+94**	X
		M		54		25161	0
apex7_gr_rcs_w5	$\langle 1500, 2, 11695, 290 \rangle$	Z		**2.17E+139**	3.69E+46	9.64E+133	X
		M		1028		48331	0
c499_gr_2pin_w6	$\langle 2070, 2, 22470, 263 \rangle$	Z		X	2.78E+47	**2.18E+55**	X
		M		0		4491	0
c499_gr_rcs_w6	$\langle 1872, 2, 18870, 462 \rangle$	Z		**2.41E+87**	7.61E+54	1.29E+84	X
		M		40		14151	0
c880_gr_rcs_w7	$\langle 4592, 2, 61745, 1024 \rangle$	Z		**1.50E+278**	1.42E+43	7.16E+255	X
		M		5		831	0
example2_gr_2pin_w6	$\langle 3603, 2, 41023, 350 \rangle$	Z		3.93E+160	7.35E+38	**7.33E+160**	X
		M		1		1971	0
example2_gr_rcs_w6	$\langle 2664, 2, 27684, 476 \rangle$	Z		**4.17E+265**	1.13E+73	6.85E+250	X
		M		167		6211	0
term1_gr_2pin_w4	$\langle 746, 2, 3964, 31 \rangle$	Z		X	2.13E+35	**6.90E+39**	X
		M		0		326771	0
term1_gr_rcs_w4	$\langle 808, 2, 3290, 57 \rangle$	Z		X	1.17E+49	**7.44E+55**	X
		M		0		341951	0
too_large_gr_rcs_w7	$\langle 3633, 2, 50373, 1069 \rangle$	Z		X	1.46E+73	**1.05E+182**	X
		M		0		1561	0
too_large_gr_rcs_w8	$\langle 4152, 2, 57495, 1330 \rangle$	Z		X	1.02E+64	**5.66E+246**	X
		M		0		1171	0
vda_gr_rcs_w9	$\langle 6498, 2, 130997, 2402 \rangle$	Z		X	2.23E+92	**5.08E+300**	X
		M		0		221	0

Table 5: Table showing the lower bounds on solution counts Z and the number of consistent samples M (only for the sampling-based solvers) output by IJGP-wc-SS/LB, IJGP-wc-IS, SampleCount and Relsat after 10 hours of CPU time for FPGA routing instances. The entries for IJGP-wc-IS, SampleCount and IJGP-wc-SS/LB contain the lower bounds computed by combining their respective sample weights with the Markov inequality based Average and Martingale schemes given in (Gogate et al., 2007).

the lower bound, the better the solver is.

First we compare the lower bounding ability of IJGP-wc-IS, IJGP-wc-SS/LB, SampleCount and Relsat on latin square instances of size 12 through 15 for which the exact counts are not known. Table 4 contains the results. IJGP-wc-SS/LB yields far better (higher) lower bounds than SampleCount as the problem size increases. Relsat underestimates the counts by several orders of magnitude as compared with IJGP-wc-SS/LB and SampleCount. As expected, IJGP-wc-IS fails on all instances. Again, we can see that the lower bounds obtained via IJGP-wc-SS/LB are based on a much larger sample size as compared with SampleCount.

Our final 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 (Simon, Berre, and Hirsch, 2005). Table 5 presents the results for these instances. IJGP-wc-SS/LB yields higher lower bounds than SampleCount and Relsat on ten out of the fifteen instances. On the remaining five instances SampleCount yields higher lower bounds than IJGP-wc-SS/LB. Relsat is always inferior to IJGP-wc-SS/LB while IJGP-wc-IS fails on all instances. SampleCount fails to yield even a single consistent sample on 6 out of the 15 instances. On the remaining nine

Problem	$\langle n, k, c, e, w \rangle$		Exact	IJGP-wc -SS/LB	IJGP-wc -SS/UB	VEC	EDBP	IJGP-wc -IS
BN_69	$\langle 777, 7, 228, 78, 47 \rangle$	Z	5.28E-054	**3.00E-55**	**3.00E-55**	1.93E-61	2.39E-57	X
		M		6.84E+5	6.84E+5			0
BN_70	$\langle 2315, 5, 484, 159, 87 \rangle$	Z	2.00E-71	**1.21E-73**	**1.21E-73**	7.99E-82	6.00E-79	X
		M		1.92E+5	1.92E+5			0
BN_71	$\langle 1740, 6, 663, 202, 70 \rangle$	Z	5.12E-111	**1.28E-111**	**1.28E-111**	7.05E-115	1.01E-114	X
		M		7.46E+4	7.46E+4			0
BN_72	$\langle 2155, 6, 752, 252, 86 \rangle$	Z	4.21E-150	**4.73E-150**	**4.73E-150**	1.32E-153	9.21E-155	X
		M		1.53E+5	1.53E+5			0
BN_73	$\langle 2140, 5, 651, 216, 101 \rangle$	Z	2.26E-113	**2.00E-115**	**2.00E-115**	6.00E-127	2.24E-118	X
		M		7.75E+4	7.75E+4			0
BN_74	$\langle 749, 6, 223, 66, 45 \rangle$	Z	3.75E-45	**2.13E-46**	**2.13E-46**	3.30E-48	5.84E-48	X
		M		2.80E+5	2.80E+5			0
BN_75	$\langle 1820, 5, 477, 155, 92 \rangle$	Z	5.88E-91	**2.19E-91**	**2.19E-91**	5.83E-97	3.10E-96	X
		M		7.72E+4	7.72E+4			0
BN_76	$\langle 2155, 7, 605, 169, 64 \rangle$	Z	4.93E-110	**1.95E-111**	**1.95E-111**	1.00E-126	3.86E-114	X
		M		2.52E+4	2.52E+4			0

Table 6: Probability of evidence (Z) computed by VEC, EDBP, IJGP-wc-IS and IJGP-wc-SS after 3 hours of CPU time for Linkage instances from the UAI 2006 evaluation. For IJGP-wc-SS and IJGP-wc-IS, we also report the number of consistent samples (M) generated in 3 hours.

instances, the number of consistent samples generated by SampleCount are far less than IJGP-wc-SS.

5.3.2. Linkage networks

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 (Ott, 1999). This is very useful in practice for identifying disease genes. The input is an ordered list of loci $L_1, \ldots, L_{k+1}$ 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 $[\theta_1, \ldots, \theta_k]$ 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 8 represents a fragment of a network that 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 (Fishelson and Geiger, 2003)).

We first evaluate the solvers on Linkage (Bayesian) networks used in the UAI 2006

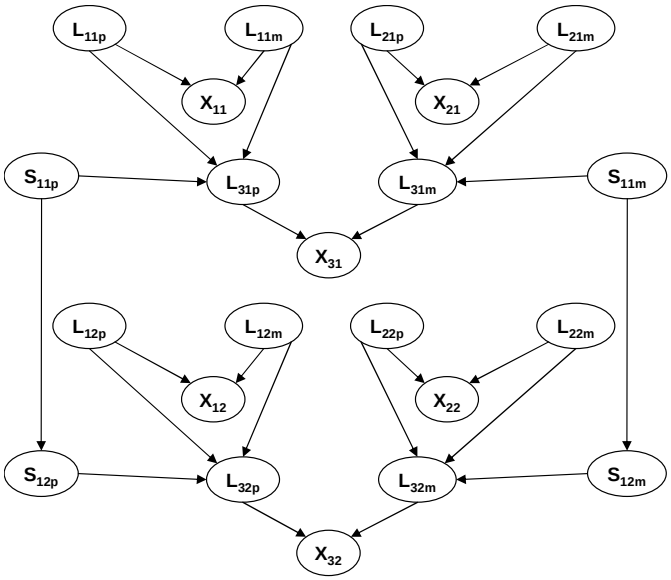


Figure 8: A fragment of a Bayesian network used in genetic linkage analysis.

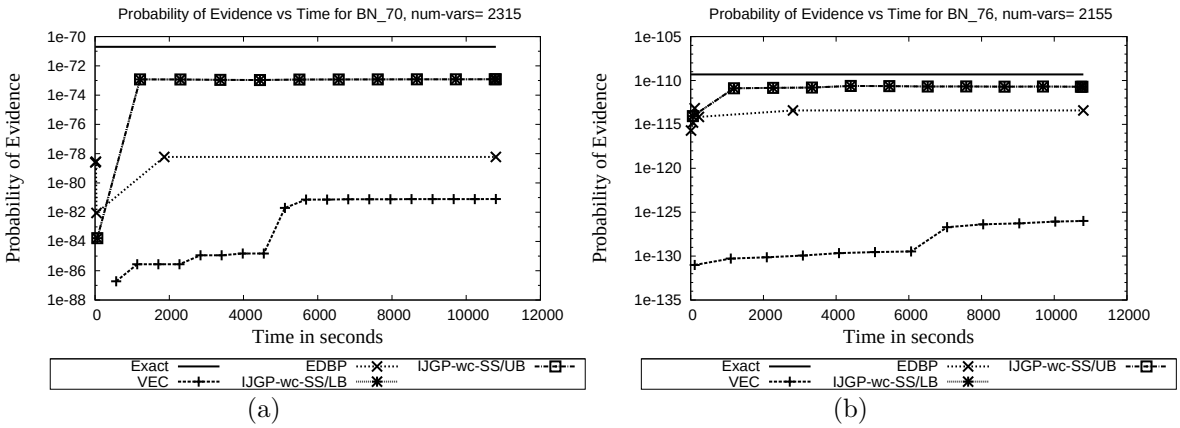


Figure 9: Convergence of probability of evidence as a function of time for two sample Linkage instances. IJGP-wc-IS is not plotted in the figures because it fails on all the instances.

Problem	$\langle n, k, c, e, w \rangle$		Exact	IJGP-wc -SS/LB	IJGP-wc -SS/UB	VEC	EDBP	IJGP-wc -IS
pedigree18	$\langle 1184, 2, 386, 0, 26 \rangle$	Z	7.18E-79	7.39E-79	7.39E-79	**7.18E-79**$^{*(64s)}$	**7.18E-79**$^{*(772s)}$	X
		M		1.30E+5	1.30E+5			0
pedigree1	$\langle 334, 2, 121, 0, 20 \rangle$	Z	7.81E-15	7.81E-15	7.81E-15	**7.81E-15**$^{*(12s)}$	**7.81E-15**$^{*(14s)}$	X
		M		3.26E+5	3.26E+5			0
pedigree20	$\langle 437, 2, 147, 0, 25 \rangle$	Z	2.34E-30	2.31E-30	2.31E-30	**2.34E-30**$^{*(1216s)}$	6.19E-31	X
		M		2.31E+5	2.31E+5			0
pedigree23	$\langle 402, 2, 130, 0, 26 \rangle$	Z	2.78E-39	2.76E-39	2.76E-39	**2.78E-39**$^{*(913s)}$	1.52E-39	X
		M		3.28E+5	3.28E+5			0
pedigree25	$\langle 1289, 2, 396, 0, 38 \rangle$	Z	1.69E-116	**1.69E-116**	**1.69E-116**	**1.69E-116**$^{*(318s)}$	**1.69E-116**$^{*(2562s)}$	X
		M		1.29E+5	1.29E+5			0
pedigree30	$\langle 1289, 2, 413, 0, 27 \rangle$	Z	1.84E-84	1.90E-84	1.90E-84	**1.85E-84**$^{*(808s)}$	**1.85E-84**$^{*(179s)}$	X
		M		1.14E+5	1.14E+5			0
pedigree37	$\langle 1032, 2, 333, 0, 25 \rangle$	Z	2.63E-117	1.18E-117	1.18E-117	**2.63E-117**$^{*(2521s)}$	5.69E-124	X
		M		4.26E+5	4.26E+5			0
pedigree38	$\langle 724, 2, 263, 0, 18 \rangle$	Z	5.64E-55	3.80E-55	3.80E-55	**5.65E-55**$^{*(735s)}$	8.41E-56	X
		M		1.63E+5	1.63E+5			0
pedigree39	$\langle 1272, 2, 354, 0, 29 \rangle$	Z	6.32E-103	6.29E-103	6.29E-103	**6.32E-103**$^{*(136s)}$	**6.32E-103**$^{*(694s)}$	X
		M		1.25E+5	1.25E+5			0
pedigree42	$\langle 448, 2, 156, 0, 23 \rangle$	Z	1.73E-31	1.73E-31	1.73E-31	**1.73E-31**$^{*(3188s)}$	8.91E-32	X
		M		3.26E+5	3.26E+5			0

Table 7: Probability of evidence Z computed by VEC, EDBP, IJGP-wc-IS and IJGP-wc-SS after 3 hours of CPU time for Linkage instances from the UAI 2008 evaluation. For IJGP-wc-SS and IJGP-wc-IS, each cell in the table also reports the number of consistent samples M generated in 3 hours. A "*" next to the output of a solver indicates that it solved the problem exactly (before the time-bound expired) followed by the number of seconds it took to solve the instance exactly.

evaluation (Bilmes and Dechter, 2006). Table 6 contains the results. We see that IJGP-wc-SS/UB and IJGP-wc-SS/LB are very accurate usually yielding a few orders of magnitude improvement over VEC and EDBP. Because the estimates output by IJGP-wc-SS/UB and IJGP-wc-SS/LB are the same on all instances, they yield an exact value of the sample mean. Figure 9 shows how the probability of evidence changes as a function of time for two sample instances. We see superior anytime performance of both IJGP-wc-SS schemes as compared with VEC and EDBP. IJGP-wc-IS fails to output a single consistent sample in 3 hours of CPU time on all the instances.

In Table 7, we present the results on the 10 linkage instances that were used in the UAI 2008 evaluation (Darwiche et al., 2008) for which the exact value of probability of evidence is known[8]. We see that VEC (as an anytime scheme) exactly solves all the 10 instances while EDBP solves 5 instances (as indicated by a $*$ in Table 7). IJGP-wc-SS/LB and IJGP-wc-SS/UB deviate only slightly from the exact value of probability of evidence and on the four instances for which EDBP does not output the exact answer, the estimates output by IJGP-wc-SS/LB and IJGP-wc-SS/UB are better than EDBP. Again, IJGP-wc-IS fails on all the instances.

5.3.3. Relational Instances

The relational instances are generated by grounding the relational Bayesian networks using the primula tool (Chavira et al., 2006). We experimented with ten Friends and

[8]The exact marginals and probability of evidence of all the Bayesian network benchmarks reported in this paper were computed using ACE (Chavira and Darwiche, 2008).

Problem	$\langle n, k, c, e, w \rangle$		Exact	IJGP-wc -SS/LB	IJGP-wc -SS/UB	VEC	EDBP	IJGP-wc -IS
Friends and Smokers								
fs-04	$\langle 262, 2, 74, 226, 12 \rangle$	Z	1.53E-05	8.11E-06	8.11E-06	**1.53E-05**$^{*(1s)}$	**1.53E-05**$^{*(2s)}$	1.52E-05
		M		1.00E+6	1.00E+6			2.17E+8
fs-07	$\langle 1225, 2, 371, 1120, 35 \rangle$	Z	1.78E-15	2.23E-16	2.23E-16	**1.78E-15**$^{*(708s)}$	1.11E-16	X
		M		1.00E+6	1.00E+6			0
fs-10	$\langle 3385, 2, 1055, 3175, 71 \rangle$	Z	7.88E-31	**2.49E-32**	**2.49E-32**	X	7.70E-34	X
		M		8.51E+5	8.51E+5			0
fs-13	$\langle 7228, 2, 2288, 6877, 119 \rangle$	Z	1.33E-51	3.26E-55	3.26E-55	X	1.63E-55	**1.33E-51**
		M		5.41E+5	5.41E+5			4.67E+7
fs-16	$\langle 13240, 2, 4232, 12712, 171 \rangle$	Z	8.63E-78	6.04E-79	6.04E-79	X	1.32E-82	**8.63E-78**
		M		1.79E+5	1.79E+5			1.37E+7
fs-19	$\langle 21907, 2, 7049, 21166, 243 \rangle$	Z	2.12E-109	**1.62E-114**	**1.62E-114**	X	X	X
		M		1.90E+5	1.90E+5			0
fs-22	$\langle 33715, 2, 10901, 32725, 335 \rangle$	Z	2.00E-146	**4.88E-147**	**4.88E-147**	X	X	X
		M		1.18E+5	1.18E+5			0
fs-25	$\langle 49150, 2, 15950, 47875, 431 \rangle$	Z	7.18E-189	**2.67E-189**	**2.67E-189**	X	X	X
		M		9.23E+4	9.23E+4			0
fs-28	$\langle 68698, 2, 22358, 67102, 527 \rangle$	Z	9.82E-237	**4.53E-237**	**4.53E-237**	X	X	X
		M		9.35E+4	9.35E+4			0
fs-29	$\langle 76212, 2, 24824, 74501, 559 \rangle$	Z	6.81E-254	**9.44E-255**	**9.44E-255**	X	X	X
		M		2.62E+4	2.62E+4			0
Mastermind								
mm_03_08_03	$\langle 1220, 2, 1193, 48, 20 \rangle$	Z	9.79E-8	9.87E-08	9.87E-08	**9.79E-08**$^{*\ (3s)}$	**9.79E-08**$^{*(11s)}$	X
		M		564101	564101			0
mm_03_08_04	$\langle 2288, 2, 2252, 64, 30 \rangle$	Z	8.77E-09	8.19E-09	8.19E-09	**8.77E-09**$^{*\ (1231s)}$	X	X
		M		35101	35101			0
mm_03_08_05	$\langle 3692, 2, 3647, 80, 42 \rangle$	Z	8.89E-11	7.27E-11	7.27E-11	**8.90E-11**$^{*\ (1503s)}$	X	X
		M		10401	10401			0
mm_04_08_03	$\langle 1418, 2, 1391, 48, 22 \rangle$	Z	8.39E-08	8.37E-08	8.37E-08	**8.39E-08**$^{*\ (7s)}$	X	X
		M		379501	379501			0
mm_04_08_04	$\langle 2616, 2, 2580, 64, 33 \rangle$	Z	2.20E-08	**1.84E-08**	**1.84E-08**	1.21E-08	X	X
		M		12901	12901			0
mm_05_08_03	$\langle 1616, 2, 1589, 48, 28 \rangle$	Z	5.29E-07	4.78E-07	4.78E-07	**5.30E-07**$^{*\ (229s)}$	**5.3E-07**$^{*(6194s)}$	X
		M		60201	60201			0
mm_06_08_03	$\langle 1814, 2, 1787, 48, 31 \rangle$	Z	1.79E-08	1.12E-08	1.12E-08	**1.80E-08**$^{*\ (2082s)}$	5.85E-09	X
		M		113301	113301			0
mm_10_08_03	$\langle 2606, 2, 2579, 48, 56 \rangle$	Z	1.92E-07	**5.01E-07**	**5.01E-07**	7.79E-08	2.39E-10	X
		M		10801	10801			0

Table 8: Probability of evidence computed by VEC, EDBP, IJGP-wc-IS and IJGP-wc-SS after 3 hours of CPU time for relational instances. For IJGP-wc-SS and IJGP-wc-IS each cell in the table also reports the number of consistent samples generated in 10 hours. A "*" next to the output of a solver indicates that it solved the problem exactly (before the time-bound expired) followed by the number of seconds it took to solve the instance exactly.

Smoker networks and six mastermind networks from this domain which have between 262 to 76,212 variables. Table 8 summarizes the results.

VEC solves 2 friends and smokers networks exactly while on the remaining instances, it fails to output any answer. EDBP solves one instance exactly while on the remaining instances it either fails or is inferior to IJGP-wc-SS. IJGP-wc-IS is better than IJGP-wc-SS on 3 instances while on the remaining instances it fails to generate a single consistent sample; especially as the instances get larger. The estimates computed by IJGP-wc-SS/LB and IJGP-wc-SS/UB on the other hand are very close to the exact probability of evidence.

VEC solves exactly six out of the eight mastermind instances while on the remaining two instances VEC is worse than IJGP-wc-SS/UB and IJGP-wc-SS/LB. EDBP solves two instances exactly while on the remaining instances it is worse than IJGP-wc-SS/LB and IJGP-wc-SS/UB.

Again, the estimates output by IJGP-wc-SS/LB and IJGP-wc-SS/UB are the same for all the relational instances indicating that our lower and upper approximations have zero bias.

5.4. Results for the Posterior Marginal Tasks

5.4.1. Setup and Evaluation Criteria

We experimented with the following four benchmark domains: (a) The linkage instances (b) The relational instances and (c) The grid instances and (d) The logistics planning instances. We measure the accuracy of the solvers using average Hellinger distance (Kokolakis and Nanopoulos, 2001). Given a mixed network with n variables, let $P(X_i)$ and $A(X_i)$ denote the exact and approximate marginals for a variable X_i, then the average Hellinger distance denoted by Δ is defined as:

$$\Delta = \frac{\sum_{i=1}^{n} \frac{1}{2} \sum_{x_i \in \mathbf{D}_i} (\sqrt{P(x_i)} - \sqrt{A(x_i)})^2}{n} \tag{39}$$

Hellinger distance lies between 0 and 1 and lower bounds the Kullback Leibler distance (Kullback and Leibler, 1951). A Hellinger distance of 0 for a solver indicates that the solver output the exact marginal distribution for each variable while a Hellinger distance of 1 indicates that the solver failed to output any solution.

We chose Hellinger distance because as pointed out in (Kokolakis and Nanopoulos, 2001), it is superior to other choices such as the Kullback-Leibler (KL) distance, the mean squared error and the relative error when zero or infinitesimally small probabilities are present. We do not use the KL distance because it lies between 0 and ∞ and in practice when the exact marginals are 0 or close to it, floating-point precision errors in the exact (or the approximate) solver may yield a false zero when the correct marginal is non-zero and vice versa yielding infinite KL distance[9]. We did compute the error using other commonly used distance measures such as the mean squared error, the relative

[9]Also see for example the results of the recent UAI evaluation (Darwiche et al., 2008). (Dechter and Mateescu, 2003) proved that IJGP (and EDBP) cannot yield marginals having infinite KL distance. However, in many cases these solvers had infinite KL distance because of precision errors.

Problem	$\langle n, k, c, e, w \rangle$		IJGP-wc-SS	IJGP	EPIS	EDBP	IJGP-wc-IS
BN_69	$\langle 777, 7, 228, 78, 47 \rangle$	Δ	**9.4E-04**	3.2E-02	1	8.0E-02	1
		M	6.84E+5				0
BN_70	$\langle 2315, 5, 484, 159, 87 \rangle$	Δ	**2.6E-03**	3.3E-02	1	9.6E-02	1
		M	1.92E+5				0
BN_71	$\langle 1740, 6, 663, 202, 70 \rangle$	Δ	**5.6E-03**	1.9E-02	1	2.5E-02	1
		M	7.46E+4				0
BN_72	$\langle 2155, 6, 752, 252, 86 \rangle$	Δ	**3.6E-03**	7.2E-03	1	1.3E-02	1
		M	1.53E+5				0
BN_73	$\langle 2140, 5, 651, 216, 101 \rangle$	Δ	**2.1E-02**	2.8E-02	1	6.1E-02	1
		M	7.75E+4				0
BN_74	$\langle 749, 6, 223, 66, 45 \rangle$	Δ	6.9E-04	**4.3E-06**	1	4.3E-02	1
		M	2.80E+5				0
BN_75	$\langle 1820, 5, 477, 155, 92 \rangle$	Δ	**8.0E-03**	6.2E-02	1	9.3E-02	1
		M	7.72E+4				0
BN_76	$\langle 2155, 7, 605, 169, 64 \rangle$	Δ	**1.8E-02**	2.6E-02	1	2.7E-02	1
		M	2.52E+4				0

Table 9: Table showing the *Hellinger distance* Δ *between the exact and approximate marginals* for IJGP-wc-SS, IJGP-wc-IS, IJGP, EPIS and EDBP for *Linkage instances from the UAI 2006 evaluation* after 3 hours of CPU time. For IJGP-wc-IS and IJGP-wc-SS, we also report the number of consistent samples M generated in 3 hours.

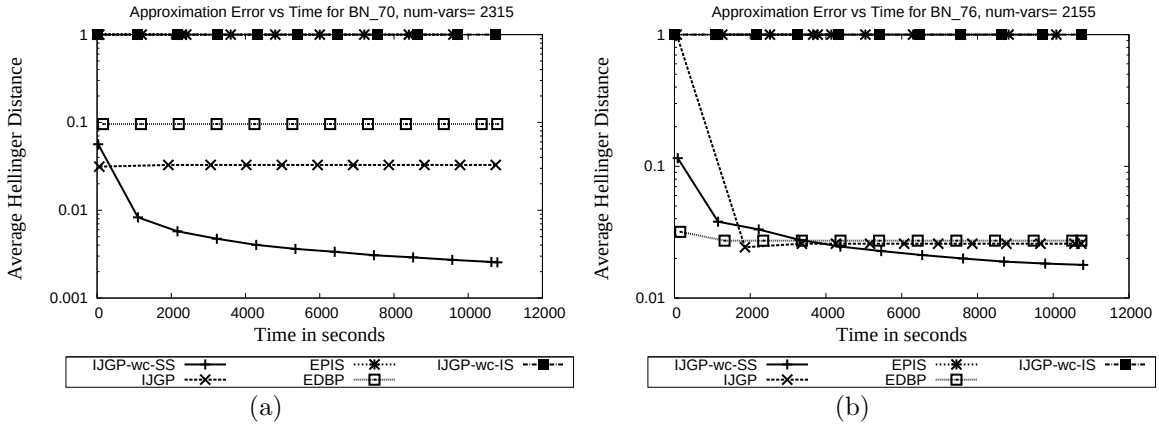


Figure 10: Time versus Hellinger distance Δ between the exact and approximate marginals for IJGP-wc-IS, IJGP-wc-SS, IJGP, EPIS and EDBP for two sample *Linkage instances*.

error and the absolute error. All error measures show similar trends, with the Hellinger distance being the most discriminative.

Finally, for the marginal task, IJGP-wc-SS/LB and IJGP-wc-SS/UB output the same marginals for all benchmarks that we experimented with and therefore we do not distinguish between them. This implies that our lower and upper approximations of the backtrack free probability are indeed quite strong and have negligible or zero bias. *Therefore, for the rest of this subsection, we will refer to IJGP-wc-SS/LB and IJGP-wc-SS/UB as IJGP-wc-SS.*

5.4.2. Linkage instances

In Table 9, we report the average Hellinger distance between exact and approximate marginals for the linkage instances from the UAI 2006 evaluation (Bilmes and Dechter, 2006). We do not report on the pedigree instances from the UAI 2008 evaluation (Darwiche et al., 2008) because their exact marginals are not known. IJGP-wc-SS is more

Problem	$\langle n, k, c, e, w \rangle$		IJGP-wc-SS	IJGP	EPIS	EDBP	IJGP-wc-IS
Friends and Smokers							
fs-04	$\langle 262, 2, 74, 226, 12 \rangle$	Δ	5.4E-05	**4.6E-08**	1	6.4E-02	3.6E-06
		M	1.00E+6				2.17E+8
fs-07	$\langle 1225, 2, 371, 1120, 35 \rangle$	Δ	**1.4E-02**	1.6E-02	1	3.0E-02	1
		M	1.00E+6				0
fs-10	$\langle 3385, 2, 1055, 3175, 71 \rangle$	Δ	1.2E-02	**6.3E-03**	1	2.7E-02	1
		M	8.51E+5				0
fs-13	$\langle 7228, 2, 2288, 6877, 119 \rangle$	Δ	2.0E-02	6.5E-03	1	2.3E-02	**1.4E-04**
		M	5.41E+5				4.67E+7
fs-16	$\langle 13240, 2, 4232, 12712, 171 \rangle$	Δ	1.2E-03	6.8E-03	1	1.7E-02	**2.1E-05**
		M	1.79E+5				1.37E+7
fs-19	$\langle 21907, 2, 7049, 21166, 243 \rangle$	Δ	**3.1E-03**	8.8E-03	1	1	1
		M	1.90E+5				0
fs-22	$\langle 33715, 2, 10901, 32725, 335 \rangle$	Δ	**2.5E-03**	8.6E-03	1	1	1
		M	1.18E+5				0
fs-25	$\langle 49150, 2, 15950, 47875, 431 \rangle$	Δ	**2.5E-03**	8.4E-03	1	1	1
		M	9.23E+4				0
fs-28	$\langle 68698, 2, 22358, 67102, 527 \rangle$	Δ	**1.3E-03**	7.4E-03	1	1	1
		M	9.35E+4				0
fs-29	$\langle 76212, 2, 24824, 74501, 559 \rangle$	Δ	**1.9E-03**	7.0E-03	1	1	1
		M	2.62E+4				0
Mastermind							
mm_03_08_03	$\langle 1220, 2, 1193, 48, 20 \rangle$	Δ	**1.1E-03**	3.8E-02	1	3.8E-01	1
		M	5.64E+5				0
mm_03_08_04	$\langle 2288, 2, 2252, 64, 30 \rangle$	Δ	**1.1E-02**	4.4E-02	1	1	1
		M	3.51E+4				0
mm_03_08_05	$\langle 3692, 2, 3647, 80, 42 \rangle$	Δ	4.0E-02	**3.2E-02**	1	1	1
		M	1.04E+4				0
mm_04_08_04	$\langle 2616, 2, 1391, 64, 33 \rangle$	Δ	**3.1E-02**	3.5E-02	1	1	1
		M	1.29E+4				0
mm_05_08_03	$\langle 1616, 2, 2580, 48, 28 \rangle$	Δ	**1.0E-02**	3.6E-02	1	4.0E-02	1
		M	6.02E+4				0
mm_06_08_03	$\langle 1814, 2, 1787, 48, 31 \rangle$	Δ	**4.7E-03**	3.3E-02	5.6E-01	3.2E-01	1
		M	1.13E+5				0
mm_10_08_03	$\langle 2606, 2, 2579, 48, 56 \rangle$	Δ	**3.9E-02**	5.3E-02	1	8.3E-02	1
		M	1.08E+4				0

Table 10: Table showing the *Hellinger distance* Δ *between the exact and approximate marginals* for IJGP-wc-SS, IJGP-wc-IS, IJGP, EPIS and EDBP for *relational instances* after 3 hours of CPU time. For IJGP-wc-IS and IJGP-wc-SS, we also report the number of consistent samples M generated in 3 hours.

accurate than IJGP which in turn is more accurate than EDBP on 7 out of the 8 instances. We can clearly see the relationship between treewidth and the performance of propagation based and sampling based techniques. When the treewidth is relatively small (on BN_74), a propagation based scheme like IJGP is more accurate than IJGP-wc-SS but as the treewidth increases, there is one to two orders of magnitude difference in the Hellinger distance. EPIS and IJGP-wc-IS do not generate even a single consistent sample in 3 hours of CPU time and therefore their average Hellinger distance is 1[10]. In Figure 10, we demonstrate the superior anytime performance of IJGP-wc-SS compared with other solvers.

[10]Unfortunately, the EPIS program does not output the number of consistent samples that were used in computing the marginals and therefore we do not report it here.

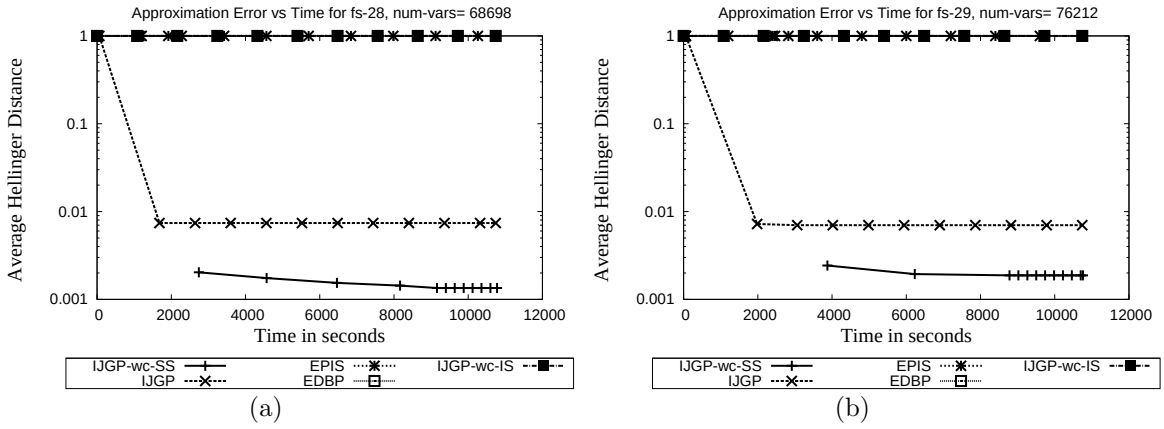


Figure 11: Time versus Hellinger distance Δ between the exact and approximate marginals for IJGP-wc-IS, IJGP-wc-SS, IJGP, EPIS and EDBP for two sample *Friends and Smokers networks*.

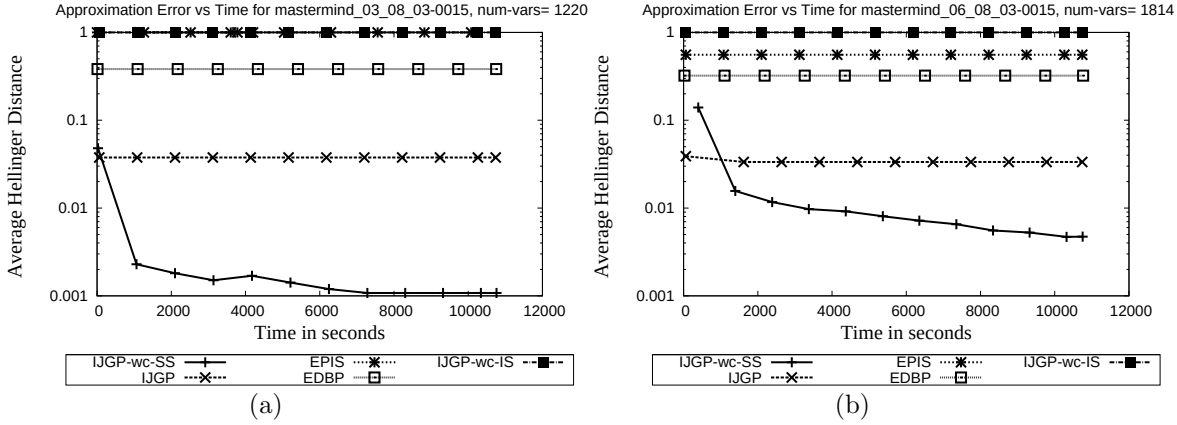


Figure 12: Time versus Hellinger distance Δ between the exact and approximate marginals for IJGP-wc-IS, IJGP-wc-SS, IJGP, EPIS and EDBP for two sample *Mastermind networks*.

5.4.3. Relational Instances

We experimented again with the 10 Friends and Smoker networks and 6 mastermind networks from the relational Bayesian networks domain (Chavira et al., 2006). Table 10 shows the Hellinger distance between the exact and approximate marginals after 3 hours of CPU time for each solver.

On the small friends and smoker networks, fs-04 to fs-13, IJGP performs better than IJGP-wc-SS. However, on large networks which have between 13240 and 76212 variables, and treewidth between 12712 to 74501, IJGP-wc-SS performs better than IJGP. EDBP is slightly worse than IJGP and runs out of memory on large instances, indicated by a Hellinger distance of 1. EPIS is not able to generate a single consistent sample in 3 hours of CPU time indicated by Hellinger distance of 1 for all instances. IJGP-wc-IS fails on all but three instances. On these three instances, IJGP-wc-IS has smaller error than IJGP-wc-SS because it generates far more consistent samples than IJGP-wc-SS (by a factor of 10-200).

Discussion: The small sample size of IJGP-wc-SS as compared with its pure sampling

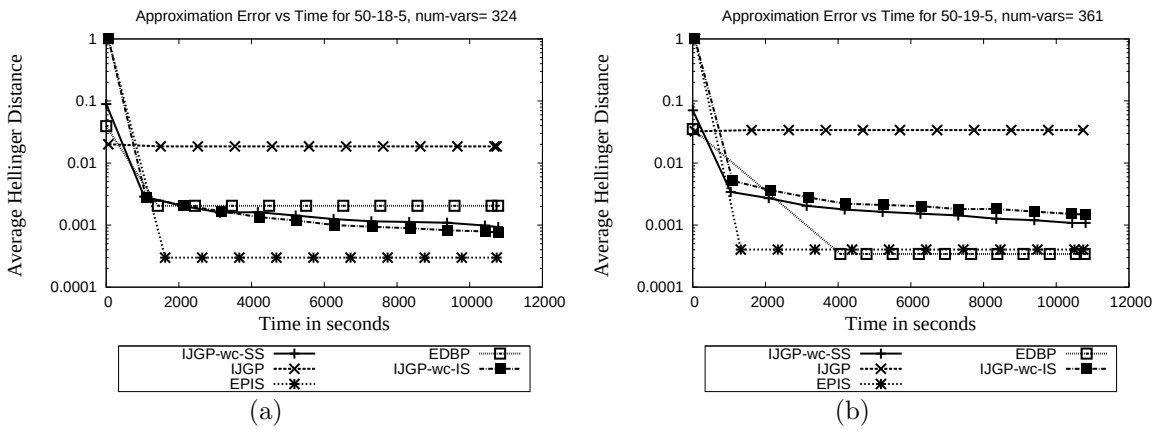


Figure 13: Time versus Hellinger distance Δ between the exact and approximate marginals for IJGP-wc-IS, IJGP-wc-SS, IJGP, EPIS and EDBP for two sample *Grid instances with deterministic ratio=50%.*

counterpart IJGP-wc-IS is due to the overhead of solving a satisfiability formula via backtracking search to generate a sample. IJGP-wc-IS, on the other hand, uses the relational consistency (Dechter, 2003; Dechter and Mateescu, 2003) power of IJGP to reduce rejection as a *pre-processing step* (Gogate and Dechter, 2005). This highlights one of the advantages of using constraint-based inference to determine the inconsistencies before sampling rather than combining search with sampling. Such constraint based inference schemes are however not scalable and as we can see they fail to yield even a single consistent sample for the larger instances (fs-19 to fs-29). Thus, to take advantage of larger sample size, we can use a simple strategy in which we run conventional sampling for a few minutes and resort to SampleSearch only when pure sampling does not produce any consistent samples.

On the mastermind networks, IJGP-wc-SS is the superior scheme followed by IJGP. EPIS fails to output even a single consistent sample in 3 hours on 6 out of the 7 instances while IJGP-wc-IS fails on all instances. EDBP is slightly worse than IJGP on 5 out of the 6 instances. Figures 11 and 12 show the anytime performance of the solvers demonstrating the clear superiority of IJGP-wc-SS.

5.4.4. Grid Networks

The Grid Bayesian networks are available from the authors of Cachet (Sang et al., 2005). A grid Bayesian network is a $s \times s$ grid, where there are two directed edges from a node to its neighbors right and down. The upper-left node is a source, and the bottom-right node is a sink. The sink node is the evidence node. The deterministic ratio p is a parameter specifying the fraction of nodes that are deterministic (functional in this case), that is, whose values are determined given the values of their parents. The grid instances are designated as $p-s$. For example, the instance $50-18$ indicates a grid of size 18 in which 50% of the nodes are deterministic or functional. Table 11 shows the Hellinger distance after 3 hours of CPU time for each solver. Time versus approximation error plots are shown for six sample instances in Figures 13 through 15.

On grids with deterministic ratio of 50%, EPIS is the best performing scheme on all but two instances. On most instances, IJGP-wc-IS yeilds marginals having smaller error

Problem	$\langle n,k,c,e,w\rangle$		IJGP-wc-SS	IJGP	EPIS	EDBP	IJGP-wc-IS
Deterministic Ratio = 50%							
50-12-5	$\langle 144,2,62,1,16\rangle$	Δ	4.3E-04	**3.2E-07**	2.6E-04	2.5E-02	1.5E-04
		M	1.90E+6				1.23E+8
50-14-5	$\langle 196,2,93,1,20\rangle$	Δ	4.9E-04	1.8E-02	**1.2E-04**	4.0E-02	2.1E-04
		M	9.37E+5				8.90E+7
50-15-5	$\langle 225,2,111,1,23\rangle$	Δ	4.9E-04	1.0E-02	**2.3E-04**	6.1E-02	6.5E-04
		M	5.24E+5				7.68E+7
50-17-5	$\langle 289,2,138,1,25\rangle$	Δ	8.0E-04	2.1E-02	**2.0E-04**	3.6E-03	1.0E-03
		M	4.34E+5				5.82E+7
50-18-5	$\langle 324,2,153,1,27\rangle$	Δ	9.3E-04	1.9E-02	**3.0E-04**	2.1E-03	7.6E-04
		M	3.46E+5				5.15E+7
50-19-5	$\langle 361,2,172,1,28\rangle$	Δ	1.1E-03	3.4E-02	4.0E-04	**3.4E-04**	1.5E-03
		M	2.87E+5				2.80E+7
Deterministic Ratio = 75%							
75-16-5	$\langle 256,2,193,1,24\rangle$	Δ	6.5E-04	**2.5E-07**	1.7E-04	7.8E-02	1.4E-04
		M	9.74E+5				7.11E+7
75-17-5	$\langle 289,2,217,1,25\rangle$	Δ	1.4E-03	**2.6E-07**	2.7E-04	1.2E-03	1.6E-04
		M	7.15E+5				5.41E+7
75-18-5	$\langle 324,2,245,1,27\rangle$	Δ	1.2E-03	3.9E-02	2.0E-04	5.0E-03	**1.9E-04**
		M	4.47E+5				5.23E+7
75-19-5	$\langle 361,2,266,1,28\rangle$	Δ	9.0E-03	4.3E-02	2.5E-04	**6.7E-05**	1.9E-04
		M	4.07E+5				3.93E+7
75-20-5	$\langle 400,2,299,1,30\rangle$	Δ	6.2E-04	**3.1E-07**	1.9E-04	1.7E-02	2.8E-04
		M	4.10E+5				2.64E+7
75-21-5	$\langle 441,2,331,1,32\rangle$	Δ	1.9E-03	**2.9E-07**	2.8E-04	1.5E-02	6.2E-04
		M	3.13E+5				2.33E+7
75-22-5	$\langle 484,2,361,1,35\rangle$	Δ	3.2E-03	2.3E-02	**2.6E-04**	2.0E-02	5.4E-04
		M	2.67E+5				2.12E+7
75-23-5	$\langle 529,2,406,1,35\rangle$	Δ	2.0E-03	4.8E-02	**2.3E-04**	2.4E-02	7.1E-04
		M	1.75E+5				1.77E+7
75-24-5	$\langle 576,2,442,1,38\rangle$	Δ	8.4E-03	4.3E-02	**2.6E-04**	3.5E-02	8.9E-04
		M	1.29E+5				2.61E+7
75-26-5	$\langle 676,2,506,1,44\rangle$	Δ	2.4E-02	5.1E-02	**3.5E-04**	5.1E-02	1.4E-03
		M	1.25E+5				2.20E+7
Deterministic Ratio = 90%							
90-20-5	$\langle 400,2,356,1,30\rangle$	Δ	1.6E-03	**2.7E-07**	2.5E-04	3.7E-02	6.5E-05
		M	8.32E+5				4.77E+7
90-22-5	$\langle 484,2,430,1,35\rangle$	Δ	4.6E-04	**2.8E-07**	1.5E-04	5.1E-02	1.0E-04
		M	4.42E+5				3.97E+7
90-23-5	$\langle 529,2,468,1,35\rangle$	Δ	2.8E-04	**3.2E-07**	3.9E-04	1.9E-02	7.0E-05
		M	6.70E+5				4.00E+7
90-24-5	$\langle 576,2,528,1,38\rangle$	Δ	5.0E-04	**3.9E-07**	3.5E-04	2.8E-02	9.2E-05
		M	7.01E+5				2.29E+7
90-25-5	$\langle 625,2,553,1,39\rangle$	Δ	**2.7E-07**	**2.7E-07**	3.4E-04	4.6E-02	**2.7E-07**
		M	7.04E+5				2.57E+7
90-26-5	$\langle 676,2,597,1,44\rangle$	Δ	1.0E-03	**1.9E-06**	2.3E-04	3.9E-02	1.9E-04
		M	4.13E+5				2.90E+7
90-34-5	$\langle 1156,2,1048,1,65\rangle$	Δ	8.6E-04	**1.8E-07**	3.9E-04	4.1E-02	6.3E-04
		M	2.80E+5				1.37E+7
90-38-5	$\langle 1444,2,1300,1,69\rangle$	Δ	1.6E-02	**4.3E-07**	1.7E-03	1.6E-01	1.0E-03
		M	1.15E+5				7.08E+6

Table 11: Table showing the *Hellinger distance* Δ *between the exact and approximate marginals* for IJGP-wc-SS, IJGP-wc-IS, IJGP, EPIS and EDBP for *Grid networks* after 3 hours of CPU time. For IJGP-wc-IS and IJGP-wc-SS, we also report the number of consistent samples M generated in 3 hours.

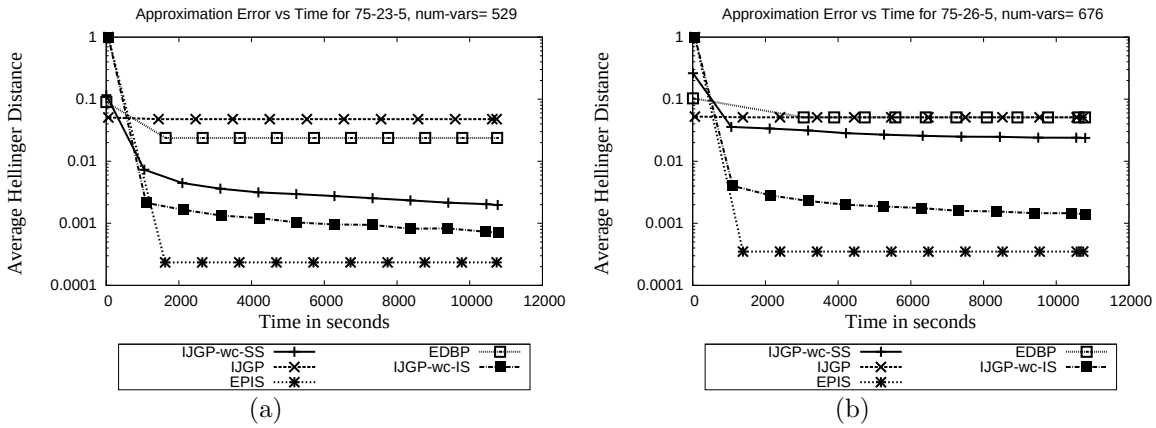


Figure 14: Time versus Hellinger distance Δ between the exact and approximate marginals for IJGP-wc-IS, IJGP-wc-SS, IJGP, EPIS and EDBP for two sample *Grid instances with deterministic ratio=75%.*

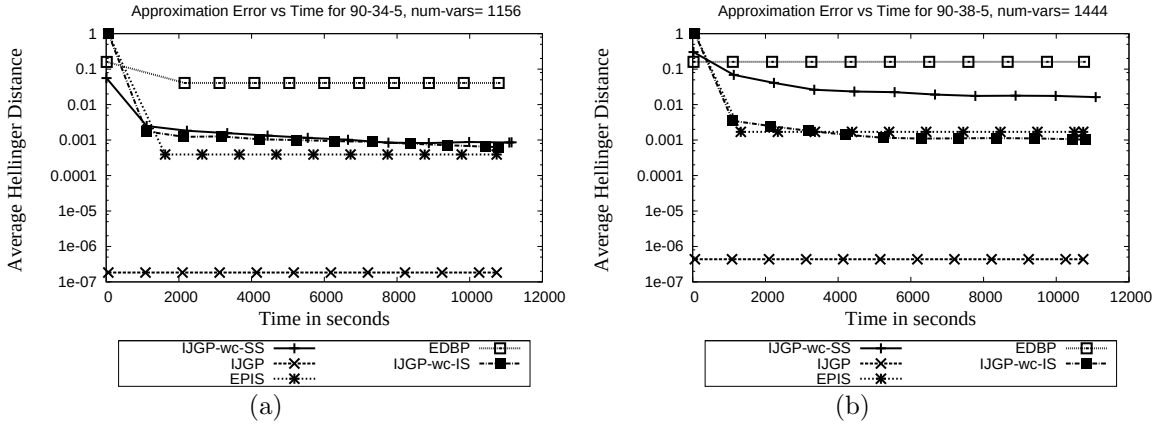


Figure 15: Time versus Hellinger distance Δ between the exact and approximate marginals for IJGP-wc-IS, IJGP-wc-SS, IJGP, EPIS and EDBP for two sample *Grid instances with deterministic ratio=90%.*

Problem	$\langle n, k, c, e, w \rangle$		IJGP-wc-SS	IJGP	EPIS	EDBP	IJGP-wc-IS
log-1	$\langle 4724, 2, 3785, 3785, 22 \rangle$	Δ	2.2E-05	**0*** $^{(2s)}$	1	1	1
		M	1.35E+8				0
log-2	$\langle 26114, 2, 24777, 24777, 51 \rangle$	Δ	**8.6E-04**	9.8E-03	1	1	1
		M	1.49E+6				0
log-3	$\langle 30900, 2, 29487, 29487, 56 \rangle$	Δ	**1.2E-04**	7.5E-03	1	1	1
		M	1.05E+5				0
log-4	$\langle 23266, 2, 20963, 20963, 52 \rangle$	Δ	**2.3E-02**	1.8E-01	1	1	1
		M	1.03E+5				0
log-5	$\langle 32235, 2, 29534, 29534, 51 \rangle$	Δ	**8.6E-03**	1.2E-02	1	1	1
		M	9.73E+3				0

Table 12: Table showing the *Hellinger distance* Δ *between the exact and approximate marginals* for IJGP-wc-SS, IJGP-wc-IS, IJGP, EPIS and EDBP for *Logistics planning instances* after 3 hours of CPU time. For IJGP-wc-IS and IJGP-wc-SS, we also report the number of consistent samples M generated in 3 hours.

than IJGP-wc-SS. On four out of the six instances, the sampling schemes yield smaller error than EDBP and IJGP. There is two orders of magnitude difference between IJGP-wc-SS and EDBP/IJGP while there is one order of magnitude difference between EPIS and IJGP-wc-IS and IJGP-wc-SS.

On grids with deterministic ratio of 75%, IJGP is best on four out of the six smaller grids (up to size 21). EPIS dominates on the larger grids (size 22-26). IJGP-wc-IS is worse than IJGP on the smaller grids (up to size 21) but dominates IJGP on larger grids. IJGP-wc-IS is consistently worse than EPIS and we suspect that this is due to the use of adaptive importance sampling in EPIS (Cheng and Druzdzel, 2000; Yuan and Druzdzel, 2006) in which proposal distribution is updated periodically based on the generated samples yielding a series of proposal distributions that with time get closer and closer to the posterior distribution[11]. We see that there is an order of magnitude difference between IJGP-wc-IS and IJGP-wc-SS because the estimates of IJGP-wc-IS are based on larger number of samples (by a factor of 60-100) as compared with IJGP-wc-SS.

On grids with deterministic ratio of 90%, IJGP is the superior scheme. IJGP-wc-IS is slightly better than EPIS which in turn is slightly better than IJGP-wc-SS. EDBP is the least accurate scheme. Again, we see that there is a two orders of magnitude difference between the sample size of IJGP-wc-IS and IJGP-wc-SS.

5.4.5. Logistics Planning instances

Our last domain is that of logistics planning. Given prior probabilities on actions and facts, the task is to compute marginal distribution of each variable. Goals and initial conditions are observed true. Bayesian networks are generated from the plan graphs, where additional nodes (all observed false) are added to represent mutex, action-effect and preconditions of actions. These benchmarks are available from the authors of Cachet (Sang et al., 2005).

Table 12 summarizes the results. IJGP-wc-IS, EPIS and EDBP fail on all instances. IJGP solves the log-1 instance exactly as indicated by a * in Table 12 while on the

[11]The difference in performance between IJGP-wc-IS and EPIS may also be due to larger sample size of EPIS but as pointed out earlier EPIS does not output the number of consistent samples used to compute the marginals.

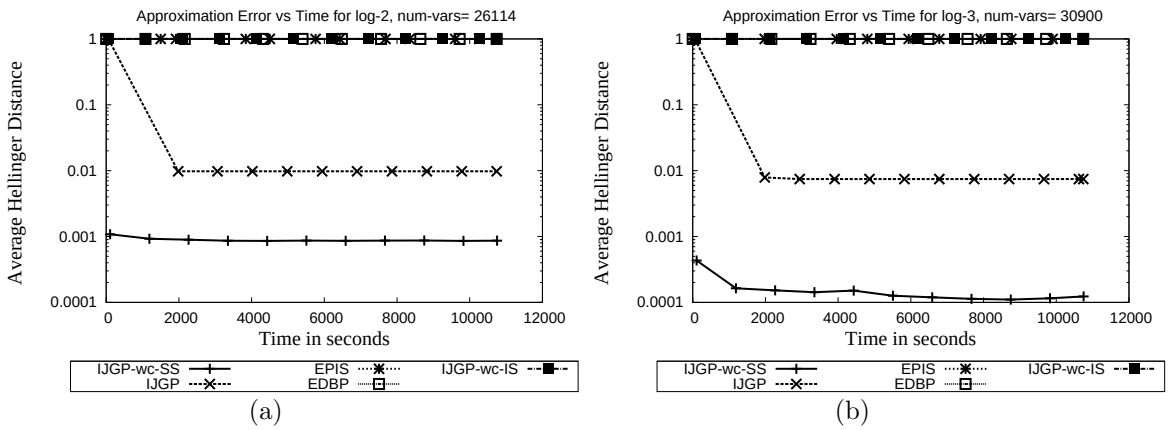


Figure 16: Time versus Hellinger distance Δ between the exact and approximate marginals for IJGP-wc-IS, IJGP-wc-SS, IJGP, EPIS and EDBP for two sample *Logistics planning instances*.

remaining instances, IJGP-wc-SS is more accurate than IJGP. In Figure 16, we demonstrate the superior anytime performance of IJGP-wc-SS as compared with the other schemes.

5.5. Summary of Experimental Evaluation

To summarize, we implemented SampleSearch on top of an advanced importance sampling technique IJGP-wc-IS presented in our previous work (Gogate and Dechter, 2005); yielding the IJGP-wc-SS technique. The search was implemented using minisat (Sorensson and Een, 2005). For model counting, we compared IJGP-wc-SS with three other approximate solution counters available in literature: ApproxCount (Wei et al., 2004), SampleCount (Gomes et al., 2007) and Relsat (Roberto J. Bayardo and Pehoushek, 2000) as well as with IJGP-wc-IS on three benchmarks: (a) Latin Square instances (b) Langford instances and (c) FPGA-routing instances. We found that on most instances, IJGP-wc-SS yields solution counts which are closer to the true counts by a few orders of magnitude than those output by SampleCount and by several orders of magnitude than those output by ApproxCount and Relsat. IJGP-wc-IS fails to generate even a single consistent sample on all the SAT instances in 10 hours of CPU time clearly demonstrating the usefulness of IJGP-wc-SS for deriving meaningful approximations in presence of significant amount of determinism.

For the problem of computing the probability of evidence in a Bayesian network and the partition function in a Markov network, we compared IJGP-wc-SS with Variable Elimination and Conditioning (VEC) (Dechter, 1999) and an advanced generalized belief propagation scheme called Edge Deletion Belief Propagation (EDBP) (Choi and Darwiche, 2006) on two benchmark domains: (a) linkage analysis and (b) relational Bayesian networks. We found that on most instances the estimates output by IJGP-wc-SS were closer to the exact answer than those output by EDBP. VEC solved some instances exactly, while on the remaining instances it was substantially inferior. IJGP-wc-IS was superior to IJGP-wc-SS whenever it was able to generate consistent samples. However, on a majority of the instances it simply failed to yield any consistent samples.

For the posterior marginal task, we experimented with linkage analysis benchmarks, partially deterministic grid benchmarks, relational benchmarks and logistics planning benchmarks. We compared the accuracy of IJGP-wc-SS using the Hellinger distance with four other schemes: two generalized belief propagation schemes of Iterative Join Graph Propagation (Dechter et al., 2002) and Edge Deletion Belief Propagation (Choi and Darwiche, 2006), an adaptive importance sampling scheme called Evidence Pre-propagated Importance Sampling (EPIS) (Yuan and Druzdzel, 2006) and IJGP-wc-IS. We found that except on the grid instances, IJGP-wc-SS consistently yielded estimates having smaller error than EDBP and IJGP. Whenever the vanilla sampling schemes IJGP-wc-IS and EPIS did not fail, they generated more consistent samples and had smaller error than IJGP-wc-SS. On the remaining instances, IJGP-wc-SS was clearly superior. Thus, we suggest the following simple strategy. We first run the given vanilla sampling scheme for a few seconds and check if it is able to generate consistent samples. If it does, then we continue with the scheme. Otherwise, we abandon it and run SampleSearch.

6. Conclusion

The paper presented the SampleSearch scheme for improving the performance of importance sampling in mixed probabilistic and deterministic graphical models. It is well known that on such graphical models, importance sampling performs quite poorly because of the rejection problem. SampleSearch remedies the rejection problem by interleaving random sampling with systematic backtracking. Specifically, when sampling variables one by one via logic sampling (Pearl, 1988), instead of rejecting a sample when its inconsistency is detected, SampleSearch backtracks to the previous variable, modifies the proposal distribution to reflect the inconsistency and continues this process until a consistent sample is found.

We showed that SampleSearch can be viewed as a systematic search technique whose value selection is stochastically guided by sampling from a distribution. This view enables us to integrate any systematic SAT/CSP solver within SampleSearch (with minor modifications). Indeed, in our experiments, we used an advanced SAT solver called minisat (Sorensson and Een, 2005). Thus, advances in the systematic search community whose primary focus is solving "yes/no" type NP-complete problems can be leveraged through SampleSearch for approximating much harder #P-complete problems in Bayesian inference.

We characterized the sampling distribution of SampleSearch using the notion of the backtrack-free distribution, which is basically a modification of the proposal distribution from which all inconsistent partial assignments along a specified order are removed. When the backtrack-free probability for a given sampled assignment is too complex to compute, we proposed two approximations, which bound the backtrack-free probability from above and below and yield asymptotically unbiased estimates of the weighted counts and marginals.

We performed an extensive empirical evaluation on several benchmark graphical models and our results clearly demonstrate that our lower and upper approximations were very accurate on most benchmarks and that overall SampleSearch was consistently

superior to other state-of-the-art schemes on domains having a substantial amount of determinism.

Specifically, on probabilistic graphical models, we showed that state-of-the-art importance sampling techniques such as EPIS (Yuan and Druzdzel, 2006) and IJGP-wc-IS (Gogate and Dechter, 2005) which reason about determinism in a limited way are unable to generate a single consistent sample on several hard linkage analysis and relational benchmarks. In such cases, SampleSearch is the only alternative importance sampling technique to date.

SampleSearch is also superior to generalized belief propagation schemes like Iterative Join Graph Propagation (IJGP) (Dechter et al., 2002) and Edge Deletion Belief Propagation (EDBP) (Choi and Darwiche, 2006). In theory, these propagation techniques are anytime, whose approximation quality can be improved by increasing their i-bound. However, their time and space complexity is exponential in i and in practice, beyond a certain i-bound (typically > 25), their memory requirement becomes a major bottleneck. Consequently, as we saw, on most benchmarks IJGP and EDBP quickly converge to an estimate which they are unable to improve with time. SampleSearch, being an importance sampling technique improves with time, and as we demonstrated yields superior anytime performance than IJGP and EDBP.

Finally, on the problem of counting solutions of a SAT/CSP, we showed that SampleSearch is slightly better than the recently proposed SampleCount (Gomes et al., 2007) technique and substantially better than ApproxCount (Wei et al., 2004) and Relsat (Roberto J. Bayardo and Pehoushek, 2000).

SampleSearch leaves plenty of room for future improvements, which are likely to make it more cost effective in practice. For instance, to generate samples, we solve the same SAT/CSP problem multiple times. Therefore, various goods and no-goods (i.e. knowledge about the problem space) learnt while generating one sample may be used to speed-up the search for a solution while generating the next sample. How to achieve this in a principled and structured way is an important theoretical and practical question. Some initial related research on solving the similar SAT problems has appeared in the bounded model checking community (Eén and Sörensson, 2003) and can be applied to improve SampleSearch's performance. A second line of improvement is a more efficient algorithm for compactly storing and combining various DFS traces used for deriving the lower and upper approximations. Currently, we store all DFS traces using an OR tree. However, the OR tree is very inefficient and we could easily use the AND/OR search space (Dechter and Mateescu, 2007) to store the traces. Borrowing ideas from the literature on ordered binary decision diagrams (OBDDs) (Bryant, 1986), we could even merge together isomorphic traces, and eliminate redundancy to further compact our representation. A third line of future research is to use adaptive importance sampling (Cheng, 1997; Ortiz and Kaelbling, 2000; Yuan and Druzdzel, 2006; Moral and Salmerón, 2005). In adaptive importance sampling, one updates the proposal distribution based on the generated samples; so that with every update the proposal gets closer and closer to the desired posterior distribution. Because we already store the DFS traces of the generated samples in SampleSearch, one could use them to dynamically update and learn the proposal distribution.

Acknowledgements

This work was supported in part by the NSF under award numbers IIS-0331707, IIS-0412854 and IIS-0713118 and by the NIH grant R01-HG004175-02.

References

R. Dechter, D. Larkin, Hybrid Processing of Beliefs and Constraints, in: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence (UAI), 112–119, 2001.

D. Larkin, R. Dechter, Bayesian Inference in the Presence of Determinism, in: Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS), 2003.

R. Dechter, R. Mateescu, Mixtures of Deterministic-Probabilistic Networks and their AND/OR Search Space, in: Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI), 120–129, 2004.

R. Mateescu, R. Dechter, Mixed Deterministic and Probabilistic Networks, Annals of Mathematics and Artificial Intelligence (AMAI); Special Issue: Probabilistic Relational Learning (to appear) .

J. Pearl, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann, 1988.

R. Dechter, Constraint Processing, Morgan Kaufmann, 2003.

A. W. Marshall, The use of multi-stage sampling schemes in Monte Carlo computations, In Symposium on Monte Carlo Methods (1956) 123–140.

R. Y. Rubinstein, Simulation and the Monte Carlo Method, John Wiley & Sons Inc., 1981.

J. Geweke, Bayesian Inference in Econometric Models Using Monte Carlo Integration, Econometrica 57 (6) (1989) 1317–39.

V. Gogate, R. Dechter, Approximate Inference Algorithms for Hybrid Bayesian Networks with Discrete Constraints, in: Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI), 209–216, 2005.

J. S. Yedidia, W. T. Freeman, Y. Weiss, Constructing Free Energy Approximations and Generalized Belief Propagation Algorithms, IEEE Transactions on Information Theory 51 (2004) 2282–2312.

R. Dechter, K. Kask, R. Mateescu, Iterative Join Graph propagation, in: Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence (UAI), Morgan Kaufmann, 128–136, 2002.

B. Bidyuk, R. Dechter, Cutset Sampling for Bayesian Networks, Journal of Artificial Intelligence Research (JAIR) 28 (2007) 1–48.

N. Sorensson, N. Een, Minisat v1.13-A SAT Solver with Conflict-Clause Minimization, in: SAT 2005 competition, 2005.

W. Wei, J. Erenrich, B. Selman, Towards Efficient Sampling: Exploiting Random Walk Strategies, in: Proceedings of the Nineteenth National Conference on Artificial Intelligence, 670–676, 2004.

C. P. Gomes, J. Hoffmann, A. Sabharwal, B. Selman, From Sampling to Model Counting, in: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), 2293–2299, 2007.

J. Roberto J. Bayardo, J. D. Pehoushek, Counting Models Using Connected Components, in: Proceedings of 17th National Conference on Artificial Intelligence (AAAI), 157–162, 2000.

R. Dechter, Bucket elimination: A unifying framework for reasoning, Artificial Intelligence 113 (1999) 41–85.

A. Choi, A. Darwiche, An Edge Deletion Semantics for Belief Propagation and its Practical Impact on Approximation Quality, in: Proceedings of The Twenty-First National Conference on Artificial Intelligence (AAAI), 1107–1114, 2006.

M. Fishelson, D. Geiger, Optimizing exact genetic linkage computations, in: Proceedings of the seventh annual international conference on Research in computational molecular biology (RECOMB), 114–121, 2003.

M. D. Chavira, A. Darwiche, M. Jaeger, Compiling Relational Bayesian networks for exact inference, International Journal of Approximate Reasoning 42 (1-2) (2006) 4–20.

C. Yuan, M. J. Druzdzel, Importance sampling algorithms for Bayesian networks: Principles and performance, Mathematical and Computer Modelling 43 (9-10) (2006) 1189–1207, ISSN 0895-7177.

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

V. Gogate, R. Dechter, Approximate Counting by Sampling the Backtrack-free Search Space, in: Proceedings of 22nd Conference on Artificial Intelligence (AAAI), 198–203, 2007b.

J. Liu, Monte-Carlo strategies in scientific computing, Springer-Verlag, New York, 2001.

E. C. Freuder, A Sufficient Condition for Backtrack-Free Search, Journal of the ACM 29 (1) (1982) 24–32.

T. Walsh, SAT v CSP, in: Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming, Springer-Verlag, London, UK, ISBN 3-540-41053-8, 441–456, 2000.

K. Pipatsrisawat, A. Darwiche, RSat 2.0: SAT Solver Description, Tech. Rep. D–153, Automated Reasoning Group, Computer Science Department, UCLA, 2007.

J. Cheng, M. J. Druzdzel, AIS-BN: An Adaptive Importance Sampling Algorithm for Evidential Reasoning in Large Bayesian Networks., Journal of Artificial Intelligence Research (JAIR) 13 (2000) 155–188.

K. Kask, R. Dechter, J. Larrosa, A. Dechter, Unifying tree decompositions for reasoning in graphical models, Artificial Intelligence 166 (1-2) (2005) 165–193.

G. Casella, C. P. Robert, Rao-Blackwellisation of sampling schemes, Biometrika 83 (1) (1996) 81–94, doi:10.1093/biomet/83.1.81.

B. Bidyuk, R. Dechter, On finding minimal w-cutset problem, in: Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence (UAI), 43–50, 2004.

W. Wei, B. Selman, A New Approach to Model Counting, in: Proceedings of Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT), 324–339, 2005.

L. G. Valiant, The complexity of enumeration and reliability problems, Siam Journal of Computation 8 (3) (1987) 105–117.

T. Sang, P. Beame, H. Kautz, Heuristics for Fast Exact Model Counting, in: Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT), 226–240, 2005.

B. Selman, H. Kautz, B. Cohen, Noise strategies for local search, in: Proceedings of the Eleventh National Conference on Artificial Intelligence, 337–343, 1994.

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

A. Darwiche, R. Dechter, A. Choi, V. Gogate, L. Otten, Results from the Probablistic Inference Evaluation of UAI'08, Available online at: http://graphmod.ics.uci.edu/uai08/Evaluation/Report, 2008.

R. Dechter, V. Gogate, L. Otten, R. Marinescu, R. Mateescu, Graphical Model Algorithms at UC Irvine, website: http://graphmod.ics.uci.edu/group/Software, 2009.

C. Gomes, D. Shmoys, Completing Quasigroups or Latin Squares: A Structured Graph Coloring Problem, in: Proceedings of the Computational Symposium on Graph Coloring and Extensions, 2002.

T. Ritter, Latin Squares: A Literature Survey, Available online at: http://www.ciphersbyritter.com/RES/LATSQ.HTM .

T. Walsh, Permutation Problems and Channelling Constraints, in: Proceedings of the 8th International Conference on Logic Programming and Automated Reasoning (LPAR), 377–391, 2001.

V. Gogate, B. Bidyuk, R. Dechter, Studies in Lower Bounding Probability of evidence using the Markov Inequality, in: Proceedings of 23rd Conference on Uncertainty in Artificial Intelligence (UAI), 141–148, 2007.

L. Simon, D. L. Berre, E. Hirsch, The SAT 2002 Competition, Annals of Mathematics and Artificial Intelligence(AMAI) 43 (2005) 307–342.

J. Ott, Analysis of Human Genetic Linkage, The Johns Hopkins University Press, Baltimore, Maryland, 1999.

J. Bilmes, R. Dechter, Evaluation of Probabilistic Inference Systems of UAI'06, Available online at http://ssli.ee.washington.edu/ bilmes/uai06InferenceEvaluation/, 2006.

M. Chavira, A. Darwiche, On Probabilistic Inference by Weighted Model Counting, Artificial Intelligence 172 (6–7) (2008) 772–799.

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

S. Kullback, R. A. Leibler, On Information and Sufficiency, The Annals of Mathematical Statistics 22 (1) (1951) 79–86.

R. Dechter, R. Mateescu, A Simple Insight into Iterative Belief Propagation's Success, Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence (UAI) (2003) 175–183.

N. Eén, N. Sörensson, Temporal Induction by Incremental SAT Solving, Electronic Notes in Theoretical Computer Science 89 (4) (2003) 543–560, ISSN 1571-0661.

R. Dechter, R. Mateescu, AND/OR Search Spaces for Graphical Models, Artificial Intelligence 171 (2-3) (2007) 73–106.

R. E. Bryant, Graph-Based Algorithms for Boolean Function Manipulation, IEEE Transancations on Computers 35 (8) (1986) 677–691.

J.-F. Cheng, Iterative Decoding, Ph.D. thesis, California Institute of Technology (Electrical Engineering), 1997.

L. Ortiz, L. Kaelbling, Adaptive Importance Sampling for Estimation in Structured Domains, in: In Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence (UAI), 446–454, 2000.

S. Moral, A. Salmerón, Dynamic importance sampling in Bayesian networks based on probability trees., International Journal of Approximate Reasoning 38 (3) (2005) 245–261.

A. Proofs

Proof. (of Theorem 2) Because, $\mathbf{B}_i^{\mathbf{x}_{i-1}} \subseteq \mathbf{A}_{N,i}^{\mathbf{x}_{i-1}} \cup \mathbf{C}_{N,i}^{\mathbf{x}_{i-1}}$, we have:

$$\begin{aligned}
\sum_{x_i' \in \mathbf{B}_i^{\mathbf{x}_{i-1}}} Q_i(x_i'|\mathbf{x}_{i-1}) &\leq \sum_{x_i' \in \mathbf{A}_{N,i}^{\mathbf{x}_{i-1}} \cup \mathbf{C}_{N,i}^{\mathbf{x}_{i-1}}} Q_i(x_i'|\mathbf{x}_{i-1}) && (40)\\
\therefore 1 - \sum_{x_i' \in \mathbf{B}_i^{\mathbf{x}_{i-1}}} Q_i(x_i'|\mathbf{x}_{i-1}) &\geq 1 - \sum_{x_i' \in \mathbf{A}_{N,i}^{\mathbf{x}_{i-1}} \cup \mathbf{C}_{N,i}^{\mathbf{x}_{i-1}}} Q_i(x_i'|\mathbf{x}_{i-1}) && (41)\\
\therefore \frac{Q_i(x_i|\mathbf{x}_{i-1})}{1 - \sum_{x_i' \in \mathbf{B}_i^{\mathbf{x}_{i-1}}} Q_i(x_i'|\mathbf{x}_{i-1})} &\leq \frac{Q_i(x_i|\mathbf{x}_{i-1})}{1 - \sum_{x_i' \in \mathbf{A}_{N,i}^{\mathbf{x}_{i-1}} \cup \mathbf{C}_{N,i}^{\mathbf{x}_{i-1}}} Q_i(x_i'|\mathbf{x}_{i-1})} && (42)\\
\therefore Q_i^F(x_i|\mathbf{x}_{i-1}) &\leq L_{N,i}^F(x_i|\mathbf{x}_{i-1}) && (43)\\
\therefore \prod_{i=1}^{n} Q_i^F(x_i|\mathbf{x}_{i-1}) &\leq \prod_{i=1}^{n} L_{N,i}^F(x_i|\mathbf{x}_{i-1}) && (44)\\
\therefore Q^F(\mathbf{x}) &\leq L_N^F(\mathbf{x}) && (45)\\
\therefore \frac{\prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})}{Q^F(\mathbf{x})} &\geq \frac{\prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})}{L_N^F(\mathbf{x})} && (46)\\
\therefore w^F(\mathbf{x}) &\geq w_L^F(\mathbf{x}) && (47)\\
\therefore \frac{1}{N} \sum_{k=1}^{N} w^F(\mathbf{x}^k) &\geq \frac{1}{N} \sum_{k=1}^{N} w_L^F(\mathbf{x}^k) && (48)\\
\therefore \widehat{Z}_N &\geq \widetilde{Z}_N^L && (49)
\end{aligned}$$

Similarly, by using $\mathbf{A}_{N,i}^{\mathbf{x}_{i-1}} \subseteq \mathbf{B}_i^{\mathbf{x}_{i-1}}$, it is easy to prove that $\widehat{Z}_N^F \leq \widetilde{Z}_N^U$. □

Proof. (of Theorem 3) From Proposition 4, it follows that U_N^F and L_N^F in the limit of infinite samples coincide with the backtrack-free distribution Q^F. Therefore,

$$\begin{aligned}
\lim_{N\to\infty} w_N^L(\mathbf{x}) &= \lim_{N\to\infty} \frac{\prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})}{L_N^F(\mathbf{x})} && (50)\\
&= \frac{\prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})}{Q^F(\mathbf{x})} && (51)\\
&= w^F(\mathbf{x}) && (52)
\end{aligned}$$

Therefore,

$$\begin{aligned}
\lim_{N\to\infty} \mathbb{E}_Q \left[\frac{1}{N} \sum_{k=1}^{N} w^L(\mathbf{x}) \right] &= \lim_{N\to\infty} \frac{1}{N} \sum_{\mathbf{x}\in\mathbf{X}} w_N^L(\mathbf{x}) Q(\mathbf{x}) \sum_{k=1}^{N}(1) && (53)\\
&= \frac{1}{N} \times N \lim_{N\to\infty} \sum_{\mathbf{x}\in\mathbf{X}} w_N^L(\mathbf{x}) Q(\mathbf{x}) && (54)\\
&= \sum_{\mathbf{x}\in\mathbf{X}} w^F(\mathbf{x}) Q(\mathbf{x}) \ldots \text{(From Equation 52)} && (55)\\
&= Z && (56)
\end{aligned}$$

Similarly, we can prove that the estimator based on U_N^F in Equation 34 is asymptotically unbiased by replacing $w_N^L(\mathbf{x})$ with $w_N^U(\mathbf{x})$ in Equations 53-56.

Finally, because the estimates $\widetilde{P}_N^U(x_i)$ and $\widetilde{P}_N^L(x_i)$ of $P(x_i)$ given in Equations 36 and 37 respectively are ratios of two asymptotically unbiased estimators, by definition, they are asymptotically unbiased too. □

Proof. (of Theorem 4) Because we store all full solutions $(x_1, \ldots, x_n)$ and all partial assignments $(x_1, \ldots, x_{i-1}, x_i')$ that were proved inconsistent during the N executions of SampleSearch, we require an additional $O(N \times n \times d)$ space to store the combined sample tree used to estimate Z and the marginals. Similarly, because we compute a sum or their ratios by visiting all nodes of this combined sample tree, the time complexity is also $O(N \times d \times n)$ □