Journal of Machine Learning Research 17 (2016) 1-54 Submitted 12/14; Revised 11/15; Published 6/16

Structure Learning in Bayesian Networks of a Moderate Size by Efficient Sampling

Ru He HRHERU@GMAIL.COM
Department of Computer Science and Department of Statistics
Iowa State University
Ames, IA 50011, USA

Jin Tian JTIAN@IASTATE.EDU
Department of Computer Science
Iowa State University
Ames, IA 50011, USA

Huaiqing Wu ISUHWU@IASTATE.EDU
Department of Statistics
Iowa State University
Ames, IA 50011, USA

Editor: Max Chickering

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 of a moderate size (with up to about 25 variables) according to the exact structure posterior. The DAG samples can then be used to construct estimators for the posterior of any feature. We theoretically prove good properties of our estimators and empirically show that our estimators considerably outperform the estimators from the previous state-of-the-art methods.

Keywords: Bayesian model averaging, Bayesian networks, DAG sampling, dynamic programming, order sampling, structure learning

1. Introduction

Bayesian networks are graphical representations of multivariate joint probability distributions and have been widely used in various data-mining tasks for probabilistic inference and causal modeling (Pearl, 2000; Spirtes et al., 2001). The core of a Bayesian network (BN) representation is its Bayesian network structure. A Bayesian network structure is a DAG (directed acyclic graph) whose nodes represent the random variables $X_1, X_2, \cdots, X_n$ in the problem domain and whose edges correspond to the direct probabilistic dependencies. Semantically, a Bayesian network structure G encodes a set of conditional independence assumptions: for each variable (node) X_i, X_i is conditionally independent of its non-descendants given its parents. With the above semantics, a Bayesian network structure provides a compact representation for joint distributions and supports efficient algorithms for answering probabilistic queries. Furthermore, with its semantics, a Bayesian network structure can often provide a deep insight into the problem domain and open the door to the cause-and-effect analysis.

In the last two decades, there have been a large number of research articles focusing on the problem of learning Bayesian network structure(s) from the data. These articles deal with a common real situation where the underlying Bayesian network is typically unknown so that it has to be learned from the observed data. One motivation for the structure learning is to use the learned structure for inference or decision making. For example, we can use the learned model to predict or classify a new instance of data. Another structure-learning motivation, which is more closely related to the semantics of Bayesian network structures, is for discovering the structure of the problem domain. For example, in the context of biological expression data, the discovery of the causal and dependence relation among different genes is often of primary interests. With the semantics of a Bayesian network structure G, the existence of an edge from node X to node Y in G can be interpreted as the fact that variable X directly influences variable Y; the existence of a directed path from node X to node Y can be interpreted as the fact that X eventually influences Y. Furthermore, under certain assumptions (Heckerman et al., 1999; Spirtes et al., 2001), the existence of a directed path from node X to node Y indicates that X causes Y. Thus, with the learned Bayesian network structure, we can answer interesting questions such as whether gene X controls gene Y which in turn controls gene Z by examining whether there is a directed path from node X via node Y to node Z in the learned structure. Just as mentioned by Friedman and Koller (2003), the extraction of these kinds of interesting structural features is often the primary goal in the discovery task.

There are several general approaches to learning BN structures. One approach is to treat learning BN structures as a model-selection problem. This approach defines a scoring criterion that measures how well a BN structure (DAG) fits the data and finds the DAG (or a set of equivalent DAGs) with the optimal score (Silander and Myllymaki, 2006; Jaakkola et al., 2010; Yuan et al., 2011; Malone et al., 2011a,b; Cussens, 2011; Yuan and Malone, 2012; Malone and Yuan, 2013; Cussens and Bartlett, 2013; Yuan and Malone, 2013). (In Bayesian approach, the score of a DAG G is simply the posterior $p(G|D)$ of G given data D.) When the data size is small as compared with the number of variables, however, the posterior $p(G|D)$ often gives significant support to a number of DAGs, 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 the posterior probability of any feature of interest is computed 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 between $2^{n(n-1)/2}$ and $n!2^{n(n-1)/2}$, 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 a moderate size with up to around 25 variables (mainly because of their space cost $O(n2^n)$). A big limitation of these algorithms is that they can only compute posteriors of modular features such as an edge but cannot 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) developed a DP algorithm that can compute the exact posterior probability of a path feature under a certain assumption. (The assumption, called the order-modular assumption, will be discussed in details soon.) This DP algorithm has (even higher) exponential time and space complexity and can only handle a Bayesian network with fewer than 20 variables (mainly because of its space cost $O(3^n)$). Because this DP algorithm can only deal with a path feature, all the other non-modular features (such as a combined path) which would interest various users still cannot be computed by any DP algorithm proposed so far. Note that generally the posterior $p(f|D)$ of a combined feature $f = (f_1, f_2, \ldots, f_J)$ cannot be obtained only from the posterior of each individual feature $p(f_j|D)$ ($j \in \{1, 2, \ldots, J\}$), because the independence among these features does not hold generally. Actually, by comparing $p(f_2|f_1, D)$ with $p(f_2|D)$, a user can know the effect of the feature f_1 upon the feature f_2; but to obtain $p(f_2|f_1, D)$ ($= p(f_1, f_2|D)/p(f_1|D)$), the user typically needs to obtain $p(f_1, f_2|D)$ first. 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 solution to computing the posterior of an arbitrary non-modular feature is drawing DAG samples $\{G_1, \ldots, G_T\}$ from the posterior $p(G|D)$, which can then be used to approximate the full Bayesian model averaging by estimating 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 the 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; Niinimaki and Koivisto, 2013). Madigan and York (1995) developed the *Structure MCMC* algorithm that uses the Metropolis-Hastings algorithm in the space of DAGs. Friedman and Koller (2003) developed the *Order MCMC* procedure that operates in the space of orders. The Order MCMC was shown to be able to considerably improve over the Structure MCMC the mixing and convergence of the Markov chain and to outperform the bootstrap approach of Friedman et al. (1999) as well. Eaton and Murphy (2007) developed the *Hybrid MCMC* method (that is, DP+MCMC method) that first runs the DP algorithm of Koivisto (2006) to develop a global proposal distribution and then runs the MCMC phase in the DAG space. Their experiments showed that the Hybrid MCMC converged faster than both the Structure MCMC and the Order MCMC, so that the Hybrid MCMC resulted in more accurate structure-learning performance. An improved MCMC algorithm (often denoted as *REV-MCMC*) traversing in the DAG space with the addition of a new edge-reversal move was developed by Grzegorczyk and Husmeier (2008) and was shown to be superior to the Structure MCMC and nearly as efficient as the Order MCMC in the mixing and convergence. Recently, Niinimaki et al. (2011) proposed the *Partial Order MCMC* method which operates in the space of partial orders. The Partial Order MCMC includes the Order MCMC as its special case (by setting the parameter bucket size b to be 1) and has been shown to be superior to the Order MCMC in terms of the mixing and the structure-learning performance when a more appropriate bucket size $b > 1$ is 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 the full Bayesian model averaging using the K*-best* Bayesian network structures was studied by Tian et al. (2010) and was shown to be competitive with the Hybrid MCMC.

Several of these state-of-the-art algorithms work in the order space, including the exact algorithms (Koivisto and Sood, 2004; Koivisto, 2006; Parviainen and Koivisto, 2011) and the approximate algorithms: the Order MCMC (Friedman and Koller, 2003) and the Partial Order MCMC (Niinimaki et al., 2011). They all assume a special form of the structure prior, termed as the *order-*

modular prior (Friedman and Koller, 2003; Koivisto and Sood, 2004), for computational convenience. (Please refer to the beginning of Section 2.1 for the definition of the order-modular prior.) Under the assumption of the order-modular prior, however, the corresponding prior $p(G)$ cannot represent some desirable priors such as a uniform prior over the DAG space [1]; the computed posterior probabilities are biased because a DAG that has a larger number of topological orders will be assigned a larger prior probability. Whether a computed posterior with the bias from the order-modular prior is inferior to its counterpart without such a bias depends on the application scenario and is beyond the scope of this paper. For the detailed discussion about this issue, please see the related papers (Friedman and Koller, 2003; Grzegorczyk and Husmeier, 2008; Parviainen and Koivisto, 2011) [2]. One method that helps the 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 uses the results of the DP algorithm of Koivisto and Sood (2004) to efficiently sample orders according to the *exact* order posterior under the assumption of the order-modular prior. Next, we develop a time-saving strategy for the process of sampling DAGs consistent with given orders. (Such a DAG-sampling process is based on sampling parents for each node as described by Friedman and Koller, 2003 by assuming a bounded node in-degree.) The resulting algorithm (called DDS) is the first algorithm that is able to sample DAGs according to the *exact* DAG posterior with the same order-modular prior assumption. We empirically show that our DDS algorithm is both considerably more accurate and considerably more efficient than the Order MCMC and the 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 the existing 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 address the limitation of the exact DP algorithms (Koivisto and Sood, 2004; Koivisto, 2006; Parviainen and Koivisto, 2011) (whose usage is restricted to modular features or path features) in order to estimate the posteriors of various *non-modular* features arbitrarily specified by users. Additionally our DDS algorithm can also be used to efficiently perform data prediction tasks in estimating $p(x|D)$ for a large number of data cases (while the exact DP algorithm has to be re-run for each data case x). Finally, we develop an algorithm (called IW-DDS) to correct the bias (due to the order-modular prior) in the DDS algorithm by extending the idea of Ellis and Wong (2008). We theoretically prove that the estimator based on our IW-DDS has several desirable properties; then we empirically show that our estimator is superior to the estimators based on the Hybrid MCMC method (Eaton and Murphy, 2007) and the K-best algorithm (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 mainly addresses the limitation of the exact DP algorithm of Tian and He (2009) (whose usage is restricted to modular features) in order to estimate the posteriors of arbitrary *non-modular* features and can additionally be used to efficiently perform data prediction tasks when an application situation prefers to avoid the bias from the order-modular prior.

1. A uniform prior over the DAG space can be represented in another form of the structure prior, termed as the *structure-modular* prior. The definition of the structure-modular prior will be given in Eq. (2) in Section 2.
2. Although a lot of discussions are about the comparison of these two kinds of priors (the order-modular prior and the structure-modular prior), there exist other kinds of priors such as a uniform prior over Markov equivalence classes. Our paper, however, will focus on learning structural features under either the order-modular prior or the structure-modular prior, because these are the two most commonly used priors in applications.

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 related DP algorithms (Koivisto and Sood, 2004; Koivisto, 2006), and the Order MCMC algorithm (Friedman and Koller, 2003). In Section 3 we present our order sampling algorithm, DDS algorithm, and IW-DDS algorithm, and prove good properties of the estimators based on our algorithms. We empirically demonstrate the advantages of our algorithms in Section 4 and conclude the paper in Section 5. Finally, Appendix A provides the proofs of all the conclusions including the propositions, theorems, and corollary referenced in the paper.

2. Bayesian Learning of Bayesian Network Structures

A Bayesian network structure is a DAG G that provides the skeleton for compactly encoding 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 $V = \{1, \ldots, n\}$ 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. (A pair (i, Pa_i) is often called a family.) Thus, a DAG G can be represented as a vector $(Pa_1, \ldots, Pa_n)$.

Assume that we are given a training data set $D = \{x^1, x^2, \ldots, x^m\}$, 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

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

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

$$p(D|G) = \prod_{i=1}^{n} p(X_i | X_{Pa_i} : D) := \prod_{i=1}^{n} score_i(Pa_i : D), \tag{1}$$

where, with appropriate parameter priors, $score_i(Pa_i : D)$ (the local score for a family (i, Pa_i)) 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 the structure prior $p(G)$ is the *structure-modular* prior (Friedman and Koller, 2003)

$$p(G) = \prod_{i=1}^{n} p_i(Pa_i), \tag{2}$$

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

Combining Eq. (1) and Eq. (2), we have

$$p_{\not\prec}(G, D) = p(D|G)p_{\not\prec}(G) = \prod_{i=1}^{n} score_i(Pa_i : D)p_i(Pa_i). \tag{3}$$

Note that the subscript $\not\prec$ is intentionally added by us to mean that each correspondingly marked probability is the one obtained under the structure-modular prior instead of the order-modular prior. This is different from the probability obtained under the order-modular prior, which will be marked by the subscript $\prec$ for the distinction.

We can compute the posterior probability of any feature 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

$$p(f|D) = \sum_{G} f(G)p(G|D). \tag{4}$$

Note that $p_{\not\prec}(f|D)$ will be obtained if $p(G|D)$ in Eq. (4) is $p_{\not\prec}(G|D)$; $p_{\prec}(f|D)$ will be obtained if $p(G|D)$ in Eq. (4) is $p_{\prec}(G|D)$. This difference is the key to understanding the bias issue which will be described in details later.

Because directly summing over all the possible DAGs by brute force is generally infeasible for any problem with $n > 6$ using a contemporary computer, one approach to computing the posterior of f is to draw DAG samples $\{G_1, \ldots, G_T\}$ 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

$$\hat{p}(f|D) = \frac{1}{T}\sum_{i=1}^{T} f(G_i). \tag{5}$$

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 (a linear order) on V represented as a vector $(U_1, \ldots, U_n)$, 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 $G = (Pa_1, \ldots, Pa_n)$ is consistent with an order $(U_1, \ldots, U_n)$, 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 the *order-modular* prior defined as follows: if G is consistent with $\prec$, then

$$p(\prec, G) = \prod_{i=1}^{n} q_i(U_i)\rho_i(Pa_i), \tag{6}$$

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

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

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

With the order-modular prior, we are interested in the posterior $p_{\prec}(f|D) = p_{\prec}(f, D)/p_{\prec}(D)$. Note that $p_{\prec}(f|D)$ can be obtained if the joint probability $p_{\prec}(f, D)$ can be computed, because $p_{\prec}(D) = p_{\prec}(f \equiv 1, D)$ 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) showed that

$$p(f, \prec, D) = \prod_{i=1}^{n} \alpha_i(U_i^{\prec}), \tag{7}$$

and

$$p_{\prec}(f, D) = \sum_{\prec} \prod_{i=1}^{n} \alpha_i(U_i^{\prec}), \tag{8}$$

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

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

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

$$\beta_i(Pa_i) = f_i(Pa_i)\rho_i(Pa_i)score_i(Pa_i : D).$$

Accordingly, the DP algorithm of Koivisto and Sood (2004) consists of the following three steps. The first step computes $\beta_i(Pa_i)$ for each $i \in V$ and each $Pa_i \subseteq V - \{i\}$. The time complexity of this step is $O(n^{k+1}C(m))$ under the assumption of the maximum in-degree k, where n is the number of variables, and $C(m)$ is the cost of computing a single local marginal likelihood $score_i(Pa_i : D)$ for m data instances. The second step computes $\alpha_i(S)$ for each $i \in V$ and each $S \subseteq V - \{i\}$. With the assumed maximum in-degree k, this step takes $O(kn2^n)$ time by using the truncated Möbius transform technique (Koivisto and Sood, 2004), which was extended from the standard fast Möbius transform algorithm (Kennes and Smets, 1991). The third step computes $p_{\prec}(f, D)$ by defining the following function (called forward contribution) for each $S \subseteq V$:

$$L(S) = \sum_{\prec \in \mathcal{L}(S)} \prod_{i \in S} \alpha_i(U_i^{\prec}), \tag{9}$$

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 $S \subseteq V$, $L(S)$ can be computed recursively using the DP technique according to the following equation (Koivisto and Sood, 2004; Koivisto, 2006):

$$L(S) = \sum_{i \in S} \alpha_i(S - \{i\})L(S - \{i\}), \tag{10}$$

starting with $L(\emptyset) = 1$ and ending with $L(V)$. From Eq. (8) and Eq. (9), we have

$$p_{\prec}(f, D) = L(V). \tag{11}$$

The third step takes $O(n2^n)$ time when $L(V)$ is computed using the above DP technique. In summary, the whole DP algorithm of Koivisto and Sood (2004) can compute the posterior of any modular feature (such as an edge feature) in $O(n^{k+1}C(m) + kn2^n)$ time and $O(n2^n)$ space, where its $O(n2^n)$ space cost limits its application to Bayesian networks with up to around 25 variables.

As the extended work of Koivisto and Sood (2004), Koivisto (2006) included the DP algorithm of Koivisto and Sood (2004) as its first three steps and appended two additional steps so that all the $n(n-1)$ edges can be computed in $O(n^{k+1}C(m) + kn2^n)$ time and $O(n2^n)$ space. The foundation of the two additional steps is the introduction of the following function (called backward contribution) for each $T \subseteq V$:

$$R(T) = \sum_{\prec' \in \mathcal{L}(T)} \prod_{i \in T} \alpha_i((V - T) \cup U_i^{\prec'}). \tag{12}$$

Essentially, $R(T)$ represents the contribution of the variables in T when they are the last $|T|$ elements in the unknown linear order on V. Like $L(S)$, $R(T)$ can also be computed recursively using some DP technique. Please refer to the paper of Koivisto (2006) for further details of the two additional steps.

Although the DP algorithms (Koivisto and Sood, 2004; Koivisto, 2006) make significant contributions to the structure learning of Bayesian networks, they have one fundamental limitation: they can only compute the posteriors of modular features. In the next section, we will show how to use the results of the DP algorithm of 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 the Order MCMC is to use the 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 to be able to compute $p(\prec, D)$, which can be obtained from Eq. (7) 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. Similarly, we define $\alpha_i'(S)$ and $L'(S)$ as the special cases of $\alpha_i(S)$ and $L(S)$ by setting each $f_i(Pa_i)$ to be the constant 1. Then from Eq. (7) and Eq. (11) we have

$$p(\prec, D) = \prod_{i=1}^{n} \alpha_i'(U_i^{\prec}), \tag{13}$$

and

$$p_{\prec}(D) = L'(V). \tag{14}$$

The Order MCMC can estimate the posterior of a modular feature as

$$\hat{p}_{\prec}(f|D) = \frac{1}{N_o} \sum_{i=1}^{N_o} p(f| \prec_i, D). \tag{15}$$

For example, from Propositions 3.1 and 3.2 stated by Friedman and Koller (2003) as well as the definitions 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

$$p((i, Pa_i)| \prec, D) = \frac{\beta_i'(Pa_i)}{\alpha_i'(U_i^{\prec})/q_i(U_i^{\prec})}, \tag{16}$$

and the posterior of the edge feature $j \to i$ given an order is

$$p(j \to i | \prec, D) = 1 - \frac{\alpha'_i(U_i^{\prec} - \{j\})/q_i(U_i^{\prec} - \{j\})}{\alpha'_i(U_i^{\prec})/q_i(U_i^{\prec})}. \tag{17}$$

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. (16). 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. (5).

3. Order Sampling Algorithm and DAG Sampling Algorithms

In this section we present our order sampling algorithm, DDS algorithm, and IW-DDS algorithm. We also prove good properties of the estimators based on our 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 $S \subseteq V$) computed from the DP algorithm of Koivisto and Sood (2004), we can draw an order sample 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:*

$$p(\sigma_k | \sigma_{k+1}, \ldots, \sigma_n, D) = \frac{L'(U_{\sigma_k}^{\prec})\alpha'_{\sigma_k}(U_{\sigma_k}^{\prec})}{L'(U_{\sigma_{k+1}}^{\prec})}, \tag{18}$$

where $\sigma_k \in V - \{\sigma_{k+1}, \ldots, \sigma_n\}$, and $U_{\sigma_i}^{\prec} = V - \{\sigma_i, \sigma_{i+1}, \ldots, \sigma_n\}$ so that $U_{\sigma_i}^{\prec}$ denotes the set of predecessors of σ_i in the order $\prec$.
Specifically for $k = n$, we essentially have

$$p(\sigma_n = i | D) = \frac{L'(V - \{i\})\alpha'_i(V - \{i\})}{L'(V)}, \tag{19}$$

where $i \in V$.

Note that all the proofs in this paper are provided in Appendix A.

It is clear that for each $k \in \{1, \ldots, n\}$, $\sum_{i \in U_{\sigma_{k+1}}^{\prec}} p(\sigma_k = i | \sigma_{k+1}, \ldots, \sigma_n, D) = 1$ because of Eq. (10) and $U_{\sigma_k}^{\prec} = U_{\sigma_{k+1}}^{\prec} - \{\sigma_k\}$. Thus, $p(\sigma_k | \sigma_{k+1}, \ldots, \sigma_n, D)$ is a probability mass function (pmf) with k possible σ_k values from $U_{\sigma_{k+1}}^{\prec}$.

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. (19).
- 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. (18).

Provided that $\alpha'_i(S)$ (for each $i \in V$ and each $S \subseteq V - \{i\}$) and $L'(S)$ (for each $S \subseteq V$) have been computed, sampling an order using the above algorithm takes only $O(n^2)$ time because sampling each element σ_k ($k \in \{1, \dots, n\}$) in the order takes $O(n)$ time.

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*

$$\prod_{k=1}^{n} p(\sigma_k | \sigma_{k+1}, \dots, \sigma_n, D) = p(\prec |D). \tag{20}$$

The key to our order sampling algorithm is our realization that the results including $\alpha'_i(S)$ and $L'(S)$ computed from the DP algorithm of Koivisto and Sood (2004) are already sufficient to guide the order sampling process. In an abstract point of view, the results computed from the DP algorithm of Koivisto and Sood (2004) are analogous to the answers provided by the #P-oracle stated in Theorem 3.3 of Jerrum et al. (1986). This theorem states that with the aid of a #P-oracle that is always able to provide the exact counting information (the exact number) of accepting configurations from a currently given configuration, a probabilistic Turing machine can serve as a uniform generator so that every accepting configuration will be reached with an equal positive probability. In our situation, instead of providing the exact counting information, the results computed from the DP algorithm of Koivisto and Sood (2004) are able to provide the exact joint probability $p(\sigma_k, \sigma_{k+1}, \dots, \sigma_n, D)$ for a subsequence $(\sigma_k, \sigma_{k+1}, \dots, \sigma_n)$ of any order $\prec$ for any $k \in \{1, 2, \dots, n\}$, which is shown in the proof of Proposition 1 in Appendix A.1. As a result, the order sampling can be efficiently performed based on the definition of the conditional probability distribution.

3.2 DDS Algorithm

After drawing an order sample, we can then easily sample a DAG by drawing parents for each node according to Eq. (16) as described by Friedman and Koller (2003) (assuming a maximum in-degree k). This naturally leads to our algorithm, termed direct DAG sampling (DDS), as follows:

- Step 1: Run the DP algorithm of Koivisto and Sood (2004) 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. (16).

The correctness of our DDS algorithm is guaranteed by the following theorem.

Theorem 3 *The N_o DAGs sampled according to the 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.*

The time complexity of the DDS algorithm is as follows. Step 1 takes $O(n^{k+1}C(m) + kn2^n)$ time (Koivisto and Sood, 2004), as discussed in Section 2.1. In Step 2, sampling each order takes $O(n^2)$ time. In Step 3, sampling each DAG takes $O(n^{k+1})$ time. Thus, the overall time complexity of our DDS algorithm is $O(n^{k+1}C(m) + kn2^n + n^2N_o + n^{k+1}N_o)$. Because typically we assume $k \geq 1$, the order sampling process (Step 2) does not affect the overall time complexity of the DDS algorithm because of its efficiency.

The time complexity of our DDS algorithm depends on the assumption of the maximum in-degree k. Such an assumption is fairly innocuous, as discussed on page 101 of Friedman and Koller (2003), because DAGs with very large families tend to have low scores. (The maximum-in-degree assumption is also justified in the context of biological expression data on page 270 of Grzegorczyk and Husmeier, 2008.) Accordingly, this assumption has been widely used in the literature (Friedman and Koller, 2003; Koivisto and Sood, 2004; Ellis and Wong, 2008; Grzegorczyk and Husmeier, 2008; Niinimaki et al., 2011; Parviainen and Koivisto, 2011) and the maximum in-degree k has been set to be no greater than 6 in all of their experiments.

Note that the DAG sampling step of the DDS algorithm takes $O(n^{k+1}N_o)$ time. This will dominate the overall running time of the DDS algorithm (even if k is assumed to be 3 or 4), when n is moderate ($n \leq 25$) and the sample size N_o reaches several thousands. Therefore, for the efficiency of our DDS algorithm, we have developed a time-saving strategy for the DAG sampling step, which will be described in details in Section 3.2.1.

Given DAG samples, $\hat{p}_{\prec}(f|D)$, an estimator of the exact posterior of any arbitrary feature f, can be constructed by Eq. (5). Let $C_{n,f}$ denote the time cost of determining the structural feature f in a DAG of n nodes. Then constructing $\hat{p}_{\prec}(f|D)$ takes $O(C_{n,f}N_o)$ time. (For example, C_{n,f_e} $= O(1)$ for an edge feature f_e; $C_{n,f_p} = O(n^2)$ for a path feature f_p.) If we only need order samples, the algorithm consisting of Steps 1 and 2 will be called direct order sampling (DOS). Given order samples, for some modular feature f such as a parent-set feature or an edge feature, $p(f| \prec_i, D)$ can be computed by Eq. (16) or (17), and then $p_{\prec}(f|D)$ can be estimated by Eq. (15). (Because computing a parent-set feature or an edge feature by Eq. (16) or (17) takes $O(1)$ time, estimating $p_{\prec}(f|D)$ by Eq. (15) only takes $O(N_o)$ time for these two features.)

As for the space cost of our DDS algorithm, note that Step 1 of our DDS takes $O(n2^n)$ memory space, which limits the application of our DDS to Bayesian networks with up to around 25 variables. Because a total order can be represented as a vector $(U_1, \ldots, U_n)$ and a DAG can be represented as a vector $(Pa_1, \ldots, Pa_n)$, both a total order and a DAG can be represented in $O(n^2)$ space [3]. Therefore, Steps 2 and 3 of our DDS take $O(n^2N_o)$ memory space, and the overall memory requirement of our DDS algorithm is $O(n2^n + n^2N_o)$.

Because of Theorem 3, the estimator $\hat{p}_{\prec}(f|D)$ based on our DDS algorithm has the following desirable properties.

Corollary 4 *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 the DDS algorithm using Eq. (5) has the following properties:*

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

3. When $n \leq 32$, in the vector representation $(Pa_1, \ldots, Pa_n)$ of a DAG, each parent set Pa_i can be represented by a 32-bit integer whose jth bit indicates whether or not node X_j is a parent of node X_i. Thus, a DAG of a moderate size n can be represented by n 32-bit integers. Similarly, when $n \leq 32$, in the vector representation $(U_1, \ldots, U_n)$ of a total order, each U_i can be represented by a 32-bit integer whose jth bit indicates whether or not node X_j is a predecessor of node X_i. Thus, a total order of a moderate size n can also be represented by n 32-bit integers.

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

(iii) If $0 < p_{\prec}(f|D) < 1$, *then the random variable*

$$\frac{\sqrt{N_o}(\hat{p}_{\prec}(f|D) - p_{\prec}(f|D))}{\sqrt{\hat{p}_{\prec}(f|D)(1 - \hat{p}_{\prec}(f|D))}}$$

has a limiting standard normal distribution.

(iv) For any $\epsilon > 0$ *and 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 4 (iv), which is essentially from the Hoeffding bound (Hoeffding, 1963; Koller and Friedman, 2009), ensures the probability that the error of the estimator $\hat{p}_{\prec}(f|D)$ from the DDS algorithm is bounded by ϵ to be at least $1-\delta$, as long as the sample size $N_o \geq (\ln(2/\delta))/(2\epsilon^2)$. This property, which the existing MCMC algorithms (Friedman and Koller, 2003; Niinimaki et al., 2011) do not have, can be used to obtain quality guarantee for the estimator from our DDS algorithm.

3.2.1 TIME-SAVING STRATEGY FOR THE DAG SAMPLING STEP OF THE DDS

The running time of the DAG sampling step (Step 3) of the DDS algorithm is $O(n^{k+1}N_o)$, which will dominate the overall running time of the DDS algorithm when n is moderate and the sample size N_o reaches several thousands. Thus, in this sub-subsection we introduce our strategy for effectively reducing the running time of the DAG sampling step so that the efficiency of the overall DDS algorithm can be achieved.

In the DAG sampling step, each sampled order $\prec_i = (\sigma_1, \ldots, \sigma_n)^{\prec_i}$ $(1 \leq i \leq N_o)$ can be represented as $\{(\sigma_1, U_{\sigma_1})^{\prec_i}, \ldots, (\sigma_n, U_{\sigma_n})^{\prec_i}\}$, where U_{σ_j} denotes the set of predecessors of σ_j in the order. For each sampled order $\prec_i$, for each $(\sigma_j, U_{\sigma_j})^{\prec_i}$ $(1 \leq j \leq n)$, we need to sample one Pa_{σ_j} of σ_j (one parent set of σ_j) from a list $\{Pa_{\sigma_j z}\}_z^{\sigma_j}$ including every parent set $Pa_{\sigma_j z} \subseteq U_{\sigma_j}^{\prec_i}$. Let Z_j be the length of such a list. Because $Z_j = \sum_{i=0}^{\min\{k,j-1\}} \binom{j-1}{i} = O(n^k)$, sampling one Pa_{σ_j} for σ_j takes $O(n^k)$ time and sampling one DAG takes $O(n^{k+1})$ time. Note that Z_j is an increasing function of j but in the following we will use the notation Z instead of Z_j for notational convenience when the context is clear. In the following we also will not use the plural for mathematical symbols to avoid notational confusion by ending "s". The reader should assume the correct singular or plural case when reading.

When $N_o > 1$, the overall running time of the DAG sampling step can be reduced as follows. Define $\theta_z^{(\sigma_j, U_{\sigma_j})^{\prec_i}}$ as

$$\begin{aligned}\theta_z^{(\sigma_j, U_{\sigma_j})^{\prec_i}} &= P((\sigma_j, Pa_{\sigma_j z})| \prec_i, D) = P((\sigma_j, Pa_{\sigma_j z})|(\sigma_j, U_{\sigma_j})^{\prec_i}, D) \\ &= \beta'_{\sigma_j}(Pa_{\sigma_j z})/[\alpha'_{\sigma_j}(U_{\sigma_j}^{\prec_i})/q_{\sigma_j}(U_{\sigma_j}^{\prec_i})],\end{aligned}$$

for $z \in \{1, \ldots, Z\}$. First, using the common strategy of sampling from a discrete distribution (Koller and Friedman, 2009), for $(\sigma_j, U_{\sigma_j})^{\prec_i}$ we can create $S_I^{(\sigma_j, U_{\sigma_j})^{\prec_i}}$, a sequence of Z probability intervals with the following form:

$$\Big(\big[0, \theta_1^{(\sigma_j,U_{\sigma_j})^{\prec_i}}\big), \\ \big[\theta_1^{(\sigma_j,U_{\sigma_j})^{\prec_i}}, \theta_1^{(\sigma_j,U_{\sigma_j})^{\prec_i}} + \theta_2^{(\sigma_j,U_{\sigma_j})^{\prec_i}}\big), \\ \ldots, \\ \big[\sum_{z=1}^{Z-2} \theta_z^{(\sigma_j,U_{\sigma_j})^{\prec_i}}, \sum_{z=1}^{Z-1} \theta_z^{(\sigma_j,U_{\sigma_j})^{\prec_i}}\big), \\ \big[\sum_{z=1}^{Z-1} \theta_z^{(\sigma_j,U_{\sigma_j})^{\prec_i}}, 1\big)\Big),$$

where the lth interval is $\big[\sum_{z=1}^{l-1} \theta_z^{(\sigma_j,U_{\sigma_j})^{\prec_i}}, \sum_{z=1}^{l} \theta_z^{(\sigma_j,U_{\sigma_j})^{\prec_i}}\big)$. Note that $S_I^{(\sigma_j,U_{\sigma_j})^{\prec_i}}$ can be created in time $O(Z)$ and sampling one Pa_{σ_j} for σ_j from a list $\{Pa_{\sigma_j z}\}_z^{\sigma_j}$ can then be achieved using binary search in time $O(\log Z)$ based on $S_I^{(\sigma_j,U_{\sigma_j})^{\prec_i}}$. Then the following observation is the key reason that the running time of the DAG sampling step can be reduced. For two sampled orders $\prec_i$ and $\prec_{i'}$ ($1 \leq i, i' \leq N_o$), even if $\prec_i \neq \prec_{i'}$, it is possible that $(\sigma_j, U_{\sigma_j})^{\prec_i} = (\sigma_j, U_{\sigma_j})^{\prec_{i'}}$ for some $j \in \{1, \ldots, n\}$. This is because for each j, (σ_j, U_{σ_j}) essentially has a multinomial distribution with N_o trials and a set of $n\binom{n-1}{j-1}$ cell probabilities $\{P((\sigma_j, U_{\sigma_j})|D)\}$. Actually, for any j, the following relation holds for each cell probability:

$$P((\sigma_j, U_{\sigma_j})|D) \propto \alpha'_{\sigma_j}(U_{\sigma_j})L'(U_{\sigma_j})R'(V - U_{\sigma_j} - \{\sigma_j\}), \tag{21}$$

where $R'(.)$ is the special case of $R(.)$ by setting $f \equiv 1$ and $R(.)$ is defined in Eq. (12). The proof of Eq. (21) is very similar to the derivation shown by Koivisto (2006) and is provided in Appendix A.5 Note that $(\sigma_j, U_{\sigma_j})^{\prec_i} = (\sigma_j, U_{\sigma_j})^{\prec_{i'}}$ implies $S_I^{(\sigma_j,U_{\sigma_j})^{\prec_i}} = S_I^{(\sigma_j,U_{\sigma_j})^{\prec_{i'}}}$. Thus, by storing the created $S_I^{(\sigma_j,U_{\sigma_j})^{\prec_i}}$ in the memory, once $(\sigma_j, U_{\sigma_j})^{\prec_i} = (\sigma_j, U_{\sigma_j})^{\prec_{i'}}$ for $i' > i$, creating $S_I^{(\sigma_j,U_{\sigma_j})^{\prec_{i'}}}$ can be avoided and sampling one Pa_{σ_j} for σ_j takes only $O(\log Z)$ time.

On one hand, our strategy will definitely save the running time for these j such that $n\binom{n-1}{j-1}$ (the number of all the possible values of (σ_j, U_{σ_j})) is smaller than N_o if every created $S_I^{(\sigma_j,U_{\sigma_j})}$ is stored. This is because the running time of sampling Pa_{σ_j} of σ_j is only $O(\log Z)$ in at least $N_o - n\binom{n-1}{j-1}$ samples out of the overall N_o samples. (In the worst case, $S_I^{(\sigma_j,U_{\sigma_j})}$ will be created for each possible (σ_j, U_{σ_j}).) For example, when $j = n$, the number of all the possible values of (σ_j, U_{σ_j}) is only n, and Z_j (the length of the list $\{Pa_{\sigma_j z}\}_z^{\sigma_j}$) achieves its maximum among all the j, so that sampling one Pa_{σ_n} for σ_n takes $O(\log Z_n)$ time in at least $N_o - n$ samples. Accordingly, when $j = n$, the worst-case running time of sampling the N_o $(\sigma_n, Pa_{\sigma_n})$ families is $O(n(Z_n + \log Z_n) + (N_o - n)\log Z_n)$ $= O(nZ_n + N_o \log Z_n)$.

On the other hand, our strategy usually can also save the running time even for these j such that the number of all the possible values of (σ_j, U_{σ_j}) is larger than N_o. This is because the probability mass usually is not uniformly distributed among the set of all the possible values of (σ_j, U_{σ_j}). Once the majority of the probability mass p_Σ is concentrated on r_j values of (σ_j, U_{σ_j}), where r_j is a number smaller than N_o, the probability that only these r_j values of (σ_j, U_{σ_j}) appear in all the N_o order samples is $(p_{\Sigma})^{N_o}$. Accordingly, with the probability $(p_{\Sigma})^{N_o}$, the running time of sampling Pa_{σ_j} for σ_j is $O(\log Z_j)$ in at least $N_o - r_j$ samples. As a result, the expected running time of sampling the N_o $(\sigma_j, Pa_{\sigma_j})$ families is below $O([r_j Z_j + N_o \log Z_j](p_{\Sigma})^{N_o} + N_o(Z_j + \log Z_j)(1-(p_{\Sigma})^{N_o}))$; the expected running time of sampling the N_o DAGs is below $O(\sum_{j=1}^{n}\{[r_j Z_j + N_o \log Z_j](p_{\Sigma})^{N_o} + N_o(Z_j + \log Z_j)(1-(p_{\Sigma})^{N_o})\})$. Typically, when m is not small, the local score $score_i(Pa_i : D)$ will not be uniform at all. Correspondingly, it is likely that the multinomial probability mass function $P((\sigma_j, U_{\sigma_j})|D)$ will concentrate dominant probability mass on a small number of (σ_j, U_{σ_j}) candidates that these $(\sigma_j, Pa_{\sigma_j})$ having large local scores are consistent with. As a result, our time-saving strategy will usually become more effective when m is not small.

Note that we also include the policy of recycling the created $S_I^{(\sigma_j, U_{\sigma_j})}$ for our strategy because it is possible that all the memory in a computer will be exhausted in order to store all the created $S_I^{(\sigma_j, U_{\sigma_j})}$, especially when n is not small but m is small. (The space complexity of storing all the $S_I^{(\sigma_j, U_{\sigma_j})}$ is $O(\sum_{j=1}^{n} n\binom{n-1}{j-1}Z_j) = O(n^{k+1}2^{n-1})$.) For this paper, we use a simple recycling method as follows. Some upper limit for the total number of the probability intervals (representing $\big[\sum_{z=1}^{l-1}\theta_z^{(\sigma_j,U_{\sigma_j})^{\prec_i}}, \sum_{z=1}^{l}\theta_z^{(\sigma_j,U_{\sigma_j})^{\prec_i}}\big)$) is pre-specified based on the memory of the computer used. Each time such an upper limit is reached during the DAG sampling step of the DDS, which indicates a large amount of memory has been used to store $S_I^{(\sigma_j, U_{\sigma_j})}$, we recycle the currently stored $S_I^{(\sigma_j, U_{\sigma_j})}$ according to their usage frequencies which serve as the estimates of $P((\sigma_j, U_{\sigma_j})|D)$. The memory for each infrequently used $S_I^{(\sigma_j, U_{\sigma_j})}$ will be reclaimed to ensure that at least a pre-specified number of probability intervals will be recycled from the memory. In addition, in order to have a better use of each created $S_I^{(\sigma_j, U_{\sigma_j})}$ before it possibly gets reclaimed, we sort the N_o sampled orders according to the posterior $p(\prec |D)$ just before executing the DAG sampling step of the DDS. The underlying rationale is that if $p(\prec_i |D)$ is relatively close to $p(\prec_{i'} |D)$, which indicates $p(\prec_i, D)$ is relatively close to $p(\prec_{i'}, D)$ (because $p_{\prec}(D)$ is a constant), it is likely that $\prec_i$ and $\prec_{i'}$ share some (σ_j, U_{σ_j}) component(s) because of Eq. (13). (The extreme situation is that if $p(\prec_i |D)$ equals $p(\prec_{i'} |D)$, it is very likely that $\prec_i$ equals $\prec_{i'}$ so that $\prec_i$ and $\prec_{i'}$ share every (σ_j, U_{σ_j}).) Thus, $\prec_i$ and $\prec_{i'}$ having similar posteriors tend to be close to each other after the sorting, so that it is likely that the common $S_I^{(\sigma_j, U_{\sigma_j})}$ will be used before the reclamation. Furthermore, as N_o increases, the probability that two orders (out of the N_o sampled orders) share some (σ_j, U_{σ_j}) component(s) increases. Accordingly, after the sorting, the probability of reusing $S_I^{(\sigma_j, U_{\sigma_j})}$ before its reclamation will also increase. As a result, the benefit of our time-saving strategy will typically increase when N_o increases.

The experimental results show that our time-saving strategy for the DAG sampling step of the DDS is very effective. Please see the discussion in Section 4 about $\hat{\mu}(T_{DAG})$ and $\hat{\sigma}(T_{DAG})$, the sample mean and the sample standard deviation of the running time of the DAG sampling step of the DDS, which are reported in Tables 2 and 4.

3.3 IW-DDS Algorithm

In this subsection we present our DAG sampling algorithm under the general structure-modular prior (Eq. 2) by effectively correcting the bias due to the use of the order-modular prior.

As mentioned in Section 1, $p_{\prec}(f|D)$ has the bias due to the assumption of the order-modular prior. This is essentially because $p_{\prec}(G|D)$ based on the order-modular prior (Eq. 6) is different from $p_{\not\prec}(G|D)$ based on the structure-modular prior (Eq. 2).

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. (6) is set to be always equal to $p_i(Pa_i)$ in Eq. (2) ($\rho_i(Pa_i) \equiv p_i(Pa_i)$), the following relation holds:

$$p_{\prec}(G|D) = \frac{p_{\not\prec}(D)}{p_{\prec}(D)} \cdot | \prec_G | \cdot p_{\not\prec}(G|D), \tag{22}$$

where $| \prec_G |$ is the number of orders that G is consistent with. (The proof of Eq. 22 is given in Appendix A.6.) Accordingly,

$$p_{\not\prec}(f|D) = \sum_G f(G) p_{\not\prec}(G|D) = \sum_G f(G) \frac{p_{\prec}(D)}{p_{\not\prec}(D)} \cdot \frac{1}{| \prec_G |} p_{\prec}(G|D).$$

Note that $p_{\prec}(D)$ can be computed by the DP algorithm of Koivisto and Sood (2004) in $O(n^{k+1}C(m) + kn2^n)$ time, and $p_{\not\prec}(D)$ can be computed by the DP algorithm of Tian and He (2009) in $O(n^{k+1}C(m) + kn2^n + 3^n)$ time. Thus, if $| \prec_{G_i} |$ is known for each sampled G_i ($i \in \{1, 2, \ldots, N_o\}$), we can use importance sampling to obtain a good estimator

$$\tilde{p}_{\not\prec}(f|D) = \frac{1}{N_o} \sum_{i=1}^{N_o} f(G_i) \frac{p_{\prec}(D)}{p_{\not\prec}(D)} \cdot \frac{1}{| \prec_{G_i} |}, \tag{23}$$

where each G_i is sampled from our DDS algorithm. Unfortunately, $| \prec_{G_i} |$ is #P-hard to compute for each G_i (Brightwell and Winkler, 1991); the state-of-the-art DP algorithm proposed by Niinimaki and Koivisto (2013) for computing $| \prec_{G_i} |$ takes $O(n2^n)$ time. Therefore, in the following we propose an estimator that can be much more efficiently computed than the estimator shown in Eq. (23).

Because $p_{\prec}(f|D)$ has the bias with respect to $p_{\not\prec}(f|D)$, a good estimator $\hat{p}_{\prec}(f|D)$ of $p_{\prec}(f|D)$ typically is not appropriate to be directly used to estimate $p_{\not\prec}(f|D)$. Noticing this problem, Ellis and Wong (2008) proposed to correct this bias for the Order MCMC method as follows: first run the Order MCMC to draw order samples; then for each unique order $\prec$ out of the sampled orders, keep drawing DAGs consistent with $\prec$ (but only keep unique DAGs) until the sum of joint probabilities of these unique DAGs, $\sum_i p(\prec, G_i, D)$, is no less than a pre-specified large proportion (such as 95%) of $p(\prec, D) = \sum_{G \subseteq \prec} p(\prec, G, D)$; finally the resulting union of all the DAG samples is treated as an importance-weighted sample for the structural discovery.

Inspired by the idea of Ellis and Wong (2008), we develop our own bias-correction strategy which is computationally more efficient and can theoretically ensure the resulting estimator to have desirable properties. (Please refer to Section 3.3.1 for detailed discussion.) Our bias-corrected algorithm, termed IW-DDS (importance-weighted DDS), is as follows:

- Step 1 (DDS step): Run the DDS algorithm to draw N_o DAG samples 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 of 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. (3).

Because checking the equality of two DAGs takes $O(n^2)$ time [4], with the use of a hash table, both the expected time cost and the space cost of the bias-correction step of the IW-DDS are $O(n^2 N_o)$. Therefore, the expected time cost of our IW-DDS algorithm is $O(n^{k+1}C(m) + kn2^n + n^2 N_o + n^{k+1} N_o)$, and the required memory space of our IW-DDS algorithm is $O(n2^n + n^2 N_o)$.

Note that when each G_i gets sampled, the corresponding joint probability $p_{\not\prec}(G_i, D)$ can be easily computed and stored with G_i. Therefore, just as constructing the estimator from the DDS, constructing the estimator $\hat{p}_{\not\prec}(f|D)$ from the IW-DDS also takes $O(C_{n,f} N_o)$ time, where $C_{n,f}$ denotes the time cost of determining the structural feature f in a DAG of n nodes.

While Ellis and Wong (2008) showed the effectiveness of their method in correcting the bias merely by the experiments, we first theoretically prove that our estimator has desirable properties as follows.

Theorem 5 *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 the IW-DDS algorithm using Eq. (24) has the following properties:*

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

(ii) $\hat{p}_{\not\prec}(f|D)$ converges almost surely to $p_{\not\prec}(f|D)$.

(iii) The convergence rate of $\hat{p}_{\not\prec}(f|D)$ is $o(a^{N_o})$ for any $0 < a < 1$.

(iv) Define the quantity $\Delta = \sum_{G\in\mathcal{G}} p_{\not\prec}(G|D)$. Then

$$\Delta \cdot \hat{p}_{\not\prec}(f|D) \le p_{\not\prec}(f|D) \le \Delta \cdot \hat{p}_{\not\prec}(f|D) + 1 - \Delta. \tag{26}$$

Note that the introduced quantity $\Delta = \sum_{G\in\mathcal{G}} p_{\not\prec}(G,D)/p_{\not\prec}(D)$ and $\Delta \in [0,1]$ essentially represents the cumulative posterior probability mass of the DAGs in $\mathcal{G}$. Eq. (26) provides a sound interval $[\Delta \cdot \hat{p}_{\not\prec}(f|D), \Delta \cdot \hat{p}_{\not\prec}(f|D) + 1 - \Delta]$ in which $p_{\not\prec}(f|D)$ *must* reside. (The "sound interval" is stronger than the "confidence interval" because there is no probability that $p_{\not\prec}(f|D)$ is outside the sound interval.) The width of the sound interval is $(1-\Delta)$, where Δ is a nondecreasing function of N_o (because if we increase the original N_o to a larger N_o' and sample additional $N_o' - N_o$ DAGs in the DDS step, the resulting $\mathcal{G}'$ is always a superset of the original $\mathcal{G}$). Thus, in the situations where m (the number of data instances) is not very small, it is possible for Δ to approach 1 by a tractable number N_o of DAG samples so that a desired small-width interval for $p_{\not\prec}(f|D)$ can be obtained. (Please refer to Section 4 for the corresponding experimental results.) Also note that Eq. (26) can be expressed in the following equivalent form:

$$-(1-\Delta)\hat{p}_{\not\prec}(f|D) \le p_{\not\prec}(f|D) - \hat{p}_{\not\prec}(f|D) \le (1-\Delta)(1-\hat{p}_{\not\prec}(f|D)), \tag{27}$$

which gives the bound for the estimation error $p_{\not\prec}(f|D) - \hat{p}_{\not\prec}(f|D)$.

4. As described in the footnote in Section 3.2, for the vector representation $(Pa_1, \ldots, Pa_n)$ of a DAG of a size $n \le 32$, each parent set Pa_i can be represented by a 32-bit integer. Accordingly, for two DAGs of a moderate size n, checking their equality takes at most n comparisons by comparing each integer component in their vector representations.

The competing state-of-the-art algorithms that are also applicable to BNs of a moderate size are the Hybrid MCMC method (Eaton and Murphy, 2007) and the K-best algorithm (Tian et al., 2010). The first competing method, the Hybrid MCMC, includes the DP algorithm of Koivisto (2006) (with time complexity $O(n^{k+1}C(m)+ kn2^n)$ and space complexity $O(n2^n)$) as its first phase and then uses the computed posteriors of all the edges to make the global proposal for its second phase (MCMC phase). When its MCMC phase eventually converges, the Hybrid MCMC will correct the bias coming from the order-prior assumption and provide DAG samples according to the DAG posterior so that the estimator $\hat{p}_{\not\prec}(f|D)$ can be constructed using Eq. (5) for any feature f. The Hybrid MCMC has been empirically shown to converge faster than both the Structure MCMC and the Order MCMC, so that more accurate structure-learning performance can be obtained (Eaton and Murphy, 2007). Note that because the REV-MCMC method (Grzegorczyk and Husmeier, 2008) is shown to be only nearly as efficient as the Order MCMC in the mixing and convergence, the Hybrid MCMC is expected to converge faster than the REV-MCMC method as long as n is moderate so that the Hybrid MCMC is applicable. (But the REV-MCMC method has its own value in learning large BNs since all these methods using some DP technique, including the Hybrid MCMC, the K-best algorithm, and our IW-DDS method, are infeasible for a large n because of the space cost.) One limitation of the Hybrid MCMC is that it cannot obtain the interval for $p_{\not\prec}(f|D)$ as specified by Theorem 5 (iv). Additionally, the convergence rate of the estimator from the Hybrid MCMC is not theoretically provided by its authors.

The second competing method, the K-best algorithm, applies some DP technique to obtain a collection $\mathcal{G}$ of DAGs with the K best scores and then uses these DAGs to construct the estimator $\hat{p}_{\not\prec}(f|D)$ by Eq. (24) and Eq. (25). One advantage of the K-best algorithm is that its estimator also has the property specified by Theorem 5 (iv) so that it can provide the sound interval for $p_{\not\prec}(f|D)$ just as our IW-DDS. However, the K-best algorithm has time complexity $O(n^{k+1}C(m)+ T'(K)n2^{n-1})$ and space complexity $O(Kn2^n)$, where $T'(K)$ is the time spent on the best-first search for K solutions and $T'(K)$ has been shown to be $O(K \log K)$ by Flerova et al. (2012). Thus, the increase in K will dramatically increase the computation cost of the K-best algorithm when n is not small. As a result, to obtain an interval width similar to that from our IW-DDS, much more time and space cost is required for the K-best. In our experiments using an ordinary desktop PC, the computation problem becomes severe for $n \geq 19$ because K can only take some small values (such as no more than 40) before the K-best algorithm exhausts the memory. Accordingly, Δ obtained from the K-best is usually smaller than that from our IW-DDS (so that the interval from the K-best is usually wider) even if K is set to reach the memory limit of a computer. (Please refer to Section 4.2 for detailed discussion.)

3.3.1 COMPARISON BETWEEN OUR BIAS-CORRECTION STRATEGY AND THAT OF ELLIS AND WONG (2008)

Our bias-correction strategy used in the IW-DDS solves the computation problem existing in the idea of Ellis and Wong (2008) and ensures the desirable properties of our estimator $\hat{p}_{\not\prec}(f|D)$ stated in Theorem 5.

Because in the Order MCMC, sampling an order is much more computationally expensive than sampling a DAG given an order, the strategy of Ellis and Wong (2008) emphasizes making the full use of each sampled order $\prec$, that is, keeping drawing DAGs consistent with each sampled $\prec$ until the sum of joint probabilities for the unique sampled DAGs, $\sum_i p(\prec, G_i, D)$, is no less than a large proportion (such as 95%) of $p(\prec, D)$. Unfortunately, such a strategy has a computational problem when the number of variables n is not small and the number of data instances m is small. Because there are a super-exponential number ($2^{\Theta(kn\log(n))}$) of DAGs (with the maximum in-degree k) consistent with each order (Friedman and Koller, 2003), it is possible that a non-negligible portion of probability mass $p(\prec, D)$ will be distributed almost uniformly to a majority of these consistent DAGs when m is small. Consequently, $N_G^{\prec}$, the required number of DAGs sampled per each sampled order $\prec$, will be extremely large, leading to large computation cost. For sampling $N_G^{\prec}$ DAGs consistent with each sampled order $\prec$, its expected time cost is $O(n^{k+1} + nk\log(n)N_G^{\prec})$ (even if a time-saving strategy as that described in Section 3.2.1 is used) and its memory requirement is $O(n^2 N_G^{\prec})$. If the memory requirement exceeds the memory of the running computer, the hard disk has to be used to temporarily store the sampled DAGs in some way. (We notice that Ellis and Wong, 2008, limited their experiments to the data sets with at most 14 variables.) If we take the data set "Child" (Tsamardinos et al., 2006) with $n = 20$ and $m = 100$ for example, for an order $\prec$ randomly sampled by our order sampling algorithm, our experiment shows that 1×10^7 DAGs (which contain 932,137 unique DAGs) need to be sampled to let the ratio $\sum_i p(\prec, G_i, D)\ /p(\prec, D)$ reach 94.071%; 1.5×10^7 DAGs (which contain 1,204,262 unique DAGs) need to be sampled to let the ratio $\sum_i p(\prec, G_i, D)\ /p(\prec, D)$ reach 94.952%. To solve this problem, based on the efficiency of our order sampling algorithm, our strategy samples only one DAG from each sampled order in the DDS step, so that the large computation cost per each sampled order is avoided for any data set. Meanwhile, unlike the strategy of Ellis and Wong (2008), our strategy does not delete the duplicate order samples. Therefore, if an order $\prec$ gets sampled j (≥ 1) times in the order sampling step, essentially j DAGs will be sampled for such a unique order in the DAG sampling step. Thus, j, the number of occurrences, implicitly serves as an importance indicator for $\prec$ among the orders.

Furthermore, the strategy of Ellis and Wong (2008) cannot guarantee that the sampled DAGs are independent, even if large computation cost is spent in sampling a huge number of DAGs per each sampled order. This is essentially because multiple DAGs sampled from a fixed order according to the strategy of Ellis and Wong (2008) are not independent. For example, given that a DAG G with an edge $X \to Y$ gets sampled from an order $\prec$, which implies that node X precedes node Y in the given order $\prec$, then the conditional probability that any DAG G' with a reverse edge $Y \to X$ gets sampled under the fixed order $\prec$ becomes zero, so that G and G' are not independent. In general, once the number of sampled orders is fixed, even if the number of sampled DAGs per each sampled order keeps increasing, every DAG that is consistent with none of the sampled orders will still have no chance of getting sampled. In contrast, the sampling strategy in our IW-DDS is able to guarantee the property that all the DAGs sampled from the DDS step are independent, which has been stated in Theorem 3. Such a property is a key to ensuring the good properties of our estimator $\hat{p}_{\not\prec}(f|D)$ stated in Theorem 5.

4. Experimental Results

We have implemented our algorithms in a C++ language tool called "BNLearner" and run several experiments to demonstrate its capabilities. (BNLearner is available at `http://www.cs.iastate.edu/~jtian/Software/BNLearner/BNLearner.htm`.) Our tested data sets include ten real data sets from the UCI Machine Learning Repository (Asuncion and Newman, 2007): "Tic-Tac-Toe," (which is also simply called "T-T-T,") "Glass," "Wine," "Housing," "Credit," "Zoo," "Letter," "Tumor," "Vehicle," and "German." Our tested data sets also include three synthetic data sets: the first one is a synthetic data set "Syn15" generated from a gold-standard 15-node Bayesian network built by us; the second one is a synthetic data set "Insur19" generated from a 19-node subnetwork of "Insurance" Bayesian network (Binder et al., 1997); the third one is a synthetic data set "Child" from a 20-node "Child" Bayesian network used by Tsamardinos et al. (2006). All the data sets contain only discrete variables (or are discretized) and have no missing values (or have their missing values filled in). For the four data sets ("Syn15," "Letter," "Insur19," and "Child"), because a large number of data instances are available, we also vary m (the number of instances) to see the corresponding learning performance. (All the data cases are also included in the tool of BNLearner.) All the experiments in this section were run under Linux on one ordinary desktop PC with a 3.0 GHz Intel Pentium processor and 2.0 GB memory if no extra specification is provided. In addition, the maximum in-degree k was assumed to be 5 for all the experiments.

4.1 Experimental Results for the DDS

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

The Partial Order MCMC (PO-MCMC) method is implemented in BEANDisco, a C++ language tool provided by Niinimaki et al. (2011). (BEANDisco is available at `http://www.cs.helsinki.fi/u/tzniinim/BEANDisco/`.) The current version of BEANDisco can only estimate the posterior of an edge feature, but as Niinimaki et al. (2011) stated, the PO-MCMC readily enables estimating the posterior of any structural feature by further sampling DAGs consistent with an order.

Because n (the number of the variables) in each investigated data case is moderate, we are able to use REBEL, a C++ language implementation of the DP algorithm of Koivisto (2006), to get the exact posterior of every edge under the assumption of the order-modular prior. (REBEL is available at `http://www.cs.helsinki.fi/u/mkhkoivi/REBEL/`.) Therefore, we can use the criterion of the sum of the absolute differences (SAD) (Eaton and Murphy, 2007) to measure the feature-learning performance for each data case:

$$\text{SAD} = \sum_f |p(f|D) - \hat{p}(f|D)|,$$

where $p(f|D)$ is the exact posterior of the investigated feature f, and $\hat{p}(f|D)$ is the corresponding estimator. In this subsection, SAD is essentially $\sum_{ij} |p_{\prec}(i \to j|D) - \hat{p}_{\prec}(i \to j|D)|$, because the investigated feature is the edge feature $i \to j$ under the order-modular prior. A smaller SAD will indicate a better performance in structure discovery. Note that the criterion SAD is closely related to another criterion MAD (the mean of the absolute differences), because MAD = SAD$/(n(n-1))$. Thus, for each data case the conclusion based on the comparison of SAD values is the same as that based on the comparison of MAD values, because $n(n-1)$ is just a constant for each data case.

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

For the setting of the PO-MCMC, according to the suggestion for the optimal setting from Niinimaki et al. (2011), we set the bucket size b to be 10 for all the data cases except Tic-Tac-Toe. The bucket size b was set to be 9 for the data case Tic-Tac-Toe, because Tic-Tac-Toe has only 10 variables and the setting $b = 10$ will cause the tool BEANDisco to throw a run-time error. We

Name	n	m	PO-MCMC		DOS		DDS	
			$\hat{\mu}$(SAD)	$\hat{\sigma}$(SAD)	$\hat{\mu}$(SAD)	$\hat{\sigma}$(SAD)	$\hat{\mu}$(SAD)	$\hat{\sigma}$(SAD)
T-T-T (Tic-Tac-Toe)	10	958	0.5174	0.1280	0.1350	0.0257	0.1547	0.0378
Glass	11	214	0.0696	0.0249	0.0230	0.0067	0.0529	0.0076
Wine	13	178	0.1616	0.0403	0.0400	0.0097	0.0839	0.0137
Housing	14	506	0.3205	0.1303	0.0650	0.0155	0.1150	0.0117
Credit	16	690	0.4549	0.2495	0.0581	0.0221	0.1071	0.0165
Zoo	17	101	0.6079	0.1809	0.1030	0.0127	0.2756	0.0137
Tumor	18	339	0.6059	0.1849	0.0877	0.0226	0.2050	0.0180
Vehicle	19	846	6.9774	8.7960	0.0200	0.0042	0.0547	0.0096
German	21	1,000	2.8802	2.0191	0.1014	0.0208	0.1298	0.0338
Syn15	15	100	0.9024	0.2258	0.1299	0.0191	0.2622	0.0190
		200	0.6449	0.1569	0.0999	0.0119	0.2228	0.0172
		500	0.3424	0.1214	0.0801	0.0183	0.1116	0.0126
		1,000	0.1558	0.0496	0.0550	0.0094	0.0724	0.0118
		2,000	0.0465	0.0209	0.0350	0.0089	0.0473	0.0071
		5,000	0.0217	0.0144	0.0226	0.0073	0.0247	0.0086
Letter	17	100	0.9530	0.1285	0.1526	0.0181	0.2948	0.0229
		200	0.3854	0.0825	0.0794	0.0210	0.1758	0.0142
		500	0.4369	0.1529	0.0732	0.0115	0.1326	0.0107
		1,000	0.3007	0.1254	0.0561	0.0134	0.0828	0.0171
		2,000	1.3740	0.9177	0.1014	0.0373	0.1386	0.0288
		5,000	0.0669	0.0139	0.0199	0.0037	0.0292	0.0088
Insur19	19	100	0.6150	0.1995	0.0977	0.0202	0.1575	0.0213
		200	0.4428	0.1319	0.0655	0.0126	0.1024	0.0162
		500	0.2757	0.1127	0.0462	0.0132	0.0594	0.0099
		1,000	0.4539	0.3031	0.0362	0.0134	0.0422	0.0134
		2,000	0.0100	0.0073	0.0067	0.0037	0.0079	0.0029
		5,000	0.0110	0.0100	0.0095	0.0046	0.0116	0.0043
Child	20	100	0.4997	0.1153	0.0779	0.0143	0.1772	0.0146
		200	0.1896	0.0528	0.0419	0.0102	0.0982	0.0101
		500	0.2385	0.0702	0.0405	0.0066	0.0816	0.0123
		1,000	0.1079	0.0525	0.0284	0.0069	0.0406	0.0080
		2,000	0.0864	0.0521	0.0216	0.0073	0.0275	0.0083
		5,000	0.0938	0.0539	0.0194	0.0058	0.0246	0.0066

Table 1: Comparison of the PO-MCMC, the DOS, and the DDS in Terms of SAD

ran the first 10,000 iterations for "burn-in," and then took 200 partial-order samples at intervals of 50 iterations. Thus, there were 20,000 iterations in total. (The time cost of each iteration in the PO-MCMC is $O(n^{k+1} + n^2 2^b n/b)$.) In the 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 $p(D, f, P_i)$ $= \sum_{\prec \supseteq P_i} \sum_{G \subseteq \prec} f(G)\, p(\prec, G)p(D|G)$. The notation $\sum_{\prec \supseteq P_i}$ means that all the total orders ($\prec$'s) that are linear extensions of the sampled partial order P_i will be included to obtain $p(D, f, P_i)$. For example, for a data set with $n = 20$, because our bucket size $b = 10$, there are $20!/(10!10!) =$ 184,756 total orders that will be included for each sampled partial order P_i. The inclusion of the information of a large number of total orders consistent with each sampled partial order gives great learning power to the PO-MCMC method; such an inclusion can be efficiently computed by the algorithm of Parviainen and Koivisto (2010) with the assumptions of the order-modular prior and the maximum in-degree k. Finally, for the PO-MCMC, the estimated posterior of each edge is computed using $\hat{p}_{\prec}(f|D) = (1/T) \sum_{i=1}^{T} p(f|D, P_i)$.

Because the to-be-learned feature is the edge feature, we can also use our DOS algorithm for the comparison. For both the DOS algorithm and the DDS algorithm, we set $N_o = 20{,}000$, that

Name	n	m	PO-MCMC	DOS	DDS	DDS					
			$\hat{\mu}(T_t)$	$\hat{\mu}(T_t)$	$\hat{\mu}(T_t)$	$\hat{\mu}(T_{DP})$	$\hat{\sigma}(T_{DP})$	$\hat{\mu}(T_{ord})$	$\hat{\sigma}(T_{ord})$	$\hat{\mu}(T_{DAG})$	$\hat{\sigma}(T_{DAG})$
T-T-T	10	958	104.30	1.96	1.73	1.42	0.0159	0.24	0.0066	0.06	0.0017
Glass	11	214	222.02	1.52	1.25	0.89	0.0136	0.25	0.0076	0.09	0.0013
Wine	13	178	374.17	2.56	2.53	1.63	0.0121	0.35	0.0039	0.53	0.0024
Housing	14	506	510.65	4.92	4.54	3.88	0.0143	0.40	0.0033	0.23	0.0020
Credit	16	690	962.97	30.90	30.93	29.57	0.2118	0.44	0.0055	0.90	0.0122
Zoo	17	101	1,331.80	13.75	21.90	12.05	0.0837	0.62	0.0039	9.20	0.1156
Tumor	18	339	1,856.03	44.99	60.21	43.33	1.0763	0.72	0.0052	16.12	0.2941
Vehicle	19	846	2,683.91	149.08	149.23	147.35	1.2863	0.65	0.0079	1.20	0.0219
German	21	1,000	4,887.33	333.43	356.17	330.64	1.3304	0.97	0.0087	24.52	0.4441
Syn15	15	100	677.29	4.82	7.13	3.49	0.0824	0.50	0.0021	3.12	0.0337
		200	677.47	5.91	8.91	4.59	0.0473	0.47	0.0044	3.83	0.0258
		500	686.51	8.56	8.44	7.41	0.1911	0.48	0.0100	0.53	0.0050
		1,000	716.31	13.01	12.52	11.78	0.0927	0.48	0.0115	0.24	0.0022
		2,000	731.50	21.70	21.24	20.58	0.5310	0.49	0.0086	0.15	0.0016
		5,000	731.05	48.01	47.26	46.63	0.4110	0.47	0.0031	0.14	0.0008
Letter	17	100	1,322.43	16.35	21.91	14.64	0.2160	0.64	0.0036	6.60	0.0497
		200	1,315.01	19.74	20.46	18.14	0.0809	0.55	0.0026	1.74	0.0234
		500	1,338.33	27.97	28.21	26.35	0.0489	0.51	0.0066	1.32	0.0130
		1,000	1,343.88	39.45	39.03	38.11	0.3011	0.47	0.0051	0.43	0.0089
		2,000	1,358.29	61.75	61.44	60.52	0.5109	0.47	0.0078	0.42	0.0063
		5,000	1,610.37	126.53	126.49	125.67	0.7967	0.52	0.0058	0.27	0.0053
Insur19	19	100	2,616.56	53.39	86.06	51.06	0.2520	0.86	0.0091	34.11	0.9630
		200	2,633.44	62.12	77.44	59.95	0.3025	0.84	0.0083	16.61	0.2278
		500	2,680.85	80.70	85.03	77.89	0.7618	0.79	0.0082	6.32	0.0535
		1,000	2,734.10	106.37	109.39	104.63	1.0958	0.89	0.0122	3.85	0.0296
		2,000	2,915.60	155.05	158.31	154.26	3.7703	0.90	0.0154	3.13	0.0155
		5,000	3,445.84	297.31	300.51	297.41	4.7737	0.96	0.0090	2.11	0.0091
Child	20	100	3,710.49	102.42	181.38	99.71	0.3497	1.07	0.0109	80.56	1.1980
		200	3,717.10	112.68	168.48	110.05	0.1793	1.04	0.0078	57.36	0.5335
		500	3,757.76	136.98	193.11	134.32	0.4652	1.07	0.0111	57.69	0.7276
		1,000	3,799.47	174.18	186.15	171.54	1.9832	1.08	0.0104	13.50	0.9244
		2,000	4,018.03	241.48	254.26	238.37	2.4952	1.15	0.0214	14.71	0.4833
		5,000	4,531.20	443.54	455.30	441.64	4.8785	1.16	0.0068	12.47	0.4113

Table 2: Comparison of the PO-MCMC, the DOS, and the DDS in Terms of Time (in Seconds)

is, 20,000 (total) orders were sampled. Theoretically, we expect that the learning performance of the DOS should be better than that of the DDS because the additional approximation coming from the DAG sampling step is avoided by the DOS. By listing the performance of the DOS, we mainly intend to examine how much the performance of the DDS decreases because of the additional approximation from sampling one DAG per order. Nevertheless, because the DDS but not the DOS is capable of learning non-modular features, the comparison between the PO-MCMC method and the DDS method is our main task.

Table 1 shows the experimental results in terms of SAD for each data case with n variables and m instances, and Table 2 lists the running-time cost corresponding to Table 1. For each of the three methods, we performed 15 independent runs for each data case. The sample mean and the sample standard deviation of the 15 SAD values of each method, denoted by $\hat{\mu}$(SAD) and $\hat{\sigma}$(SAD), are listed in Table 1. Correspondingly, the sample mean of the total running time T_t of each method, denoted by $\hat{\mu}(T_t)$, is shown in Table 2. (Precisely speaking, the reported total running time T_t of the DDS method includes both the time of running the three steps of the DDS and the relatively tiny $O(N_o)$ time cost of computing $\hat{p}_{\prec}(f|D)$ for each edge f using Eq. 5 at the end. Similarly, the reported total running time T_t of the DOS method also includes the relatively tiny $O(N_o)$ time cost of computing $\hat{p}_{\prec}(f|D)$ for each edge f using Eq. 17 and Eq. 15 at the end.) In addition, the sample mean and the sample standard deviation of the running time of the three steps of the DDS (including the DP step, the order sampling step, and the DAG sampling step), denoted by $\hat{\mu}(T_{DP})$, $\hat{\sigma}(T_{DP})$, $\hat{\mu}(T_{ord})$, $\hat{\sigma}(T_{ord})$, $\hat{\mu}(T_{DAG})$, and $\hat{\sigma}(T_{DAG})$ respectively, are listed in the last six columns of Table 2. Note that we still show $\hat{\mu}(T_{DP})$, the mean running time of the DP step of the DDS in the 15 independent runs, though the DP step is not a random algorithm at all. The running time of the DP step is not exactly the same in each run because of the randomness from uncontrolled factors such as the internal status of the computer. By showing $\hat{\mu}(T_{DP})$, we can clearly see the percentage of the total running time that the DP step typically takes by comparing $\hat{\mu}(T_{DP})$ and $\hat{\mu}(T_t)$.

Tables 1 and 2 clearly illustrate the performance advantage of our DDS method over the PO-MCMC method. The overall time cost of our DDS based on 20,000 DAG samples is much smaller than the corresponding cost of the PO-MCMC method based on 20,000 MCMC iterations in the partial-order space. Using much shorter time, our DDS method has its $\hat{\mu}$(SAD) much smaller than $\hat{\mu}$(SAD) from the PO-MCMC method for 28 out of all the 33 data cases. The five exceptional cases are Glass, Syn15 with m = 2,000, Syn15 with m = 5,000, Insur19 with m = 2,000, and Insur19 with m = 5,000. (In both Glass and Insur19 with m = 2,000, $\hat{\mu}$(SAD) using our DDS method is still smaller than that using the PO-MCMC method, but their difference is not large compared with $\hat{\sigma}$(SAD) from the PO-MCMC method.) Furthermore, because both $\hat{\mu}$(SAD) and $\hat{\sigma}$(SAD) are given in Table 1, by the two-sample t test with unequal variances (Casella and Berger, 2002), we can conclude with strong evidence (at the significance level 5×10^{-3}) that the real mean of SAD using our DDS method is smaller than the real mean of SAD using the PO-MCMC method for each of the 28 data cases. For the exceptional data case Glass, the p-value of the t test is 0.012, so that we can conclude at the significance level 0.05 that the real mean of SAD using our DDS method is smaller than that using the PO-MCMC method. For each of the other four exceptions, by the same t test we cannot reject (with the p-value > 0.2) the null hypothesis that there is no difference in the real means of SAD from the two methods. (Corresponding to Table 1, the comparison is also re-demonstrated using boxplots in the supplementary material for all the 33 data cases.) Thus, the advantage of our DDS algorithm over the PO-MCMC method in learning Bayesian networks of a moderate n can be clearly seen, though the value of the PO-MCMC method still remains for larger n for which our DDS algorithm is infeasible.

In terms of the total running time of the DDS algorithm, Table 2 shows that the running time of the DP step always accounts for the largest portion. The running time of the DAG sampling step is less than 81 seconds to get 20,000 DAG samples for all the 33 cases. Though both the order sampling step and the DAG sampling step involve randomness, the variability of their running time is actually small. This can be seen from the ratio of $\hat{\sigma}(T_{ord})$ to $\hat{\mu}(T_{ord})$ (which is always less than 3.04% for all the 33 cases) and the ratio of $\hat{\sigma}(T_{DAG})$ to $\hat{\mu}(T_{DAG})$ (which is always less than 6.85% for all the 33 cases). The ratio of $\hat{\mu}(T_{DAG})$ to $\hat{\mu}(T_{ord})$ ranges from 0.25 to 75.29 across the 33 cases, which is much smaller than the upper bound of the ratio of $O(n^{k+1}N_o)$ to $O(n^2N_o)$. This indicates that our time-saving strategy introduced in Section 3.2.1 can effectively reduce the running time of the DAG sampling step. In addition, the running time of the DAG sampling step often decreases further when m increases, which can be clearly seen from all the four data sets (Syn15, Letter, Insur19, and Child) with different values of m. Take the data set Letter for example, when m increases from 100 to 1,000, the corresponding $\hat{\mu}(T_{DAG})$ decreases from 6.60 to 0.43

second, a 93.48% of decrease. In summary, the effectiveness of our time-saving strategy introduced in Section 3.2.1 has been clearly shown in Table 2.

In addition, Table 1 clearly shows that the learning performance of the DOS is better than that of the DDS, as expected theoretically. Note that $\hat{\mu}$(SAD) from the DOS is significantly smaller than $\hat{\mu}$(SAD) from the DDS for 28 out of the 33 data cases. The five other cases are Tic-Tac-Toe, Syn15 with m = 5,000, and Insur19 with m = 1,000, 2,000, and 5,000. For each of these 28 data cases, by the two-sample t test with unequal variances (Casella and Berger, 2002), we can conclude (at the significance level 0.05) that the real mean of SAD using the DOS is smaller than that using the DDS. This shows that the additional approximation from the DAG sampling step will usually make the DDS perform worse than the DOS in learning modular features. Nevertheless, the DDS but not the DOS has the ability of learning arbitrary non-modular features, which is the main goal of this paper.

Finally, we present the experimental results for the DDS by varying the sample size. We first choose the data case Letter with $m = 500$ as an example. For the DDS, we tried the sample size N_o = 5,000 $\cdot i$, where $i \in \{1, 2, \ldots, 10\}$. For each i, we independently ran the DDS 15 times to get the sample mean and the sample standard deviation of SAD for the (directed) edge features. For the PO-MCMC, with the bucket size $b = 10$, we ran totally 5,000 $\cdot i$ MCMC iterations in the partial-order space, where $i \in \{1, 2, \ldots, 10\}$. For each i, we discarded the first 2,500 $\cdot i$ MCMC iterations for "burn-in" and set the thinning parameter to be 50, so that $50 \cdot i$ partial orders finally got sampled. Again, for each i, we independently ran the PO-MCMC 15 times to get the sample mean and the sample standard deviation of SAD for the edge features.

Figure 1 shows the SAD performance of the two methods with each i in terms of the edge features, where an error bar represents one sample standard deviation $\hat{\sigma}$(SAD) across 15 runs from a method (the PO-MCMC or the DDS) at each i. For example, Figure 1 shows that when $i = 4$, $\hat{\mu}$(SAD) $= 0.1326$ and $\hat{\sigma}$(SAD) $= 0.0107$ from our DDS algorithm; $\hat{\mu}$(SAD) $= 0.4369$ and $\hat{\sigma}$(SAD) $= 0.1529$ from the PO-MCMC method. This exactly matches the results previously shown in Table 1. Correspondingly, Figure 2 shows $\hat{\mu}(T_t)$ (the sample mean of the total running time) of the PO-MCMC and the DDS with each i, where the running time is in seconds. The advantage of our DDS can be clearly seen from Figures 1 and 2. For each $i \in \{1, 2, \ldots, 10\}$, the real mean of SAD from the DDS is significantly smaller than that from the PO-MCMC with the p-value $< 5 \times 10^{-3}$ returned by the two-sample t test (with unequal variances). Meanwhile, the total running time of the DDS is much shorter than that of the PO-MCMC. For example, $\hat{\mu}(T_t)$ of the DDS increases with respect to i and reaches only 29.55 seconds at $i = 10$. This is shorter than 9% of $\hat{\mu}(T_t)$ of the PO-MCMC at $i = 1$, which is 336.09 seconds. Therefore, the learning performance of the DDS with each sample size is significantly better than that of the PO-MCMC for the data case Letter with $m = 500$.

We also performed the experiment with the same experimental settings for the data cases Tic-Tac-Toe, Wine, Child with $m = 500$, and German. Please refer to the supplementary material for the experimental results. (The supplementary material is available at `http://www.cs.iastate.edu/~jtian/Software/BNLearner/BN_Learning_Sampling_Supplement.pdf`.) The same conclusion about the learning performance can be clearly drawn by examining the figures shown in the supplementary material.

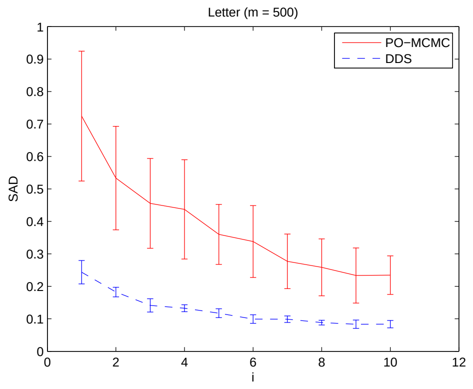

Figure 1: Plot of the SAD Performance of the PO-MCMC and the DDS for Letter ($m = 500$)

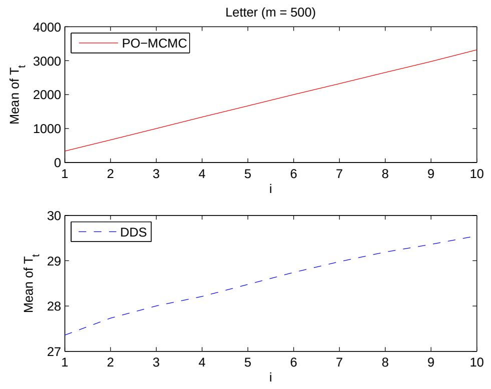

Figure 2: Plot of the Total Running Time of the PO-MCMC and the DDS for Letter ($m = 500$)

4.2 Experimental Results for the IW-DDS

In this subsection, we compare our IW-DDS algorithm with the Hybrid MCMC (that is, DP+MCMC) method (Eaton and Murphy, 2007) and the K-best algorithm (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 the Hybrid MCMC (called BDAGL) and the implementation of the K-best (called KBest) are both made available online by their corresponding authors. (BDAGL is available at `http://www.cs.ubc.ca/~murphyk/Software/BDAGL/`; KBest is available at `http://www.cs.iastate.edu/~jtian/Software/UAI-10/KBest.htm`.)

Because n in each investigated data case is moderate, we are able to use POSTER, a C++ language implementation of the DP algorithm of Tian and He (2009), to get $p_{\not\prec}(i \rightarrow j|D)$, the exact posterior of each edge $i \rightarrow j$ under the structure-modular prior. (POSTER is available at `http://www.cs.iastate.edu/~jtian/Software/UAI-09/Poster.htm`.) Therefore, we can use the SAD criterion ($\sum_{ij} |p_{\not\prec}(i \rightarrow j|D) - \hat{p}_{\not\prec}(i \rightarrow j|D)|$) to measure the performance of these three methods in the structure learning for each data case.

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

As for the DP+MCMC, we note that most part of its implementation in BDAGL tool is written in Matlab, whereas both the K-best and the IW-DDS are implemented in C++. In order to make relatively fair comparison in terms of the running time, we used REBEL tool, a C++ implementation of the DP algorithm of Koivisto (2006), to perform the computation in the DP phase of the DP+MCMC; but for fair comparison 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 used the Matlab implementation of BDAGL [5] and ran it under Windows 7 on

5. The original BDAGL was found to get an out-of-memory error for any data case with more than 19 variables in our experiments. This is because the original BDAGL intends to pre-compute the local scores of all the $n2^{n-1}$ possible families and store them in an array for the later usage in both the DP phase and the MCMC

Name	n	m	DP	DP+MCMC		K-best	IW-DDS		K-best	IW-DDS	
			SAD	$\hat{\mu}$(SAD)	$\hat{\sigma}$(SAD)	SAD	$\hat{\mu}$(SAD)	$\hat{\sigma}$(SAD)	Δ	$\hat{\mu}(\Delta)$	$\hat{\sigma}(\Delta)$
T-T-T	10	958	0.1651	15.0079	2.9877	1.4194	0.0227	0.0102	6.943E-01	9.935E-01	8.636E-04
Glass	11	214	1.5444	0.3587	0.4599	0.0904	0.0381	0.0019	9.780E-01	9.901E-01	6.368E-04
Wine	13	178	1.4786	0.4605	0.2968	0.2011	0.1041	0.0075	9.023E-01	9.670E-01	1.700E-03
Housing	14	506	5.6478	8.0000	3.3408	9.1179	4.8276	0.0624	2.880E-02	1.096E-01	1.200E-03
Credit	16	690	4.0580	5.0261	2.3482	5.1492	2.9148	0.0336	5.010E-02	1.793E-01	1.700E-03
Zoo	17	101	8.2142	32.4189	10.0953	35.1215	19.7025	4.0767	3.815E-08	1.652E-11	1.211E-11
Tumor	18	339	5.1536	17.5104	7.9198	20.5793	10.3139	0.6950	2.940E-05	3.619E-06	3.586E-07
Vehicle	19	846	3.5759	3.8234	0.4011	3.3683	0.4648	0.0194	5.387E-01	9.432E-01	2.600E-03
German	21	1,000	3.7261	5.0207	3.2223	5.5902	0.9891	0.0449	7.800E-02	7.422E-01	7.200E-03
Syn15	15	100	4.9321	12.8705	7.6384	11.8685	10.1341	0.1495	1.604E-04	1.183E-04	4.664E-06
		200	3.2557	4.5090	2.5875	7.5232	4.9079	0.0583	2.700E-03	7.300E-03	1.265E-04
		500	6.9798	5.5466	1.9175	4.4379	4.2965	0.1619	1.526E-01	2.388E-01	4.900E-03
		1,000	1.3000	0.3974	0.3299	0.0848	0.0498	0.0048	9.699E-01	9.843E-01	8.909E-04
		2,000	1.7192	1.8263	1.7095	0.3701	0.1081	0.0147	8.521E-01	9.703E-01	3.300E-03
		5,000	1.9473	0.0304	0.0094	8.89E-04	0.0022	0.0002	9.998E-01	9.994E-01	9.821E-05
Letter	17	100	9.2140	27.1507	4.0940	24.4313	15.8780	0.2764	2.908E-04	1.621E-04	5.336E-06
		200	7.2855	15.1587	3.5615	9.4512	6.7936	0.1191	7.800E-03	1.220E-02	3.341E-04
		500	6.0961	3.4637	4.6789	1.7237	0.6347	0.0119	6.948E-01	8.808E-01	3.000E-03
		1,000	0.6394	0.1761	0.0166	0.0837	0.0766	0.0039	9.834E-01	9.864E-01	8.678E-04
		2,000	2.3913	3.5085	3.1132	2.0976	0.1338	0.0213	6.859E-01	9.756E-01	2.700E-03
		5,000	0.8407	0.1182	0.0442	0.0160	0.0072	0.0005	9.948E-01	9.972E-01	2.077E-04
Insur19	19	100	5.3356	9.4318	3.9576	11.7779	6.5062	0.0891	6.000E-03	2.010E-02	4.767E-04
		200	5.9844	5.5465	2.7295	2.2572	1.4630	0.0557	4.495E-01	7.049E-01	1.120E-02
		500	1.8274	0.3605	0.2287	0.4970	0.1328	0.0105	7.464E-01	9.379E-01	3.000E-03
		1,000	1.7386	0.2186	0.0762	0.7498	0.0623	0.0111	5.866E-01	9.785E-01	2.600E-03
		2,000	1.2737	0.1217	0.0380	0.0174	0.0062	0.0012	9.900E-01	9.976E-01	4.864E-04
		5,000	1.9511	0.1765	0.0594	0.0103	0.0092	0.0010	9.970E-01	9.973E-01	2.086E-04
Child	20	100	6.9783	11.8987	3.1086	11.6189	7.0304	0.0909	7.848E-04	2.700E-03	1.066E-04
		200	3.2826	4.7066	4.3749	5.0729	2.8510	0.0192	4.800E-03	1.506E-01	1.200E-03
		500	2.5580	2.4716	1.3489	1.5304	0.5416	0.0305	3.582E-01	8.222E-01	5.100E-03
		1,000	2.4708	2.6061	2.2909	0.7066	0.1499	0.0198	7.013E-01	9.545E-01	3.400E-03
		2,000	2.3330	1.4286	1.2290	1.5279	0.0662	0.0161	6.509E-01	9.828E-01	2.800E-03
		5,000	2.0365	1.2533	1.7313	0.8783	0.0150	0.0013	8.209E-01	9.940E-01	4.696E-04

Table 3: Comparison of the DP+MCMC, the K-best, and the IW-DDS in Terms of SAD

an ordinary laptop with 2.40 GHz Intel Core i5 CPU and 4.0 GB memory. The MCMC used the pure global proposal (with the local proposal choice $\beta = 0$), because such a setting was reported by 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 the burn-in period. Then we set the thinning parameter to be 3 to get the final 30,000 DAG samples. As a result, the time statistics of the DP phase (the numbers before the + sign in Table 4) but not the MCMC phase (the numbers after the + sign) can be directly compared with those 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.

phase. To solve this out-of-memory problem, we have updated the original Matlab code in BDAGL and provided the BDAGL-New package which is also available at `http://www.cs.iastate.edu/~jtian/Software/BNLearner/BNLearner.htm`. The main update is that, with the assumed maximum in-degree, only the local scores of all the families whose sizes are no more than the assumed maximum in-degree are pre-computed and stored in a hash table. With BDAGL-New, the experiments for all the data cases in this paper can be performed without any error.

Name	n	m	DP+MCMC	K-best	IW-DDS	IW-DDS					
			$\hat{\mu}(T_t)$	T_t	$\hat{\mu}(T_t)$	$\hat{\mu}(T_{DP})$	$\hat{\sigma}(T_{DP})$	$\hat{\mu}(T_{ord})$	$\hat{\sigma}(T_{ord})$	$\hat{\mu}(T_{DAG})$	$\hat{\sigma}(T_{DAG})$
T-T-T	10	958	1.29 + 1,032.40	8.37	3.28	1.27	0.0060	1.57	0.0109	0.44	0.0128
Glass	11	214	1.04 + 1,037.26	18.13	1.72	0.98	0.0210	0.50	0.0054	0.24	0.0018
Wine	13	178	2.44 + 1,127.80	141.20	3.52	2.15	0.0199	0.63	0.0061	0.74	0.0057
Housing	14	506	4.83 + 1,421.00	323.37	5.67	4.21	0.0813	0.63	0.0042	0.52	0.0043
Credit	16	690	33.73 + 1,476.90	2,073.41	35.20	29.30	0.3322	0.81	0.0055	4.94	1.6191
Zoo	17	101	22.33 + 2,107.50	5,531.81	26.16	12.49	0.1853	1.11	0.0048	12.05	0.1898
Tumor	18	339	60.86 + 1,799.99	18,640.09	87.35	39.49	0.4419	1.19	0.0184	46.31	0.2937
Vehicle	19	846	207.27 + 1,886.10	17,126.45	171.75	160.55	0.9310	1.49	0.0175	9.67	0.0665
German	21	1,000	600.19 + 1,849.90	10,981.29	540.61	392.25	2.8656	1.68	0.0207	146.65	1.4290
Syn15	15	100	5.21 + 1,284.00	889.97	9.41	3.56	0.0667	0.81	0.0048	4.80	0.0273
		200	6.28 + 1,286.20	901.23	10.29	4.76	0.2090	0.80	0.0111	4.53	0.0256
		500	9.64 + 1,336.80	899.42	9.59	7.50	0.0821	0.85	0.0051	1.08	0.0128
		1,000	13.69 + 1,364.60	910.76	13.35	12.15	0.7578	0.85	0.0156	0.33	0.0038
		2,000	22.64 + 1,372.10	907.02	21.98	20.81	0.5200	0.85	0.0061	0.30	0.0019
		5,000	54.79 + 1,356.70	932.96	52.07	47.80	1.0887	3.99	0.0235	0.28	0.0023
Letter	17	100	25.53 + 1,572.60	7,639.02	20.27	15.83	0.0601	1.15	0.0062	2.83	0.0292
		200	30.01 + 1,576.80	7,966.25	25.87	20.22	0.1769	1.09	0.0069	4.21	0.0304
		500	39.58 + 1,598.90	8,257.60	31.88	29.84	0.1575	0.95	0.0055	1.01	0.0123
		1,000	52.85 + 1,575.60	8,380.57	44.48	42.93	0.4835	0.98	0.0093	0.56	0.0100
		2,000	77.48 + 1,591.00	7,619.29	69.14	66.34	0.5184	2.01	0.0241	0.78	0.0068
		5,000	157.69 + 1,636.40	8,134.85	136.95	134.94	1.6127	1.36	0.0097	0.65	0.0067
Insur19	19	100	101.47 + 1,828.00	6,745.41	112.28	55.85	0.1540	1.41	0.0117	54.71	0.5438
		200	113.79 + 1,896.10	6,783.76	82.52	68.12	0.4187	1.45	0.0120	12.90	0.1329
		500	137.18 + 1,864.40	6,894.15	98.96	91.44	0.4547	1.43	0.0128	6.07	0.0358
		1,000	168.55 + 1,862.30	6,966.90	133.87	123.42	0.8007	1.44	0.0157	9.00	0.0601
		2,000	226.36 + 1,781.70	7,277.60	185.02	177.66	1.3165	3.30	0.0483	4.06	0.0356
		5,000	380.78 + 1,814.80	7,061.76	336.03	329.78	4.5841	1.89	0.0220	4.34	0.0713
Child	20	100	203.44 + 1,785.10	15,085.86	223.62	106.01	0.3976	1.67	0.0176	115.60	1.3183
		200	215.70 + 1,760.80	14,222.86	225.76	119.82	1.8604	1.69	0.0107	104.09	0.9659
		500	248.17 + 1,818.70	14,016.94	214.91	149.73	0.8344	1.67	0.0183	63.48	0.5783
		1,000	292.40 + 1,817.20	15,504.82	232.11	193.18	1.1041	1.72	0.0135	37.20	0.3176
		2,000	371.12 + 1,841.40	16,109.91	306.42	268.78	2.3209	1.82	0.0165	35.81	0.3680
		5,000	589.99 + 1,846.40	15,372.61	524.63	483.90	6.7403	1.93	0.0160	38.80	0.5411

Table 4: Comparison of the DP+MCMC, the K-best, and the IW-DDS in Terms of Time (in Seconds)

For our IW-DDS, we set $N_o = 30{,}000$. We performed 20 independent runs for each data case to get the results. For the K-best, note that its SAD is fixed because there is no randomness in the computed results. So we only ran it once to get the result. We set K to be 100 for Tic-Tac-Toe, Glass, Wine, Housing, Credit, Zoo, Syn15, and Letter, that is, we got the 100 best DAGs from Tic-Tac-Toe, Glass, Wine, Housing, Credit, Zoo, each of the six cases of Syn15, and each of the six cases of Letter. We set K to be only 80 for Tumor because our experiments showed that for Tumor the K-best program ran out of memory with $K > 80$. Because of the same out-of-memory issue, we set K to be only 40 for Vehicle and Insur19; we set K to be only 20 for Child and 9 for German. The fact that K can only take a value no greater than 40 for $n \geq 19$ in our experiments confirms our claim about the computation problem of the K-best algorithm in terms of its space cost.

Table 3 shows the experimental results in terms of SAD for each data case, and Table 4 shows the running-time cost corresponding to Table 3. The column named DP in Table 3 shows the SAD ($\sum_{ij} |p_{\not\prec}(i \to j|D) - p_{\prec}(i \to j|D)|$), where each edge posterior $p_{\not\prec}(i \to j|D)$ is computed by the exact DP method of Tian and He (2009), and each edge posterior $p_{\prec}(i \to j|D)$ is computed by the exact DP method of Koivisto (2006). The SAD values reported in this column indicate the bias due to the assumption of the order-modular prior. Next to the DP column, the SAD values of the three methods are listed in Table 3. Because the DP+MCMC method and the IW-DDS method are random, both the sample mean $\hat{\mu}$(SAD) and the sample standard deviation $\hat{\sigma}$(SAD) of the 20 SAD values are shown for these two methods. The outcome of the K-best algorithm is not random, so that only its SAD is shown. Finally Table 3 also shows the cumulative posterior probability mass Δ for both the K-best algorithm and the IW-DDS method. Again, because the IW-DDS method is random, both the sample mean and the sample standard deviation of the 20 Δ values, denoted by $\hat{\mu}(\Delta)$ and $\hat{\sigma}(\Delta)$, are listed for the IW-DDS method in Table 3. As for Table 4, its presentation format is very similar to that of Table 2. Table 4 first shows the sample mean of the total running time T_t (denoted by $\hat{\mu}(T_t)$) of both the DP+MCMC method and the IW-DDS method as well as the total running time T_t of the K-best algorithm. The last six columns of Table 4, similar to the last six columns of Table 2, list the sample mean and the sample standard deviation of the running time of the three steps of the DDS in the IW-DDS algorithm.

Tables 3 and 4 clearly demonstrate the advantage of our method over the other two methods. With much shorter computation time, our method has its $\hat{\mu}$(SAD) less than that from the DP+MCMC for 32 out of the 33 data cases. The only exceptional case is Syn15 with $m = 200$. Furthermore, based on the two-sample t test with unequal variances, we can conclude at the significance level 0.05 that the real mean of SAD using our method is less than that using the DP+MCMC for each of the 31 cases; the two exceptional cases are Syn15 with $m = 100$ and Syn15 with $m = 200$. (For 30 out of these 31 cases, the p-value of the two-sample t test is less than 0.01.) Meanwhile, $\hat{\sigma}$(SAD) using our method is always much smaller than that using the DP+MCMC for each of the 33 cases, which indicates higher stability of the performance of our method. Similarly, with much shorter computation time, our method has its $\hat{\mu}$(SAD) less than the SAD from the K-best for 32 out of the 33 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 at the significance level 5×10^{-4} that the real mean of SAD using our method is less than the SAD using the K-best for each of these 32 cases. (Corresponding to Table 3, the comparison of the three methods is also re-illustrated using boxplots in the supplementary material for all the 33 data cases.)

There are several other interesting things shown in Tables 3 and 4. In terms of the SAD, for very small m, $\hat{\mu}$(SAD) using the DP+MCMC method is even larger than the SAD from the DP phase (Koivisto, 2006) itself. For example, for the data case Zoo, the SAD from the DP phase is 8.2142, but $\hat{\mu}$(SAD) obtained after the MCMC phase of the DP+MCMC method is 32.4189. Similar situations also occur in Syn15, Letter, Insur19, and Child when $m = 100$. This indicates that for very small m, the MCMC phase of the DP+MCMC method is unable to reduce the bias from the DP method of Koivisto (2006) for all these cases based on 190,000 MCMC iterations. As for the running time, note that $\hat{\mu}(T_{DP})$ of our IW-DDS is always less than the running time of the DP phase of the DP+MCMC method. This is because the DP step of our method uses the DP algorithm of Koivisto and Sood (2004), that is, the first three steps of the DP algorithm of Koivisto (2006); while the DP phase of the DP+MCMC method uses all the five steps of the DP algorithm of Koivisto (2006). In other words, compared with the DP algorithm of Koivisto and Sood (2004), the DP algorithm of Koivisto (2006) includes a larger constant factor hidden in the $O(n^{k+1}C(m) + kn2^n)$ notation though these two DP algorithms have the same time complexity. This difference will make the total running time of our IW-DDS even less than the running time of the DP phase of the DP+MCMC method when the remaining steps of the IW-DDS run faster than the last two steps of the DP algorithm of Koivisto (2006). For example, for the data case Child with $m = 5{,}000$, $\hat{\mu}(T_t)$ of the IW-DDS is 524.63 seconds while the corresponding running time of the DP phase of the DP+MCMC method is 589.99 seconds. Actually, Table 4 shows that there are 21 out of the 33 cases where $\hat{\mu}(T_t)$ of the IW-DDS is less than the running time of the DP phase of the DP+MCMC method. In addition, just as shown in Section 4.1, the effectiveness of our time-saving strategy can also be clearly seen from Table 4. For example, the ratio of $\hat{\mu}(T_{DAG})$ to $\hat{\mu}(T_{ord})$ ranges from 0.07 to 87.29 across the 33 cases, which is much smaller than the upper bound of the ratio of $O(n^{k+1}N_o)$ to $O(n^2 N_o)$.

Table 3 also shows the resulting cumulative probability mass Δ from the K-best and our IW-DDS for all the data cases. (Note that Δ is computed by the formula $\Delta = \sum_{G\in\mathcal{G}} p_{\not\prec}(G, D)/p_{\not\prec}(D)$, where $p_{\not\prec}(D)$ is computed using POSTER tool.) In the table, $\hat{\mu}(\Delta)$ from our IW-DDS is greater than Δ from the K-best for 28 out of the 33 data cases. The five exceptional cases are Zoo, Tumor, Syn15 with $m = 100$, Syn15 with $m = 5{,}000$, and Letter with $m = 100$. Interestingly, for four out of the five exceptional cases (as well as the other 28 cases), $\hat{\mu}$(SAD) from our IW-DDS is significantly smaller than SAD from the K-best. One possible reason is that the K best DAGs tend to have the same or similar local structures (families (i, Pa_i)'s) that have relatively large local scores, but a large number of DAGs sampled from our IW-DDS include various local structures for each node i. When Δ is far below 1, the inclusion of various local structures seems to be more effective in improving the structure-learning performance.

In addition, Table 3 shows that when m is not very small (such as no smaller than 1,000), Δ from our IW-DDS with $N_o = 30{,}000$ can reach a large percentage (such as greater than 90%) in most of our data cases. As a result, we can obtain a sound interval for $p_{\not\prec}(f|D)$ with a small width (such as less than 0.1) for any feature f.

To further demonstrate that our IW-DDS can obtain a large Δ efficiently when m is not very small, we increased N_o from 100,000 to 600,000 with each increment 100,000 to see its performance for the data cases Letter with $m = 500$, Child with $m = 500$, and German. Again, we performed 20 independent runs for each data case to get the results. Figures 3, 5, and 7 show the increase in Δ with respect to the increase in N_o for these three data cases. Correspondingly, Figures 4, 6, and 8 indicate the increase in $\hat{\mu}(T_t)$, $\hat{\mu}(T_{ord})$ and $\hat{\mu}(T_{DAG})$ with respect to the increase in N_o for these three data cases, where the running time is in seconds. These figures clearly show that our IW-DDS can efficiently achieve a large Δ. Take the data case German for example, with the time cost $\hat{\mu}(T_t) = 1{,}493.02$ seconds, our IW-DDS can collect $N_o = 600{,}000$ DAG samples so that the corresponding mean of Δ can reach 91.74%. Therefore, for any feature f in the data case German, our IW-DDS can provide a sound interval for $p_{\not\prec}(f|D)$ with a width of 0.0826. Note that the K-best can only provide a meaningless sound interval for $p_{\not\prec}(f|D)$ with a huge width of 0.922 because its Δ can only reach 0.078 in the data case German before running out of the memory. Also note that the ratio of $\hat{\mu}(T_{DAG})$ to $\hat{\mu}(T_{ord})$ decreases from 56.45 to 30.95 when N_o increases from 100,000 to 600,000. (The rate of increase of $\hat{\mu}(T_{ord})$ is a constant with respect to N_o, but the rate of increase of $\hat{\mu}(T_{DAG})$ actually decreases as N_o increases.) This confirms our statement in Section 3.2.1 that the benefit from our time-saving strategy will typically increase when N_o increases.

Finally, we present the experimental results for the IW-DDS by varying the sample size. As in Section 4.1, the experiments were performed for the five data cases Tic-Tac-Toe, Wine, Letter with $m = 500$, Child with $m = 500$, and German. For the IW-DDS, we tried the sample size $N_o = 5{,}000 \cdot i$, where $i \in \{1, 2, \ldots, 10\}$. For each i, we independently ran the IW-DDS 20 times to get the sample mean and the sample standard deviation of SAD for the (directed) edge features. For

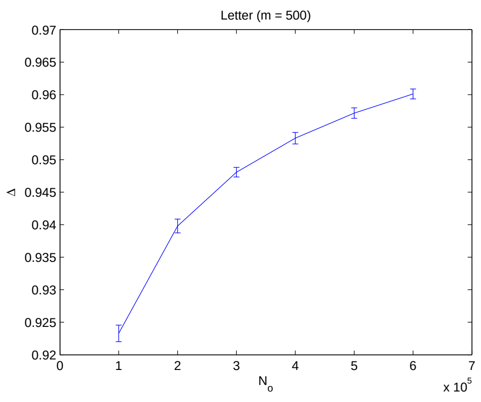

Figure 3: Plot of Δ versus N_o for Letter ($m = 500$)

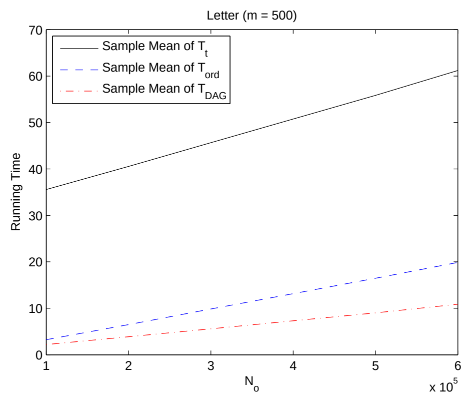

Figure 4: Plot of the Running Time versus N_o for Letter ($m = 500$)

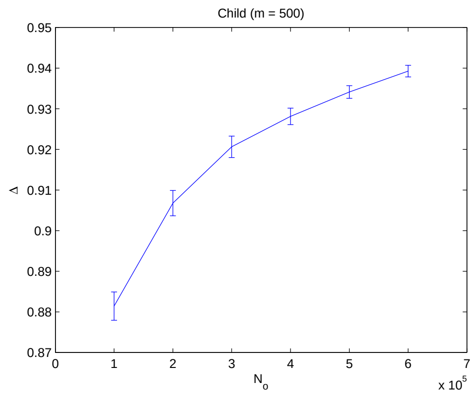

Figure 5: Plot of Δ versus N_o for Child ($m = 500$)

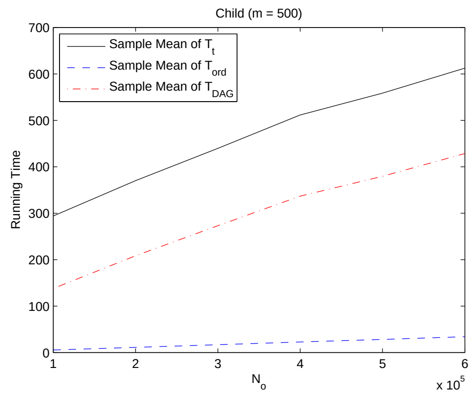

Figure 6: Plot of the Running Time versus N_o for Child ($m = 500$)

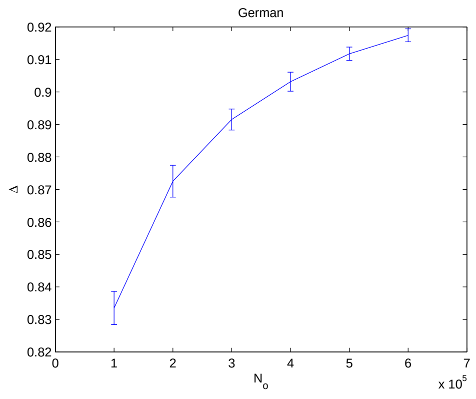

Figure 7: Plot of Δ versus N_o for German

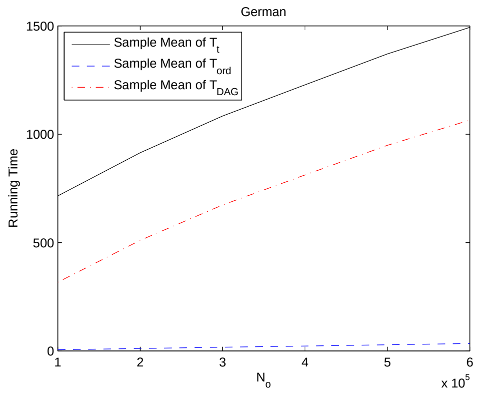

Figure 8: Plot of the Running Time versus N_o for German

the DP+MCMC, we ran totally 50,000 $\cdot i$ MCMC iterations, where $i \in \{1, 2, \ldots, 10\}$. For each i, we discarded the first 25,000 $\cdot i$ MCMC iterations for "burn-in" and set the thinning parameter to be 5, so that 5,000 $\cdot i$ DAGs got sampled. Again, for each i, we independently ran the MCMC 20 times to get the sample mean and the sample standard deviation of SAD for the edge features. As for the K-best, different experimental settings were used for different data cases because of the out-of-memory issue. For the two data cases Tic-Tac-Toe and Wine, we ran the K-best program with $K = 20 \cdot i$, where $i \in \{1, 2, \ldots, 10\}$. (The setting of $K = 20 \cdot i$ guarantees that for these two data cases the running time of the K-best is longer than the running time of the IW-DDS at each i, which will be demonstrated soon.) For the data case Letter with $m = 500$, we only ran the K-best with $K = 162$ because the K-best program will run out of memory when $K > 162$ because of its expensive space cost. The corresponding result of the K-best would be compared with the result of the IW-DDS at $i = 10$ (that is, the IW-DDS with $N_o =$ 50,000). For the same out-of-memory issue, we only set $K = 20$ for Child with $m = 500$ and set $K = 9$ for German when running the K-best program. Note that because there is no randomness in the outcome of the K-best algorithm, we always ran the K-best program only once to get its fixed SAD for the edge features.

The experimental results of comparing the three methods based on the data case Tic-Tac-Toe are shown in Figures 9 and 10. Figure 9 shows the SAD performance of the three methods with each $i \in \{1, 2, \ldots, 10\}$ in terms of the edge features, where an error bar represents one sample standard deviation $\hat{\sigma}$(SAD) across 20 runs from the DP+MCMC or the IW-DDS at each i. Figure 10 shows $\hat{\mu}(T_t)$ (the sample mean of the total running time) of the DP+MCMC and the IW-DDS as well as T_t (the total running time) of the K-best with each i, where the running time is in seconds. The advantage of our IW-DDS can be clearly seen from Figures 9 and 10. Comparing with the DP+MCMC, for each $i \in \{1, 2, \ldots, 10\}$, the IW-DDS uses shorter running time but has its real mean of SAD significantly smaller than that from the DP+MCMC, with the p-value $< 1 \times 10^{-10}$ from the two-sample t test with unequal variances. Comparing with the K-best, for each $i \in \{1, 2, \ldots, 10\}$, the IW-DDS uses shorter running time but has its real mean of SAD significantly smaller than the SAD from the K-best, with the p-value $< 1 \times 10^{-35}$ from the one-sample t test. Therefore, the learning performance of the IW-DDS is significantly better than that of the other two methods at each i for the data case Tic-Tac-Toe.

The experimental results based on the data case Letter with $m = 500$ are shown in Figures 11 and 12. Just as the description for Figures 9 and 10, Figure 11 shows the SAD performance of the three methods in terms of the edge features, and Figure 12 shows the corresponding time cost of the three methods. The only difference is that in both Figure 11 and Figure 12, the corresponding result of the K-best with $K = 162$ is marked as a star and compared with that of the IW-DDS at $i = 10$. The advantage of our IW-DDS can be clearly seen from Figures 11 and 12. Comparing with the DP+MCMC, for each $i \in \{1, 2, \ldots, 10\}$, the IW-DDS uses much shorter running time but has its real mean of SAD significantly smaller than that from the DP+MCMC, with the p-value < 0.013 from the two-sample t test with unequal variances. ($\hat{\mu}(T_t)$ of the IW-DDS at $i = 10$ is only 33.00 seconds, which is even less than the running time 39.58 seconds of the DP phase of the DP+MCMC method.) Note that $\hat{\sigma}$(SAD) from the DP+MCMC is shown to be very large. The DP+MCMC has its $\hat{\sigma}$(SAD) even larger than its $\hat{\mu}$(SAD) (the sample mean of SAD) when $i \geq 8$, which indicates that the performance of the DP+MCMC is not stable based on the 500,000 MCMC iterations. Comparing with the K-best, the IW-DDS with $N_o =$ 50,000 uses much shorter running time but has its real mean of SAD significantly smaller than the SAD from the K-best with $K = 162$, because the p-value from the corresponding one-sample t test is less than 1×10^{-35}.

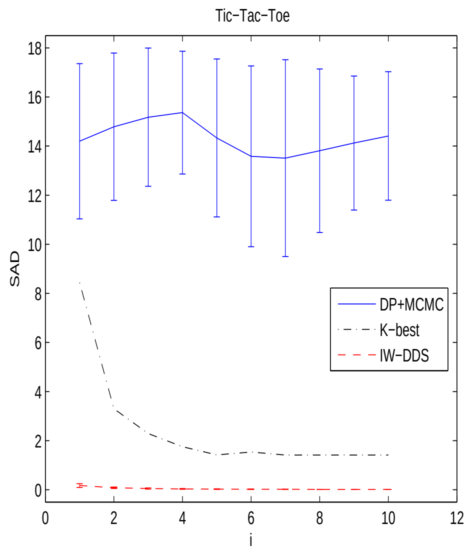

Figure 9: Plot of the SAD Performance of the DP+MCMC, the K-best, and the IW-DDS for Tic-Tac-Toe

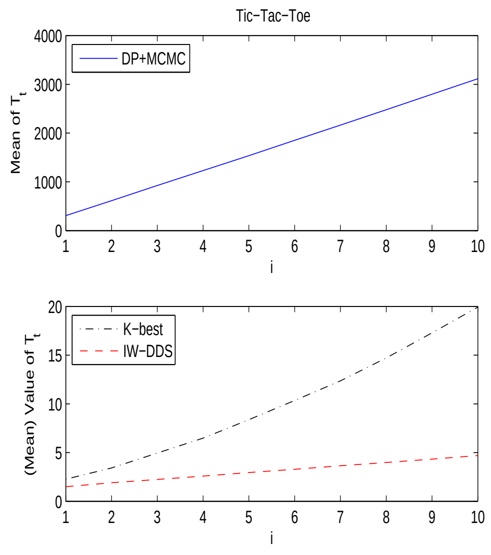

Figure 10: Plot of the Total Running Time of the DP+MCMC, the K-best, and the IW-DDS for Tic-Tac-Toe

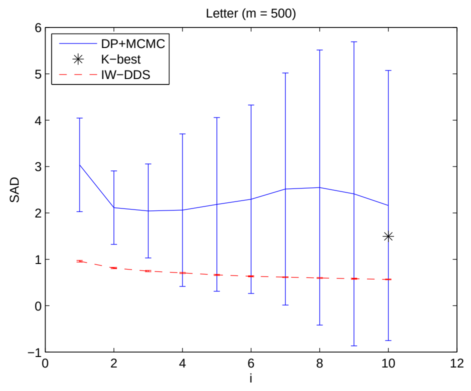

Figure 11: Plot of the SAD Performance of the DP+MCMC, the K-best, and the IW-DDS for Letter ($m = 500$)

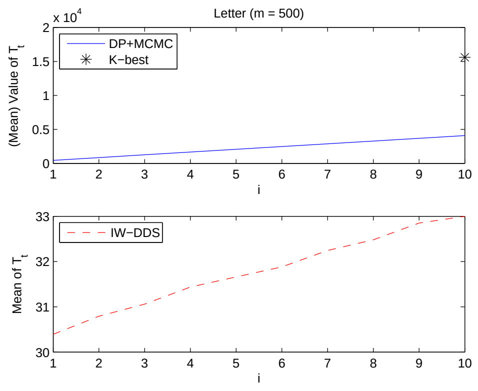

Figure 12: Plot of the Total Running Time of the DP+MCMC, the K-best, and the IW-DDS for Letter ($m = 500$)

Therefore, the learning performance of the IW-DDS is also significantly better than that of the other two methods for the data case Letter with $m = 500$.

The experimental results for the other three data cases Wine, Child with $m = 500$, and German are represented similarly in the supplementary material. The same conclusion about the learning performance can be clearly drawn by examining the figures shown in the supplementary material.

4.3 Learning Performance on Non-modular Features

In Sections 4.1 and 4.2 we did not provide experimental results on the learning performance on 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 approximate method cannot be precisely measured. We did not do so in Section 4.1 because the current PO-MCMC tool (BEANDisco) only supports the estimation of the posterior of an edge feature so that the comparison of our method and the PO-MCMC can only be made for the edge feature. (Thus, we did not make the comparison for the path feature, which is one particular non-modular feature, though the DP algorithm of Parviainen and Koivisto, 2011 can compute the exact posterior of a path feature $p_{\prec}(f|D)$.) Our idea is that by showing that our algorithms have significantly better performance in computing fundamental structural features (directed edge features), 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 that our algorithms will also be superior in computing other complicated structural 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). Because n is small, by enumerating all the DAGs, we were able to compute $p_{\not\prec}(f|D)$, the true posterior probability for any interesting non-modular feature f. For demonstration, we investigated the following five interesting non-modular features. Feature f_1, a directed path feature from node x to node y, denoted by $x \sim> y$, represents the situation that variable x eventually influences variable y. Feature f_2, a limited-length directed path feature $x \sim> y$ that has its path length no more than 2, represents that variable x can influence variable y via at most one intermediate variable. Feature f_3, a combined path feature $x \sim> y \sim> z$, can be interpreted as the situation that variable x eventually influences variable y which in turn eventually influences variable z. Feature f_4, a combined path feature $y <\sim x \sim> z$ with $y \neq z$, means that variable x eventually influences both variable y and variable z. Feature f_5, a combined path feature $y <\sim x \not\sim> z$ with $x \neq z$, represents that variable x eventually influences variable y but not variable z. Please see Figures 13 to 18 for the example features in the data set Coronary. Then we compared the SAD performance on the (directed) edge feature with the corresponding SAD performance [6] on each feature f_j ($j \in \{1, 2, 3, 4, 5\}$) from the DP+MCMC, the K-best, and the IW-DDS. The experimental results on both data sets show that if the SAD of the IW-DDS is significantly smaller than the SAD of the competing method (the DP+MCMC or the K-best) for the edge feature, then the SAD of the IW-DDS will also be significantly smaller than the SAD of the competing method for each investigated non-modular feature f_j using the same set of DAG samples. Thus, our expectation is supported by the experiments. The detailed experimental results are as follows.

The following is our experimental design for the data set Coronary with $n = 6$ and $m = 1841$. For the IW-DDS, we tried the sample size $N_o = 2{,}500 \cdot i$, where $i \in \{1, 2, \ldots, 20\}$. For each i, we

6. More specifically, SAD = $(\sum_{xy} |p_{\not\prec}(x \sim> y|D) - \hat{p}_{\not\prec}(x \sim> y|D)|)$ for the path feature $x \sim> y$; SAD = $(\sum_{xy} |p_{\not\prec}(x \sim> y|D) - \hat{p}_{\not\prec}(x \sim> y|D)|)$ for the path feature $x \sim> y$ whose length is no more than 2; SAD = $(\sum_{xyz} |p_{\not\prec}(x \sim> y \sim> z|D) - \hat{p}_{\not\prec}(x \sim> y \sim> z|D)|)$ for the combined feature $x \sim> y \sim> z$; SAD = $(\sum_{xyz, y\neq z} |p_{\not\prec}(y <\sim x \sim> z|D) - \hat{p}_{\not\prec}(y <\sim x \sim> z|D)|)$ for the combined feature $y <\sim x \sim> z$ with $y \neq z$; SAD = $(\sum_{xyz, x\neq z} |p_{\not\prec}(y <\sim x \not\sim> z|D) - \hat{p}_{\not\prec}(y <\sim x \not\sim> z|D)|)$ for the combined feature $y <\sim x \not\sim> z$ with $x \neq z$.

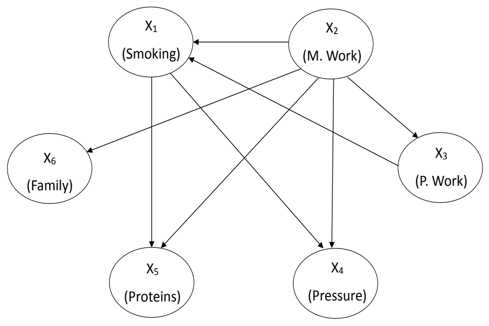

Figure 13: Example Edges in Coronary

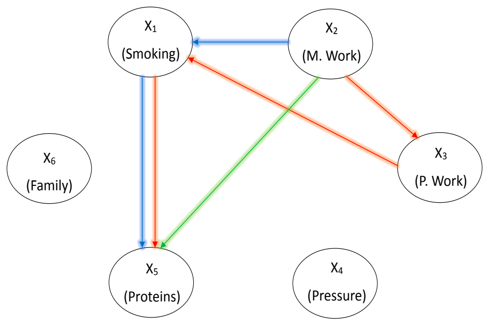

Figure 14: Example f_1 Feature in Coronary

Figure 13 shows every (directed) edge feature e with its true posterior $p_{\not\prec}(e|D) > 0.5$. Figure 14 shows the directed path feature $X_2 \sim> X_5$, which has the largest true posterior 0.8348 among all the possible f_1 features. The directed path feature $X_2 \sim> X_5$ includes many possible ways going from X_2 to X_5. For clarity, only three possible ways that have every constituent edge with true posterior > 0.5 are shown in Figure 14. These three possible ways are $X_2 \to X_5$, $X_2 \to X_1 \to X_5$, and $X_2 \to X_3 \to X_1 \to X_5$. The figure is best viewed in color.

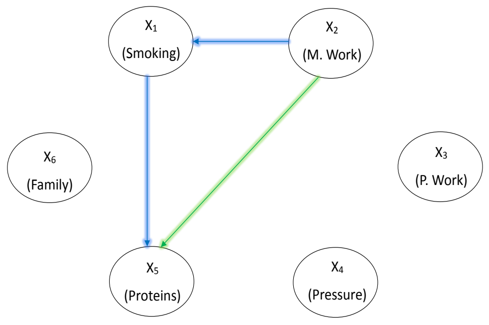

Figure 15: Example f_2 Feature in Coronary

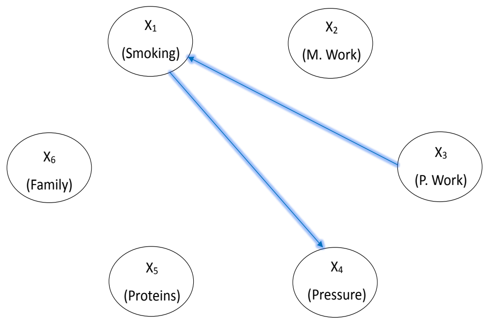

Figure 16: Example f_3 Feature in Coronary

Figure 15 shows the limited-length directed path feature $X_2 \sim> X_5$, which has the largest true posterior 0.8348 among all the possible f_2 features. The limited-length directed path feature $X_2 \sim> X_5$ includes many possible ways going from X_2 to X_5. For clarity, only two possible ways that have every constituent edge with true posterior > 0.5 are shown in Figure 15. These two possible ways are $X_2 \to X_5$, and $X_2 \to X_1 \to X_5$. Figure 16 shows the combined path feature $X_3 \sim> X_1 \sim> X_4$, which has the largest true posterior 0.5044 among all the possible f_3 features. The combined path feature $X_3 \sim> X_1 \sim> X_4$ includes many possible ways going from X_3 via X_1 to X_4. For clarity, only one possible way ($X_3 \to X_1 \to X_4$) that has every constituent edge with true posterior > 0.5 is shown in Figure 16.

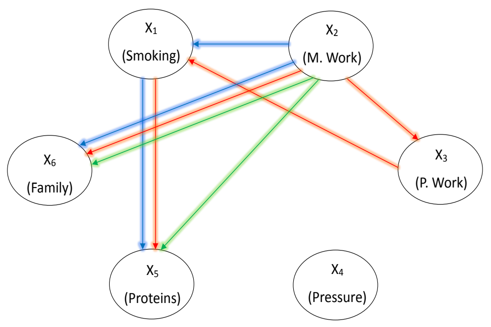

Figure 17: Example f_4 Feature in Coronary

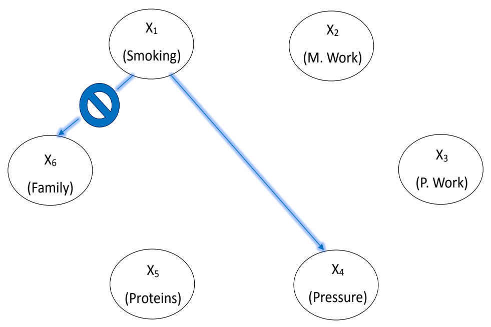

Figure 18: Example f_5 Feature in Coronary

Figure 17 shows the combined path feature $X_5 <\sim X_2 \sim> X_6$, which has the largest true posterior 0.6020 among all the possible f_4 features. The combined path feature $X_5 <\sim X_2 \sim> X_6$ includes many possible ways. For clarity, only three possible ways that have every constituent edge with true posterior > 0.5 are shown in Figure 17. These three possible ways are $X_6 \leftarrow X_2 \rightarrow X_5$, $X_6 \leftarrow X_2 \rightarrow X_1 \rightarrow X_5$, and $X_6 \leftarrow X_2 \rightarrow X_3 \rightarrow X_1 \rightarrow X_5$. The figure is best viewed in color. Figure 18 shows the combined path feature $X_4 <\sim X_1 \not\sim> X_6$, which has the largest true posterior 0.5139 among all the possible f_5 features. The combined path feature $X_4 <\sim X_1 \not\sim> X_6$ excludes any possible way going from X_1 to X_6 but includes many possible ways going from X_1 to X_4. For clarity, only one possible way ($X_1 \rightarrow X_4$) that has every constituent edge with true posterior > 0.5 is shown in Figure 18.

independently ran the IW-DDS 20 times to get the sample mean and the sample standard deviation of SAD for the (directed) edge feature and the five non-modular features (f_1, f_2, f_3, f_4, and f_5). For the DP+MCMC, we ran 25,000 $\cdot i$ MCMC iterations, where $i \in \{1, 2, \ldots, 20\}$. For each i, we discarded the first 12,500 $\cdot i$ MCMC iterations for "burn-in" and set the thinning parameter to be 5, so that 2,500 $\cdot i$ DAGs got sampled. For each i, we independently ran the MCMC 20 times to get the sample mean and the sample standard deviation of SAD for the edge feature, f_1, f_2, f_3, f_4, and f_5. For the K-best, we ran the K-best program with $K = 10 \cdot i$, where $i \in \{1, 2, \ldots, 20\}$. For each i, we ran the K-best just once to get SAD for the edge feature, f_1, f_2, f_3, f_4, and f_5, because there is no randomness in the outcome of the K-best algorithm. [7]

The experimental results for the data set Coronary are demonstrated in Figures 19 to 24. Figure 19 shows the SAD performance of the three methods with each i for the edge feature, where an error bar represents one sample standard deviation across 20 runs for the DP+MCMC or the IW-DDS at each i. Correspondingly, Figures 20 to 24 show the SAD performance of the three methods with each i for the five investigated non-modular features (f_1, f_2, f_3, f_4, and f_5) respectively. From Figure 19 and each of Figures 20 to 24, one can clearly see that if the SAD of the IW-DDS is significantly smaller than the SAD of the competing method (the DP+MCMC or the K-best) for the edge feature, then the SAD of the IW-DDS will also be significantly smaller than the SAD of the competing method for each of the five investigated non-modular features. More

7. The purpose of the experimental setting for the DP+MCMC and the K-best is merely to testify the claimed "if-then" conditional statement: if the SAD performance from the IW-DDS is significantly better than the SAD performance from the competing method for an edge feature, then the SAD performance from the IW-DDS will also be significantly better than that from the competing method for each investigated non-modular feature using the same set of DAG samples.

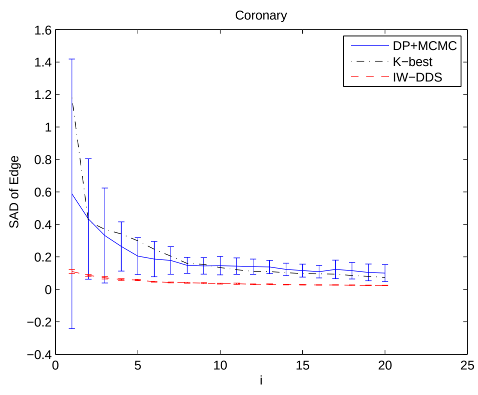

Figure 19: SAD of the Learned Edge Features for Coronary

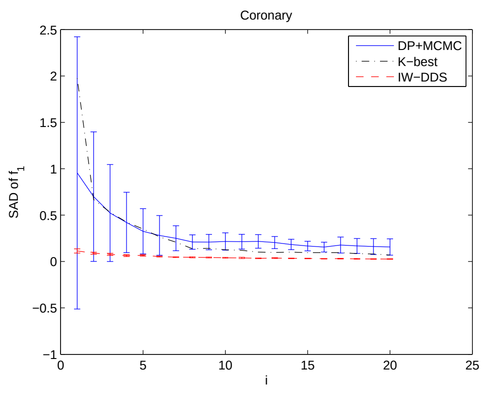

Figure 20: SAD of the Learned f_1 Features for Coronary

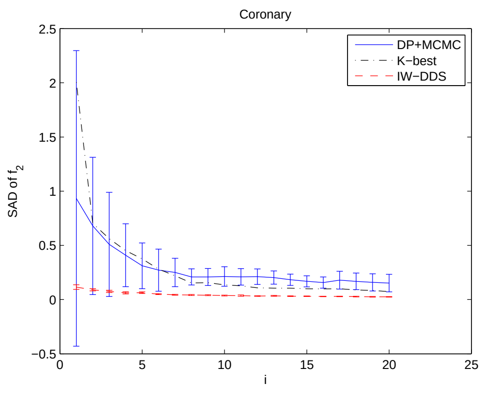

Figure 21: SAD of the Learned f_2 Features for Coronary

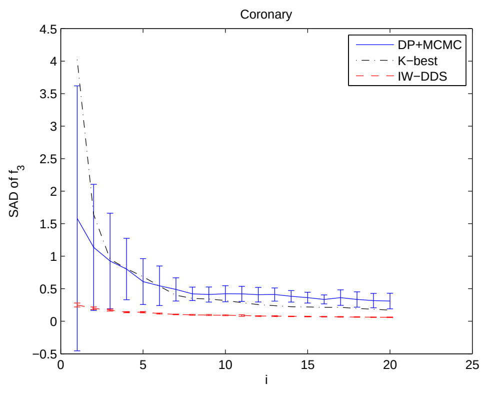

Figure 22: SAD of the Learned f_3 Features for Coronary

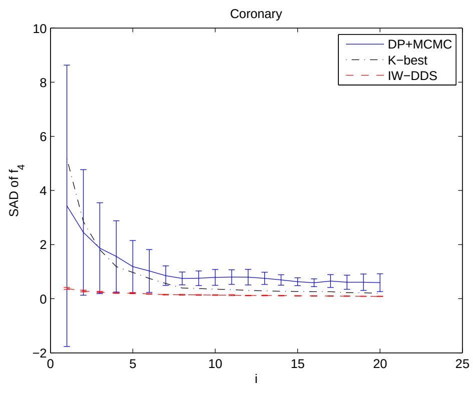

Figure 23: SAD of the Learned f_4 Features for Coronary

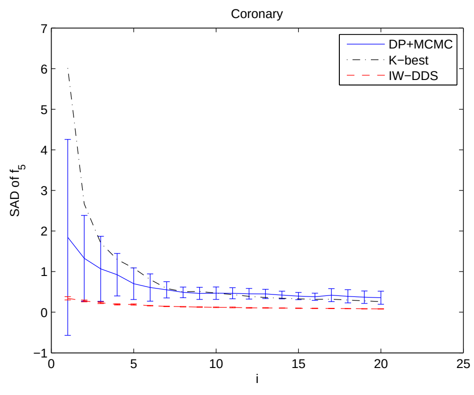

Figure 24: SAD of the Learned f_5 Features for Coronary

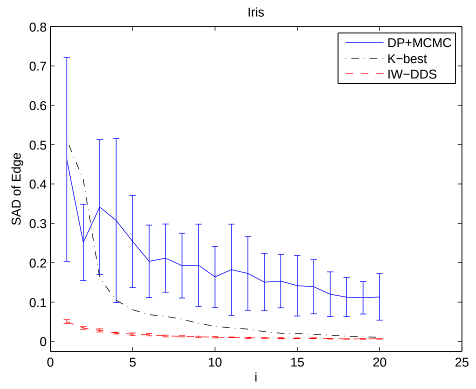

Figure 25: SAD of the Learned Edge Features for Iris

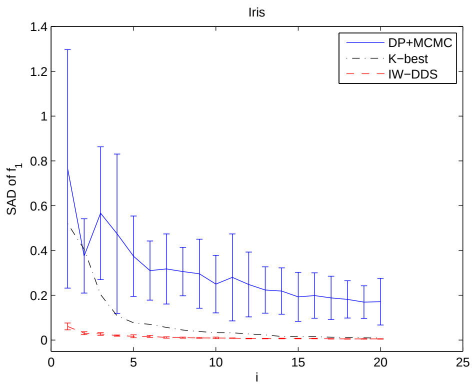

Figure 26: SAD of the Learned f_1 Features for Iris

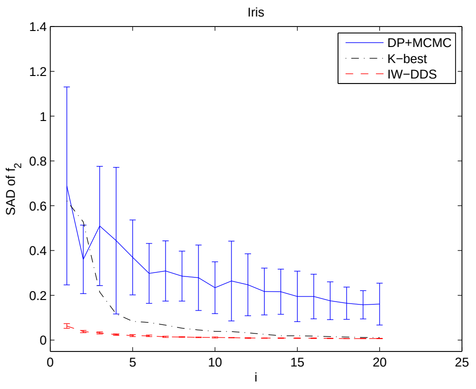

Figure 27: SAD of the Learned f_2 Features for Iris

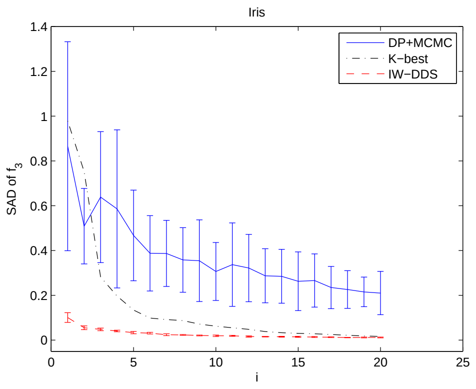

Figure 28: SAD of the Learned f_3 Features for Iris

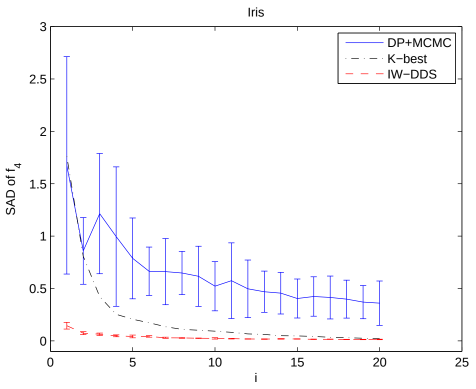

Figure 29: SAD of the Learned f_4 Features for Iris

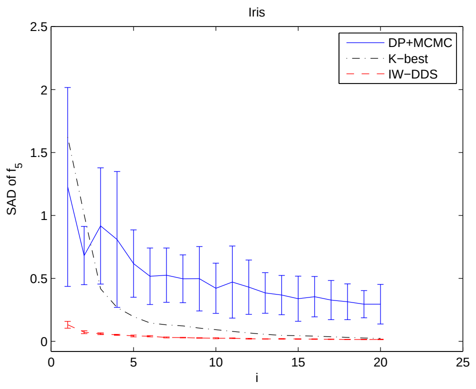

Figure 30: SAD of the Learned f_5 Features for Iris

specifically, comparing with the DP+MCMC, at each $i \in \{1, 2, \ldots 20\}$, for the edge feature, the real mean of SAD from the IW-DDS is significantly smaller than that from the DP+MCMC with the p-value < 0.01 from the two-sample t test with unequal variances. Consistently, at each $i \in \{1, 2, \ldots 20\}$, for each investigated non-modular feature f_j, the real mean of SAD from the IW-DDS is also significantly smaller than that from the DP+MCMC with the p-value < 0.01 from the same t test. Comparing with the K-best, at each $i \in \{1, 2, \ldots 20\}$, for the edge feature, the real mean of SAD from the IW-DDS is significantly smaller than the SAD from the K-best with the p-value $< 1 \times 10^{-20}$ from the one-sample t test. Consistently, at each $i \in \{1, 2, \ldots 20\}$, for each investigated non-modular feature f_j, the real mean of SAD from the IW-DDS is also significantly smaller than the SAD from the K-best with the p-value $< 1 \times 10^{-20}$ from the same t test.

We also performed the same kind of experiments for the data set Iris with $n = 5$ and $m = 150$. The results are demonstrated in Figures 25 to 30. Comparing with the DP+MCMC, at each $i \in \{1, 2, \ldots 20\}$, just as the comparison result for the edge feature, for each investigated f_j, the real mean of SAD from the IW-DDS is significantly smaller than that from the DP+MCMC with the p-value $< 1 \times 10^{-5}$ from the two-sample t test with unequal variances. Comparing with the K-best, at each $i \in \{1, 2, \ldots 20\}$, just as the comparison result for the edge feature, for each investigated f_j, the real mean of SAD from the IW-DDS is also significantly smaller than the SAD from the K-best with the p-value $< 1 \times 10^{-9}$ from the one-sample t test. Thus, a conclusion similar to that from Coronary can be drawn: if the SAD of the IW-DDS is significantly smaller than the SAD of the competing method for the edge feature, then the SAD of the IW-DDS will also be significantly smaller than the SAD of the competing method for each of the five investigated non-modular features.

4.4 Performance Guarantee for the DDS Algorithm

To testify the quality guarantee from Corollary 4 (iv) for the estimator based on the DDS algorithm, we performed experiments based on two data cases (Letter with $m = 100$ and Tic-Tac-Toe), which have relatively large $\hat{\mu}$(SAD) or $\hat{\mu}$(MAD) ($=$ $\hat{\mu}$(SAD) $/(n(n-1))$) from the DDS algorithm shown in Table 1. Based on hypothesis testing, we can conclude with very strong evidence that the performance guarantee for our estimator holds for both data cases. The details of the experiments are as follows.

For the first set of experiments, we choose the data case Letter with $m = 100$, which has the largest $\hat{\mu}$(SAD) ($= 0.2948$) from the DDS among all the 33 data cases shown in Table 1. We first consider the setting of the parameters specified as $\epsilon = 0.02$ and $\delta = 0.05$, which serves as our performance requirement. By setting the DAG sample size $N_o = \lceil (\ln(2/\delta))/(2\epsilon^2) \rceil = 4612$, we intend to show that the estimator $\hat{p}_{\prec}(f|D)$ coming from our DDS has the performance guarantee such that the Hoeffding inequality $P(|\hat{p}_{\prec}(f|D) - p_{\prec}(f|D)| \geq \epsilon) \leq \delta$ holds. Each directed edge feature f is investigated here because the posterior of each edge $p_{\prec}(f|D)$ can be easily obtained by the DP algorithm of Koivisto (2006). For each edge f, we call the event of $|\hat{p}_{\prec}(f|D) - p_{\prec}(f|D)| \geq \epsilon$ as the event of violation (of the pre-specified estimation error bound) in the learning of f. Define the indicator variable W for the event of violation in the learning of f. Thus, W is a Bernoulli random variable with the success probability $p_{vio} = P(|\hat{p}_{\prec}(f|D) - p_{\prec}(f|D)| \geq \epsilon)$. We independently repeat the DDS algorithm (with the same N_o) $R = 400$ times and use the average of W as the estimator $\hat{p}_{vio}$ for each edge. Note that the mean of $\hat{p}_{vio}$ is p_{vio} and the variance of $\hat{p}_{vio}$ is $p_{vio}(1 - p_{vio})/R$ because $R\hat{p}_{vio}$ has a binomial distribution with the trial number R and the success probability p_{vio}. Because we expect that p_{vio} will be small so that $p_{vio}(1 - p_{vio})$ will be large as compared with p_{vio}, we choose large $R = 400$ to make the variance of $\hat{p}_{vio}$ relatively small with respect to the mean of $\hat{p}_{vio}$. Figure 31 shows the histogram of $\hat{p}_{vio}$ for each of all the $n(n-1) = 272$ directed edges. For each of the 272 edges, it can be clearly seen that the corresponding $\hat{p}_{vio}$ is much smaller than $\delta = 0.05$ marked by the vertical bar. For 240 out of the 272 edges, the corresponding $\hat{p}_{vio}$'s are exactly equal to 0. Even for the largest $\hat{p}_{vio} = 0.015$, corresponding to 6 successes among 400 trials, we can use the one-sided hypothesis testing to reject the null hypothesis that $p_{vio} \geq 0.05$ and to conclude that $p_{vio} < 0.05$ with the p-value less than 2×10^{-4}. Therefore, the Hoeffding inequality holds for the learning of each edge in this parameter setting.

Next, we consider another setting of the parameters with $\epsilon = 0.01$ and $\delta = 0.02$, which has a more demanding performance requirement. By setting the DAG sample size $N_o = \lceil (\ln(2/\delta))/(2\epsilon^2) \rceil$ $= 23026$, we want to show that the estimator $\hat{p}_{\prec}(f|D)$ coming from our DDS has the performance guarantee satisfying the Hoeffding inequality. With the same logic, we independently repeat the DDS algorithm (with the same N_o) $R = 1250$ times and use the average of W as the estimator $\hat{p}_{vio}$ for each edge. (Here we choose even larger R because we expect that p_{vio} will be smaller in this parameter setting.) Figure 32 shows the histogram of $\hat{p}_{vio}$ for each of all the 272 directed edges. For each edge, it can be clearly seen that the corresponding $\hat{p}_{vio}$ is much smaller than $\delta = 0.02$. Even for the largest $\hat{p}_{vio} = 0.004$, corresponding to 5 successes among 1250 trials, we can use the one-sided hypothesis testing to reject the null hypothesis that $p_{vio} \geq 0.02$ and to conclude that p_{vio} < 0.02 with the p-value less than 2×10^{-6}. Therefore, the Hoeffding inequality also holds in this parameter setting.

For the second set of experiments, we choose the data case Tic-Tac-Toe, which has the largest $\hat{\mu}$(MAD) (= $\hat{\mu}$(SAD) $/(n(n-1)) = 0.1547/90$) from the DDS among all the 33 data cases shown in Table 1. The same kind of experiments are performed for this data case. For the parameter setting with $\epsilon = 0.02$ and $\delta = 0.05$, the corresponding result is shown in Figure 33. For each of the 90 edges, the corresponding $\hat{p}_{vio}$ is clearly much smaller than $\delta = 0.05$. Even for the largest $\hat{p}_{vio} = 0.0125$, corresponding to 5 successes among 400 trials, we can use the one-sided hypothesis testing to conclude that $p_{vio} < 0.05$ with the p-value less than 6×10^{-5}. For the parameter setting with $\epsilon = 0.01$ and $\delta = 0.02$, the corresponding result is shown in Figure 34. For each edge, it can be clearly seen that the corresponding $\hat{p}_{vio}$ is much smaller than $\delta = 0.02$. Even for the largest $\hat{p}_{vio}$ $= 0.0056$, corresponding to 7 successes among 1250 trials, we can use the one-sided hypothesis testing to conclude that $p_{vio} < 0.02$ with the p-value less than 3×10^{-5}. Thus, the Hoeffding inequality also holds in the set of experiments for this data case.

Finally, for the data case Tic-Tac-Toe, we fix $\epsilon = 0.02$ but increase N_o from 2,000 to 10,000 by an increment of 1,000 each time. For each N_o, we plot the Hoeffding bound $2e^{-2N_o\epsilon^2}$ for the probability of violation $p_{vio} = P(|\hat{p}_{\prec}(f|D) - p_{\prec}(f|D)| \geq \epsilon)$ in Figure 35. (Note that the Hoeffding bound decreases at an exponential rate as N_o increases.) Then for each N_o, we also plot both the maximum and the mean of all the $n(n-1)$ $\hat{p}_{vio}$'s in Figure 35. Again $\hat{p}_{vio}$ for each edge is the average of W by independently running the DDS algorithm (with the same N_o) R times. We set $R = \max\{400, \lceil 10/(e^{-2N_o\epsilon^2}) \rceil\}$ and use larger R for larger N_o because we expect that $\hat{p}_{vio}$ will be smaller for larger N_o. From Figure 35, we can clearly see that $\hat{p}^*_{vio}$, the maximum of all the $n(n-1)$ $\hat{p}_{vio}$'s, is always far below the Hoeffding bound for each N_o. Furthermore, for each N_o we can use the one-sided hypothesis testing to reject the null hypothesis that $p_{vio} \geq 2e^{-2N_o\epsilon^2}$ and to conclude that $p_{vio} < 2e^{-2N_o\epsilon^2}$ with the p-value less than 3×10^{-4}. Therefore, the Hoeffding inequality holds for each N_o.

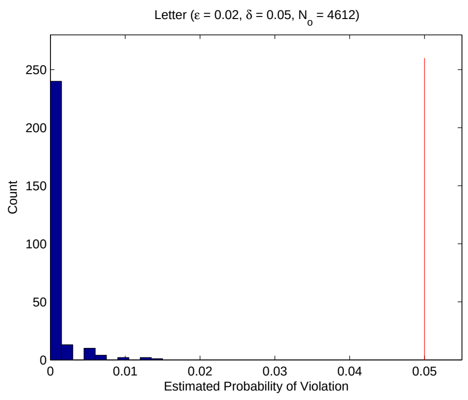

Figure 31: Histogram of Estimated Probabilities of Violation in Edge Learning for Letter ($m = 100$) with $\epsilon = 0.02$, $\delta = 0.05$, and $N_o = 4612$

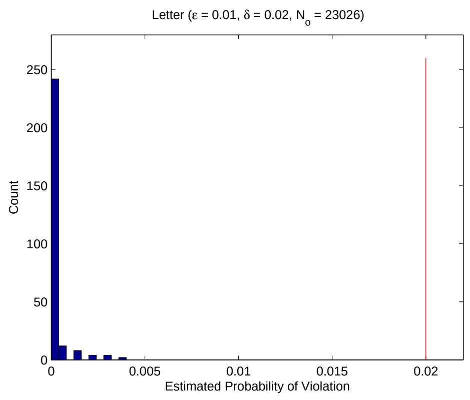

Figure 32: Histogram of Estimated Probabilities of Violation in Edge Learning for Letter ($m = 100$) with $\epsilon = 0.01$, $\delta = 0.02$, and $N_o = 23026$

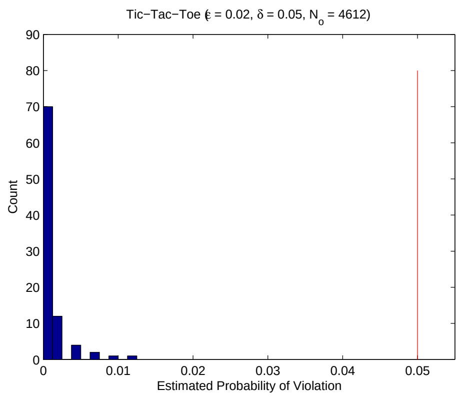

Figure 33: Histogram of Estimated Probabilities of Violation in Edge Learning for Tic-Tac-Toe with $\epsilon = 0.02$, $\delta = 0.05$, and $N_o = 4612$

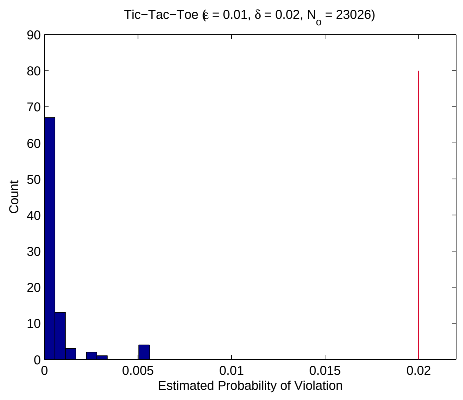

Figure 34: Histogram of Estimated Probabilities of Violation in Edge Learning for Tic-Tac-Toe with $\epsilon = 0.01$, $\delta = 0.02$, and $N_o = 23026$

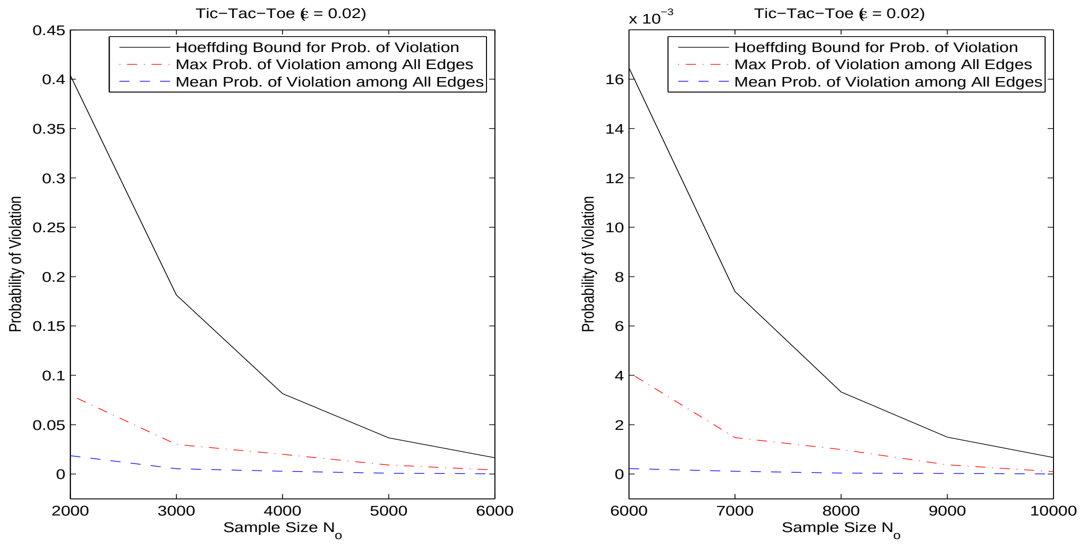

Figure 35: Plot of Probability of Violation versus N_o for Tic-Tac-Toe with $\epsilon = 0.02$

5. Conclusion

We develop new algorithms for efficiently sampling Bayesian network structures (DAGs). The sampled DAGs can then be used to build estimators for the posteriors of any features of interests. Theoretically we show that our estimators have several desirable properties. For example, unlike the existing MCMC algorithms, the estimators based on the DDS algorithm satisfy the Hoeffding bound and therefore enjoy the quality guarantee of the estimation with the given number of samples. Empirically we show that our estimators considerably outperform the previous state-of-the-art estimators with or without assuming the order-modular prior.

Our algorithms are capable of estimating the posteriors of arbitrary (non-modular) features (the DDS under the order-modular prior, and the IW-DDS under the structure-modular prior); while the exact algorithms are available for computing modular features under the order-modular prior with time $O(n^{k+1}C(m) + kn2^n)$ and space $O(n2^n)$ (Koivisto and Sood, 2004; Koivisto, 2006); computing path features under the order-modular prior with time $O(n^{k+1}C(m) + n3^n)$ and space $O(3^n)$ (Parviainen and Koivisto, 2011); and computing modular features under the structure-modular prior with time $O(n^{k+1}C(m) + kn2^n + 3^n)$ and space $O(n2^n)$ (Tian and He, 2009). The bottleneck of our algorithms is their first computation step, the DP algorithm of Koivisto and Sood (2004) whose space cost is $O(n2^n)$. Therefore, the application of our algorithms is limited to the data sets on which the DP algorithm of Koivisto and Sood (2004) is able to run—up to around 25 variables in current desktops, while a parallel implementation of the DP algorithm has been demonstrated on a data set with 33 variables using a cluster including totally 2,048 processors and 8,192 GB memory (Chen et al., 2014).

Appendix A. Proofs of Propositions, Theorems, and Corollary

This appendix provides the proofs of the propositions, theorems, and corollary in the paper.

A.1 Proof of Proposition 1

We first prove a lemma for Proposition 1.

Let an order $\prec$ be represented as $(\sigma_1, \dots, \sigma_n)$, where σ_i is the ith element in the order.

Lemma 6 *The probability that the last $n-k+1$ elements along the order are $\sigma_k, \sigma_{k+1}, \dots, \sigma_n$ respectively is given by*

$$p(\sigma_k, \sigma_{k+1}, \dots, \sigma_n, D) = L'(U^{\prec}_{\sigma_k}) \prod_{i=k}^{n} \alpha'_{\sigma_i}(U^{\prec}_{\sigma_i}),$$

where $U^{\prec}_{\sigma_i} = V - \{\sigma_i, \sigma_{i+1}, \dots, \sigma_n\}$.

Proof

$$\begin{aligned}
&p(\sigma_k, \sigma_{k+1}, \dots, \sigma_n, D) \\
&= \sum_{\prec' \in \mathcal{L}(U^{\prec}_{\sigma_k})} p(\prec', \sigma_k, \sigma_{k+1}, \dots, \sigma_n, D) \\
&= \prod_{i=k}^{n} \alpha'_{\sigma_i}(U^{\prec}_{\sigma_i}) \sum_{\prec' \in \mathcal{L}(U^{\prec}_{\sigma_k})} \prod_{i \in U^{\prec}_{\sigma_k}} \alpha'_i(U^{\prec'}_i) \quad \text{(from Eq. 13)} \\
&= L'(U^{\prec}_{\sigma_k}) \prod_{i=k}^{n} \alpha'_{\sigma_i}(U^{\prec}_{\sigma_i}) \quad \text{(from Eq. 9)}.
\end{aligned}$$

■

Proposition 1 can be directly proved by the conclusion of Lemma 6 according to the definition of the conditional probability.

A.2 Proof of Proposition 2

Proof

From Proposition 1 and $L'(U^{\prec}_{\sigma_1} = \emptyset) = 1$, we have

$$\begin{aligned}
&\prod_{k=1}^{n} p(\sigma_k | \sigma_{k+1}, \dots, \sigma_n, D) \\
&= \frac{\prod_{i=1}^{n} \alpha'_{\sigma_i}(U^{\prec}_{\sigma_i})}{L'(V)} \\
&= \frac{p(\prec, D)}{p_{\prec}(D)} \quad \text{(from Eq. 13 and Eq. 14)} \\
&= p(\prec | D),
\end{aligned}$$

which proves Eq. (20). ■

A.3 Proof of Theorem 3

Proof

First, we show that each DAG G sampled according to our DDS algorithm has its pmf $p_s(G)$ equal to the exact posterior $p_{\prec}(G|D)$ by assuming the order-modular prior.

On one hand, from the following derivation, we can get the exact form for $p_{\prec}(G|D)$:

$$\begin{aligned} & p_{\prec}(f|D) \\ &= \sum_{\prec} p(\prec|D) p(f|\prec, D) \\ &= \sum_{\prec} p(\prec|D) \sum_{G \subseteq \prec} f(G) p(G|\prec, D) \\ &= \sum_{\prec} \sum_{G \subseteq \prec} f(G) p(\prec, G|D) \\ &= \sum_{G} f(G) \sum_{\prec \text{s.t.} G \subseteq \prec} p(\prec, G|D). \end{aligned} \tag{28}$$

Thus, for each possible DAG G_i, by setting $f(G)$ to be the indicator function $I[G = G_i]$ and then relating Eq. (28) with Eq. (4), we know that $p_{\prec}(G_i|D) = \sum_{\prec \text{s.t.} G_i \subseteq \prec} p(\prec, G_i|D)$ for each G_i.

On the other hand, the event that a DAG G gets sampled according to our DDS algorithm occurs if and only if one of the orders that G is consistent with gets sampled in Step 2 of the DDS algorithm and then G gets sampled from the sampled order in Step 3 of the DDS algorithm. Therefore, based on the total probability formula, $p_s(G) = \sum_{\prec \text{s.t.} G \subseteq \prec} p(\prec|D) p(G|\prec, D) = \sum_{\prec \text{s.t.} G \subseteq \prec} p(\prec, G|D)$ $= p_{\prec}(G|D)$.

Second, because N_o orders are sampled independently and each DAG per sampled order is sampled independently, $p_s(G_1, G_2, \ldots, G_{N_o}|D) = \prod_{i=1}^{N_o} [\sum_{\prec \text{s.t.} G \subseteq \prec} p(\prec|D) p(G_i|\prec, D)] = \prod_{i=1}^{N_o}$ $p_{\prec}(G_i|D)$.

Therefore, Theorem 3 is proved.

■

A.4 Proof of Corollary 4

Proof

For each G_i in the DAG set $\{G_1, G_2, \ldots, G_{N_o}\}$ sampled from the DDS algorithm, $f(G_i) \in \{0, 1\}$. Because $G_1, G_2, \ldots, G_{N_o}$ are iid with pmf $p_{\prec}(G|D)$ (by Theorem 3), $f(G_1), f(G_2), \ldots,$ $f(G_{N_o})$ are iid with Bernoulli pmf.

For each G_i, the following is true for $E(f(G_i))$, the expectation of $f(G_i)$:

$$\begin{aligned} & E(f(G_i)) \\ &= \sum_{G} f(G) p_{\prec}(G|D) \\ &= p_{\prec}(f|D). \end{aligned}$$

Thus, $f(G_1), f(G_2), \ldots, f(G_{N_o})$ are iid with Bernoulli($p_{\prec}(f|D)$). In other words, $f(G_1),$ $f(G_2), \ldots, f(G_{N_o})$ are independent; and for each G_i, $P(f(G_i) = 1) = p_{\prec}(f|D)$ and $P(f(G_i) = 0) = 1 - p_{\prec}(f|D)$.

(i) Proof that $\hat{p}_{\prec}(f|D)$ is an unbiased estimator of $p_{\prec}(f|D)$, that is, $E(\hat{p}_{\prec}(f|D)) = p_{\prec}(f|D)$.

$$\begin{aligned}
&E\left(\hat{p}_{\prec}(f|D)\right)\\
&= E\left(\frac{1}{N_o}\sum_{i=1}^{N_o} f(G_i)\right)\\
&= \frac{1}{N_o}\sum_{i=1}^{N_o} E\left(f(G_i)\right)\\
&= \frac{1}{N_o} N_o p_{\prec}(f|D)\\
&= p_{\prec}(f|D).
\end{aligned}$$

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

Because $f(G_1), f(G_2), \ldots, f(G_{N_o})$ are iid with $E(f(G_i)) = p_{\prec}(f|D) < \infty$ and

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

based on the strong law of large numbers (Theorem 5.5.9 of Casella and Berger, 2002), $\hat{p}_{\prec}(f|D)$ converges almost surely to $p_{\prec}(f|D)$ (as $N_o \to \infty$).

Note that, by Theorem 2.5.1 of Athreya and Lahiri (2006), the property that $\hat{p}_{\prec}(f|D)$ converges almost surely to $p_{\prec}(f|D)$ implies that $\hat{p}_{\prec}(f|D)$ converges in probability to $p_{\prec}(f|D)$, that is, $\hat{p}_{\prec}(f|D)$ is a consistent estimator of $p_{\prec}(f|D)$.

(iii) Proof that if $0 < p_{\prec}(f|D) < 1$, then the random variable

$$\frac{\sqrt{N_o}(\hat{p}_{\prec}(f|D) - p_{\prec}(f|D))}{\sqrt{\hat{p}_{\prec}(f|D)(1-\hat{p}_{\prec}(f|D))}}$$

has a limiting standard normal distribution, denoted by

$$\frac{\sqrt{N_o}(\hat{p}_{\prec}(f|D) - p_{\prec}(f|D))}{\sqrt{\hat{p}_{\prec}(f|D)(1-\hat{p}_{\prec}(f|D))}} \longrightarrow^{d} \mathcal{N}(0,1).$$

Because $f(G_1), f(G_2), \ldots, f(G_{N_o})$ are iid with Bernoulli($p_{\prec}(f|D)$), for each G_i, the following is true for Var($f(G_i)$), the variance of $f(G_i)$:

$$\begin{aligned}
&\text{Var}(f(G_i))\\
&= E(f(G_i))(1 - E(f(G_i)))\\
&= p_{\prec}(f|D)(1 - p_{\prec}(f|D))\\
&< \infty.
\end{aligned}$$

Because $0 < p_{\prec}(f|D) < 1$, Var($f(G_i)$) is also strictly greater than 0.

Again, we have already known that $E(f(G_i)) = p_{\prec}(f|D) < \infty$.

Thus, by the central limit theorem (Theorem 5.5.15 of Casella and Berger, 2002),

$$\frac{\sqrt{N_o}(\hat{p}_\prec(f|D) - E(f(G)))}{\sqrt{\text{Var}(f(G))}} \longrightarrow^d \mathcal{N}(0,1),$$

that is,

$$\frac{\sqrt{N_o}(\hat{p}_\prec(f|D) - p_\prec(f|D))}{\sqrt{p_\prec(f|D)(1-p_\prec(f|D))}} \longrightarrow^d \mathcal{N}(0,1).$$

Because $\hat{p}_\prec(f|D)$ converges in probability to $p_\prec(f|D)$, denoted by $\hat{p}_\prec(f|D) \longrightarrow^p p_\prec(f|D)$, by the continuous mapping theorem (Theorem 5.5.4 of Casella and Berger, 2002),

$$\sqrt{\hat{p}_\prec(f|D)(1-\hat{p}_\prec(f|D))} \longrightarrow^p \sqrt{p_\prec(f|D)(1-p_\prec(f|D))}.$$

Finally, by Slutsky's theorem (Theorem 5.5.17 of Casella and Berger, 2002),

$$\frac{\sqrt{N_o}(\hat{p}_\prec(f|D) - p_\prec(f|D))}{\sqrt{\hat{p}_\prec(f|D)(1-\hat{p}_\prec(f|D))}} \longrightarrow^d \mathcal{N}(0,1).$$

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

Because $f(G_1),\ f(G_2), \ldots, f(G_{N_o})$ are iid with Bernoulli($p_\prec(f|D)$), the Hoeffding bound (Hoeffding, 1963; Koller and Friedman, 2009) holds:

$$P(|\hat{p}_\prec(f|D) - p_\prec(f|D)| \geq \epsilon) \leq 2e^{-2N_o\epsilon^2}.$$

This is equivalent to

$$P(|\hat{p}_\prec(f|D) - p_\prec(f|D)| < \epsilon) \geq 1 - 2e^{-2N_o\epsilon^2} \geq 1 - \delta$$

for $N_o \geq (\ln(2/\delta))/(2\epsilon^2)$.

■

A.5 Proof of Equation (21)

Proof For any $j \in \{1, \ldots, n\}$,

$$\begin{aligned}
&P((\sigma_j, U_{\sigma_j})|D) \\
&\propto P((\sigma_j, U_{\sigma_j}), D) \\
&= \sum_{\substack{(\sigma_1,\ldots,\sigma_{j-1}) \\ \in\mathcal{L}(U_{\sigma_j})}} \sum_{\substack{(\sigma_{j+1},\ldots,\sigma_n) \\ \in\mathcal{L}(V-U_{\sigma_j}-\{\sigma_j\})}} P(\sigma_1, \ldots, \sigma_{j-1}, \sigma_j, \sigma_{j+1}, \ldots, \sigma_n, D) \\
&= \sum_{\substack{(\sigma_1,\ldots,\sigma_{j-1}) \\ \in\mathcal{L}(U_{\sigma_j})}} \sum_{\substack{(\sigma_{j+1},\ldots,\sigma_n) \\ \in\mathcal{L}(V-U_{\sigma_j}-\{\sigma_j\})}} \prod_{i=1}^{n} \alpha'_{\sigma_i}(U_{\sigma_i}) \\
&= \alpha'_{\sigma_j}(U_{\sigma_j}) L'(U_{\sigma_j}) R'(V - U_{\sigma_j} - \{\sigma_j\}).
\end{aligned}$$

■

A.6 Proof of Equation (22)

Proof

$$\begin{aligned}
& p_{\prec}(G|D) \\
&= \sum_{\prec \text{s.t.} G \subseteq \prec} p(\prec, G|D) \\
&= \frac{1}{p_{\prec}(D)} \sum_{\prec \text{s.t.} G \subseteq \prec} p(\prec, G, D) \\
&= \frac{1}{p_{\prec}(D)} \sum_{\prec \text{s.t.} G \subseteq \prec} p(\prec, G) p(D|G) \\
&= \frac{1}{p_{\prec}(D)} \sum_{\prec \text{s.t.} G \subseteq \prec} \left(\prod_{i=1}^{n} q_i(U_i) \rho_i(Pa_i) \right) p(D|G) \\
&= \frac{1}{p_{\prec}(D)} \sum_{\prec \text{s.t.} G \subseteq \prec} \left(\prod_{i=1}^{n} p_i(Pa_i) \right) p(D|G) \\
&= \frac{1}{p_{\prec}(D)} \sum_{\prec \text{s.t.} G \subseteq \prec} p_{\not\prec}(G) p(D|G) \\
&= \frac{1}{p_{\prec}(D)} \cdot | \prec_G | \cdot p_{\not\prec}(G, D) \\
&= \frac{p_{\not\prec}(D)}{p_{\prec}(D)} \cdot | \prec_G | \cdot p_{\not\prec}(G|D).
\end{aligned}$$

Because both $p_{\not\prec}(D) > 0$ and $p_{\prec}(D) > 0$, $p_{\prec}(G|D) \propto | \prec_G | \cdot p_{\not\prec}(G|D)$, which also was shown by Ellis and Wong (2008).

■

A.7 Proof of Theorem 5

Proof

Let Ω denote the set of all the DAGs. Define $\mathcal{U}^+ = \{G \in \Omega : p_{\not\prec}(G, D) > 0\}$. $\mathcal{U}^+$ will equal Ω if the user chooses a prior such that $p_{\not\prec}(G) > 0$ for every DAG G (such as a uniform DAG prior $p_{\not\prec}(G) \equiv 1$). If the user, however, has some additional domain knowledge so that he or she sets some prior to exclude some DAGs a priori, $\mathcal{U}^+$ will be a proper subset of Ω. Note that $p_{\not\prec}(f|D) = p_{\not\prec}(f, D)/p_{\not\prec}(D)$, where $p_{\not\prec}(f, D) = \sum_G f(G) p_{\not\prec}(G, D) = \sum_{G \in \mathcal{U}^+} f(G) p_{\not\prec}(G, D)$, and $p_{\not\prec}(D) = p_{\not\prec}(f \equiv 1, D) = \sum_G p_{\not\prec}(G, D) = \sum_{G \in \mathcal{U}^+} p_{\not\prec}(G, D)$.

Let $I[\cdot]$ denote the indicator function. Rewrite $\hat{p}_{\not\prec}(f|D)$ in Eq. (24) as $\hat{p}_{\not\prec}(f, D)/\hat{p}_{\not\prec}(D)$, where $\hat{p}_{\not\prec}(f, D) = \sum_{G \in \mathcal{G}} f(G) p_{\not\prec}(G, D) = \sum_G f(G)\, p_{\not\prec}(G, D) I[G \in \mathcal{G}] = \sum_{G \in \mathcal{U}^+} f(G) p_{\not\prec}(G, D) I[G \in \mathcal{G}]$, and $\hat{p}_{\not\prec}(D) = \hat{p}_{\not\prec}(f \equiv 1, D) = \sum_{G \in \mathcal{G}} p_{\not\prec}(G, D) = \sum_G p_{\not\prec}(G, D) I[G \in \mathcal{G}] = \sum_{G \in \mathcal{U}^+} p_{\not\prec}(G, D) I[G \in \mathcal{G}]$.

Note that by Theorem 3, $P(G \in \mathcal{G}) = 1 - (1 - p_{\prec}(G|D))^{N_o}$, where $p_{\prec}(G|D)$ is the exact posterior of G under the order-modular prior assumption. Also note that by Eq. (22), for any $G \in \Omega$, $p_{\not\prec}(G, D) > 0$ implies $p_{\prec}(G|D) > 0$.

(i) Proof that $\hat{p}_{\not\prec}(f|D)$ is an asymptotically unbiased estimator of $p_{\not\prec}(f|D)$, that is,

$$\lim_{N_o\to\infty} E(\hat{p}_{\not\prec}(f|D)) = p_{\not\prec}(f|D). \tag{29}$$

For notational convenience, let γ denote $\hat{p}_{\not\prec}(f,D)$ and τ denote $\hat{p}_{\not\prec}(D)$. Define $g(\gamma,\tau)=\gamma/\tau$ so that $g(\gamma,\tau)$ is $\hat{p}_{\not\prec}(f|D)$.

Note that

$$\begin{aligned}
&E(\gamma)\\
&= E(\hat{p}_{\not\prec}(f,D))\\
&= E\left(\sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)I[G\in\mathcal{G}]\right)\\
&= \sum_{G\in\mathcal{U}^+} E(f(G)p_{\not\prec}(G,D)I[G\in\mathcal{G}])\\
&= \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)P(G\in\mathcal{G})\\
&= \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)(1-(1-p_{\prec}(G|D))^{N_o}).
\end{aligned}$$

Thus,

$$\begin{aligned}
&\lim_{N_o\to\infty} E(\gamma)\\
&= \lim_{N_o\to\infty}\left(\sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)(1-(1-p_{\prec}(G|D))^{N_o})\right)\\
&= \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)\lim_{N_o\to\infty}(1-(1-p_{\prec}(G|D))^{N_o})\\
&= \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)\\
&= p_{\not\prec}(f,D).
\end{aligned}$$

Similarly, by setting $f\equiv 1$, we have

$$\begin{aligned}
&E(\tau)\\
&= E(\hat{p}_{\not\prec}(D))\\
&= \sum_{G\in\mathcal{U}^+} p_{\not\prec}(G,D)(1-(1-p_{\prec}(G|D))^{N_o}),
\end{aligned}$$

and

$$\lim_{N_o\to\infty} E(\tau) = p_{\not\prec}(D).$$

Next, by Taylor's theorem (with the Lagrange form of the remainder),

$$\begin{aligned}
&g(\gamma, \tau) \\
&= g(E(\gamma), E(\tau)) + \left[\frac{\partial g(\gamma, \tau)}{\partial \gamma}\right]_{\gamma=E(\gamma),\tau=E(\tau)} (\gamma^* - E(\gamma)) \\
&\quad + \left[\frac{\partial g(\gamma, \tau)}{\partial \tau}\right]_{\gamma=E(\gamma),\tau=E(\tau)} (\tau^* - E(\tau)),
\end{aligned}$$

where $\gamma^* = E(\gamma) + \theta(\gamma - E(\gamma))$, $\tau^* = E(\tau) + \theta(\tau - E(\tau))$, and θ is a random variable such that $0 < \theta < 1$.

Examine the components separately:

$$g(E(\gamma), E(\tau)) = E(\gamma)/E(\tau);$$

$$\begin{aligned}
&\left[\frac{\partial g(\gamma, \tau)}{\partial \gamma}\right]_{\gamma=E(\gamma),\tau=E(\tau)} \\
&= \left[\tau^{-1}\right]_{\gamma=E(\gamma),\tau=E(\tau)} \\
&= (E(\tau))^{-1};
\end{aligned}$$

$$\begin{aligned}
&\left[\frac{\partial g(\gamma, \tau)}{\partial \tau}\right]_{\gamma=E(\gamma),\tau=E(\tau)} \\
&= \left[-\gamma(\tau)^{-2}\right]_{\gamma=E(\gamma),\tau=E(\tau)} \\
&= -E(\gamma)(E(\tau))^{-2}.
\end{aligned}$$

Because neither $E(\gamma)$ nor $E(\tau)$ is random,

$$\begin{aligned}
&E\left(g(\gamma, \tau)\right) \\
&= E(\gamma)/E(\tau) + (E(\tau))^{-1}E(\gamma^* - E(\gamma)) - E(\gamma)(E(\tau))^{-2}E(\tau^* - E(\tau)) \\
&= E(\gamma)/E(\tau) + (E(\tau))^{-1}E(\theta(\gamma - E(\gamma))) - E(\gamma)(E(\tau))^{-2}E(\theta(\tau - E(\tau))).
\end{aligned}$$

Consider the limit of each component separately:

$$\begin{aligned}
&\lim_{N_o \to \infty} (E(\gamma)/E(\tau)) \\
&= \left(\lim_{N_o \to \infty} E(\gamma)\right) / \left(\lim_{N_o \to \infty} E(\tau)\right) \\
&= p_{\not\prec}(f, D)/p_{\not\prec}(D) \\
&= p_{\not\prec}(f|D);
\end{aligned}$$

$$\begin{aligned}
&\lim_{N_o\to\infty}(E(\tau))^{-1}\\
&=\left(\lim_{N_o\to\infty}E(\tau)\right)^{-1}\\
&=(p_{\not\prec}(D))^{-1};
\end{aligned}$$

$$\begin{aligned}
&\lim_{N_o\to\infty}-E(\gamma)(E(\tau))^{-2}\\
&=-\left(\lim_{N_o\to\infty}E(\gamma)\right)\left(\lim_{N_o\to\infty}E(\tau)\right)^{-2}\\
&=-p_{\not\prec}(f,D)(p_{\not\prec}(D))^{-2}.
\end{aligned}$$

Note that both $(p_{\not\prec}(D))^{-1}$ and $-p_{\not\prec}(f,D)(p_{\not\prec}(D))^{-2}$ are constant real numbers.
Finally, we intend to prove the following two equalities:

$$\lim_{N_o\to\infty}E(\theta(\gamma-E(\gamma)))=0, \tag{30}$$

$$\lim_{N_o\to\infty}E(\theta(\tau-E(\tau)))=0. \tag{31}$$

Once this is done,

$$\begin{aligned}
&\lim_{N_o\to\infty}E\left(g(\gamma,\tau)\right)\\
&=\lim_{N_o\to\infty}(E(\gamma)/E(\tau))+\lim_{N_o\to\infty}(E(\tau))^{-1}\cdot 0+\lim_{N_o\to\infty}\left(-E(\gamma)(E(\tau))^{-2}\right)\cdot 0\\
&=p_{\not\prec}(f|D).
\end{aligned}$$

The whole proof of Eq. (29) will then be done.
The proof of Eq. (30) is as follows.
Based on the definition of the limit, proving Eq. (30) is equivalent to proving

$$\lim_{N_o\to\infty}|E(\theta(\gamma-E(\gamma)))|=0. \tag{32}$$

Because

$$\begin{aligned}
&|E(\theta(\gamma-E(\gamma)))|\\
&\leq E\left(|\theta(\gamma-E(\gamma))|\right)\\
&=E\left(|\theta|\cdot|\gamma-E(\gamma)|\right)\\
&\leq E\left(|\gamma-E(\gamma)|\right)\quad\text{(because } 0<\theta<1\text{)},
\end{aligned}$$

and $|E(\theta(\gamma-E(\gamma)))|\geq 0$, to prove Eq. (32), it is sufficient to prove

$$\lim_{N_o\to\infty}E\left(|\gamma-E(\gamma)|\right)=0. \tag{33}$$

Note that

$$\begin{aligned}
&E\left(|\gamma - E(\gamma)|\right) \\
&= E\left(\left|\sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)I[G\in\mathcal{G}] \;-\sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)P(G\in\mathcal{G})\right|\right) \\
&= E\left(\left|\sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)(I[G\in\mathcal{G}] - P(G\in\mathcal{G}))\right|\right) \\
&\le E\left(\sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)\left|I[G\in\mathcal{G}] - P(G\in\mathcal{G})\right|\right) \\
&= \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)E\left(|I[G\in\mathcal{G}] - P(G\in\mathcal{G})|\right) \\
&= \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)[(1-P(G\in\mathcal{G}))P(G\in\mathcal{G}) \\
&\qquad + (P(G\in\mathcal{G})-0)(1-P(G\in\mathcal{G}))] \\
&= \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)[2P(G\in\mathcal{G})(1-P(G\in\mathcal{G}))] \\
&= \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)[2(1-(1-p_{\prec}(G|D))^{N_o}) \times (1-p_{\prec}(G|D))^{N_o}].
\end{aligned}$$

Because for any $G \in \mathcal{U}^+$, $p_{\prec}(G|D) > 0$, we have

$$\begin{aligned}
&\lim_{N_o\to\infty} \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)[2(1-(1-p_{\prec}(G|D))^{N_o}) \times (1-p_{\prec}(G|D))^{N_o}] \\
&= \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)\lim_{N_o\to\infty}[2(1-(1-p_{\prec}(G|D))^{N_o}) \times (1-p_{\prec}(G|D))^{N_o}] \\
&= \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)\cdot 0 \\
&= 0.
\end{aligned}$$

Eq. (33) is proved, and the proof of Eq. (30) is done.

Setting $f \equiv 1$ in Eq. (30) leads to Eq. (31).

Thus, the whole proof of Eq. (29) is done.

(ii) Proof that $\hat{p}_{\not\prec}(f|D)$ converges almost surely to $p_{\not\prec}(f|D)$, denoted by $\hat{p}_{\not\prec}(f|D) \longrightarrow^{a.s.} p_{\not\prec}(f|D)$.

Recall that Ω denotes the set of all the DAGs, that is, $\Omega = \{G_1, G_2, \ldots, G_{W^*}\}$, where W^* is $|\Omega|$, the number of all the DAGs. Note that W^* is a finite positive integer though it is super-exponential in the number of variables n.

Let $\mathcal{F}$ be $\mathcal{P}(\Omega)$, the power set of Ω. Thus, $\mathcal{F}$ is a σ-algebra on Ω (Athreya and Lahiri, 2006). Define for any $A \in \mathcal{F}$, $\mu(A) = \sum_{G_j\in A} p_{\prec}(G_j|D)$. It is well-known that μ is a probability measure on $\mathcal{F}$ so that $(\Omega, \mathcal{F}, \mu)$ is a probability space (Athreya and Lahiri, 2006).

For each $i \geq 1$, let $\Omega_i = \Omega$, $\mathcal{F}_i = \mathcal{F}$, and $\mu_i = \mu$. Let $\Omega^\infty = \{(G^{(1)}, G^{(2)}, \ldots) : G^{(i)} \in \Omega_i, i \geq 1\}$. Let a cylinder set $\underset{\sim}{A} = A_1 \times A_2 \times \cdots \times A_k \times \Omega_{k+1} \times \Omega_{k+2} \times \cdots$, where there exists $1 \leq k < \infty$ such that $A_i \in \mathcal{F}_i$ for $1 \leq i \leq k$ and $A_i = \Omega_i$ for $i > k$. Let $\mathcal{F}^\infty = \sigma < \{\underset{\sim}{A} : \underset{\sim}{A}$ is a cylinder set $\} >$, that is, $\mathcal{F}^\infty$ is a σ-algebra generated by the set of all the $\underset{\sim}{A}$'s. $\mathcal{F}^\infty$ is a σ-algebra on Ω^∞ and is called a product σ-algebra.

Define, for each $(A_1, A_2, \ldots, A_k, \Omega_{k+1}, \Omega_{k+2}, \ldots) \in \mathcal{F}^\infty$, $\mu^\infty(A_1, A_2, \ldots, A_k, \Omega_{k+1}, \Omega_{k+2}, \ldots) = \mu_1(A_1) \times \mu_2(A_2) \times \cdots \times \mu_k(A_k)$. By Kolmogorov's consistency theorem, $(\Omega^\infty, \mathcal{F}^\infty, \mu^\infty)$ is a probability space (Athreya and Lahiri, 2006).

Let $\Omega^{\infty,0} = \{(G^{(1)}, G^{(2)}, \ldots) \in \Omega^\infty :$ there exists $G \in \mathcal{U}^+$ such that for any $i \geq 1, G^{(i)} \neq G\}$. Let $\Omega^{\infty,1} = \Omega^\infty - \Omega^{\infty,0}$. Thus, $\Omega^{\infty,1} = \{(G^{(1)}, G^{(2)}, \ldots) \in \Omega^\infty :$ for any $G \in \mathcal{U}^+,$ there exists $i \geq 1$ such that $G^{(i)} = G\}$.

Define, for each $\omega^\infty \in \Omega^\infty$,

$$\hat{p}_{\not\to}^{N_o}(\omega^\infty) = \frac{\sum_{G \in \mathcal{U}^+} f(G) p_{\not\to}(G, D) I[G \in \mathcal{G}^{N_o}(\omega^\infty)]}{\sum_{G \in \mathcal{U}^+} p_{\not\to}(G, D) I[G \in \mathcal{G}^{N_o}(\omega^\infty)]},$$

where $\mathcal{G}^{N_o}(\omega^\infty)$ is the DAG set that includes the first N_o coordinates of ω^∞. By the definition, we know that $\hat{p}_{\not\to}^{N_o}(\omega^\infty) = \hat{p}_{\not\to}(f|D)$.

For each $\omega^\infty \in \Omega^{\infty,1}$, for each $G \in \mathcal{U}^+$, let $N(G, \omega^\infty)$ be the smallest integer such that $G^{(N(G,\omega^\infty))} = G$. Let $N(\omega^\infty) = \max_{G \in \mathcal{U}^+} N(G, \omega^\infty)$. Then for each $N_o \geq N(\omega^\infty)$, for each $G \in \mathcal{U}^+$, $I[G \in \mathcal{G}^{N_o}(\omega^\infty)] = 1$.

Accordingly, for each $\omega^\infty \in \Omega^{\infty,1}$, there exists $N(\omega^\infty)$ such that for each $N_o \geq N(\omega^\infty)$,

$$\begin{aligned}
&\hat{p}_{\not\to}^{N_o}(\omega^\infty) \\
&= \frac{\sum_{G \in \mathcal{U}^+} f(G) p_{\not\to}(G, D)}{\sum_{G \in \mathcal{U}^+} p_{\not\to}(G, D)} \\
&= p_{\not\to}(f|D).
\end{aligned}$$

This implies that $\lim_{N_o \to \infty} \hat{p}_{\not\to}^{N_o}(\omega^\infty) = p_{\not\to}(f|D)$ for each $\omega^\infty \in \Omega^{\infty,1}$.

Finally, we intend to prove the following equality:

$$\mu^\infty(\Omega^{\infty,1}) = 1. \tag{34}$$

Once this is done, the whole proof that $\hat{p}_{\not\to}(f|D) \longrightarrow^{a.s.} p_{\not\to}(f|D)$ is done.

Proving Eq. (34) is equivalent to proving

$$\mu^\infty(\Omega^{\infty,0}) = 0. \tag{35}$$

For any $G \in \Omega$, let $\Omega^{\infty,0,G} = \{(G^{(1)}, G^{(2)}, \ldots) \in \Omega^\infty :$ for any $i \geq 1, G^{(i)} \neq G\}$. Thus, $\Omega^{\infty,0} = \bigcup_{G \in \mathcal{U}^+} \Omega^{\infty,0,G}$. Accordingly, $\mu^\infty(\Omega^{\infty,0}) \leq \sum_{G \in \mathcal{U}^+} \mu^\infty(\Omega^{\infty,0,G})$.

For each $j \geq 1$, let $\Omega^{\infty,0,G,j} = \{(G^{(1)}, G^{(2)}, \ldots) \in \Omega^\infty :$ for any $i \in \{1, \ldots, j\}, G^{(i)} \neq G\}$. Note that $\Omega^{\infty,0,G,1} \supseteq \Omega^{\infty,0,G,2} \supseteq \ldots \supseteq \Omega^{\infty,0,G,j-1} \supseteq \Omega^{\infty,0,G,j}$ for each $j \geq 1$. Thus, $\Omega^{\infty,0,G} = \bigcap_{j=1}^\infty \Omega^{\infty,0,G,j}$.

Finally, for any $G \in \mathcal{U}^+$,

$$\begin{aligned}
&\mu^\infty(\Omega^{\infty,0,G}) \\
&= \mu^\infty(\bigcap_{j=1}^{\infty} \Omega^{\infty,0,G,j}) \\
&= \lim_{j\to\infty} \mu^\infty(\Omega^{\infty,0,G,j}) \\
&= \lim_{j\to\infty} \mu_1(\Omega - \{G\}) \times \mu_2(\Omega - \{G\}) \times \cdots \times \mu_j(\Omega - \{G\}) \\
&= \lim_{j\to\infty} [1 - \mu(\{G\})]^j \\
&= \lim_{j\to\infty} [1 - p_{\prec}(G|D)]^j \\
&= 0.
\end{aligned}$$

Thus, $\sum_{G\in\mathcal{U}^+} \mu^\infty(\Omega^{\infty,0,G}) = 0$, so that Eq. (35) is proved. The whole proof is done.

Note that, by Theorem 2.5.1 of Athreya and Lahiri (2006), the property that $\hat{p}_{\not\prec}(f|D)$ converges almost surely to $p_{\not\prec}(f|D)$ implies that $\hat{p}_{\not\prec}(f|D)$ converges in probability to $p_{\not\prec}(f|D)$, that is, $\hat{p}_{\not\prec}(f|D)$ is a consistent estimator of $p_{\not\prec}(f|D)$.

(iii) Proof that the convergence rate of $\hat{p}_{\not\prec}(f|D)$ is $o(a^{N_o})$ for any $0 < a < 1$.

In the proof of (ii), we have shown that for each $\omega^\infty \in \Omega^{\infty,1}$, there exists $N(\omega^\infty)$ such that for each $N_o \geq N(\omega^\infty)$, $\hat{p}^{N_o}_{\not\prec}(\omega^\infty) = p_{\not\prec}(f|D)$. This means that for any $0 < a < 1$, $(a^{N_o})^{-1}[\hat{p}^{N_o}_{\not\prec}(\omega^\infty) - p_{\not\prec}(f|D)] = 0$. Thus, $\lim_{N_o\to\infty}(a^{N_o})^{-1}[\hat{p}^{N_o}_{\not\prec}(\omega^\infty) - p_{\not\prec}(f|D)] = 0$ so that the proof is done.

(iv) Proof that if the quantity $\Delta = \sum_{G\in\mathcal{G}} p_{\not\prec}(G|D)$, then $\Delta \cdot \hat{p}_{\not\prec}(f|D) \leq p_{\not\prec}(f|D) \leq \Delta \cdot \hat{p}_{\not\prec}(f|D) + 1 - \Delta$.

The proof is essentially the same as the proof of Proposition 1 of Tian et al. (2010) which proves Eq. (27), an equivalent form of Eq. (26). The direct proof of Eq. (26) is also provided in the supplementary material.

■

References

Arthur Asuncion and David Newman. UCI machine learning repository, 2007.

Krishna Athreya and Soumendra Lahiri. Measure Theory and Probability Theory. Springer, 2006.

John Binder, Daphne Koller, Stuart Russell, and Keiji Kanazawa. Adaptive probabilistic networks with hidden variables. Machine Learning, 29:213–244, 1997.

Graham Brightwell and Peter Winkler. Counting linear extensions is #P-complete. In Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing, pages 175–181, 1991.

George Casella and Roger Berger. Statistical Inference. Duxbury, 2002.

Yetian Chen, Jin Tian, Olga Nikolova, and Srinivas Aluru. A parallel algorithm for exact Bayesian structure discovery in Bayesian networks. CoRR, abs/1408.1664, 2014. URL `http://arxiv.org/abs/1408.1664`.

Gregory Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. Machine Learning, 9:309–347, 1992.

James Cussens. Bayesian network learning with cutting planes. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pages 153–160, 2011.

James Cussens and Mark Bartlett. Advances in Bayesian network learning using integer programming. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pages 182–191, 2013.

Denver Dash and Gregory Cooper. Model averaging for prediction with discrete Bayesian networks. Journal of Machine Learning Research, 5:1177–1203, 2004.

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

David Edwards. Introduction to Graphical Modelling. Springer, 2nd edition, 2000.

Byron Ellis and Wing Wong. Learning causal Bayesian network structures from experimental data. Journal of the American Statistical Association, 103:778–789, 2008.

Natalia Flerova, Emma Rollon, and Rina Dechter. Bucket and mini-bucket schemes for m best solutions over graphical models. In Graph Structures for Knowledge Representation and Reasoning, volume 7205, pages 91–118. 2012.

Nir Friedman and Daphne Koller. Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. Machine Learning, 50:95–125, 2003.

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

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

David Heckerman, Dan Geiger, and David Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning, 20:197–243, 1995.

David Heckerman, Christopher Meek, and Gregory Cooper. A Bayesian approach to causal discovery. In Computation, Causation, and Discovery, pages 141–166, 1999.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association, 58:13–30, 1963.

Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. Learning Bayesian network structure using LP relaxations. In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), pages 358–365, 2010.

Mark Jerrum, Leslie Valiant, and Vijay Vazirani. Random generation of combinatorial structures from a uniform distribution. Theoretical Computer Science, 43:169–188, 1986.

Robert Kennes and Philippe Smets. Computational aspects of the Möbius transformation. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pages 401–416, 1991.

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

Mikko Koivisto and Kismat Sood. Exact Bayesian structure discovery in Bayesian networks. Journal of Machine Learning Research, 5:549–573, 2004.

Daphne Koller and Nir Friedman. Probabilistic Graphical Models: Principles and Techniques. The MIT Press, 2009.

David Madigan and Jeremy York. Bayesian graphical models for discrete data. International Statistical Review, 63:215–232, 1995.

Brandon Malone and Changhe Yuan. Evaluating anytime algorithms for learning optimal Bayesian networks. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pages 381–390, 2013.

Brandon Malone, Changhe Yuan, and Eric Hansen. Memory-efficient dynamic programming for learning optimal Bayesian networks. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), pages 1057–1062, 2011a.

Brandon Malone, Changhe Yuan, Eric Hansen, and Susan Bridges. Improving the scalability of optimal Bayesian network learning with external-memory frontier breadth-first branch and bound search. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pages 479–488, 2011b.

Marina Meila and Tommi Jaakkola. Tractable Bayesian learning of tree belief networks. Statistics and Computing, 16:77–92, 2006.

Teppo Niinimaki and Mikko Koivisto. Annealed importance sampling for structure learning in Bayesian networks. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pages 1579–1585, 2013.

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

Pekka Parviainen and Mikko Koivisto. Bayesian structure discovery in Bayesian networks with less space. In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), pages 589–596, 2010.

Pekka Parviainen and Mikko Koivisto. Ancestor relations in the presence of unobserved variables. In Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), pages 581–596, 2011.

Judea Pearl. Causality: Models, Reasoning, and Inference. Cambridge University Press, NY, 2000.

Tomi Silander and Petri Myllymaki. 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, 2006.

Peter Spirtes, Clark Glymour, and Richard Scheines. Causation, Prediction, and Search. MIT Press, Cambridge, MA, 2nd edition, 2001.

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

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

Ioannis Tsamardinos, Laura Brown, and Constantin Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. Machine Learning, 65:31–78, 2006.

Changhe Yuan and Brandon Malone. An improved admissible heuristic for learning optimal Bayesian networks. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pages 924–933, 2012.

Changhe Yuan and Brandon Malone. Learning optimal Bayesian networks: A shortest path perspective. Journal of Artificial Intelligence Research, 48:23–65, 2013.

Changhe Yuan, Brandon Malone, and Xiaojian Wu. Learning optimal Bayesian networks using A* search. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pages 2186–2191, 2011.