Artificial Intelligence 175 (2011) 694–729

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

SampleSearch: Importance sampling in presence of determinism

Vibhav Gogate [a,*,1], Rina Dechter [b]

[a] Computer Science & Engineering, University of Washington, Seattle, WA 98195, USA
[b] Donald Bren School of Information and Computer Sciences, University of California, Irvine, Irvine, CA 92697, USA

ARTICLE INFO

Article history:
Received 31 July 2009
Received in revised form 1 October 2010
Accepted 21 October 2010
Available online 5 November 2010

Keywords:
Probabilistic inference
Approximate inference
Importance sampling
Markov chain Monte Carlo
Bayesian networks
Markov networks
Satisfiability
Model counting
Constraint satisfaction

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 [1–4]. The mixed networks framework treats probabilistic graphical models such as Bayesian and Markov networks [5], and deterministic graphical models such as constraint networks [6] 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 belief updating or posterior estimation in a Bayesian or Markov network.

It is straightforward to design importance sampling algorithms [7–9] 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 an expectation using a special distribution called the proposal (or importance) 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.

* Corresponding author.
E-mail addresses: vgogate@cs.washington.edu (V. Gogate), dechter@ics.uci.edu (R. Dechter).
[1] This work was done when the author was a graduate student at University of California, Irvine.

0004-3702/$ – see front matter
doi:10.1016/j.artint.2010.10.009

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.

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.[2] 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. In particular, 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. We implemented SampleSearch on top of IJGP-wc-IS [10], a powerful importance sampling technique which uses a generalized belief propagation algorithm [11] called Iterative Join Graph Propagation (IJGP) [12,13] to construct a proposal distribution and w-cutset (Rao-Blackwellised) sampling [14] to reduce the variance. The search was implemented using the *minisat SAT solver* [15]. 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 [16], SampleCount [17] and Relsat [18] 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) [19], an advanced generalized belief propagation scheme called Edge Deletion Belief Propagation (EDBP) [20] as well as with IJGP-wc-IS on linkage analysis [21] and relational [22] 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 [12,13] and Edge Deletion Belief Propagation [20] and an adaptive importance sampling scheme called Evidence Pre-propagated Importance Sampling (EPIS) [23]. 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 paper is based on earlier conference papers [24,25]. The present article contains more detailed and general analysis, full proofs, new bounding approximations (described in Section 4.2.1), as well as new experimental results.

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.

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

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 $\mathbf{x}_\mathbf{S}$.

$\sum_{\mathbf{x} \in \mathbf{X}}$ denotes the sum over the possible values of variables in $\mathbf{X}$, namely, $\sum_{x_1 \in X_1} \times \sum_{x_2 \in X_2} \times \cdots \times \sum_{x_n \in X_n}$. 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. Markov, Bayesian and constraint networks

Definition 1 *(Graphical models and Markov networks).* A discrete graphical model, denoted by $\mathcal{G}$ (or a Markov network, denoted by $\mathcal{T}$) is a 3-tuple $\langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a finite set of variables, $\mathbf{D} = \{\mathbf{D}_1, \ldots, \mathbf{D}_n\}$ is a finite set of domains where $\mathbf{D}_i$ is the domain of variable X_i and $\mathbf{F} = \{F_1, \ldots, F_m\}$ is a finite set of discrete-valued functions (or potentials). Each function F_i is defined over a subset $\mathbf{S}_i \subseteq \mathbf{X}$ of variables. A graphical model $\mathcal{G}$ represents a joint distribution $P_\mathcal{G}$ over the variables $\mathbf{X}$, given by:

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

where Z is the normalization constant and 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. 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 $\mathcal{G} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ is an undirected graph $G(\mathbf{X}, \mathbf{E})$ which has variables of $\mathcal{G}$ as its vertices and has an edge between any two variables that appear in a scope of a function $F \in \mathbf{F}$.

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

Definition 4 *(Constraint networks).* A constraint network is a graphical model $\mathcal{R} = \langle \mathbf{X}, \mathbf{D}, \mathbf{C} \rangle$ where $\mathbf{C} = \{C_1, \ldots, C_m\}$ is a set of constraints. Each constraint C_i is a 0/1 function defined over a subset of variables $\mathbf{S}_i$, called its scope. Given an assignment $\mathbf{S}_i = \mathbf{s}_i$, a constraint is satisfied if $C_i(\mathbf{s}_i) = 1$. A constraint can also be expressed by a pair $\langle R_i, \mathbf{S}_i \rangle$ where R_i is a relation defined over the variables $\mathbf{S}_i$ and contains all tuples $\mathbf{S}_i = \mathbf{s}_i$ for which $C_i(\mathbf{s}_i) = 1$. 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. A constraint network represents a uniform distribution over its solutions.

Propositional satisfiability

A special case of a constraint network is the *propositional satisfiability problem* (SAT). A propositional 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 (denoted by $\wedge$) of *clauses* $Cl_1 \wedge \cdots \wedge Cl_t$ (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 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 [3,4]. 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 5 *(Mixed network).* A mixed network is a four-tuple $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$ where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of random variables, $\mathbf{D} = \{\mathbf{D}_1, \ldots, \mathbf{D}_n\}$ is a set of domains where $\mathbf{D}_i$ is the domain of X_i, $\mathbf{F} = \{F_1, \ldots, F_m\}$ is a set of non-negative real valued functions where each F_i is defined over a subset of variables $\mathbf{S}_i \subseteq \mathbf{X}$ (its scope) and $\mathbf{C} = \{C_1, \ldots, C_p\}$ is a set of constraints (or 0/1 functions). A mixed network represents a joint distribution over $\mathbf{X}$ given by:

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

where $sol(\mathbf{C})$ is the set of solutions of $\mathbf{C}$ and $Z = \sum_{\mathbf{x} \in sol(\mathbf{C})} \prod_{i=1}^{m} F_i(\mathbf{x})$ is the normalizing constant. The **primal graph** of a mixed network has variables as its vertices and an edge between any two variables that appear in the scope of a function $F \in \mathbf{F}$ or a constraint $C \in \mathbf{C}$.

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

Definition 6 *(The weighted counting task).* Given a mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$, the weighted counting task is to compute the normalization constant given by:

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

Equivalently, if we represent the constraints in $\mathbf{C}$ 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 7 *(Marginal task).* Given a mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$, the marginal task is to compute the marginal distribution of each variable. Namely, for each variable X_i and $x_i \in \mathbf{D}_i$, compute:

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

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

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

H_i can also be expressed as a relation. The set of constraints $\mathbf{C}'$ 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 [7,9] 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. For more details on the theoretical results presented in this subsection, we refer the reader to [8,26].

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 [5].

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.

Throughout the paper, we will often use the notion of biased and unbiased estimators, which we define below.

Definition 9 *(Unbiased and asymptotically unbiased estimator).* Given a probability distribution Q, a statistics θ of Q, and N samples drawn from Q, a function $\widehat{\theta_N}$, defined over the samples is an unbiased estimator of θ if $\mathbb{E}_Q[\widehat{\theta_N}] = \theta$. Similarly, a function $\widetilde{\theta_N}$ 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 denote an unbiased estimator of a statistics θ by $\widehat{\theta}$, an asymptotically unbiased estimator by $\widetilde{\theta}$ and an arbitrary estimator by $\bar{\theta}$.

The notion of unbiasedness and asymptotic unbiasedness is important because it helps to characterize the performance of an estimator which we explain briefly below. The mean-squared error of an estimator $\bar{\theta}$ is given by:

$$MSE(\bar{\theta}) = \mathbb{E}_Q\left[(\bar{\theta} - \theta)^2\right] \tag{5}$$

$$= \mathbb{E}_Q\left[\bar{\theta}^2\right] - 2\mathbb{E}_Q[\bar{\theta}]\theta + \theta^2 \tag{6}$$

$$= \left[\mathbb{E}_Q\left[\bar{\theta}^2\right] - \mathbb{E}_Q[\bar{\theta}]^2\right] + \left[\mathbb{E}_Q[\bar{\theta}]^2 - 2\mathbb{E}_Q[\bar{\theta}]\theta + \theta^2\right] \tag{7}$$

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

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

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

$$V_Q[\bar{\theta}] = \mathbb{E}_Q\left[\bar{\theta}^2\right] - \mathbb{E}_Q[\bar{\theta}]^2$$

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

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

In other words, the mean squared error of an estimator is equal to bias squared plus variance. 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.

2.3.1. Estimating weighted counts

Consider the expression for weighted counts (see Definition 6).

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

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 Eq. (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:

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

where

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

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

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

We can estimate the variance of $\widehat{Z}_N$ by:

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

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

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

We can show that:

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. $\mathbb{E}_Q[\widehat{V_Q}[\widehat{Z}_N]] = V_Q[\widehat{Z}_N] = V_Q[w(\mathbf{x})]/N$.

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 $Q(\mathbf{x}) \propto \prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})$, then for any sample $\mathbf{x}$, we have $w(\mathbf{x}) = Z$ yielding an optimal (zero variance) estimator. However, making $Q(\mathbf{x}) \propto \prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})$ 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 $\prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})$.

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 Eq. (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, Eq. (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 using the notion of properly weighted samples.

Definition 10 *(A properly weighted sample).* (See [26].) A set of weighted samples $\{\mathbf{x}^k, w(\mathbf{x}^k)\}_{k=1}^N$ drawn from a distribution G are said to be properly weighted with respect to a distribution P if for any discrete function H,

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

where c is a normalization constant common to all samples.

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

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

Substituting Eq. (15) in Eq. (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:

Proposition 1. *Given* $w(\mathbf{x}) = \frac{\delta_{x_i}(\mathbf{x})\prod_{i=1}^{m}F_i(\mathbf{x})\prod_{j=1}^{p}C_j(\mathbf{x})}{Q(\mathbf{x})}$*, the set of weighted samples* $\{\mathbf{x}^k, w(\mathbf{x}^k)\}_{k=1}^N$ *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,

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

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 Eq. (11)) would be zero and the estimate of the marginals (given by Eq. (19)) would be ill-defined. Clearly, if Q properly encodes all the zeros in $\mathcal{M}$, then we would have no rejection.

Definition 11 *(Zero equivalence).* A distribution P is zero equivalent to a distribution P', iff their flat constraint networks (see Definition 8) 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.

Algorithm 1: Sampling from the backtrack-free distribution.

Input: A mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$, 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

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

Definition 12 *(Consistent and globally consistent partial sample).* Given a set of constraints $\mathbf{C}$ defined over $\mathbf{X} = \{X_1, \ldots, X_n\}$, a partial sample $(x_1, \ldots, x_i)$ is consistent if it does not violate any constraint in $\mathbf{C}$. A partial sample $(x_1, \ldots, x_i)$ is globally consistent if it can be extended to a solution of $\mathbf{C}$ (i.e. it can be extended to a full assignment to all n variables that satisfies all constraints in $\mathbf{C}$).

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

Definition 13 *(Backtrack-free distribution of Q w.r.t. $\mathbf{C}$).* Given a mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$ and a proposal distribution $Q = \{Q_1, \ldots, Q_n\}$ representing $Q(\mathbf{X}) = \prod_{i=1}^{n} Q_i(X_i|X_1, \ldots, X_{i-1})$ along an ordering o, the backtrack-free distribution $Q^F = \{Q_1^F, \ldots, Q_n^F\}$ of Q along o w.r.t. $\mathbf{C}$ where $Q^F(\mathbf{X}) = \prod_{i=1}^{n} Q_i^F(X_i|X_1, \ldots, X_{i-1})$ is defined by:

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

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:

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

We borrow the term *backtrack-free* from the constraint satisfaction literature [6,27]. 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 8). The proposal distribution presented in Definition 13 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 Eq. (2)) by replacing Q by Q^F in Eq. (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.

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 $\mathbf{x} = (x_1, \ldots, x_n)$

1 SET $i = 1$, $D'_i = D_i$ (copy domains), $Q'_1(X_1) = Q_1(X_1)$ (copy distribution), $\mathbf{x} = \emptyset$;
2 **while** $1 \leqslant i \leqslant n$ **do**
  // Forward phase
3 **if** D'_i *is not empty* **then**
4 Sample $X_i = x_i$ from Q'_i and remove it from D'_i;
5 **if** $(x_1, \ldots, x_i)$ *violates any constraint in* **C** **then**
6 SET $Q'_i(X_i = x_i|x_1, \ldots, x_{i-1}) = 0$ and normalize Q'_i;
7 Goto step 3;
8 $\mathbf{x} = \mathbf{x} \cup x_i$, $i = i + 1$, $D'_i = D_i$, $Q'_i(X_i|x_1, \ldots, x_{i-1}) = Q_i(X_i|x_1, \ldots, x_{i-1})$;
  // Backward phase
9 **else**
10 $\mathbf{x} = \mathbf{x} \backslash x_{i-1}$;
11 SET $Q'_{i-1}(X_{i-1} = x_{i-1}|x_1, \ldots, x_{i-2}) = 0$ and normalize $Q'_{i-1}(X_{i-1}|x_1, \ldots, x_{i-2})$;
12 SET $i = i - 1$;
13 **if** $i = 0$ **then**
14 return inconsistent;
15 **else**
16 return $\mathbf{x}$;

Similarly, we can estimate the posterior marginals by replacing the weight $w(\mathbf{x})$ in Eq. (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 Eq. (20) is an unbiased estimate of Z while $\widetilde{P_N}(x_i)$ defined in Eq. (22) is an asymptotically unbiased estimate of the posterior marginals $P(x_i)$.

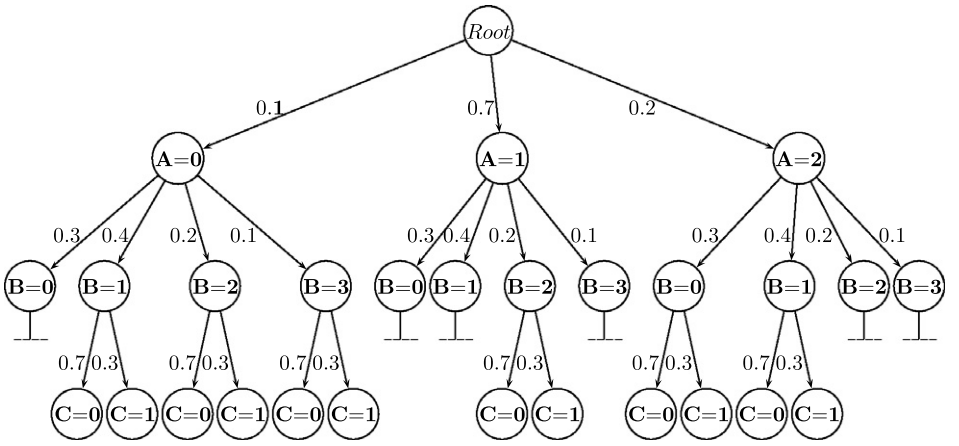
In practice, one could use any constraint solver as a substitute for the oracle in Algorithm 1. However, generating samples using an exact solver would be inefficient in many cases. Next, we present the SampleSearch scheme which integrates backtracking search with sampling. In essence, we integrate more 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. Indeed, in our empirical work, we will use an advanced search scheme based on the minisat solver [15].

Given a mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$ and a proposal distribution $Q(\mathbf{X})$, the traditional 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 **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 in which case the algorithm will further modify $Q_i(X_i|x_1, \ldots, x_{i-1})$. 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 **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 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.



Proposal Distribution Q

$Q = Q(A) \times Q(B|A) \times Q(C|A,B)$
$Q(A) = (0.1, 0.7, 0.2)$
$Q(B|A) = Q(B) = (0.3, 0.4, 0.2, 0.1)$
$Q(C|A,B) = Q(C) = (0.7, 0.3)$

Constraints

$A \neq B,\ A = 1 \rightarrow B \neq 0$
$B = 3 \rightarrow C \neq 0,\ B = 3 \rightarrow C \neq 1$
$A = 1 \rightarrow B \neq 3,\ A = 2 \rightarrow B \neq 3$

Fig. 1. A full OR search tree given a set of constraints and a proposal distribution.

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 $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$ and a proposal distribution Q, the sampling distribution I of SampleSearch equals the backtrack-free probability distribution Q^F of Q w.r.t. $\mathbf{C}$, i.e. $\forall i\ Q_i^F = I_i$.*

To prove this theorem, we need the following proposition:

Proposition 2. *Given a mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$, 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. $\mathbf{C}$, SampleSearch samples values without replacement from the domain $\mathbf{D}_i$ of X_i until a globally consistent extension $(x_1, \ldots, x_{i-1}, x_i)$ is generated.*

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 1).

Example 1. Consider the complete search tree corresponding to the proposal distribution and to the constraints given in Fig. 1. 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 Fig. 2. 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 Fig. 2, (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) $\langle B = 0, B = 3, B = 2 \rangle | A = 0$, 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

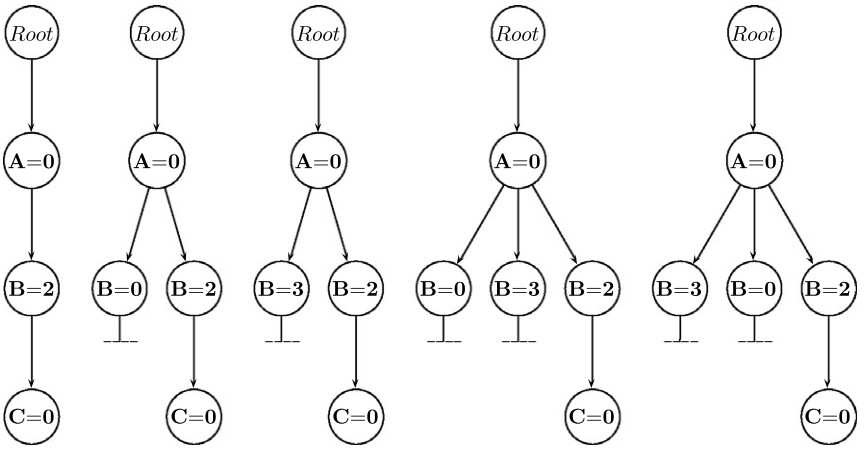


Fig. 2. Five possible traces of SampleSearch which lead to the sample $(A = 0,\ B = 2,\ C = 0)$. The children of each node are specified from left to right in the order in which they are generated.

from left to right ($B = 3, B = 0, B = 2$). Clearly, the probability $I_B(B = 2|A = 0)$ equals 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 $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C}\rangle$, 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. $\mathbf{C}$, 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 $\mathbf{R}_i^{\mathbf{x}_{i-1}} = \{x_i \in \mathbf{D}_i^{\mathbf{x}_{i-1}} \mid (x_1, \ldots, x_{i-1}, x_i)$ *is globally consistent*$\}$ and $\mathbf{B}_i^{\mathbf{x}_{i-1}} = \mathbf{D}_i^{\mathbf{x}_{i-1}} \setminus \mathbf{R}_i^{\mathbf{x}_{i-1}}$.

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 $\mathbf{x}_{i-1}$ by SampleSearch.

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 $\mathbf{x}_{i-1}$:

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

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

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

Substituting Eq. (24) in Eq. (23), we get:

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

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:

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

From Eqs. (25) and (26), we get:

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

$Q_i(x_i|\mathbf{x}_{i-1})$ does not depend on the indices j and k in Eq. (27) and therefore we can rewrite Eq. (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 Eq. (28) does not depend on x_i and therefore it follows that if $(x_1, \ldots, x_{i-1}, x_i)$ is globally consistent:

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

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 Eq. (30) is by definition equal to $Q_i^F(x_i|\mathbf{x}_{i-1})$ (see Definition 13). $\square$

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

Once we have a 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 13, 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 [6] or any other 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 a 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 Eq. (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, Fig. 2 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.

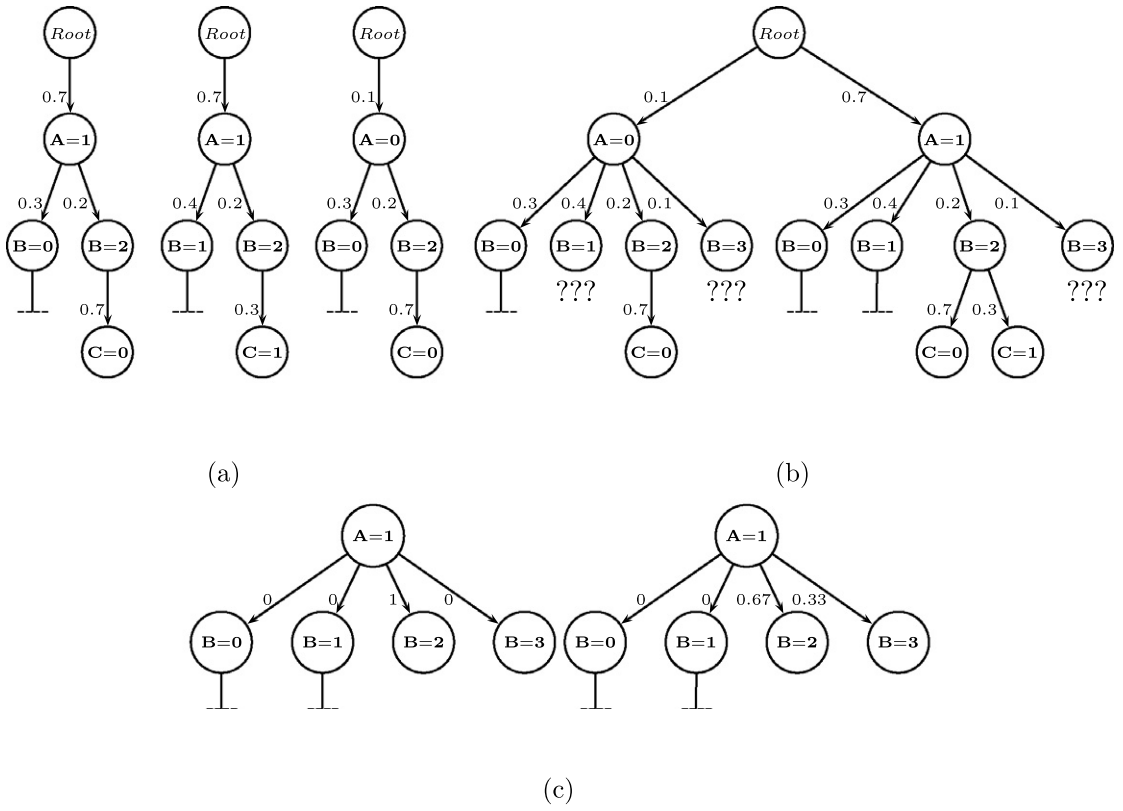


Fig. 3. (a) Three DFS-traces. (b) Combined information from the three DFS-traces given in (a). (c) Two possible approximations of $I(B|A=1)$.

Example 2. Consider the three traces given in Fig. 3(a). We can combine the information from the three traces as shown in Fig. 3(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 Fig. 3(b)). Setting the unexplored nodes to either inconsistent or consistent gives us the two different approximations shown in Fig. 3(c).

Generalizing Example 2, 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.[3]

Definition 14 *(Upper and lower approximations of Q^F by U_N^F and L_N^F).* Given a mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$, 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 $\mathbf{x}_{i-1} = (x_1, \ldots, x_{i-1})$ generated in one of the N independent runs, we define two sets:

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

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

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

[3] 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.

$$L_N^F(\mathbf{x}) = \prod_{i=1}^{n} L_{N,i}^F(x_i|\mathbf{x}_{i-1}) \quad \text{where } L_{N,i}^F(x_i|\mathbf{x}_{i-1}) = \frac{Q_i(x_i|\mathbf{x}_{i-1})}{1 - \sum_{x'_i \in \mathbf{A}_{N,i}^{\mathbf{x}_{i-1}} \cup \mathbf{C}_{N,i}^{\mathbf{x}_{i-1}}} Q_i(x'_i|\mathbf{x}_{i-1})} \tag{33}$$

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

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

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 Appendix A.

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

Theorem 3. *The estimates $\widetilde{Z}_N^U$ and $\widetilde{Z}_N^L$ of Z given in Eqs.* (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 Eqs.* (36) *and* (37) *respectively are asymptotically unbiased.*

Theorem 4. *Given N samples output by SampleSearch for a mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$, 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 Eqs.* (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 [6] instead of naive backtracking and easily show that:

Proposition 5. *Given a mixed network $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C} \rangle$ 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.* **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[4] and use minisat [15] as our SAT solver. However, we have to make minisat (or any other state-of-the-art SAT solver e.g. RSAT [29]) *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 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 a SAT formula, (b) computing probability of evidence and partition function in Bayesian and Markov networks respectively, and (c) computing posterior marginals in Bayesian and Markov networks.

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 operates on top of an advanced importance sampling algorithm IJGP-wc-IS presented in [10]. We call the resulting scheme IJGP-wc-SS. IJGP-wc-IS uses the generalized belief propagation scheme Iterative Join Graph Propagation (IJGP) to construct a proposal distribution and the w-cutset sampling framework [14] to reduce the variance. 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 [8,30]. In principle, one could use the prior distribution as the proposal distribution, such as in Likelihood weighting [31,32]. However, when the evidence is unlikely, the prior is a very bad approximation of the posterior [5,30]. In this case, the variance of the sample weights will be large and a few samples with large weights will dominate the mean, yielding an inefficient sampling scheme.
  Several schemes have been proposed to address this problem and below we briefly review two different but complementary approaches. In the first approach, which is often referred to as *adaptive importance sampling* [30,33], the sampling algorithm periodically updates the proposal distribution using the generated samples. As more and more samples are drawn, the hope is that the updated proposal distribution would get closer and closer to the posterior distribution; yielding a low variance sampling scheme. In the second approach, the idea is to use a state-of-the-art approximation algorithm, e.g., Belief propagation [5] to construct a proposal distribution [34,23,10]. In IJGP-wc-IS, we use the latter approach.
  In particular, IJGP-wc-IS uses $Q = \{Q_1, \ldots, Q_n\}$, obtained from the output of Iterative Join Graph Propagation (IJGP) [12, 13] which was shown to yield good performance in earlier studies [23,10]. IJGP is a generalized belief propagation [11] technique for approximating the posterior distribution in graphical models. It uses the same message passing scheme as *join tree propagation* [35], but applies it over the clusters of a *join graph* rather than a *join tree*, iteratively. A join graph is a decomposition of the 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. IJGP is exponential in its i-bound and its accuracy

[4] It is easy to convert any (relational) constraint network to a CNF formula. In our implementation, we use the direct encoding described in [28].

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 [12] and run IJGP on JG;
3 Create a w-cutset $\mathbf{K} \subseteq \mathbf{X}$ using the greedy scheme described in [14,38]. 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 following heuristic scheme [10]. First, we select 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**
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;

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 [12] and compute, in advance, the space required by IJGP by summing over the space required by the individual messages.[5] 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 [14]. The idea 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 $\mathbf{R}$ conditioned on $\mathbf{K} = \mathbf{k}$ using exact inference techniques such as bucket elimination [19], we need to sample only the variables in $\mathbf{K}$. From the Rao-Blackwell theorem [26,36], 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 Eq. (38) in polynomial time (exponential in the constant w) using bucket elimination [19].

  It was demonstrated that the higher the w-bound [14], 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 heuristics:* We experimented with three different variable ordering heuristics for constructing the join graph of IJGP: min-fill ordering, min-degree ordering and the hmetis ordering.[6] We performed sampling along the reverse order in which the join graph was constructed. Intuitively, this makes sense because IJGP is akin to variable elimination and sampling is akin to search, and it is known that the best ordering for elimination is the reverse ordering for search and vice versa. In case of Bayesian networks, we also experimented with topological ordering for sampling. We found (as was also observed before) that the min-fill ordering gives the best performance and therefore for brevity, we only compare the performance of min-fill based IJGP-wc-IS and IJGP-wc-SS with the other solvers. We evaluate the ordering heuristics in Section 5.5.

The 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.[7] 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 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 Eqs. (34) and (35) as IJGP-wc-SS/UB and IJGP-wc-SS/LB respectively. Note that IJGP-wc-

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

[6] This ordering heuristic due to [37] is based on hyper-graph partitioning. To create the partitioning, we use the hmetis software available at: http://www-users.cs.umn.edu/karypis/metis/hmetis and hence the name.

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

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 [12,13] in which we start with an i-bound of 1, run IJGP until convergence or until 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 [39] introduced an approximate solution counting scheme called ApproxCount. ApproxCount is based on the formal result of [40] 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 [41]. To reduce the variance, Wei and Selman [39] 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 [16], which is an extension of the well-known local search SAT solver Walksat [42]. We experimented with an anytime version of ApproxCount in which we report the cumulative average accumulated over several runs.

SampleCount [17] 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.[8]

In our experiments, we used an implementation of ApproxCount and SampleCount available from the respective authors [16,17]. Following the recommendations made in [17], 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 importance sampling algorithm for computing marginals in Bayesian networks [23]. The algorithm uses loopy belief propagation [5,43] to construct the proposal distribution. In our experiments, we used the anytime implementation of EPIS submitted to the UAI 2008 evaluation [44].

4. Edge Deletion Belief Propagation (EDBP) [20] 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 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 [20].

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 [19]. 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 [19]. In our experiments we select the w-cutset in such a way that bucket elimination would require less than 1.5 GB of space. Again, this is done to ensure that bucket elimination terminates in a reasonable amount of time and uses bounded space. Exact weighted counts can be computed by summing over the exact solution output by bucket elimination for all possible instantiations of the w-cutset. When VEC is terminated before completion, it outputs a partial sum yielding a lower bound on the weighted counts.

As pre-processing, the algorithm performs SAT-based variable domain pruning that often yields significant performance gains in practice. Here, we first convert all zero probabilities and constraints in the problem to a CNF formula F. Then, for each variable-value pair, we construct a new CNF formula F' by adding a unit clause corresponding to the pair to F and check using minisat [15] if F' is consistent or not. If F' is inconsistent then we delete the value from the domain of the variable. The implementation of this scheme is available publicly from our software website [45].

[8] In the original paper, SampleCount [17] was investigated for lower bounding solution counts. Here, we evaluate the unbiased solution counts computed by the algorithm.

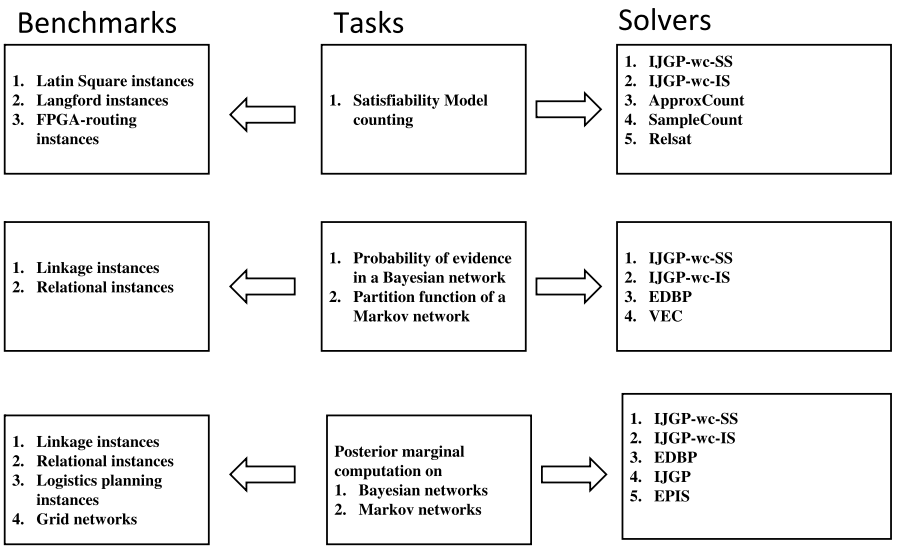


Fig. 4. Chart showing the scope of our experimental study.

Table 1
Query types handled by various solvers.

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

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

6. Relsat[9] [18] 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.

7. ACE[10] is a package for exact inference in Bayesian and Markov networks; currently it is state-of-the-art. It first compiles the Bayesian or Markov network into an Arithmetic Circuit (AC) [46] and then uses the AC to answer various queries over the network. ACE uses the c2d compiler [47] to compile the network into a d-DNNF [48] and then extracts the AC from the d-DNNF. Note that unlike other exact schemes described until now, ACE is not an anytime scheme. We therefore report only the time required by ACE to solve the instance, and use these times as a baseline for comparison.

The benchmarks and the solvers for the different task types are shown in Fig. 4. Table 1 summarizes different query types that can be handled by the various solvers. A '√' indicates that the algorithm is able to approximately estimate the query while a lack of √ indicates otherwise.

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 k, 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 after incorporating evidence and removing irrelevant variables). 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.

[9] Available at http://www.bayardo.org/resources.html.
[10] Available at http://reasoning.cs.ucla.edu/ace/.

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.

Problem	$\langle n, k, c, w \rangle$		Exact	Sample-Count	Approx-Count	Relsat	IJGP-wc-SS/LB	IJGP-wc-SS/UB	IJGP-wc-IS
ls8-norm	⟨512, 2, 5584, 255⟩	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	⟨729, 2, 9009, 363⟩	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	⟨1000, 2, 13820, 676⟩	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	⟨1331, 2, 20350, 956⟩	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

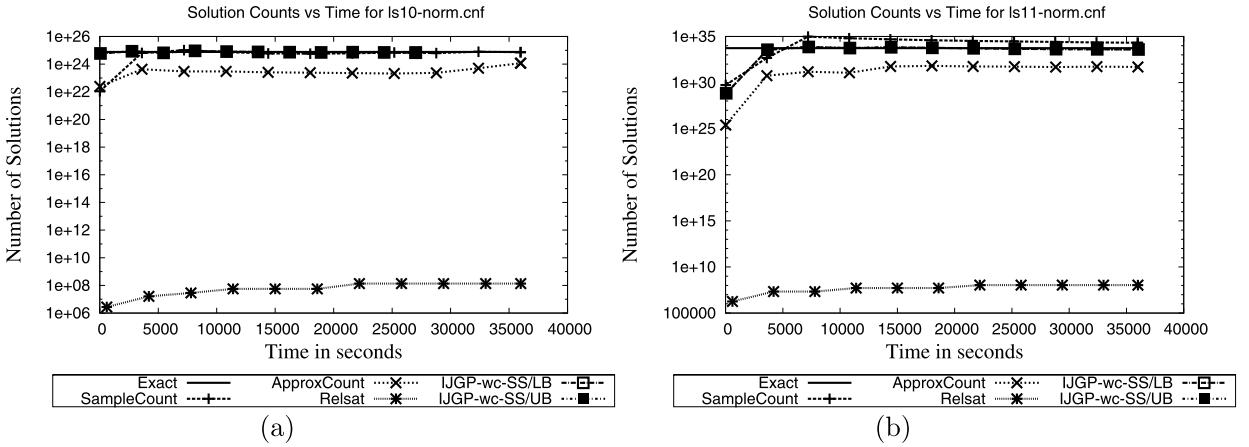


Fig. 5. Time versus solution counts for two sample Latin square instances. IJGP-wc-IS is not plotted in the figures because it fails on all the instances.

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 column. In a normalized Latin square the first row and column are fixed. The task here is to count the number of normalized Latin squares of a given order. The Latin squares were modeled as SAT formulas using the extended encoding given in [49]. The exact counts for these formulas are known up to order 11 [50].

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 than with SampleCount and ApproxCount. In Fig. 5(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 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 [51].

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

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.

Problem	$\langle n, k, c, w \rangle$		Exact	Sample-Count	Approx-Count	Relsat	IJGP-wc-SS/LB	IJGP-wc-SS/UB	IJGP-wc-IS
lang12	⟨576, 2, 13584, 383⟩	Z	2.16E+5	1.93E+05	2.95E+04	**2.16E+05**$^{*(297\ s)}$	2.16E+05	2.16E+05	X
		M		2720	4668		999991	999991	0
lang16	⟨1024, 2, 32320, 639⟩	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	⟨1444, 2, 54226, 927⟩	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	⟨1600, 2, 63280, 1023⟩	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	⟨2116, 2, 96370, 1407⟩	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	⟨2304, 2, 109536, 1535⟩	Z	9.37E+16	1.17E+13	4.15E+11	X	8.74E+15	**1.40E+16**	X
		M		25	33		271	271	0

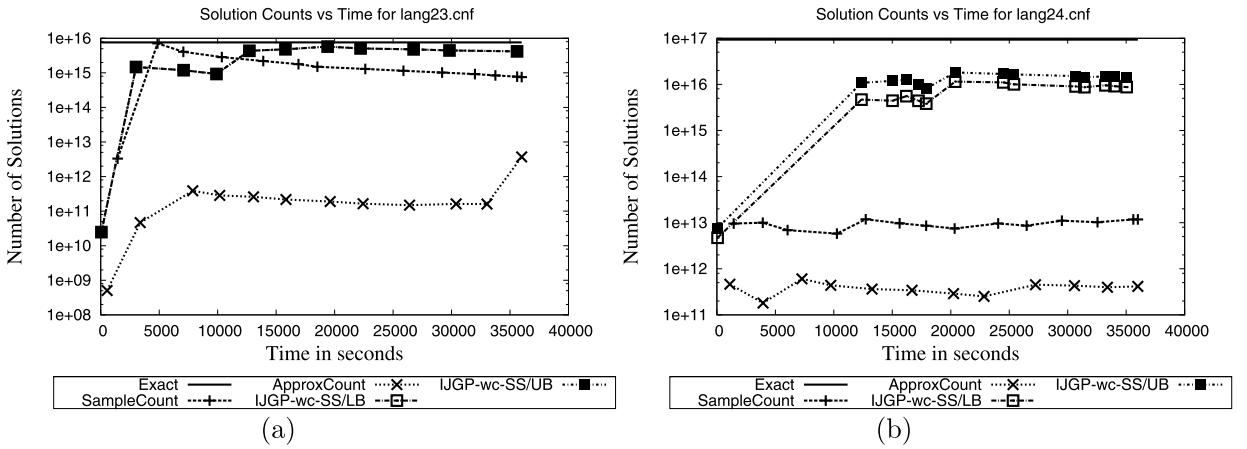


Fig. 6. Time versus solution counts for two sample Langford instances. IJGP-wc-IS and Relsat are not plotted in the figures because they fail on the given instances.

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 Fig. 6(a) and (b), we show how the estimates vary with time for the two largest instances. We clearly see the superior anytime performance of IJGP-wc-SS/LB and IJGP-wc-SS/UB.[11]

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 various sampling schemes for generating good lower-bound approximations whose quality can be compared (the higher the better). Specifically, 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 [52]. These lower bounding schemes [17,52] 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.

[11] An anonymous reviewer pointed out that the number of solutions of the Langford problem can be estimated using a specialized sampling scheme (he/she also provided a python implementation). This scheme suffers from the rejection problem, but the rejection rate is very small. The scheme often yields sample means which are closer to the true mean than the sample means output by SampleSearch, SampleCount and ApproxCount.

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 [52].

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

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. Comparing lower-bounds facilitates a comparative evaluation even on instances for which exact weighted counts are not available.[12]

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.

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 [53]. 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 instances, the number of consistent samples generated by SampleCount are far smaller than IJGP-wc-SS.

Next, we present results for Bayesian and Markov networks benchmarks. For the rest of the paper, note a slight change in the content of each table. In the second column, we also report the time required by ACE to compute the weighted counts or marginals for the instance. The time-bound for ACE was set to 3 hrs.

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 [54]. 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. Fig. 7 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

[12] We still cannot evaluate the quality of the marginals when the exact solution is not known because the Markov inequality based schemes [17,52] cannot lower bound marginal probabilities.

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 [52].

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

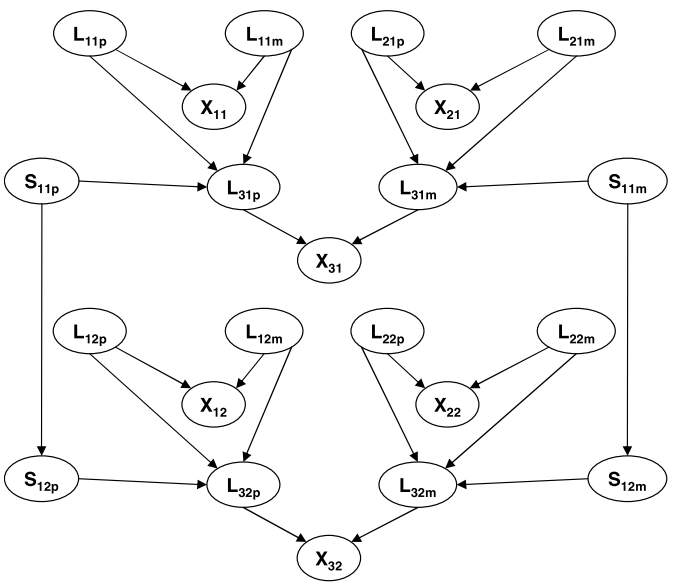


Fig. 7. A fragment of a Bayesian network used in genetic linkage analysis.

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.

Problem	$\langle n, k, c, e, w\rangle$ ACE time		Exact	IJGP-wc-SS/LB	IJGP-wc-SS/UB	VEC	EDBP	IJGP-wc-IS
BN_69	⟨777, 7, 228, 78, 39⟩	Z	5.28E−054	**3.00E−55**	**3.00E−55**	1.93E−61	2.39E−57	X
	ACE (timeout)	M		6.84E+5	6.84E+5			0
BN_70	⟨2315, 5, 484, 159, 35⟩	Z	2.00E−71	**1.21E−73**	**1.21E−73**	7.99E−82	6.00E−79	X
	ACE (233 s)	M		1.92E+5	1.92E+5			0
BN_71	⟨1740, 6, 663, 202, 53⟩	Z	5.12E−111	**1.28E−111**	**1.28E−111**	7.05E−115	1.01E−114	X
	ACE (timeout)	M		7.46E+4	7.46E+4			0
BN_72	⟨2155, 6, 752, 252, 65⟩	Z	4.21E−150	**4.73E−150**	**4.73E−150**	1.32E−153	9.21E−155	X
	ACE (timeout)	M		1.53E+5	1.53E+5			0
BN_73	⟨2140, 5, 651, 216, 67⟩	Z	2.26E−113	**2.00E−115**	**2.00E−115**	6.00E−127	2.24E−118	X
	ACE (timeout)	M		7.75E+4	7.75E+4			0
BN_74	⟨749, 6, 223, 66, 35⟩	Z	3.75E−45	**2.13E−46**	**2.13E−46**	3.30E−48	5.84E−48	X
	ACE (timeout)	M		2.80E+5	2.80E+5			0
BN_75	⟨1820, 5, 477, 155, 37⟩	Z	5.88E−91	**2.19E−91**	**2.19E−91**	5.83E−97	3.10E−96	X
	ACE (timeout)	M		7.72E+4	7.72E+4			0
BN_76	⟨2155, 7, 605, 169, 53⟩	Z	4.93E−110	**1.95E−111**	**1.95E−111**	1.00E−126	3.86E−114	X
	ACE (timeout)	M		2.52E+4	2.52E+4			0

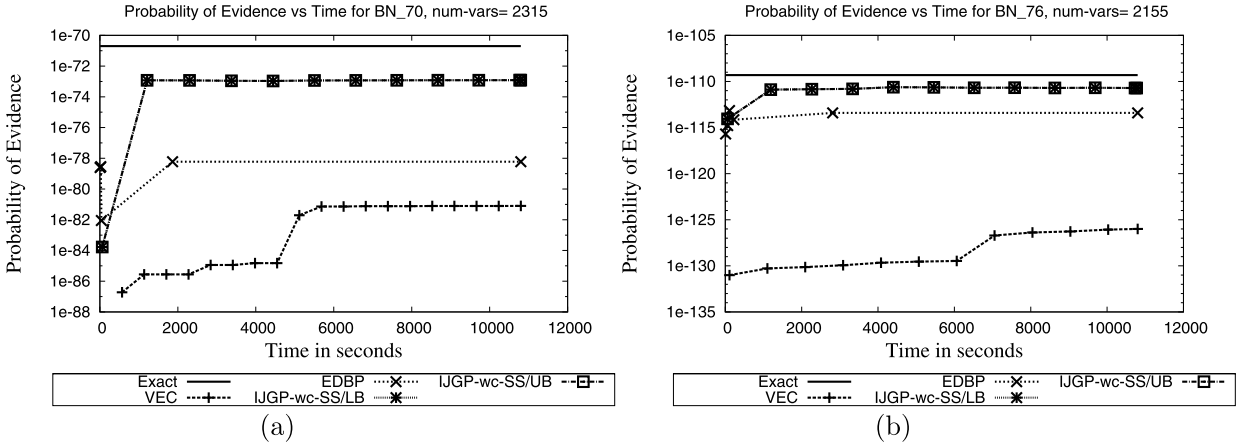


Fig. 8. Convergence of probability of evidence as a function of time for two sample Linkage instances. IJGP-wc-IS is not plotted in the figures because it fails on all the instances.

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 [21]).

We first evaluate the solvers on Linkage (Bayesian) networks used in the UAI 2006 evaluation [55]. Table 6 contains the results. The exact results for these instances are available from the UAI 2006 evaluation website. 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. Fig. 8 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 18 linkage instances that were used in the UAI 2008 evaluation [44] for which the exact value of probability of evidence is known.[13] We see that VEC (as an anytime scheme) exactly solves 10 instances (as indicated by a "*" in Table 7). On 7 out of the remaining 8 instances, IJGP-wc-SS/LB and IJGP-wc-SS/UB are better than VEC. EDBP exactly solves 5 instances. On the remaining 13 instances, IJGP-wc-SS/LB and IJGP-wc-SS/UB are better than

[13] The exact value of probability of evidence for instances that ACE and VEC were unable to solve was obtained by running the Bucket elimination (BE) with external memory (BEEM) algorithm [56]. BEEM uses external memory, such as disk storage, to increase the amount of memory available to BE, thereby significantly improving BE's scalability.

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.

Problem	$\langle n, k, c, e, w \rangle$ ACE time		Exact	IJGP-wc-SS/LB	IJGP-wc-SS/UB	VEC	EDBP	IJGP-wc-IS
pedigree18	⟨1184, 2, 386, 0, 26⟩ ACE (10 s)	Z M	7.18E−79	7.39E−79 1.30E+5	7.39E−79 1.30E+5	**7.18E−79***(64 s)	**7.18E−79***(772 s)	X 0
pedigree1	⟨334, 2, 121, 0, 20⟩ ACE (2 s)	Z M	7.81E−15	7.81E−15 3.26E+5	7.81E−15 3.26E+5	**7.81E−15***(12 s)	**7.81E−15***(14 s)	X 0
pedigree20	⟨437, 2, 147, 0, 25⟩ ACE (timeout)	Z M	2.34E−30	2.31E−30 2.31E+5	2.31E−30 2.31E+5	**2.34E−30***(1216 s)	6.19E−31	X 0
pedigree23	⟨402, 2, 130, 0, 26⟩ ACE (8 s)	Z M	2.78E−39	2.76E−39 3.28E+5	2.76E−39 3.28E+5	**2.78E−39***(913 s)	1.52E−39	X 0
pedigree25	⟨1289, 2, 396, 0, 38⟩ ACE (timeout)	Z M	1.69E−116	**1.69E−116** 1.29E+5	**1.69E−116** 1.29E+5	**1.69E−116***(318 s)	**1.69E−116***(2562 s)	X 0
pedigree30	⟨1289, 2, 413, 0, 27⟩ ACE (8 s)	Z M	1.84E−84	1.90E−84 1.14E+5	1.90E−84 1.14E+5	**1.85E−84***(808 s)	**1.85E−84***(179 s)	X 0
pedigree37	⟨1032, 2, 333, 0, 25⟩ ACE (52 s)	Z M	2.63E−117	1.18E−117 4.26E+5	1.18E−117 4.26E+5	**2.63E−117***(2521 s)	5.69E−124	X 0
pedigree38	⟨724, 2, 263, 0, 18⟩ ACE (340 s)	Z M	5.64E−55	3.80E−55 1.63E+5	3.80E−55 1.63E+5	**5.65E−55***(735 s)	8.41E−56	X 0
pedigree39	⟨1272, 2, 354, 0, 29⟩ ACE (timeout)	Z M	6.32E−103	6.29E−103 1.25E+5	6.29E−103 1.25E+5	**6.32E−103***(136 s)	**6.32E−103***(694 s)	X 0
pedigree42	⟨448, 2, 156, 0, 23⟩ ACE (timeout)	Z M	1.73E−31	1.73E−31 3.26E+5	1.73E−31 3.26E+5	**1.73E−31***(3188 s)	8.91E−32	X 0
pedigree31	⟨1183, 2, 0, 45⟩ ACE (timeout)	Z M	1.98E−70	**2.08E−70** 6.7E+4	**2.08E−70** 6.7E+4	1.67E−76	1.34E−70	X 0
pedigree34	⟨1160, 1, 0, 59⟩ ACE (timeout)	Z M	5.9E−65	3.84E−65 1.2E+5	3.84E−65 1.2E+5	2.58E−76	**4.30E−65**	X 0
pedigree13	⟨1077, 1, 0, 51⟩ ACE (timeout)	Z M	5.44E−32	**7.03E−32** 8.1E+4	**7.03E−32** 8.1E+4	2.17E−37	6.53E−34	X 0
pedigree9	⟨1118, 2, 0, 41⟩ ACE (timeout)	Z M	3.43E−79	2.93E−79 8.0E+4	2.93E−79 8.0E+4	8.00E−82	**3.13E−79**	X 0
pedigree19	⟨793, 2, 0, 23⟩ ACE (timeout)	Z M	9.4E−60	6.76E−60 9.5E+4	6.76E−60 9.5E+4	**7.97E−60**	3.35E−60	X 0
pedigree7	⟨1068, 1, 0, 56⟩ ACE (timeout)	Z M	1.49E−65	**1.33E−65** 8.3E+4	**1.33E−65** 8.3E+4	1.66E−72	2.93E−66	X 0
pedigree51	⟨1152, 1, 0, 51⟩ ACE (timeout)	Z M	1.34E−74	**2.47E−74** 1.0E+5	**2.47E−74** 1.0E+5	5.56E−85	6.16E−76	X 0
pedigree44	⟨811, 1, 0, 29⟩ ACE (timeout)	Z M	3.36E−64	**3.39E−64** 1.7E+5	**3.39E−64** 1.7E+5	2.23E−64	7.69E−66	X 0

VEC. Overall, IJGP-wc-SS/LB and IJGP-wc-SS/UB deviate only slightly from the exact value of probability of evidence. 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 [22]. We experimented with ten Friends and Smoker networks and six mastermind networks from this domain which have between 262 to 76212 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.

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.

Problem	$\langle n, k, c, e, w \rangle$ ACE time		Exact	IJGP-wc-SS/LB	IJGP-wc-SS/UB	VEC	EDBP	IJGP-wc-IS
Friends and smokers								
fs-04	⟨262, 2, 74, 226, 12⟩	*Z*	1.53E−05	8.11E−06	8.11E−06	**1.53E−05**$^{*(1\,s)}$	**1.53E−05**$^{*(2\,s)}$	1.52E−05
	ACE (4 s)	*M*		1.00E+6	1.00E+6			2.17E+8
fs-07	⟨1225, 2, 371, 1120, 35⟩	*Z*	1.78E−15	2.23E−16	2.23E−16	**1.78E−15**$^{*(708\,s)}$	1.11E−16	X
	ACE (4 s)	*M*		1.00E+6	1.00E+6			0
fs-10	⟨3385, 2, 1055, 3175, 71⟩	*Z*	7.88E−31	**2.49E−32**	**2.49E−32**	X	7.70E−34	X
	ACE (9 s)	*M*		8.51E+5	8.51E+5			0
fs-13	⟨7228, 2, 2288, 6877, 117⟩	*Z*	1.33E−51	3.26E−55	3.26E−55	X	1.63E−55	**1.33E−51**
	ACE (9 s)	*M*		5.41E+5	5.41E+5			4.67E+7
fs-16	⟨13240, 2, 4232, 12712, 171⟩	*Z*	8.63E−78	6.04E−79	6.04E−79	X	1.32E−82	**8.63E−78**
	ACE (14 s)	*M*		1.79E+5	1.79E+5			1.37E+7
fs-19	⟨21907, 2, 7049, 21166, 243⟩	*Z*	2.12E−109	**1.62E−114**	**1.62E−114**	X	X	X
	ACE (22 s)	*M*		1.90E+5	1.90E+5			0
fs-22	⟨33715, 2, 10901, 32725, 315⟩	*Z*	2.00E−146	**4.88E−147**	**4.88E−147**	X	X	X
	ACE (49 s)	*M*		1.18E+5	1.18E+5			0
fs-25	⟨49150, 2, 15950, 47875, 431⟩	*Z*	7.18E−189	**2.67E−189**	**2.67E−189**	X	X	X
	ACE (74 s)	*M*		9.23E+4	9.23E+4			0
fs-28	⟨68698, 2, 22358, 67102, 527⟩	*Z*	9.82E−237	**4.53E−237**	**4.53E−237**	X	X	X
	ACE (148 s)	*M*		9.35E+4	9.35E+4			0
fs-29	⟨76212, 2, 24824, 74501, 559⟩	*Z*	6.81E−254	**9.44E−255**	**9.44E−255**	X	X	X
	ACE (167 s)	*M*		2.62E+4	2.62E+4			0
Mastermind								
mm_03_08_03	⟨1220, 2, 1193, 48, 20⟩	*Z*	9.79E−8	9.87E−08	9.87E−08	**9.79E−08**$^{*(3\,s)}$	**9.79E−08**$^{*(11\,s)}$	X
	ACE (7 s)	*M*		564101	564101			0
mm_03_08_04	⟨2288, 2, 2252, 64, 30⟩	*Z*	8.77E−09	8.19E−09	8.19E−09	**8.77E−09**$^{*(1231\,s)}$	X	X
	ACE (12 s)	*M*		35101	35101			0
mm_03_08_05	⟨3692, 2, 3647, 80, 42⟩	*Z*	8.89E−11	7.27E−11	7.27E−11	**8.90E−11**$^{*(1503\,s)}$	X	X
	ACE (35 s)	*M*		10401	10401			0
mm_04_08_03	⟨1418, 2, 1391, 48, 22⟩	*Z*	8.39E−08	8.37E−08	8.37E−08	**8.39E−08**$^{*(7\,s)}$	X	X
	ACE (9 s)	*M*		379501	379501			0
mm_04_08_04	⟨2616, 2, 2580, 64, 33⟩	*Z*	2.20E−08	**1.84E−08**	**1.84E−08**	1.21E−08	X	X
	ACE (19 s)	*M*		12901	12901			0
mm_05_08_03	⟨1616, 2, 1589, 48, 28⟩	*Z*	5.29E−07	4.78E−07	4.78E−07	**5.30E−07**$^{*(229\,s)}$	**5.3E−07**$^{*(6194\,s)}$	X
	ACE (12 s)	*M*		60201	60201			0
mm_06_08_03	⟨1814, 2, 1787, 48, 31⟩	*Z*	1.79E−08	1.12E−08	1.12E−08	**1.80E−08**$^{*(2082\,s)}$	5.85E−09	X
	ACE (13 s)	*M*		113301	113301			0
mm_10_08_03	⟨2606, 2, 2579, 48, 56⟩	*Z*	1.92E−07	**5.01E−07**	**5.01E−07**	7.79E−08	2.39E−10	X
	ACE (27 s)	*M*		10801	10801			0

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, (c) the grid instances, and (d) the logistics planning instances. We measure the accuracy of the solvers using average Hellinger distance [57]. 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 divergence [58]. 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.

As pointed out in [57], Hellinger distance is superior to other choices such as the Kullback–Leibler (KL) divergence, the mean squared error and the relative error when zero or infinitesimally small probabilities are present. We do not use the KL divergence because it lies between 0 and ∞ and in practice when the exact marginals are 0 or close to it, floating-point

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.

Problem	$\langle n, k, c, e, w\rangle$ ACE time		IJGP-wc-SS	IJGP	EPIS	EDBP	IJGP-wc-IS
BN_69	⟨777, 7, 228, 78, 39⟩	Δ	**9.4E−04**	3.2E−02	1	8.0E−02	1
	ACE (timeout)	M	6.84E+5				0
BN_70	⟨2315, 5, 484, 159, 35⟩	Δ	**2.6E−03**	3.3E−02	1	9.6E−02	1
	ACE (233 s)	M	1.92E+5				0
BN_71	⟨1740, 6, 663, 202, 53⟩	Δ	**5.6E−03**	1.9E−02	1	2.5E−02	1
	ACE (timeout)	M	7.46E+4				0
BN_72	⟨2155, 6, 752, 252, 65⟩	Δ	**3.6E−03**	7.2E−03	1	1.3E−02	1
	ACE (timeout)	M	1.53E+5				0
BN_73	⟨2140, 5, 651, 216, 67⟩	Δ	**2.1E−02**	2.8E−02	1	6.1E−02	1
	ACE (timeout)	M	7.75E+4				0
BN_74	⟨749, 6, 223, 66, 35⟩	Δ	6.9E−04	**4.3E−06**	1	4.3E−02	1
	ACE (timeout)	M	2.80E+5				0
BN_75	⟨1820, 5, 477, 155, 37⟩	Δ	**8.0E−03**	6.2E−02	1	9.3E−02	1
	ACE (timeout)	M	7.72E+4				0
BN_76	⟨2155, 7, 605, 169, 53⟩	Δ	**1.8E−02**	2.6E−02	1	2.7E−02	1
	ACE (timeout)	M	2.52E+4				0

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 divergence.[14] We did compute the error using all other commonly used distance measures such as the mean squared error, the relative 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 [55]. We do not report on the pedigree instances from the UAI 2008 evaluation [44] because their exact marginals are not known. We can see that IJGP-wc-SS is more 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.[15] In Fig. 9, we demonstrate the superior anytime performance of IJGP-wc-SS compared with other solvers.

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 [22]. 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 12 and 559, 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 many 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 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

[14] Also see for example the results of the recent UAI evaluation [44]. Dechter and Mateescu [59] 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.

[15] The EPIS program does not output the number of consistent samples that were used in computing the marginals. It outputs an invalid marginal distribution for all variables (for example, it will output a marginal distribution of $(0, 0, 0)$ for a variable having 3 values in its domain) when it generates no consistent samples within the stipulated time-bound.

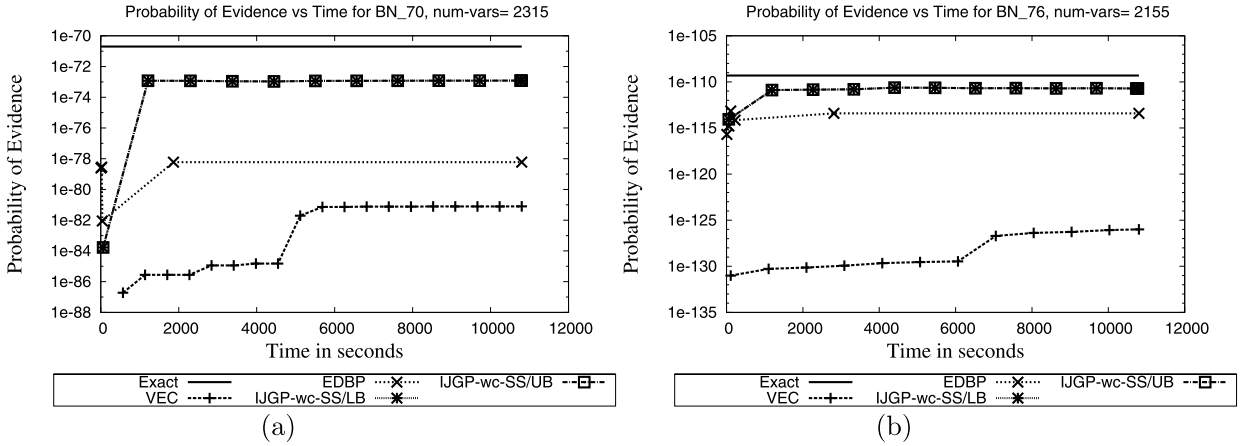


Fig. 9. Time versus Hellinger distance Δ between the exact and approximate marginals for IJGP-wc-IS, IJGP-wc-SS, IJGP, EPIS and EDBP for two sample *Linkage instances*.

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.

Problem	$\langle n, k, c, e, w \rangle$ ACE time		IJGP-wc-SS	IJGP	EPIS	EDBP	IJGP-wc-IS
Friends and smokers							
fs-04	⟨262, 2, 74, 226, 12⟩	Δ	5.4E−05	**4.6E−08**	1	6.4E−02	3.6E−06
	ACE (4 s)	M	1.00E+6				2.17E+8
fs-07	⟨1225, 2, 371, 1120, 35⟩	Δ	**1.4E−02**	1.6E−02	1	3.0E−02	1
	ACE (4 s)	M	1.00E+6				0
fs-10	⟨3385, 2, 1055, 3175, 71⟩	Δ	1.2E−02	**6.3E−03**	1	2.7E−02	1
	ACE (9 s)	M	8.51E+5				0
fs-13	⟨7228, 2, 2288, 6877, 117⟩	Δ	2.0E−02	6.5E−03	1	2.3E−02	**1.4E−04**
	ACE (10 s)	M	5.41E+5				4.67E+7
fs-16	⟨13240, 2, 4232, 12712, 171⟩	Δ	1.2E−03	6.8E−03	1	1.7E−02	**2.1E−05**
	ACE (14 s)	M	1.79E+5				1.37E+7
fs-19	⟨21907, 2, 7049, 21166, 243⟩	Δ	**3.1E−03**	8.8E−03	1	1	1
	ACE (23 s)	M	1.90E+5				0
fs-22	⟨33715, 2, 10901, 32725, 315⟩	Δ	**2.5E−03**	8.6E−03	1	1	1
	ACE (49 s)	M	1.18E+5				0
fs-25	⟨49150, 2, 15950, 47875, 431⟩	Δ	**2.5E−03**	8.4E−03	1	1	1
	ACE (74 s)	M	9.23E+4				0
fs-28	⟨68698, 2, 22358, 67102, 527⟩	Δ	**1.3E−03**	7.4E−03	1	1	1
	ACE (149 s)	M	9.35E+4				0
fs-29	⟨76212, 2, 24824, 74501, 559⟩	Δ	**1.9E−03**	7.0E−03	1	1	1
	ACE (168 s)	M	2.62E+4				0
Mastermind							
mm_03_08_03	⟨1220, 2, 1193, 48, 20⟩	Δ	**1.1E−03**	3.8E−02	1	3.8E−01	1
	ACE (7 s)	M	5.64E+5				0
mm_03_08_04	⟨2288, 2, 2252, 64, 30⟩	Δ	**1.1E−02**	4.4E−02	1	1	1
	ACE (12 s)	M	3.51E+4				0
mm_03_08_05	⟨3692, 2, 3647, 80, 42⟩	Δ	4.0E−02	**3.2E−02**	1	1	1
	ACE (35 s)	M	1.04E+4				0
mm_04_08_04	⟨2616, 2, 1391, 64, 33⟩	Δ	**3.1E−02**	3.5E−02	1	1	1
	ACE (19 s)	M	1.29E+4				0
mm_05_08_03	⟨1616, 2, 2580, 48, 28⟩	Δ	**1.0E−02**	3.6E−02	1	4.0E−02	1
	ACE (12 s)	M	6.02E+4				0
mm_06_08_03	⟨1814, 2, 1787, 48, 31⟩	Δ	**4.7E−03**	3.3E−02	5.6E−01	3.2E−01	1
	ACE (13 s)	M	1.13E+5				0
mm_10_08_03	⟨2606, 2, 2579, 48, 56⟩	Δ	**3.9E−02**	5.3E−02	1	8.3E−02	1
	ACE (27 s)	M	1.08E+4				0

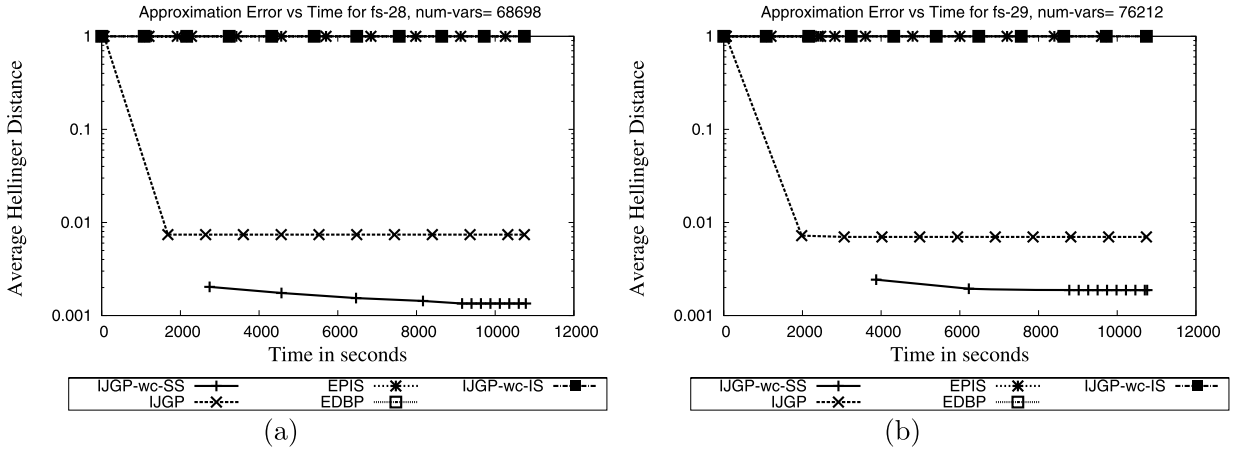


Fig. 10. Time versus Hellinger distance Δ between the exact and approximate marginals for IJGP-wc-IS, IJGP-wc-SS, IJGP, EPIS and EDBP for two sample *Friends and Smokers networks*.

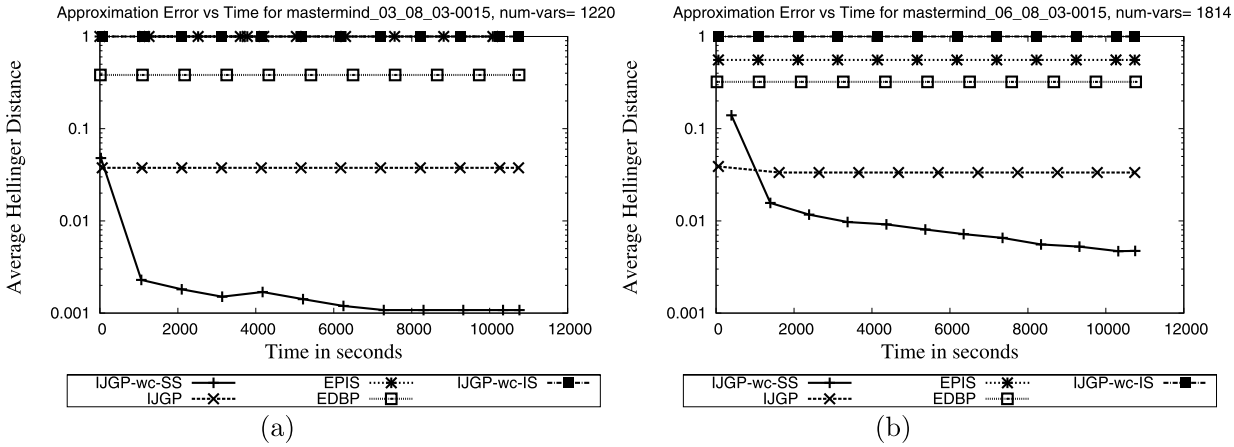


Fig. 11. Time versus Hellinger distance Δ between the exact and approximate marginals for IJGP-wc-IS, IJGP-wc-SS, IJGP, EPIS and EDBP for two sample *Mastermind networks*.

the relational consistency [6,59] power of IJGP to reduce rejection as a *pre-processing step* [10]. This highlights that sometimes using constraint-based inference to determine the inconsistencies before sampling is more cost-effective to 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. Figs. 10 and 11 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 [41]. 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 a function of 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 Figs. 12 through 14.

On grids with deterministic ratio of 50%, EPIS is the best performing scheme on all but two instances. On most instances, IJGP-wc-IS yields marginals having smaller error than IJGP-wc-SS. On four out of the six instances, the sampling schemes

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.

Problem	$\langle n, k, c, e, w \rangle$ ACE time		IJGP-wc-SS	IJGP	EPIS	EDBP	IJGP-wc-IS
Deterministic ratio = 50%							
50-12-5	⟨144, 2, 62, 1, 16⟩	Δ	4.3E−04	**3.2E−07**	2.6E−04	2.5E−02	1.5E−04
	ACE (3 s)	M	1.90E+6				1.23E+8
50-14-5	⟨196, 2, 93, 1, 20⟩	Δ	4.9E−04	1.8E−02	**1.2E−04**	4.0E−02	2.1E−04
	ACE (3 s)	M	9.37E+5				8.90E+7
50-15-5	⟨225, 2, 111, 1, 23⟩	Δ	4.9E−04	1.0E−02	**2.3E−04**	6.1E−02	6.5E−04
	ACE (6 s)	M	5.24E+5				7.68E+7
50-17-5	⟨289, 2, 138, 1, 25⟩	Δ	8.0E−04	2.1E−02	**2.0E−04**	3.6E−03	1.0E−03
	ACE (211 s)	M	4.34E+5				5.82E+7
50-18-5	⟨324, 2, 153, 1, 27⟩	Δ	9.3E−04	1.9E−02	**3.0E−04**	2.1E−03	7.6E−04
	ACE (timeout)	M	3.46E+5				5.15E+7
50-19-5	⟨361, 2, 172, 1, 28⟩	Δ	1.1E−03	3.4E−02	4.0E−04	**3.4E−04**	1.5E−03
	ACE (timeout)	M	2.87E+5				2.80E+7
Deterministic ratio = 75%							
75-16-5	⟨256, 2, 193, 1, 24⟩	Δ	6.5E−04	**2.5E−07**	1.7E−04	7.8E−02	1.4E−04
	ACE (7 s)	M	9.74E+5				7.11E+7
75-17-5	⟨289, 2, 217, 1, 25⟩	Δ	1.4E−03	**2.6E−07**	2.7E−04	1.2E−03	1.6E−04
	ACE (9 s)	M	7.15E+5				5.41E+7
75-18-5	⟨324, 2, 245, 1, 27⟩	Δ	1.2E−03	3.9E−02	2.0E−04	5.0E−03	**1.9E−04**
	ACE (11 s)	M	4.47E+5				5.23E+7
75-19-5	⟨361, 2, 266, 1, 28⟩	Δ	9.0E−03	4.3E−02	2.5E−04	**6.7E−05**	1.9E−04
	ACE (14 s)	M	4.07E+5				3.93E+7
75-20-5	⟨400, 2, 299, 1, 30⟩	Δ	6.2E−04	**3.1E−07**	1.9E−04	1.7E−02	2.8E−04
	ACE (11 s)	M	4.10E+5				2.64E+7
75-21-5	⟨441, 2, 331, 1, 32⟩	Δ	1.9E−03	**2.9E−07**	2.8E−04	1.5E−02	6.2E−04
	ACE (60 s)	M	3.13E+5				2.33E+7
75-22-5	⟨484, 2, 361, 1, 35⟩	Δ	3.2E−03	2.3E−02	**2.6E−04**	2.0E−02	5.4E−04
	ACE (78 s)	M	2.67E+5				2.12E+7
75-23-5	⟨529, 2, 406, 1, 35⟩	Δ	2.0E−03	4.8E−02	**2.3E−04**	2.4E−02	7.1E−04
	ACE (420 s)	M	1.75E+5				1.77E+7
75-24-5	⟨576, 2, 442, 1, 38⟩	Δ	8.4E−03	4.3E−02	**2.6E−04**	3.5E−02	8.9E−04
	ACE (228 s)	M	1.29E+5				2.61E+7
75-26-5	⟨676, 2, 506, 1, 44⟩	Δ	2.4E−02	5.1E−02	**3.5E−04**	5.1E−02	1.4E−03
	ACE (timeout)	M	1.25E+5				2.20E+7
Deterministic ratio = 90%							
90-20-5	⟨400, 2, 356, 1, 30⟩	Δ	1.6E−03	**2.7E−07**	2.5E−04	3.7E−02	6.5E−05
	ACE (8 s)	M	8.32E+5				4.77E+7
90-22-5	⟨484, 2, 430, 1, 35⟩	Δ	4.6E−04	**2.8E−07**	1.5E−04	5.1E−02	1.0E−04
	ACE (7 s)	M	4.42E+5				3.97E+7
90-23-5	⟨529, 2, 468, 1, 35⟩	Δ	2.8E−04	**3.2E−07**	3.9E−04	1.9E−02	7.0E−05
	ACE (9 s)	M	6.70E+5				4.00E+7
90-24-5	⟨576, 2, 528, 1, 38⟩	Δ	5.0E−04	**3.9E−07**	3.5E−04	2.8E−02	9.2E−05
	ACE (6 s)	M	7.01E+5				2.29E+7
90-25-5	⟨625, 2, 553, 1, 39⟩	Δ	**2.7E−07**	**2.7E−07**	3.4E−04	4.6E−02	**2.7E−07**
	ACE (7 s)	M	7.04E+5				2.57E+7
90-26-5	⟨676, 2, 597, 1, 44⟩	Δ	1.0E−03	**1.9E−06**	2.3E−04	3.9E−02	1.9E−04
	ACE (10 s)	M	4.13E+5				2.90E+7
90-34-5	⟨1156, 2, 1048, 1, 65⟩	Δ	8.6E−04	**1.8E−07**	3.9E−04	4.1E−02	6.3E−04
	ACE (25 s)	M	2.80E+5				1.37E+7
90-38-5	⟨1444, 2, 1300, 1, 69⟩	Δ	1.6E−02	**4.3E−07**	1.7E−03	1.6E−01	1.0E−03
	ACE (136 s)	M	1.15E+5				7.08E+6

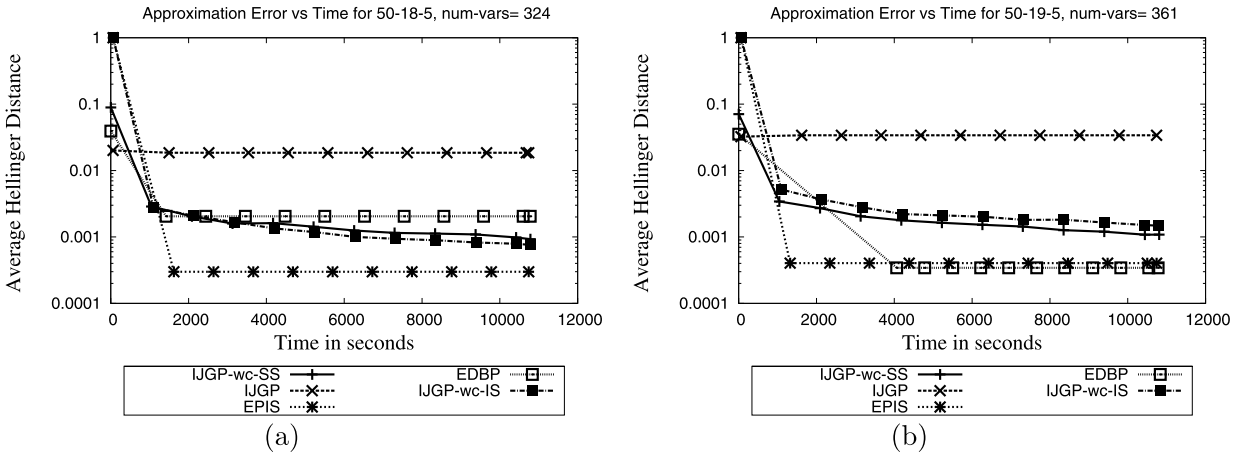


Fig. 12. Time versus Hellinger distance Δ between the exact and approximate marginals for IJGP-wc-IS, IJGP-wc-SS, IJGP, EPIS and EDBP for two sample *Grid instances with deterministic ratio* = 50%.

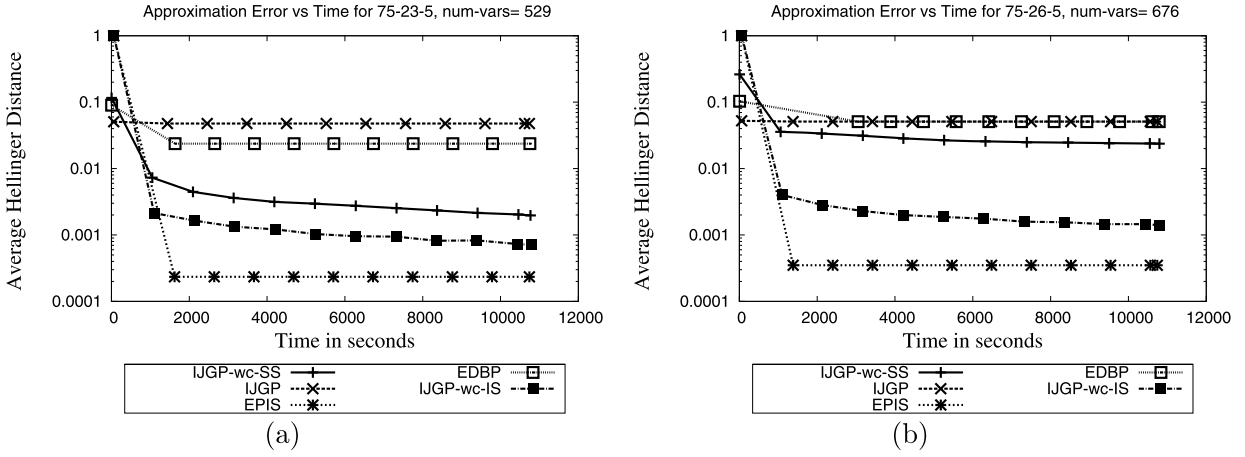


Fig. 13. Time versus Hellinger distance Δ between the exact and approximate marginals for IJGP-wc-IS, IJGP-wc-SS, IJGP, EPIS and EDBP for two sample *Grid instances with deterministic ratio* = 75%.

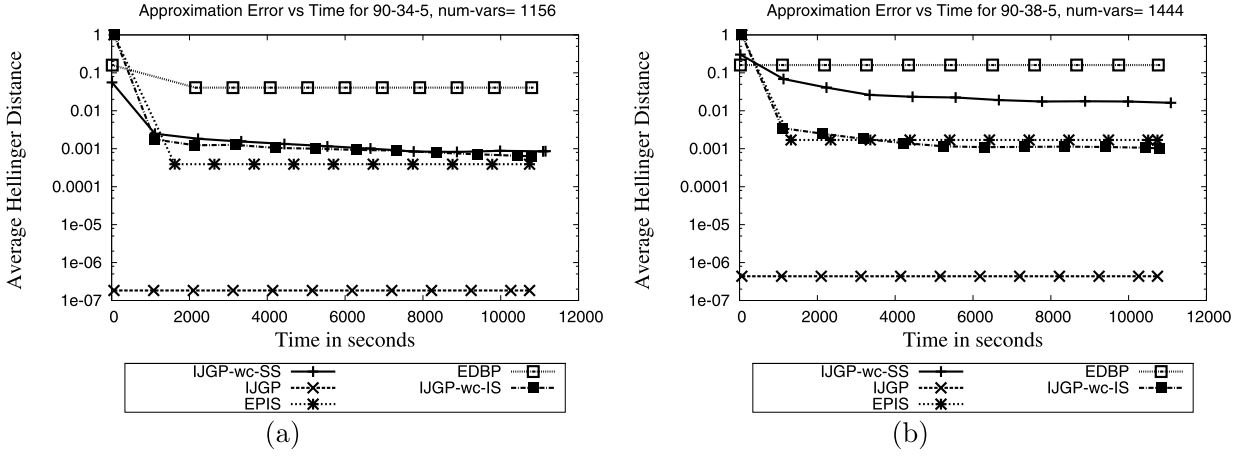


Fig. 14. Time versus Hellinger distance Δ between the exact and approximate marginals for IJGP-wc-IS, IJGP-wc-SS, IJGP, EPIS and EDBP for two sample *Grid instances with deterministic ratio* = 90%.

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.

Problem	$\langle n, k, c, e, w\rangle$ ACE time		IJGP-wc-SS	IJGP	EPIS	EDBP	IJGP-wc-IS
log-1	⟨4724, 2, 3785, 3785, 22⟩	Δ	2.2E−05	**0**$^{*(2\ s)}$	1	1	1
	ACE (1 s)	M	1.35E+8				0
log-2	⟨26114, 2, 24777, 24777, 51⟩	Δ	**8.6E−04**	9.8E−03	1	1	1
	ACE (58 s)	M	1.49E+6				0
log-3	⟨30900, 2, 29487, 29487, 56⟩	Δ	**1.2E−04**	7.5E−03	1	1	1
	ACE (23 s)	M	1.05E+5				0
log-4	⟨23266, 2, 20963, 20963, 52⟩	Δ	**2.3E−02**	1.8E−01	1	1	1
	ACE (68 s)	M	1.03E+5				0
log-5	⟨32235, 2, 29534, 29534, 51⟩	Δ	**8.6E−03**	1.2E−02	1	1	1
	ACE (727 s)	M	9.73E+3				0

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 this is because of the overhead of the exact inference step of w-cutset sampling and also because of the min-fill ordering used by IJGP-wc-IS. We found that the topological ordering (which is used by EPIS) performs better than the min-fill ordering. Specifically, we found that when we set $w = 0$ and use topological ordering, the performance of IJGP-wc-IS and EPIS is almost the same (results not shown).

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.

The poor performance of IJGP-wc-SS as compared with EPIS and IJGP-wc-IS is because of its search overhead. On grid networks, rejection is not an issue for the IJGP-wc-IS and EPIS solvers because the deterministic portion is easy for inference. Although, it may seem on surface that both EPIS and IJGP-wc-IS do not reason about determinism, it is not the case. It is known that Loopy Belief propagation, when run until convergence makes the constraint portion of the mixed network relationally-arc-consistent [59]. Therefore, if the constraint network has small treewidth, Loopy BP (or IJGP) may yield a proposal distribution that is either backtrack-free or almost backtrack-free. Note however that enforcing relational consistency may reduce but would not completely eliminate the rejection problem. Typically, to guarantee that a backtrack-free distribution is obtained, one has to use a consistency enforcement scheme whose time and space complexity is bounded by the treewidth of the constraint portion of the mixed network (see [60], Chapter 2 for details). Overall, when the treewidth of the constraint portion is large, SampleSearch is the only practical alternative available to date.

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 [41].

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 remaining instances, IJGP-wc-SS is more accurate than IJGP. In Fig. 15, we demonstrate the superior anytime performance of IJGP-wc-SS as compared with the other schemes.

5.5. Impact of ordering heuristics

Table 13 shows the impact of using different variable ordering heuristics on the performance of IJGP-wc-SS measured in terms of the Hellinger distance between the exact and the approximate marginals. For brevity, we show the results for a few sample instances from each domain. We can clearly see that except for the Grid instances, on average, the min-fill ordering performs better than the other schemes. The topological ordering scheme performs the best on the grid instances. hmetis and min-degree ordering are the worst performing schemes.

5.6. Summary of experimental evaluation

We implemented SampleSearch on top of an advanced importance sampling technique IJGP-wc-IS presented in [10]; yielding a scheme called IJGP-wc-SS. The search was implemented using minisat [15]. For model counting, we compared IJGP-wc-SS with three other approximate solution counters available in literature: ApproxCount [16], SampleCount [17] and

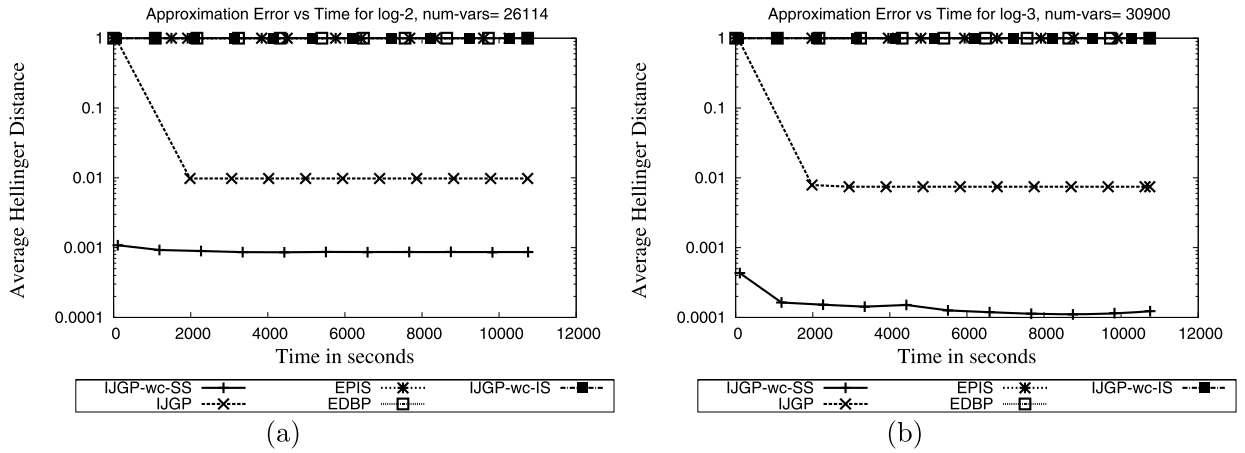


Fig. 15. Time versus Hellinger distance Δ between the exact and approximate marginals for IJGP-wc-IS, IJGP-wc-SS, IJGP, EPIS and EDBP for two sample *Logistics planning instances*.

Table 13
Table showing the effect of the four ordering heuristics: min-fill, min-degree, hmetis and topological on the *Hellinger distance* Δ *between the exact and approximate marginals* computed by IJGP-wc-SS. The time-bound used was 3 hours. The best performing scheme is highlighted in each row. For each ordering heuristic, we report its treewidth w.

Problem	$\langle n, k, e \rangle$		IJGP-wc-SS			
			min-fill	min-degree	topological	hmetis
Linkage						
BN_69	⟨777, 7, 78⟩	Δ	**9.4E−04**	2.0E−03	4.7E−03	2.2E−03
		w	39	38	122	39
BN_70	⟨2315, 5, 159⟩	Δ	**2.6E−03**	1.4E−02	7.5E−03	8,3E−03
		w	35	56	144	51
BN_75	⟨1820, 5, 155⟩	Δ	8.0E−03	**2.2E−03**	2.1E−02	5.5E−03
		w	37	41	178	46
BN_76	⟨2155, 7, 169⟩	Δ	**1.8E−02**	1.5E−01	3.2E−02	**1.8E−02**
		w	37	40	333	40
Grids						
50-18-5	⟨324, 2, 1⟩	Δ	9.3E−04	7.1E−03	**3.3E−04**	2.3E−03
		w	27	27	21	30
50-19-5	⟨361, 2, 1⟩	Δ	1.1E−03	1.3E−03	3.5E−04	1.8E−03
		w	28	27	21	28
75-24-5	⟨576, 2, 1⟩	Δ	4.3E−02	3.8E−02	**1.9E−03**	2.2E−02
		w	38	40	37	38
75-26-5	⟨676, 2, 1⟩	Δ	2.4E−02	8.0E−02	**8E−04**	4.5E−02
		w	44	48	38	46
90-34-5	⟨1156, 2, 1⟩	Δ	**8.6E−04**	1.6E−03	1.4E−03	9.4E−04
		w	65	65	51	61
90-38-5	⟨1444, 2, 1⟩	Δ	1.6E−02	1.6E−02	**3.0E−03**	4.5E−02
		w	69	69	56	69
Relational						
fs-28	⟨68698, 2, 67102⟩	Δ	1.1E−03	1.3E−03	**6.4E−04**	2.7E−03
		w	527	527	632	719
fs-29	⟨76212, 2, 74501⟩	Δ	**1.9E−03**	2.1E−03	6.8E−03	3.4E−03
		w	559	559	803	799
mm_06_08_03-0015	⟨1814, 2, 48⟩	Δ	**4.7E−03**	6.1E−03	1.9E−02	8.5E−03
		w	31	31	173	35
mm_10_08_03-0015	⟨2606, 2, 48⟩	Δ	**3.9E−02**	6.5E−02	8.8E−02	5.6E−02
		w	56	56	185	48

Relsat [18] 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 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) [19] and an advanced generalized belief propagation scheme called Edge Deletion Belief Propagation (EDBP) [20] 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 [12] and Edge Deletion Belief Propagation [20], an adaptive importance sampling scheme called Evidence Pre-propagated Importance Sampling (EPIS) [23] 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 IJGP-wc-IS and EPIS did not fail, they generated more consistent samples and performed better than IJGP-wc-SS. On the remaining instances, IJGP-wc-SS was clearly superior.

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 overcomes the rejection problem by interleaving random sampling with systematic backtracking. Specifically, when sampling variables one by one via logic sampling [5], 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 the integration of any systematic SAT/CSP solver within SampleSearch (with minor modifications). Indeed, in our experiments, we used an advanced SAT solver called minisat [15]. 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 as the backtrack-free distribution, which is 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 accurate on most benchmarks. 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 [23] and IJGP-wc-IS [10] 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) [12] and Edge Deletion Belief Propagation (EDBP) [20]. 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 their memory requirement becomes a major bottleneck beyond a certain i-bound (typically > 22). Consequently, on most benchmarks, we observed that IJGP and EDBP quickly converge to an estimate which they are unable to improve with time. On the other hand, as we demonstrated SampleSearch improves with time and 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 [17] technique and substantially better than ApproxCount [16] and Relsat [18].

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 [61] 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. Instead, we can easily use the AND/OR

search space [62]. Borrowing ideas from the literature on ordered binary decision diagrams (OBDDs) [63], 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 [30,33,64]. 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 number IIS-0713118 and by the NIH grant R01-HG004175-02.

Appendix 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:

$$\sum_{x_i' \in \mathbf{B}_i^{\mathbf{x}_{i-1}}} Q_i\left(x_i'|\mathbf{x}_{i-1}\right) \leqslant \sum_{x_i' \in \mathbf{A}_{N,i}^{\mathbf{x}_{i-1}} \cup \mathbf{C}_{N,i}^{\mathbf{x}_{i-1}}} Q_i\left(x_i'|\mathbf{x}_{i-1}\right) \tag{40}$$

$$\therefore 1 - \sum_{x_i' \in \mathbf{B}_i^{\mathbf{x}_{i-1}}} Q_i\left(x_i'|\mathbf{x}_{i-1}\right) \geqslant 1 - \sum_{x_i' \in \mathbf{A}_{N,i}^{\mathbf{x}_{i-1}} \cup \mathbf{C}_{N,i}^{\mathbf{x}_{i-1}}} Q_i\left(x_i'|\mathbf{x}_{i-1}\right) \tag{41}$$

$$\therefore \frac{Q_i(x_i|\mathbf{x}_{i-1})}{1 - \sum_{x_i' \in \mathbf{B}_i^{\mathbf{x}_{i-1}}} Q_i(x_i'|\mathbf{x}_{i-1})} \leqslant \frac{Q_i(x_i|\mathbf{x}_{i-1})}{1 - \sum_{x_i' \in \mathbf{A}_{N,i}^{\mathbf{x}_{i-1}} \cup \mathbf{C}_{N,i}^{\mathbf{x}_{i-1}}} Q_i(x_i'|\mathbf{x}_{i-1})} \tag{42}$$

$$\therefore Q_i^F(x_i|\mathbf{x}_{i-1}) \leqslant L_{N,i}^F(x_i|\mathbf{x}_{i-1}) \tag{43}$$

$$\therefore \prod_{i=1}^{n} Q_i^F(x_i|\mathbf{x}_{i-1}) \leqslant \prod_{i=1}^{n} L_{N,i}^F(x_i|\mathbf{x}_{i-1}) \tag{44}$$

$$\therefore Q^F(\mathbf{x}) \leqslant L_N^F(\mathbf{x}) \tag{45}$$

$$\therefore \frac{\prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})}{Q^F(\mathbf{x})} \geqslant \frac{\prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})}{L_N^F(\mathbf{x})} \tag{46}$$

$$\therefore w^F(\mathbf{x}) \geqslant w_L^F(\mathbf{x}) \tag{47}$$

$$\therefore \frac{1}{N}\sum_{k=1}^{N} w^F\left(\mathbf{x}^k\right) \geqslant \frac{1}{N}\sum_{k=1}^{N} w_L^F\left(\mathbf{x}^k\right) \tag{48}$$

$$\therefore \widehat{Z}_N \geqslant \widetilde{Z}_N^L \tag{49}$$

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 $\widehat{Z}_N^F \leqslant \widetilde{Z}_N^U$. □

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,

$$\lim_{N\to\infty} w_N^L(\mathbf{x}) = \lim_{N\to\infty} \frac{\prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})}{L_N^F(\mathbf{x})} \tag{50}$$

$$= \frac{\prod_{i=1}^{m} F_i(\mathbf{x}) \prod_{j=1}^{p} C_j(\mathbf{x})}{Q^F(\mathbf{x})} \tag{51}$$

$$= w^F(\mathbf{x}) \tag{52}$$

Therefore,

$$\lim_{N\to\infty} \mathbb{E}_Q\left[\frac{1}{N}\sum_{k=1}^{N} w^L(\mathbf{x})\right] = \lim_{N\to\infty} \frac{1}{N}\sum_{\mathbf{x}\in\mathbf{X}} w_N^L(\mathbf{x}) Q(\mathbf{x}) \sum_{k=1}^{N}(1) \tag{53}$$

$$= \frac{1}{N} \times N \lim_{N\to\infty} \sum_{\mathbf{x}\in\mathbf{X}} w_N^L(\mathbf{x}) Q(\mathbf{x}) \tag{54}$$

$$= \sum_{\mathbf{x} \in \mathbf{X}} w^F(\mathbf{x}) Q(\mathbf{x}) \ldots \quad \big(\text{from Eq. (52)}\big) \tag{55}$$

$$= Z \tag{56}$$

Similarly, we can prove that the estimator based on U_N^F in Eq. (34) is asymptotically unbiased by replacing $w_N^L(\mathbf{x})$ with $w_N^U(\mathbf{x})$ in Eqs. (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 Eqs. (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 $O(N \times d \times n)$. □

References

[1] R. Dechter, D. Larkin, Hybrid processing of beliefs and constraints, in: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence (UAI), 2001, pp. 112–119.
[2] D. Larkin, R. Dechter, Bayesian inference in the presence of determinism, in: Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS), 2003.
[3] 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), 2004, pp. 120–129.
[4] R. Mateescu, R. Dechter, Mixed deterministic and probabilistic networks, Annals of Mathematics and Artificial Intelligence (AMAI), Special Issue: Probabilistic Relational Learning 54 (1–3) (2008) 3–51.
[5] J. Pearl, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann, 1988.
[6] R. Dechter, Constraint Processing, Morgan Kaufmann, 2003.
[7] A.W. Marshall, The use of multi-stage sampling schemes in Monte Carlo computations, in: Symposium on Monte Carlo Methods, 1956, pp. 123–140.
[8] R.Y. Rubinstein, Simulation and the Monte Carlo Method, John Wiley & Sons Inc., 1981.
[9] J. Geweke, Bayesian inference in econometric models using Monte Carlo integration, Econometrica 57 (6) (1989) 1317–1339.
[10] 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), 2005, pp. 209–216.
[11] 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.
[12] R. Dechter, K. Kask, R. Mateescu, Iterative join graph propagation, in: Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence (UAI), Morgan Kaufmann, 2002, pp. 128–136.
[13] R. Mateescu, K. Kask, V. Gogate, R. Dechter, Join-graph propagation algorithms, Journal of Artificial Intelligence Research 37 (2009) 279–328.
[14] B. Bidyuk, R. Dechter, Cutset sampling for Bayesian networks, Journal of Artificial Intelligence Research (JAIR) 28 (2007) 1–48.
[15] N. Sorensson, N. Een, Minisat v1.13–A SAT solver with conflict–clause minimization, in: SAT 2005 Competition, 2005.
[16] W. Wei, J. Erenrich, B. Selman, Towards efficient sampling: exploiting random walk strategies, in: Proceedings of the Nineteenth National Conference on Artificial Intelligence, 2004, pp. 670–676.
[17] 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), 2007, pp. 2293–2299.
[18] J. Roberto, J. Bayardo, J.D. Pehoushek, Counting models using connected components, in: Proceedings of 17th National Conference on Artificial Intelligence (AAAI), 2000, pp. 157–162.
[19] R. Dechter, Bucket elimination: A unifying framework for reasoning, Artificial Intelligence 113 (1999) 41–85.
[20] 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), 2006, pp. 1107–1114.
[21] M. Fishelson, D. Geiger, Optimizing exact genetic linkage computations, in: Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology (RECOMB), 2003, pp. 114–121.
[22] 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.
[23] C. Yuan, M.J. Druzdzel, Importance sampling algorithms for Bayesian networks: Principles and performance, Mathematical and Computer Modelling (ISSN 0895-7177) 43 (90–10) (2006) 1189–1207.
[24] V. Gogate, R. Dechter, SampleSearch: A scheme that searches for consistent samples, in: Proceedings of the 11th Conference on Artificial Intelligence and Statistics (AISTATS), 2007, pp. 147–154.
[25] V. Gogate, R. Dechter, Approximate counting by sampling the backtrack-free search space, in: Proceedings of 22nd Conference on Artificial Intelligence (AAAI), 2007, pp. 198–203.
[26] J. Liu, Monte-Carlo Strategies in Scientific Computing, Springer-Verlag, New York, 2001.
[27] E.C. Freuder, A sufficient condition for backtrack-free search, Journal of the ACM 29 (1) (1982) 24–32.
[28] 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, 2000, pp. 441–456.
[29] K. Pipatsrisawat, A. Darwiche, RSAT 2.0: SAT solver description, Tech. Rep. D-153, Automated Reasoning Group, Computer Science Department, UCLA, 2007.
[30] 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.
[31] R.D. Shachter, M.A. Peot, Simulation approaches to general probabilistic inference on belief networks, in: Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence (UAI), 1990, pp. 221–234.
[32] R.M. Fung, K.-C. Chang, Weighing and integrating evidence for stochastic simulation in Bayesian networks, in: Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence (UAI), 1990, pp. 209–220.
[33] L. Ortiz, L. Kaelbling, Adaptive importance sampling for estimation in structured domains, in: Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence (UAI), 2000, pp. 446–454.

[34] R. Fung, B. del Favero, Backward simulation in Bayesian networks, in: Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI), 1994, pp. 227–234.
[35] K. Kask, R. Dechter, J. Larrosa, A. Dechter, Unifying tree decompositions for reasoning in graphical models, Artificial Intelligence 166 (1–2) (2005) 165–193.
[36] G. Casella, C.P. Robert, Rao-blackwellisation of sampling schemes, Biometrika 83 (1) (1996) 81–94, doi:10.1093/biomet/83.1.81.
[37] A. Darwiche, M. Hopkins, Using recursive decomposition to construct elimination orders, jointrees, and dtrees, in: Trends in Artificial Intelligence, in: Lecture Notes in AI, Springer-Verlag, 2001, pp. 180–191.
[38] B. Bidyuk, R. Dechter, On finding minimal w-cutset problem, in: Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence (UAI), 2004, pp. 43–50.
[39] W. Wei, B. Selman, A new approach to model counting, in: Proceedings of Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT), 2005, pp. 324–339.
[40] L.G. Valiant, The complexity of enumeration and reliability problems, SIAM Journal of Computation 8 (3) (1987) 105–117.
[41] T. Sang, P. Beame, H. Kautz, Heuristics for fast exact model counting, in: Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT), 2005, pp. 226–240.
[42] B. Selman, H. Kautz, B. Cohen, Noise strategies for local search, in: Proceedings of the Eleventh National Conference on Artificial Intelligence, 1994, pp. 337–343.
[43] K.P. Murphy, Y. Weiss, M.I. Jordan, Loopy belief propagation for approximate inference: an empirical study, in: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI), 1999, pp. 467–475.
[44] A. Darwiche, R. Dechter, A. Choi, V. Gogate, L. Otten, Results from the probabilistic inference evaluation of UAI'08, available online at: http://graphmod.ics.uci.edu/uai08/Evaluation/Report, 2008.
[45] 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.
[46] A. Darwiche, A differential approach to inference in Bayesian networks, Journal of the ACM 50 (3) (2003) 280–305.
[47] A. Darwiche, New advances in compiling CNF into decomposable negation normal form, in: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI), 2004, pp. 328–332.
[48] A. Darwiche, P. Marquis, A knowledge compilation map, Journal of Artificial Intelligence Research (JAIR) 17 (2002) 229–264.
[49] 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.
[50] T. Ritter, Latin squares: A literature survey, available online at: http://www.ciphersbyritter.com/RES/LATSQ.HTM.
[51] T. Walsh, Permutation problems and channelling constraints, in: Proceedings of the 8th International Conference on Logic Programming and Automated Reasoning (LPAR), 2001, pp. 377–391.
[52] 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), 2007, pp. 141–148.
[53] L. Simon, D.L. Berre, E. Hirsch, The SAT 2002 competition, Annals of Mathematics and Artificial Intelligence (AMAI) 43 (2005) 307–342.
[54] J. Ott, Analysis of Human Genetic Linkage, The Johns Hopkins University Press, Baltimore, Maryland, 1999.
[55] J. Bilmes, R. Dechter, Evaluation of probabilistic inference systems of UAI'06, available online at http://ssli.ee.washington.edu/bilmes/uai06Inference Evaluation/, 2006.
[56] K. Kask, R. Dechter, A. Gelfand, BEEM: bucket elimination with external memory, in: 26th Conference on Uncertainty in Artificial Intelligence (UAI), 2010, pp. 268–276.
[57] G. Kokolakis, P. Nanopoulos, Bayesian multivariate micro-aggregation under the Hellinger distance criterion, Research in Official Statistics 4 (2001) 117–125.
[58] S. Kullback, R.A. Leibler, On information and sufficiency, The Annals of Mathematical Statistics 22 (1) (1951) 79–86.
[59] R. Dechter, R. Mateescu, A simple insight into iterative belief propagation's success, in: Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence (UAI), 2003, p. 175–183.
[60] V. Gogate, Sampling algorithms for probabilistic and deterministic graphical models, PhD thesis, University of California, Irvine, 2009.
[61] N. Eén, N. Sörensson, Temporal induction by incremental SAT solving, Electronic Notes in Theoretical Computer Science (ISSN 1571-0661) 89 (4) (2003) 543–560.
[62] R. Dechter, R. Mateescu, AND/OR search spaces for graphical models, Artificial Intelligence 171 (2–3) (2007) 73–106.
[63] R.E. Bryant, Graph-based algorithms for Boolean function manipulation, IEEE Transactions on Computers 35 (8) (1986) 677–691.
[64] S. Moral, A. Salmerón, Dynamic importance sampling in Bayesian networks based on probability trees, International Journal of Approximate Reasoning 38 (3) (2005) 245–261.