Bayesian Learning in Bayesian Networks of Moderate Size by Efficient Sampling

Ru He and Jin Tian
Department of Computer Science
Iowa State University
Ames, IA 50011
$\{$*rhe, jtian*$\}$@*iastate.edu*

Abstract

We study the Bayesian model averaging approach to learning Bayesian network structures (DAGs) from data. We develop new algorithms including the first algorithm that is able to efficiently sample DAGs according to the exact structure posterior. The DAG samples can then be used to construct the estimators for the posterior of any feature. Our estimators have several good properties; for example, unlike the existing MCMC-based algorithms, quality guarantee can be provided for our estimators when assuming the order-modular prior. We empirically show that our algorithms considerably outperform previous state-of-the-art methods.

1 INTRODUCTION

Learning the structures of Bayesian networks (BNs) from data has been an active research problem. One approach to the problem is to treat it as a model selection problem. We define a criterion that measures how well a network structure (DAG) fits the data and find the optimal DAG (or a set of equivalent DAGs) (Silander and Myllymaki, 2006). However, when the data size is small relative to the number of variables, the posterior $p(G|D)$ of DAG G given data D often gives significant support to a number of structures, and using a single maximum-a-posteriori (MAP) model could lead to unwarranted conclusions (Friedman and Koller, 2003). It is therefore desirable to use the Bayesian model averaging approach by which we compute the posterior probability of any hypothesis of interest by averaging over all the possible DAGs (Heckerman et al., 1999).

Bayesian model averaging is, however, computationally challenging because the number of possible network structures is at least $2^{n(n-1)/2}$, super-exponential in the number of variables n. Tractable algorithms have been developed for special cases of averaging over trees (Meila and Jaakkola, 2006) and averaging over DAGs given a node ordering (Dash and Cooper, 2004). Since 2004, dynamic programming (DP) algorithms have been developed for computing exact posterior probabilities of structural features such as edges or subnetworks (Koivisto and Sood, 2004; Koivisto, 2006; Tian and He, 2009). These algorithms have exponential time and space complexity and are capable of handling Bayesian networks of moderate size with up to around 25 variables (mainly due to their space cost $O(n2^n)$). A limitation of these algorithms is that they can only compute posteriors of modular features such as an edge but can not compute non-modular features such as a path ("is there a path from node X to node Y"), a combined path ("is there a path from node X via node Y to node Z" or "is there a path from node X to node Y and no path from node X to node Z"), or a limited-length path ("is there a path of length at most 3 from node X to node Y"). Recently, Parviainen and Koivisto (2011) have developed a DP algorithm that can compute the exact posterior probability of a path feature. This DP algorithm has (even higher) exponential time and space complexity and can only handle a Bayesian network with less than 20 variables (mainly due to its space cost $O(3^n)$). Since this DP algorithm can only deal with a path feature, all the other non-modular features (such as a combined path or a limited-length path) still can not be computed by any DP algorithm proposed so far. Another limitation of all these DP algorithms is that it is very expensive for them to perform data prediction tasks. They can compute the exact posterior of a new observational data case $p(x|D)$ but the algorithms have to be re-run for each new data case x.

One approach to computing the posterior of an arbitrary non-modular feature is drawing DAG (directed acyclic graph) samples $\{G_1, \ldots, G_T\}$ from the posterior $p(G|D)$, which can then be used to approximate the full Bayesian model averaging to estimate the posterior of an arbitrary feature f as $p(f|D) \approx \frac{1}{T}\sum_{i=1}^{T} f(G_i)$, or the posterior predictive distribution as $p(x|D) \approx \frac{1}{T}\sum_{i=1}^{T} p(x|G_i)$. A number of algorithms have been developed for drawing sample DAGs using the bootstrap technique (Friedman et al., 1999) or Markov Chain Monte Carlo (MCMC) techniques (Madigan and York, 1995; Friedman and Koller, 2003; Eaton and Murphy, 2007; Grzegorczyk and Husmeier, 2008; Niinimaki et al., 2011). Madigan and York (1995) developed a *Structure MCMC* algorithm that uses the Metropolis-Hastings algorithm in the space of DAGs. Friedman and Koller (2003) developed *Order MCMC* procedure that operates in the space of orders. Order MCMC was shown to be able to considerably improve over Structure MCMC the mixing and convergence of the Markov chain and to outperform the bootstrap approach in (Friedman et al., 1999) as well. Eaton and Murphy (2007) developed *Hybrid MCMC* method (i.e., DP+MCMC method) that first runs the DP algorithm (Koivisto, 2006) to develop a global proposal distribution and then runs MCMC in the DAG space. Their experiments showed that Hybrid MCMC converged faster than Structure MCMC and Order MCMC and resulted in more accurate structure learning performance. Recently, Niinimaki et al. (2011) have proposed *Partial Order MCMC* method which operates in the space of partial orders. Partial Order MCMC includes Order MCMC as its special case (by setting the parameter bucket size b to be 1) and was shown to be superior to Order MCMC in terms of mixing and structural learning performance when a more appropriate bucket size $b > 1$ was set. One common drawback of these MCMC algorithms is that there is no guarantee on the quality of the approximation in finite runs. The approach to approximating full Bayesian model averaging using the K*-best* Bayesian network structures was studied in (Tian et al., 2010) and was shown to be at least as good as Hybrid MCMC.

Several of these state-of-the-art algorithms work in the order space, including the exact algorithms in (Koivisto and Sood, 2004; Koivisto, 2006; Parviainen and Koivisto, 2011) and the approximate algorithms Order MCMC (Friedman and Koller, 2003) and Partial Order MCMC (Niinimaki et al., 2011). They all assume a special form of the structure prior, termed as *order-modular* prior (Friedman and Koller, 2003; Koivisto and Sood, 2004), for computational convenience. However, the assumption of order-modular prior has the consequence that the corresponding prior $p(G)$ cannot represent some desirable priors such as a uniform prior over the DAG space; and the computed posterior probabilities are biased since a DAG that has a larger number of topological orders will be assigned a larger prior probability. (The detailed discussion about this bias issue can be found in (Friedman and Koller, 2003; Grzegorczyk and Husmeier, 2008; Parviainen and Koivisto, 2011).) One method that helps Order MCMC (Friedman and Koller, 2003) to correct this bias was proposed by Ellis and Wong (2008).

In this paper, first we develop a new algorithm that can efficiently sample orders according to the *exact* order posterior under the assumption of order-modular prior. Next, since a DAG consistent with a given order can be efficiently sampled as described in (Friedman and Koller, 2003) (by assuming fixed node-indegree), our order sampling algorithm leads to the first algorithm (called DDS) that can efficiently sample DAGs according to the *exact* DAG posterior with the same order-modular prior assumption. Our DDS algorithm has the same time and space complexity as the exact DP algorithms (Koivisto and Sood, 2004; Koivisto, 2006), and is shown to be considerably more accurate and more efficient than Order MCMC and Partial Order MCMC when n is moderate so that our DDS algorithm is applicable. Moreover, the estimator based on our DDS algorithm has several desirable properties; for example, unlike these MCMC algorithms, the quality of our estimator can be guaranteed by controlling the number of DAGs sampled by our DDS algorithm. The main application of our DDS algorithm is to serve as a complement to the exact DP algorithms (Koivisto and Sood, 2004; Koivisto, 2006; Parviainen and Koivisto, 2011) (which can only work for modular features or path features) to estimate the posteriors of arbitrary non-modular features such as combined paths or limited-length paths, or to efficiently perform data prediction tasks in estimating $p(x|D)$ for a large number of data cases. Finally, we develop an algorithm (called IW-DDS) to correct the bias (due to the order-modular prior) in the DDS algorithm using an idea refined from the one in Ellis and Wong (2008). The estimator based on our IW-DDS is consistent and asymptotically unbiased; and we empirically show that our estimator is superior to the estimators based on Hybrid MCMC (Eaton and Murphy, 2007) and K-best method (Tian et al., 2010), two state-of-the-art algorithms that can estimate the posterior of any feature without the order-modular prior assumption. Analogously, our IW-DDS algorithm serves as a complement to the exact DP algorithm (Tian and He, 2009) to estimate the posteriors of arbitrary non-modular features or to efficiently perform data prediction tasks.

The rest of the paper is organized as follows. In Section 2 we briefly review the Bayesian approach to learning Bayesian networks from data, the DP algorithms (Koivisto and Sood, 2004; Koivisto, 2006) and Order MCMC algorithm (Friedman and Koller, 2003). In Section 3 we present our order sampling algorithm, DDS algorithm and IW-DDS algorithm. We empirically demonstrate the advantages of our algorithms in Section 4. Section 5 concludes the paper.

2 BAYESIAN LEARNING OF BAYESIAN NETWORKS

A Bayesian network is a DAG G that encodes a joint probability distribution over a set $X = \{X_1, \ldots, X_n\}$ of random variables with each node of the DAG representing a variable in X. For convenience we typically work on the index set $V = \{1, \dots, n\}$ and represent a variable X_i by its index i. We use $X_{Pa_i} \subseteq X$ to represent the parent set of X_i in a DAG G and use $Pa_i \subseteq V$ to represent the corresponding index set. Thus, a DAG G can be represented as a vector $(Pa_1, \dots, Pa_n)$.

Assume that we are given a training data set $D = \{x^1, x^2, \dots, x^m\}$, where each x^i is a particular instantiation over the set of variables X. We only consider situations where the data are complete, that is, every variable in X is assigned a value. In the Bayesian approach to learning Bayesian networks from the training data D, we compute the posterior probability of a DAG G as

$$p(G|D) = \frac{p(D|G)p(G)}{p(D)} = \frac{p(D|G)p(G)}{\sum_G p(D|G)p(G)}. \qquad (1)$$

Assuming global and local parameter independence, and parameter modularity, $p(D|G)$ can be decomposed into a product of local marginal likelihood (often called local scores) as (Cooper and Herskovits, 1992; Heckerman et al., 1995)

$$p(D|G) = \prod_{i=1}^{n} p(x_i|x_{Pa_i} : D) := \prod_{i=1}^{n} score_i(Pa_i : D), \qquad (2)$$

where, with appropriate parameter priors, $score_i(Pa_i : D)$ has a closed form solution. In this paper we will assume that these local scores can be computed efficiently from data. The standard assumption for structure prior $p(G)$ is structure-modular prior (Friedman and Koller, 2003):

$$p(G) = \prod_{i=1}^{n} p_i(Pa_i). \qquad (3)$$

where p_i is some nonnegative function over the subsets of $V - \{i\}$.

Combing Eq. (2) and Eq. (3), we have

$$p_{\not\prec}(G, D) = p(D|G)p(G) = \prod_{i=1}^{n} score_i(Pa_i : D)p_i(Pa_i). \qquad (4)$$

Note that the subscript $\not\prec$ is intentionally added to mean that the corresponding probability is the one obtained without order-modular prior assumption. This is different from the probability computed with order-modular prior assumption, which will be marked by the subscript $\prec$ for the distinction.

We can compute the posterior probability of any hypothesis of interest by averaging over all the possible DAGs. For example, we are often interested in computing the posteriors of structural features. Let f be a structural feature represented by an indicator function such that $f(G)$ is 1 if the feature is present in G and 0 otherwise. By the full Bayesian model averaging, we have the posterior of f as

$$p(f|D) = \sum_G f(G)p(G|D), \qquad (5)$$

Note that $p_{\not\prec}(f|D)$ will be obtained if $p(G|D)$ in Eq. (5) is $p_{\not\prec}(G|D)$; $p_{\prec}(f|D)$ will be obtained if $p(G|D)$ in Eq. (5) is $p_{\prec}(G|D)$. This difference is the key to understanding the bias issue which will be described in details later.

Since summing over all the possible DAGs is generally infeasible, one approach to computing the posterior of f is to draw DAG samples $\{G_1, \dots, G_T\}$ from the posterior $p_{\not\prec}(G|D)$ or $p_{\prec}(G|D)$, which can then be used to estimate the posterior $p_{\not\prec}(f|D)$ or $p_{\prec}(f|D)$ as

$$\hat{p}(f|D) = \frac{1}{T}\sum_{i=1}^{T} f(G_i). \qquad (6)$$

2.1 THE DP ALGORITHMS

The DP algorithms (Koivisto and Sood, 2004; Koivisto, 2006) work in the order space rather than the DAG space. We define an *order* $\prec$ of variables as a total order (linear order) on V represented as a vector $(U_1, \dots, U_n)$ where U_i is the set of predecessors of i in the order $\prec$. To be more clear we may use $U_i^{\prec}$. We say that a DAG $G = (Pa_1, \dots, Pa_n)$ is consistent with an order $(U_1, \dots, U_n)$, denoted by $G \subseteq \prec$, if $Pa_i \subseteq U_i$ for each i. If S is a subset of V, we let $\mathcal{L}(S)$ denote the set of linear orders on S. In the following we will largely follow the notation from Koivisto (2006).

The algorithms working in the order space assume *order-modular* prior defined as follows: if G is consistent with $\prec$, then

$$p(\prec, G) = \prod_{i=1}^{n} q_i(U_i)\rho_i(Pa_i). \qquad (7)$$

where each q_i and ρ_i is some function from the subsets of $V - \{i\}$ to the nonnegative real numbers. (If G is not consistent with $\prec$, then $p(\prec, G) = 0$.)

A *modular feature* is defined as:

$$f(G) = \prod_{i=1}^{n} f_i(Pa_i), \qquad (8)$$

where $f_i(Pa_i)$ is an indicator function returning a 0/1 value. For example, an edge $j \to i$ can be represented by setting $f_i(Pa_i) = 1$ if and only if $j \in Pa_i$ and setting $f_l(Pa_l) = 1$ for all $l \neq i$.

With the order-modular prior, we are interested in the posterior $p_{\prec}(f|D) = p_{\prec}(f, D)/p_{\prec}(D)$. $p_{\prec}(f|D)$ can be obtained if the joint probability $p_{\prec}(f, D)$ can be computed (since $p_{\prec}(D) = p_{\prec}(f \equiv 1, D)$ where $f \equiv 1$, meaning that f always equals 1, can be easily achieved by setting each $f_i(Pa_i)$ to be the constant 1). Koivisto and Sood (2004) show that

$$p(f, \prec, D) = \prod_{i=1}^{n} \alpha_i(U_i^{\prec}), \tag{9}$$

and

$$p_{\prec}(f, D) = \sum_{\prec} \prod_{i=1}^{n} \alpha_i(U_i^{\prec}), \tag{10}$$

where the function α_i is defined for each $i \in V$ and each $S \subseteq V - \{i\}$ as

$$\alpha_i(S) = q_i(S) \sum_{Pa_i \subseteq S} \beta_i(Pa_i), \tag{11}$$

in which the function β_i is defined for each $i \in V$ and each $Pa_i \subseteq V - \{i\}$ as

$$\beta_i(Pa_i) = f_i(Pa_i)\rho_i(Pa_i)score_i(Pa_i : D). \tag{12}$$

Now the summation over the order space is computed by defining the following function for each $S \subseteq V$:

$$L(S) = \sum_{\prec \in \mathcal{L}(S)} \prod_{i \in S} \alpha_i(U_i^{\prec}), \tag{13}$$

where $U_i^{\prec}$ is the set of variables in S ahead of i in the order $\prec \in \mathcal{L}(S)$. It can be shown that for every $S \subseteq V$ the corresponding $L(S)$ can be computed recursively using the DP technique according to the following equation (Koivisto and Sood, 2004; Koivisto, 2006):

$$L(S) = \sum_{i \in S} \alpha_i(S - \{i\}) L(S - \{i\}), \tag{14}$$

starting with $L(\emptyset) = 1$ and ending with $L(V)$. Combining Eq. (10) and Eq. (13), we have

$$p_{\prec}(f, D) = L(V). \tag{15}$$

The DP algorithm (Koivisto and Sood, 2004) can only compute the posteriors of modular features. In this paper, we will show how to use the results of DP algorithm (Koivisto and Sood, 2004) to efficiently draw DAG samples, which can then be used to compute the posteriors of arbitrary features.

2.2 ORDER MCMC

The idea of Order MCMC is to use Metropolis-Hastings algorithm to draw order samples $\{\prec_1, \ldots, \prec_{N_o}\}$ that have $p(\prec|D)$ as the invariant distribution, where N_o is the number of sampled orders. For this purpose we need be able to compute $p(\prec, D)$, which can be obtained from Eq. (9) by setting $f \equiv 1$. Let $\beta'_i(Pa_i)$ denote $\beta_i(Pa_i)$ resulted from setting each $f_i(Pa_i)$ to be the constant 1 . We define $\alpha'_i(S)$ and $L'(S)$ similarly. Then from Eq. (9) and (15) we have

$$p(\prec, D) = \prod_{i=1}^{n} \alpha'_i(U_i^{\prec}), \tag{16}$$

and

$$p_{\prec}(D) = L'(V). \tag{17}$$

Order MCMC can estimate the posterior of a modular feature as

$$\hat{p}_{\prec}(f|D) = \frac{1}{N_o} \sum_{i=1}^{N_o} p(f| \prec_i, D). \tag{18}$$

For example, from Proposition 3.1 in (Friedman and Koller, 2003) as well as the definition of β'_i and α'_i, the posterior of a particular choice of parent set $Pa_i \subseteq U_i^{\prec}$ for node i given an order is

$$p(Pa_i| \prec, D) = \frac{\beta'_i(Pa_i)}{\alpha'_i(U_i^{\prec})/q_i(U_i^{\prec})}. \tag{19}$$

In order to compute arbitrary non-modular features, we further draw DAG samples after drawing N_o order samples. Given an order, a DAG can be sampled by drawing parents for each node according to Eq. (19). Given DAG samples $\{G_1, \ldots, G_T\}$, we can then estimate any feature posterior $p_{\prec}(f|D)$ using $\hat{p}_{\prec}(f|D)$ shown in Eq. (6).

3 ORDER SAMPLING ALGORITHM AND DAG SAMPLING ALGORITHMS

3.1 ORDER SAMPLING ALGORITHM

In this subsection, we show that using the results including $\alpha'_i(S)$ (for each $i \in V$ and each $S \subseteq V - \{i\}$) and $L'(S)$ (for each $S \subseteq V$) of the DP algorithm (Koivisto and Sood, 2004), we can draw order samples efficiently by drawing each element in the order one by one. Let an order $\prec$ be represented as $(\sigma_1, \ldots, \sigma_n)$ where σ_i is the ith element in the order.

Proposition 1 *The conditional probability that the kth ($1 \leq k \leq n$) element in the order is σ_k given that the $n - k$ elements after it along the order are $\sigma_{k+1}, \ldots, \sigma_n$ respectively is as follows:*

$$p(\sigma_k|\sigma_{k+1}, \ldots, \sigma_n, D) = \frac{L'(U_{\sigma_k}^{\prec})\alpha'_{\sigma_k}(U_{\sigma_k}^{\prec})}{L'(U_{\sigma_{k+1}}^{\prec})}, \tag{20}$$

where $\sigma_k \in V - \{\sigma_{k+1}, \ldots, \sigma_n\}$, and $U^{\prec}_{\sigma_i} = V - \{\sigma_i, \sigma_{i+1}, \ldots, \sigma_n\}$.
Specifically for $k = n$, we essentially have

$$p(\sigma_n = i|D) = \frac{L'(V - \{i\})\alpha'_i(V - \{i\})}{L'(V)}, \tag{21}$$

where $i \in V$.

Note that all the proofs in this paper are omitted due to the space constraint. [1]

Note that it is clear that for each $k \in \{1, \ldots, n\}$, $\sum_{i \in U^{\prec}_{\sigma_{k+1}}} p(\sigma_k = i|\sigma_{k+1}, \ldots, \sigma_n, D) = 1$ because of Eq.(14) and $U^{\prec}_{\sigma_k} = U^{\prec}_{\sigma_{k+1}} - \{\sigma_k\}$. Thus, $p(\sigma_k|\sigma_{k+1}, \ldots, \sigma_n, D)$ is a probability mass function (pmf) with k possible σ_k values from $U^{\prec}_{\sigma_{k+1}}$.

Based on Proposition 1, we propose the following order sampling algorithm to sample an order $\prec$:

- Sample σ_n, the last element of the order $\prec$, according to Eq. (21).
- For each k from $n - 1$ down to 1: given the sampled $(\sigma_{k+1}, \ldots, \sigma_n)$, sample σ_k, the kth element of the order $\prec$, according to Eq. (20).

The following proposition guarantees the correctness of our order sampling algorithm.

Proposition 2 *An order $\prec$ sampled according to our order sampling algorithm has its pmf equal to the exact posterior $p(\prec|D)$ under the order-modular prior, because*

$$\prod_{k=1}^{n} p(\sigma_k|\sigma_{k+1}, \ldots, \sigma_n, D) = p(\prec|D). \tag{22}$$

3.2 DDS ALGORITHM

After drawing an order sample, then we can easily sample a DAG by drawing parents for each node according to Eq. (19) as described in (Friedman and Koller, 2003) (by assuming fixed node-indegree). This naturally leads to our algorithm, termed Direct DAG Sampling (DDS), as follows:

- Step 1: Run DP algorithm (Koivisto and Sood, 2004) (i.e., the first three steps of DP algorithm (Koivisto, 2006)) with each $f_i(Pa_i)$ set to be the constant 1.
- Step 2 (Order Sampling Step): Sample N_o orders such that each order $\prec$ is independently sampled according to our order sampling algorithm.
- Step 3 (DAG Sampling Step): For each sampled order $\prec$, one DAG is independently sampled by drawing a parent set for each node of the DAG according to Eq. (19).

The following theorem guarantees the correctness of our DDS algorithm.

Theorem 1 *The N_o DAGs sampled according to DDS algorithm are independent and identically distributed (iid) with the pmf equal to the exact posterior $p_{\prec}(G|D)$ under the order-modular prior.*

Given DAG samples, $\hat{p}_{\prec}(f|D)$, the estimator for the exact posterior of any arbitrary feature f, can be constructed by Eq. (6).

The time complexity of DDS algorithm is as follows. Step 1 takes $O(n^{k+1}C(m) + kn2^n)$ time (Koivisto and Sood, 2004), where n is the number of nodes, k is the assumed constant maximum in-degree, and $C(m)$ is the cost of computing a single local marginal likelihood $score_i(Pa_i : D)$ for m data instances. In Step 2, each order sampling takes $O(n^2)$ time. In Step 3, each DAG sampling takes $O(n^{k+1})$ time. Thus, the overall time complexity of our DDS algorithm is $O(n^{k+1}C(m) + kn2^n + n^2N_o + n^{k+1}N_o)$. Since typically we assume $k \geq 1$, the order sampling process (Step 2) does not affect the overall time complexity of DDS algorithm because of its efficiency. (We have also used some strategy for Step 3 which can often greatly reduce its real running time when m is not small. The details are omitted due to the space constraint.) The space complexity of our DDS algorithm is $O(n2^n)$, the same as the one of DP algorithm (Koivisto and Sood, 2004).

The estimator $\hat{p}_{\prec}(f|D)$ based on our DDS algorithm has several desirable properties due to Theorem 1.

Corollary 1 *For any structural feature f, with respect to the exact posterior $p_{\prec}(f|D)$, the estimator $\hat{p}_{\prec}(f|D)$ based on the N_o DAG samples from DDS algorithm using Eq. (6) has the following properties:*

(i) $\hat{p}_{\prec}(f|D)$ is an unbiased estimator for $p_{\prec}(f|D)$.

(ii) $\hat{p}_{\prec}(f|D)$ converges almost surely to $p_{\prec}(f|D)$.

(iii) $\hat{p}_{\prec}(f|D)$ is a consistent estimator for $p_{\prec}(f|D)$, i.e., $\hat{p}_{\prec}(f|D)$ converges in probability to $p_{\prec}(f|D)$.

(iv) For any $\epsilon > 0$, any $0 < \delta < 1$, if $N_o \geq \ln(2/\delta)/(2\epsilon^2)$, then $p(|\hat{p}_{\prec}(f|D) - p_{\prec}(f|D)| < \epsilon) \geq 1 - \delta$.

In particular, Corollary 1 (iv), which is essentially from Hoeffding bound (Hoeffding, 1963; Koller and Friedman, 2009), states that in order to ensure that the probability that the error of the estimator $\hat{p}_{\prec}(f|D)$ from DDS algorithm is bounded by ϵ is at least $1 - \delta$, we just need to require the sample size $N_o \geq \ln(2/\delta)/(2\epsilon^2)$. This property, which the

[1] All the material that is omitted in this paper due to the space constraint has been provided in the supplementary material by the link `http://www.cs.iastate.edu/~rhe/BySampling/`.

MCMC algorithms do not have, can be used to obtain quality guarantee for the estimator from our DDS algorithm.

3.3 IW-DDS ALGORITHM

In this section we present our DAG sampling algorithm under the general structure-modular prior (Eq. (3)) by effectively correcting the bias due to the use of order-modular prior.

As mentioned in Section 1, $p_{\prec}(f|D)$ has the bias due to the assumption of order-modular prior. This is essentially because $p_{\prec}(G|D)$, which equals $\sum_{\prec s.t. G \subseteq \prec} p(G, \prec |D)$, does not equal $p_{\not\prec}(G|D)$ which is based on the standard structure-modular prior assumption instead of order-modular prior assumption. In fact, with the common setting that $q_i(U_i)$ always equals 1 ($q_i(U_i) \equiv 1$), if $\rho_i(Pa_i)$ in Eq. (7) is set to be always equal to $p_i(Pa_i)$ in Eq. (3) ($\rho_i(Pa_i) \equiv p_i(Pa_i)$), the following relation holds (Ellis and Wong, 2008):

$$p_{\prec}(G|D) \propto |\prec_G| \cdot p_{\not\prec}(G|D) \tag{23}$$

where $|\prec_G|$ is the number of orders that G is consistent with and $|\prec_G|$ is #P hard to compute (Brightwell and Winkler, 1991).

Noticing this problem, Ellis and Wong (2008) propose to correct this bias for Order MCMC method as follows: first run Order MCMC to draw order samples; then for each unique order $\prec$ out of the sampled orders, keep drawing DAGs consistent with $\prec$ until the sum of joint probabilities for the unique sampled DAGs, $\sum_i p(G_i, \prec, D)$, is no less than a pre-specified large proportion (such as 95%) of $p(\prec, D) = \sum_{G \subseteq \prec} p(G, \prec, D)$; finally the resulting union of all the DAG samples is treated as an importance-weighted sample for the structural discovery.

By refining the idea of Ellis and Wong (2008), here we propose our bias-corrected algorithm, termed IW-DDS (Importance-weighted DDS), as follows:

- Step 1 (DDS Step): Run DDS algorithm with the setting that $q_i(U_i) \equiv 1$ and $\rho_i(Pa_i) \equiv p_i(Pa_i)$.
- Step 2 (Bias Correction Step): Make the union set $\mathcal{G}$ of all the sampled DAGs by eliminating the duplicate DAGs.

Given $\mathcal{G}$, $\hat{p}_{\not\prec}(f|D)$, the estimator for the exact posterior of any feature f, can then be constructed as

$$\hat{p}_{\not\prec}(f|D) = \sum_{G \in \mathcal{G}} f(G) \hat{p}_{\not\prec}(G|D), \tag{24}$$

where

$$\hat{p}_{\not\prec}(G|D) = \frac{p_{\not\prec}(G, D)}{\sum_{G \in \mathcal{G}} p_{\not\prec}(G, D)}, \tag{25}$$

and $p_{\not\prec}(G, D)$ is given in Eq. (4).

Note that the time and space complexity of our IW-DDS algorithm are the same as the ones of our DDS algorithm.

While Ellis and Wong (2008) show the effectiveness of their methods in correcting the bias merely by the experiments, we first prove good properties of $\hat{p}_{\not\prec}(f|D)$ based on our IW-DDS as follows.

Theorem 2 *For any structural feature f, with respect to the exact posterior $p_{\not\prec}(f|D)$, the estimator $\hat{p}_{\not\prec}(f|D)$ based on the DAG samples from IW-DDS algorithm using Eq. (24) has the following two properties:*

(i) $\hat{p}_{\not\prec}(f|D)$ is a consistent estimator for $p_{\not\prec}(f|D)$, i.e., $\hat{p}_{\not\prec}(f|D)$ converges in probability to $p_{\not\prec}(f|D)$.

(ii) $\hat{p}_{\not\prec}(f|D)$ is an asymptotically unbiased estimator for $p_{\not\prec}(f|D)$.

The proof for (i) utilizes Theorem 5.5.13 and Slutsky's Theorem (Theorem 5.5.17) in (Casella and Berger, 2002). Proof for (ii) mainly depends on Taylor's Theorem (with Lagrange form of the remainder). The details of the proofs are omitted due to the space constraint.

The competing state-of-the-art algorithms also applicable in BNs of moderate size are Hybrid MCMC (Eaton and Murphy, 2007) whose first phase runs DP algorithm (Koivisto, 2006) with time complexity $O(n^{k+1}C(m) + kn2^n)$ and space complexity $O(n2^n)$; and K-best (Tian et al., 2010) with time complexity $O(n^{k+1}C(m) + K^2 n 2^{n-1})$ and space complexity $O(Kn2^n)$. When $n > 17$, K can only take some moderate value (such as no more than 100) in order to make K-best method feasible in current desktop computers.

4 EXPERIMENTAL RESULTS

We have implemented our algorithm in C++ language and we run several experiments to demonstrate its capabilities. The tested data sets include one real data set "Letter" from the UCI Machine Learning Repository (Asuncion and Newman, 2007), one synthetic data set "Syn15" generated from a gold-standard 15-node Bayesian network built by us, and one synthetic data set "Child" used in (Tsamardinos et al., 2006). All the data sets contain discrete variables (or are discretized) and have no missing values. All the experiments in this section were run under Linux on one ordinary desktop PC with a 3.0GHz Intel Pentium processor and 2.0GB of memory if no extra specification is provided. In addition, the maximum in-degree k is assumed to be 5 for all the experiments.

4.1 EXPERIMENTAL RESULTS FOR DDS

In this subsection, we compare our DDS algorithm with Partial Order MCMC method (Niinimaki et al., 2011), the state-of-the-art learning method under the order-modular prior.

Partial Order MCMC (PO-MCMC) method is implemented in BEANDisco[2], a C++ language tool provided by Niinimaki et al. (2011). The current version of BEANDisco can only estimate the posterior of edge feature, but as Niinimaki et al. (2011) have stated, PO-MCMC readily enables estimating the posterior of any structural feature by further sampling DAGs consistent with an order.

Since n (the size of the problems) is moderate, we are able to use REBEL, a C++ language implementation of the DP algorithm in (Koivisto, 2006), to get the exact posterior of every edge under the assumption of order-modular prior. Thus we can use the criterion of the sum of the absolute differences (SAD) (Eaton and Murphy, 2007) to measure the feature learning performance, where SAD is $\sum_{ij} |p_{\prec}(i \to j|D) - \hat{p}_{\prec}(i \to j|D)|$, $p_{\prec}(i \to j|D)$ is the exact posterior of edge $i \to j$ and $\hat{p}_{\prec}(i \to j|D)$ is the corresponding estimator. The smaller SAD will indicate the better performance in structure discovery.

For the fair comparison, for our algorithms, we used the K2 score (Heckerman et al., 1995) and we set $q_i(U_i) = 1$ and $\rho_i(Pa_i) = 1/\binom{n-1}{|Pa_i|}$ for each i, U_i, Pa_i, where $|Pa_i|$ is the size of the set Pa_i, since such a setting is used in both BEANDisco and REBEL.

For the setting of PO-MCMC, we set bucket size b to be 10, which is based on the suggestion for the optimal setting from Niinimaki et al. (2011). We run the first $10,000$ iterations for "burn-in", and then took 200 partial order samples at intervals of 50 iterations, thus, $20,000$ iterations in total. (The time cost of each iteration in PO-MCMC is $O(n^{k+1} + n^2 2^b n/b)$.) In PO-MCMC, for each sampled partial order P_i, $p(f|D, P_i)$ is obtained by $p(D, f, P_i)/p(D, P_i) = p(D, f, P_i)/p(D, f \equiv 1, P_i)$, where $p(D, f, P_i) = \sum_{\prec \supseteq P_i} \sum_{G \subseteq \prec} f(G)p(G, \prec)p(D|G)$. (Notation $\sum_{\prec \supseteq P_i}$ means that each total order $\prec$ that is a linear extension of the sampled partial order P_i will be included to obtain $p(D, f, P_i)$.) Finally, the estimated posterior of each edge is computed using $\hat{p}_{\prec}(f|D) = (1/T) \sum_{i=1}^{T} p(f|D, P_i)$.

For DDS algorithm, we just set $N_o = 20,000$, that is, $20,000$ (total) orders were sampled.

Table 1 shows experimental results in terms of both SAD and total running time T_t for each data case with n variables and m instances. For each of two methods, we performed 15 independent runs for each data case. The sample mean and the sample standard deviation of SAD of each method, denoted by $\hat{\mu}$(SAD) and $\hat{\sigma}$(SAD) respectively, are listed along each column in Table 1. Correspondingly, the sample mean of the total running time T_t of each method, denoted by $\hat{\mu}(T_t)$, is also shown in Table 1. In addition, the sample mean of the running time of DAG sampling step of DDS, denoted by $\hat{\mu}(T_{DAG})$, is listed along the last column in Table 1.

Table 1 clearly illustrates the performance advantage of our DDS method over PO-MCMC method. The overall time costs of our DDS based on $20,000$ DAG samples are much smaller than the corresponding ones of PO-MCMC method based on $20,000$ MCMC iterations in the partial order space. Using much smaller time, $\hat{\mu}$(SAD) and $\hat{\sigma}$(SAD) using our DDS method are much smaller than the ones using PO-MCMC method for 16 out of all the 18 data cases. The two exceptional cases are Syn15 with $m = 2,000$ and Syn15 with $m = 5,000$. Furthermore, since both $\hat{\mu}$(SAD) and $\hat{\sigma}$(SAD) are given, by the two-sample t test (Casella and Berger, 2002) with unequal sample variance, we can conclude with strong evidence (with level 0.001) that the real mean of SAD using our DDS method is smaller than the real mean of SAD using PO-MCMC method for each of these 16 data cases. For each of these two exceptions, by the same t test we can conclude (with p-value > 0.2) the null hypothesis that there is no significant difference in the real means of SAD.

4.2 EXPERIMENTAL RESULTS FOR IW-DDS

In this subsection, we compare our IW-DDS algorithm with Hybrid MCMC (i.e., DP+MCMC) method (Eaton and Murphy, 2007) and K-best method (Tian et al., 2010), two state-of-the-art methods that can estimate the posteriors of any features without the order-modular prior assumption. The implementation of Hybrid MCMC method (called BDAGL) and the implementation of K-best method are both made available online by their corresponding authors.

Again, since n is moderate, we are able to use POSTER, a C++ language implementation of the DP algorithm (Tian and He, 2009) to get the exact posterior of each edge under the assumption of structure-modular prior instead of order-modular prior. Thus we can also use SAD criterion ($\sum_{ij} |p_{\not\prec}(i \to j|D) - \hat{p}_{\not\prec}(i \to j|D)|$) to measure the performance of these three methods in the structural learning.

For the fair comparison, for our algorithm we used the BDeu score (Heckerman et al., 1995) with equivalent sample size 1 and set $p_i(Pa_i)(\equiv \rho_i(Pa_i)) \equiv 1$, since these settings are also used in POSTER and the implementation of K-best method.

As for DP+MCMC methods, we note that the most part of its implementation (in BDAGL tool) is written in Matlab, which is different from the tool of K-best method

[2]BEANDisco is available at http://www.cs.helsinki.fi/u/tzniinim/BEANDisco/.

Table 1: The Comparison of PO-MCMC & DDS in Terms of SAD and Time (Time Is in Seconds)

Name	n	m	PO-MCMC		DDS		PO-MCMC	DDS	DDS
			$\hat{\mu}$(SAD)	$\hat{\sigma}$(SAD)	$\hat{\mu}$(SAD)	$\hat{\sigma}$(SAD)	$\hat{\mu}(T_t)$	$\hat{\mu}(T_t)$	$\hat{\mu}(T_{DAG})$
Syn15	15	100	0.9024	0.2258	0.2622	0.0190	677.29	6.94	3.09
		200	0.6449	0.1569	0.2228	0.0172	677.47	8.74	3.78
		500	0.3424	0.1214	0.1116	0.0126	686.51	8.24	0.53
		1,000	0.1558	0.0496	0.0724	0.0118	716.31	12.48	0.24
		2,000	0.0465	0.0209	0.0473	0.0071	731.50	21.18	0.15
		5,000	0.0217	0.0144	0.0247	0.0086	731.05	48.21	0.14
Letter	17	100	0.9530	0.1285	0.2948	0.0229	1,322.43	23.40	8.19
		200	0.3854	0.0825	0.1758	0.0142	1,315.01	20.44	1.78
		500	0.4369	0.1529	0.1326	0.0107	1,338.33	28.46	1.35
		1,000	0.3007	0.1254	0.0828	0.0171	1,343.88	39.47	0.43
		2,000	1.3740	0.9177	0.1386	0.0288	1,358.29	62.63	0.42
		5,000	0.0669	0.0139	0.0292	0.0088	1,610.37	130.26	0.28
Child	20	100	0.4997	0.1153	0.1772	0.0146	3,710.49	197.62	96.64
		200	0.1896	0.0528	0.0982	0.0101	3,717.10	183.31	71.19
		500	0.2385	0.0702	0.0816	0.0123	3,757.76	207.17	70.62
		1,000	0.1079	0.0525	0.0406	0.0080	3,799.47	199.13	25.15
		2,000	0.0864	0.0521	0.0275	0.0083	4,018.03	269.05	26.34
		5,000	0.0938	0.0539	0.0246	0.0066	4,531.20	483.74	31.51

and our tool which are written in C++. In order to make the relatively fair comparison in terms of running time, we used REBEL tool, a C++ implementation of DP algorithm (Koivisto, 2006), to perform the computation in the DP phase (but we changed its scoring criterion into the BDeu score with equivalent sample size 1 and set $q_i(U_i) \equiv 1$ and $\rho_i(Pa_i) \equiv 1$). To perform the computation in the MCMC phase, we had to use Matlab implementation and we ran it under Windows 7 on an ordinary laptop with 2.40 Intel Core i5 CPU and 4.0 GB memory. The MCMC used the pure global proposal (with local proposal choice β = 0) since such a setting was reported in (Eaton and Murphy, 2007) to have the best performance for edge discovery when up to about $190,000$ MCMC iterations were performed in their experimental results. We ran totally 190,000 MCMC iterations each time and discarded the first 100,000 iterations as burn-in period. Then we set the thinning parameter as 3 to get the final 30,000 DAG samples. As a result, the time statistics of the DP phase (the number before + sign) but not the MCMC phase (the number after + sign) can be directly compared with the ones of the other two methods. For each data case, we performed 20 independent MCMC runs based on the DP outcome from REBEL to get the results.

For our method, we set $N_o = 30,000$. We performed 20 independent runs for each data case to get the results. With reference to K-best method, note that SAD of K-best method is fixed since there is no randomness in the computed results. So we only run it once to get the result. We set K to be 100 for Syn15 and Letter, i.e., we got the 100 best DAGs from each of six cases of Syn15 and each of six cases of Letter. We set K to be only 20 for Child because our experiments showed that for Child K-best program run out of memory with the setting of $K > 20$.

Table 2 shows experimental results in terms of both SAD and total running time T_t for each data case. The table clearly demonstrates that the advantage of our method over the other two methods. By using much less computation time, $\hat{\mu}$(SAD) using our method is less than the corresponding one using DP+MCMC method for 17 out of the 18 data cases. The only exceptional case is Syn15 with $m = 200$. Furthermore, based on the two-sample t test with unequal variance, we can conclude with level 0.05 that the real mean of SAD using our method is less than the corresponding one using DP+MCMC method for 16 out of the 18 cases. The two exceptional cases are Syn15 with $m = 100$ and Syn15 with $m = 200$. Similarly, using much less computation time, $\hat{\mu}$(SAD) of our method is less than the one using K-best method for 17 out of 18 cases. The only exception is Syn15 with m = 5,000. Furthermore, based on the one-sample t test (Casella and Berger, 2002), we can conclude with level 1×10^{-4} that the real mean of SAD using our method is less than the corresponding one using K-best method for each of these 17 cases.

4.3 LEARNING PERFORMANCE OF NON-MODULAR FEATURES

In Section 4.1 and 4.2 we did not provide experimental results on the learning performance of non-modular features. We did not do so in Section 4.2 because there is no known method to compute the true/exact posterior probability of any non-modular feature $p_{\not\prec}(f|D)$ except by the brute force enumeration over all the (super-exponential number of) DAGs so that the quality of the corresponding $\hat{p}_{\not\prec}(f|D)$ learned from any method cannot be precisely measured. We did not do so in Section 4.1 because the

Table 2: The Comparison of DP+MCMC, K-best & IW-DDS in Terms of SAD and Time (Time Is in Seconds)

Name	n	m	DP+MCMC		K-best	IW-DDS		DP+MCMC	K-best	IW-DDS	IW-DDS
			$\hat{\mu}$(SAD)	$\hat{\sigma}$(SAD)	SAD	$\hat{\mu}$(SAD)	$\hat{\sigma}$(SAD)	$\hat{\mu}(T_t)$	T_t	$\hat{\mu}(T_t)$	$\hat{\mu}(T_{DAG})$
Syn15	15	100	12.8705	7.6384	11.8685	10.1216	0.1650	4.96 + 1,284.00	901.83	9.21	4.81
		200	4.5090	2.5875	7.5232	4.9225	0.0605	5.98 + 1,286.20	913.09	9.97	4.53
		500	5.5466	1.9175	4.4379	4.2333	0.1854	8.92 + 1,336.80	911.28	9.46	1.14
		1,000	0.3974	0.3299	0.0848	0.0497	0.0061	13.70 + 1,364.60	922.62	12.92	0.39
		2,000	1.8263	1.7095	0.3701	0.0999	0.0187	22.71 + 1,372.10	918.88	21.96	0.36
		5,000	0.0304	0.0094	8.89E-4	0.0021	0.0002	48.72 + 1,356.70	944.82	52.01	0.34
Letter	17	100	27.1507	4.0940	24.4313	15.9160	0.3181	24.37 + 1,572.60	7,650.88	19.28	2.89
		200	15.1587	3.5615	9.4512	6.7787	0.1032	28.20 + 1,576.80	7,978.11	24.14	4.25
		500	3.4637	4.6789	1.7237	0.6364	0.0101	36.79 + 1,598.90	8,269.46	28.92	1.06
		1,000	0.1761	0.0166	0.0837	0.0739	0.0028	48.60 + 1,575.60	8,392.43	40.31	0.61
		2,000	3.5085	3.1132	2.0976	0.1328	0.0193	72.09 + 1,591.00	7,631.15	64.47	0.83
		5,000	0.1182	0.0442	0.0160	0.0073	0.0004	136.88 + 1,636.40	8,146.71	131.12	0.71
Child	20	100	11.8987	3.1086	11.6189	7.0372	0.1084	200.43 + 1,785.10	15,097.72	237.13	134.66
		200	4.7066	4.3749	5.0729	2.8646	0.0242	207.91 + 1,760.80	14,234.72	239.38	126.25
		500	2.4716	1.3489	1.5304	0.5315	0.0274	234.10 + 1,818.70	14,028.80	218.00	79.28
		1,000	2.6061	2.2909	0.7066	0.1507	0.0172	274.21 + 1,817.20	15,516.68	231.26	53.90
		2,000	1.4286	1.2290	1.5279	0.0758	0.0537	346.99 + 1,841.40	16,121.77	295.95	47.21
		5,000	1.2533	1.7313	0.8783	0.0153	0.0015	555.79 + 1,846.40	15,384.47	508.86	53.11

current PO-MCMC tool (BEANDisco) only supports to estimate the posterior of edge feature so that the comparison of our method and PO-MCMC can only be limited in terms of edge feature. (Thus, we did not make the comparison in terms of path feature (which is one particular non-modular feature), though there exists the DP algorithm (Parviainen and Koivisto, 2011) that can compute the exact posterior of a path feature $p_{\prec}(f|D)$.) Our idea is that by showing our algorithms have significantly better performance in computing one feature (directed edge feature), which should be due to the better quality of our DAG samples with respect to the corresponding $p_{\not\prec}(G|D)$ or $p_{\prec}(G|D)$, we expect they will also be superior in computing other features using the same set of DAG samples.

To verify our expectation, we performed the experiments on the real data set "Iris" (with $n = 5$) from the UCI Machine Learning Repository (Asuncion and Newman, 2007) and the well-studied data set "Coronary" (Coronary Heart Disease) (with $n = 6$) (Edwards, 2000). Since n is small, by enumerating all the DAGs we were able to compute $p_{\not\prec}(f_i|D)$, the true posterior probability for several interesting non-modular features: f_1, a directed path feature; f_2, a limited-length directed path feature that has its path length no more than 2; f_3, a combined directed path feature that represents a directed path from node i via node j to node k. Then we compared SAD performance of (directed) edge feature from DP+MCMC, K-best and IW-DDS with the corresponding SAD performance of feature f_i ($i \in \{1, 2, 3\}$) from these three methods. The experimental results on both data sets show that if SAD of IW-DDS is significantly smaller than SAD of the competing method (DP+MCMC or K-best) in terms of edge feature, then SAD of IW-DDS will also be smaller (usually significantly smaller) than SAD of the competing method in terms of each feature f_i. Thus, our expectation is supported by the experiments. The detailed experimental results are omitted due to the space constraint.

4.4 PERFORMANCE GUARANTEE FOR DDS ALGORITHM

To testify the quality guarantee for the estimator based on DDS algorithm (Corollary 1 (iv)), we performed the experiments based on two data cases (Syn15 with $m = 100$ and Letter with $m = 100$) which have relatively large $\hat{\mu}$(SAD) shown in Table 1. Based on the hypothesis testing, we can conclude with very strong evidence that performance guarantee for our estimator holds on both data cases. The details are omitted due to the space constraint.

5 CONCLUSION

We develop new algorithms for efficiently sampling DAGs in problems of moderate size. The sampled DAGs can then be used to build the estimators for the posteriors of any features. The corresponding estimators have good properties; for example, performance guarantee can be provided for our estimator when assuming the order-modular prior. Our algorithms serve as the complements to the exact algorithms (Koivisto and Sood, 2004; Parviainen and Koivisto, 2011; Tian and He, 2009) to estimate the posteriors of arbitrary non-modular features. We empirically show that our algorithms perform considerably better than previous state-of-the-art methods with or without assuming order-modular prior.

References

Asuncion, A. and Newman, D. J. (2007). UCI machine learning repository.

Athreya, K. B. and Lahiri, S. N. (2006). *Measure Theory and Probability Theory*. Springer.

Brightwell, G. and Winkler, P. (1991). Counting linear extensions is #P-complete. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 175–181.

Casella, G. and Berger, R. L. (2002). *Statistical Inference*. Duxbury.

Cooper, G. F. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347.

Dash, D. and Cooper, G. F. (2004). Model averaging for prediction with discrete Bayesian networks. *Journal of Machine Learning Research*, 5:1177–1203.

Eaton, D. and Murphy, K. (2007). Bayesian structure learning using dynamic programming and MCMC. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 101–108.

Edwards, D. (2000). *Introduction to Graphical Modelling*. Springer, 2nd edition.

Ellis, B. and Wong, W. H. (2008). Learning causal Bayesian network structures from experimental data. *Journal of American Statistical Association*, 103:778–789.

Friedman, N., Goldszmidt, M., and Wyner, A. (1999). Data analysis with Bayesian networks: A bootstrap approach. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 196–205.

Friedman, N. and Koller, D. (2003). Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1-2):95–125.

Grzegorczyk, M. and Husmeier, D. (2008). Improving the structure MCMC sampler for Bayesian networks by introducing a new edge reversal move. *Machine Learning*, 71:265–305.

Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243.

Heckerman, D., Meek, C., and Cooper, G. (1999). A Bayesian approach to causal discovery. In Glymour, C. and Cooper, G., editors, *Computation, Causation, and Discovery*, pages 141–165.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):pp. 13–30.

Koivisto, M. (2006). Advances in exact Bayesian structure discovery in Bayesian networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 241–248.

Koivisto, M. and Sood, K. (2004). Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573.

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

Madigan, D. and York, J. (1995). Bayesian graphical models for discrete data. *International Statistical Review*, 63:215–232.

Meila, M. and Jaakkola, T. (2006). Tractable Bayesian learning of tree belief networks. *Statistics and Computing*, 16:77–92.

Niinimaki, T., Parviainen, P., and Koivisto, M. (2011). Partial order MCMC for structure discovery in Bayesian networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 557–564.

Parviainen, P. and Koivisto, M. (2011). Ancestor relations in the presence of unobserved variables. In *Proceedings of the European conference on machine learning and knowledge discovery in databases (ECML PKDD)*, pages 581–596.

Silander, T. and Myllymaki, P. (2006). A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 445–452.

Tian, J. and He, R. (2009). Computing posterior probabilities of structural features in Bayesian networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 538–547.

Tian, J., He, R., and Ram, L. (2010). Bayesian model averaging using the k-best Bayesian network structures. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 589–597.

Tsamardinos, I., Brown, L., and Aliferis, C. (2006). The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65:31–78.