Niinimäki, Teppo; Parviainen, Pekka; Koivisto, Mikko

# Structure discovery in Bayesian networks by sampling partial orders

*Published in:*
Journal of Machine Learning Research

Published: 01/04/2016

*Document Version*
Publisher's PDF, also known as Version of record

Journal of Machine Learning Research 17 (2016) 1-47 Submitted 4/15; Revised 12/15; Published 4/16

# Structure Discovery in Bayesian Networks by Sampling Partial Orders*

**Teppo Niinimäki** TEPPO.NIINIMAKI@HELSINKI.FI
*Helsinki Institute for Information Technology*
*Department of Computer Science*
*P.O. Box 68 (Gustaf Hällströminkatu 2b)*
*FI-00014 University of Helsinki, Finland*

**Pekka Parviainen** PEKKA.PARVIAINEN@AALTO.FI
*Helsinki Institute for Information Technology*
*Department of Computer Science*
*P.O. Box 15400 (Konemiehentie 2)*
*FI-00076 Aalto, Finland*

**Mikko Koivisto** MIKKO.KOIVISTO@HELSINKI.FI
*Helsinki Institute for Information Technology*
*Department of Computer Science*
*P.O. Box 68 (Gustaf Hällströminkatu 2b)*
*FI-00014 University of Helsinki, Finland*

**Editor:** Edo Airoldi

## Abstract

We present methods based on Metropolis-coupled Markov chain Monte Carlo ($MC^3$) and annealed importance sampling (AIS) for estimating the posterior distribution of Bayesian networks. The methods draw samples from an appropriate distribution of partial orders on the nodes, continued by sampling directed acyclic graphs (DAGs) conditionally on the sampled partial orders. We show that the computations needed for the sampling algorithms are feasible as long as the encountered partial orders have relatively few down-sets. While the algorithms assume suitable modularity properties of the priors, arbitrary priors can be handled by dividing the importance weight of each sampled DAG by the number of topological sorts it has—we give a practical dynamic programming algorithm to compute these numbers. Our empirical results demonstrate that the presented partial-order-based samplers are superior to previous Markov chain Monte Carlo methods, which sample DAGs either directly or via linear orders on the nodes. The results also suggest that the convergence rate of the estimators based on AIS are competitive to those of $MC^3$. Thus AIS is the preferred method, as it enables easier large-scale parallelization and, in addition, supplies good probabilistic lower bound guarantees for the marginal likelihood of the model.

**Keywords:** annealed importance sampling, directed acyclic graph, fast zeta transform, linear extension, Markov chain Monte Carlo

∗. Preliminary versions of this work have appeared in the proceedings of AISTATS 2010, UAI 2011, and IJCAI 2013 (Parviainen and Koivisto, 2010; Niinimäki et al., 2011; Niinimäki and Koivisto, 2013b). All correspondence should be addressed to the first author.

## 1. Introduction

The Bayesian paradigm to structure learning in Bayesian networks is concerned with the posterior distribution of the underlying directed acyclic graph (DAG) given data on the variables associated with the nodes of the graph (Buntine, 1991; Cooper and Herskovits, 1992; Madigan and York, 1995). The paradigm is appealing as it offers an explicit way to incorporate prior knowledge as well as full characterization of posterior uncertainty about the quantities of interest, including proper treatment of any non-identifiability issues. However, a major drawback of the Bayesian paradigm is its large computational requirements. Indeed, because the number of DAGs grows very rapidly with the number of nodes, exact computation of the posterior distribution becomes impractical already when there are more than about a dozen of nodes.

There are two major approaches to handle the posterior distribution of DAGs without explicitly computing and representing the entire distribution. One is to summarize the posterior distribution by a relatively small number of summary statistics, of which perhaps the most extensively used is a *mode* of the distribution, that is, a maximum a posteriori DAG (Cooper and Herskovits, 1992). Other useful statistics are the marginal posterior probabilities of so-called *structural features*, such as individual arcs or larger subgraphs (Buntine, 1991; Cooper and Herskovits, 1992; Friedman and Koller, 2003). When the interest is in how well the chosen Bayesian model fits the data, say in comparison to some alternative model, then the key quantity is the *marginal likelihood* of the model—also known as the *integrated likelihood*, *evidence*, or the *normalizing constant*—which is simply the marginal probability (density) of the observed data. Provided that the model satisfies certain modularity conditions, all these statistics can be computed in a dynamic programming fashion, and thereby avoiding exhaustive traversing through individual DAGs. Specifically, assuming a very basic form of modularity one can find a mode over $n$-node DAGs in $O(2^n n^2)$ time (Koivisto and Sood, 2004; Ott et al., 2004; Singh and Moore, 2005; Silander and Myllymäki, 2006) and the arc posterior probabilities and the marginal likelihood in $O(3^n n)$ time (Tian and He, 2009). Assuming a more convoluted form of modularity, also the latter quantities can be computed in $O(2^n n^2)$ time (Koivisto and Sood, 2004; Koivisto, 2006). In practice, these algorithms scale up to about 25 nodes. For mode finding, there are also algorithms based on the A$^*$ search heuristic (Yuan and Malone, 2013) and integer linear programming (Bartlett and Cussens, 2013), which often can solve even larger problem instances.

The other approach is to approximate the posterior distribution by collecting a sample of DAGs, each of which assigned a weight reflecting how representative the DAG is of the posterior distribution. Having such a collection at hand, various quantities, including the aforementioned statistics, can be estimated by appropriate weighted averages. Principled implementations of the approach have used the Markov chain Monte Carlo (MCMC) methodology in various forms: Madigan and York (1995) simulated a Markov chain that moves in the space of DAGs by simple arc changes such that the chain's stationary distribution is the posterior distribution. Friedman and Koller (2003) obtained a significantly faster mixing chain by operating, not directly on DAGs, but in the much smaller and smoother space of node orderings, or *linear orders* on the nodes more formally. The sampler, called *order-MCMC* in the sequel, requires the prior to be of a particular form that favors DAGs that are compatible with many node orderings, thus introducing a "bias." Ellis and Wong

(2008) enhanced order-MCMC by presenting a sophisticated sampler based on tempering techniques, and a heuristic for removing the bias. Also other refinements to Madigan and York's sampler have been presented (Eaton and Murphy, 2007; Grzegorczyk and Husmeier, 2008; Corander et al., 2008), however with somewhat more limited advantages over order-MCMC. More recently, Battle et al. (2010) extended Madigan and York's sampler in yet another direction by applying annealed importance sampling (AIS) (Neal, 2001) to sample fully specified Bayesian networks (i.e., DAGs equipped with the associated conditional distributions).

While the current MCMC methods for structure learning seem to work well in many cases, they also leave the following central questions open:

1. *Smoother sampling spaces.* Can we take the idea of Friedman and Koller (2003) further: are there sampling spaces that yield still a better tradeoff between computation time and accuracy?

2. *Arbitrary structure priors.* Can we efficiently remove the bias due to sampling node orderings? Specifically, is it computationally feasible to estimate posterior expectations under an arbitrary structure prior, yet exploiting the smoother sampling space of node orderings? (The method of Ellis and Wong (2008) relies on heuristic arguments and becomes computationally infeasible when the data set is small.)

3. *Efficient parallel computation.* Can we efficiently and easily exploit parallel computation, that is, to run the algorithm in parallel on thousands of processors, preferably without frequent synchronization or communication between the parallel processes. (Existing MCMC methods are designed rather for a small number of very long runs, and thus do not enable large-scale parallelization.)

4. *Quality guarantees.* Can we measure how accurate the algorithm's output is? For instance, is it a lower bound, an upper bound, or an approximation to within some multiplicative or additive term? (Existing MCMC methods offer such guarantees only in the limit of running the algorithm infinitely many steps.)

In this article, we make a step toward answering these questions in the affirmative. Specifically, we advance the state of the art by presenting three new ideas, published in a preliminary form in three conference proceedings (Parviainen and Koivisto, 2010; Niinimäki et al., 2011; Niinimäki and Koivisto, 2013b), and their combination that has not been investigated prior to the present work. The next paragraphs give an overview of our contributions.

We address the first question by introducing a sampling space, in which each state is a partial order on the nodes. Compared to the space of node orderings (i.e., linear orders on the nodes), the resulting sampling space is smaller and the induced sampling distribution is smoother. We will also show that going from linear orders to partial orders does not increase the computational cost per sampled order, as long as the partial orders are sufficiently thin, that is, they have relatively few so-called downsets. These algorithmic results build on and extend our earlier work on finding an *optimal* DAG subject to a given partial order constraint (Parviainen and Koivisto, 2013).

To address the second question, we take a somewhat straightforward approach: per sampled partial order, we draw one or several DAGs independently from the corresponding conditional distribution, and assign each DAG a weight that compensates the difference of the structure prior of interest and the "proxy prior" we employ to make order-based sampling efficient. Specifically, we show that the number of linear extensions (or, topological sorts) of a given DAG—that we need for the weight—can be computed sufficiently fast in practice for moderate-size DAGs, even though the problem is #P-hard in general (Brightwell and Winkler, 1991).

Our third contribution applies both to the third and the fourth question. Motivated by the desire for accuracy guarantees, we seek a sampler such that we know exactly from which distribution the samples are drawn. Here, the annealed importance sampling (AIS) method of Neal (2001) provides an appealing solution. It enables drawing independent and identically distributed samples and computing the associated importance weights, so that the expected value of each weighted sample matches the quantity of interest. Due to the independence of the samples, already a small number of samples may suffice, not only for producing an accurate estimate, but also for finding a relatively tight, high-confidence lower bound on the true value (Gomes et al., 2007; Gogate et al., 2007; Gogate and Dechter, 2011). Furthermore, the independence of the samples renders the approach embarrassingly parallel, requiring interaction of the parallel computations only at the very end when the independent samples are collected in a Monte Carlo estimator. We note that Battle et al. (2010) adopted AIS for quite different reasons. Namely, due to the structure of their model, they had to sample fully specified Bayesian networks whose posterior distribution is expected to be severely multimodal, in which case AIS is a good alternative to the usual MCMC methods.

Finally, we evaluate the significance of the aforementioned advances empirically. As a benchmark we use Friedman and Koller's (2003) simple Markov chain on the space of node orderings, however, equipped with the Metropolis-coupling technique (Geyer, 1991) to enhance the chain's mixing properties. Our implementation of the *Metropolis-coupled MCMC* ($\mathrm{MC}^3$) method for the space of node orderings also serves as a proxy of a related implementation[1] of Ellis and Wong (2008). Our experimental study aims to answer two main questions: First, does the squeezing of the space of linear orders into a space of partial orders yield a significantly faster mixing Markov chain when we already use the Metropolis coupling technique to help mixing? This question was left unanswered in our preliminary work (Niinimäki et al., 2011) that only considered a single simple Markov chain similar to that of Friedman and Koller (2003). Second, are the Monte Carlo estimators based on AIS competitive to the $\mathrm{MC}^3$-based estimators when we sample partial orders instead of linear orders? This question was left unanswered in our preliminary work (Niinimäki and Koivisto, 2013b) that only considered sampling linear orders.

The remainder of this article is organized as follows. We begin in Section 2 with an introduction to some basic properties of graphs and partial orders. The section also contains some more advanced algorithmic results, which will serve as building blocks of our method for learning Bayesian networks. In Section 3, we review the Bayesian formulation of the structure learning problem in Bayesian networks, and also outline the approach of Friedman and Koller (2003) based on sampling node orderings. In Section 4, we extend the idea of

1. The software used in the work of Ellis and Wong (2008) is not publicly available (W. H. Wong, personal communication, January 29, 2013).

sampling node orderings to partial orders and formulate the two sampling methods, MC$^3$ and AIS, in that context. We dedicate Section 5 to the description of fast algorithms for computing several quantities needed by the samplers. Experimental results are reported in Section 6. We conclude and discuss directions for future research in Section 7.

## 2. Partial Orders and DAGs

This section introduces concepts, notation, and algorithms associated with graphs and partial orders. In later sections, we will apply them as building blocks in the context of structure learning in Bayesian networks. The content of the first subsection will be needed already in Section 3. The content of the latter subsections will be needed only in Section 5, and the reader may wish to skip them on the first read.

### 2.1 Antisymmetric Binary Relations

We use the standard concepts of graphs and partial orders. Our notation is however somewhat nonstandard, which warrants a special attention. Let $N$ be a finite set, and let $R \subseteq N \times N$ be a binary relation on $N$. We shall denote an element $(u, v)$ of $R$ by $uv$ for short. We say that $R$ is

$$\begin{aligned} \textit{reflexive} &\quad \text{if } u \in N \text{ implies } uu \in R\,; \\ \textit{antisymmetric} &\quad \text{if } uv, vu \in R \text{ implies } u = v\,; \\ \textit{transitive} &\quad \text{if } uv, vw \in R \text{ implies } uw \in R\,; \\ \textit{total} &\quad \text{if } u, v \in N \text{ implies } uv \in R \text{ or } vu \in R\,. \end{aligned}$$

If $R$ has the first three properties, it is called a *partial order*. If it has all the four properties, it is called a *total* or *linear order*. The set $N$ is the *ground set* of the order and the pair $(N, R)$ is called a linearly or partially ordered set.

We will also consider relations that are antisymmetric but that need not have any other of the above the properties. We say that $R$ is

$$\begin{aligned} \textit{acyclic} &\quad \text{if there are no elements } u_1, \ldots, u_l \text{ such that} \\ &\qquad u_1 = u_l \text{ and } u_{i-1}u_i \in R \text{ for each } i = 2, \ldots, l\,. \end{aligned}$$

Note that acyclicity implies irreflexivity, that is, that $uu \notin R$ for all $u \in N$. If $R$ is acyclic, we call the pair $(N, R)$ a *directed acyclic graph* (DAG), $N$ its *node set*, and $R$ its *arc set*.

When the set $N$ is clear from the context—as it will often be in our applications—we identify the structure $(N, R)$ with the relation $R$. We will typically use the letters $P$, $L$, and $A$ for a partial order, linear order, and a DAG, respectively.

The following relationships among the three objects are central in later sections of this article. Let $R$ and $Q$ be relations on $N$. We say that $Q$ is an *extension* of $R$ if simply $R \subseteq Q$. If $Q$ is a linear order, then we may also call it a *linear extension* of $R$. Note that in the literature, a linear extension of a DAG is sometimes called a topological sort of the DAG. Sometimes we will be interested in the *number of linear extensions* of $R$, which we denote by $\ell(R)$. Furthermore, we say that $R$ and $Q$ are *compatible* with each other if they have a common linear extension $L \supseteq R, Q$. While this relationship is symmetric, in our applications one of the $R, Q$ will be a partial order and the other one a DAG. Also,

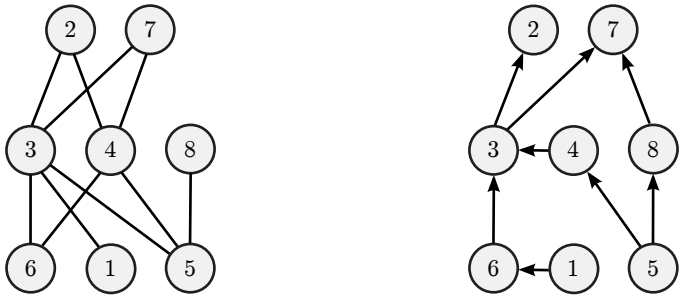



Figure 1: The Hasse diagram of a partial order on eight nodes (left) and a DAG compatible with the partial order (right).

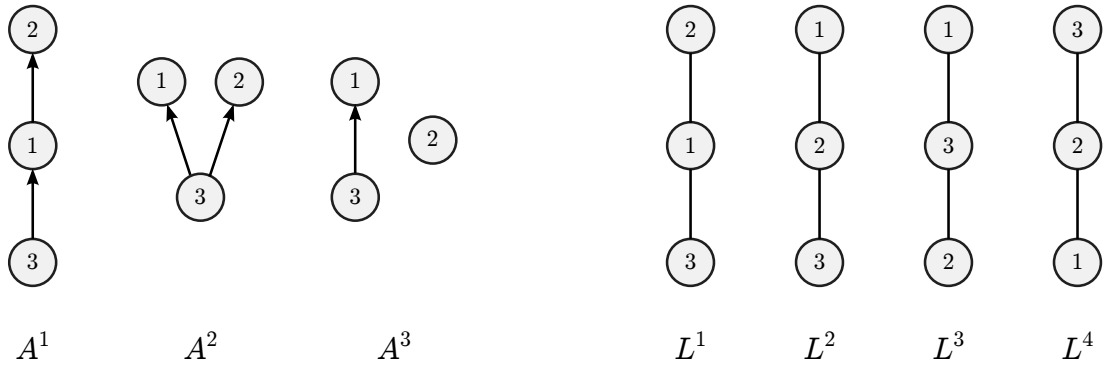



Figure 2: Three DAGs (left) and the Hasse diagrams of four linear orders (right), some of which are compatible with some of the DAGs. DAG $A^1$ has only one linear extension: $L^1$. DAG $A^2$ has two linear extensions: $L^1$ and $L^2$. DAG $A^3$ has three linear extensions: $L^1$, $L^2$ and $L^3$. None of the DAGs is compatible with $L^4$.

we say that $Q$ is the *transitive closure* of $R$ if $Q$ is the minimal transitive relation that contains $R$. Finally, we say that $R$ is the *transitive reduction* of $Q$ if $R$ is the minimal relation with the same transitive closure as $Q$. The transitive reduction of a partial order $Q$ is sometimes called the *covering relation* and visualized graphically by means of the Hasse diagram. Figures 1 and 2 illustrate some of these concepts.

Some of the above described relationships can be characterized locally, in terms of the in-neighbors of each element of $N$. To this end we let $R_v$ denote the set of elements that precede $v$ in $R$, formally

$$R_v = \{u : uv \in R, u \neq v\}\,.$$

We will call the elements of $R_v$ the *parents* of $v$ and the set $R_v$ the *parent set* of $v$. If $R$ is a partial order we may call the parents also the *predecessors* of $v$. We observe that if $Q$ is reflexive, then $Q$ is an extension of $R$ if and only if $R_v \subseteq Q_v$ for all $v \in N$.

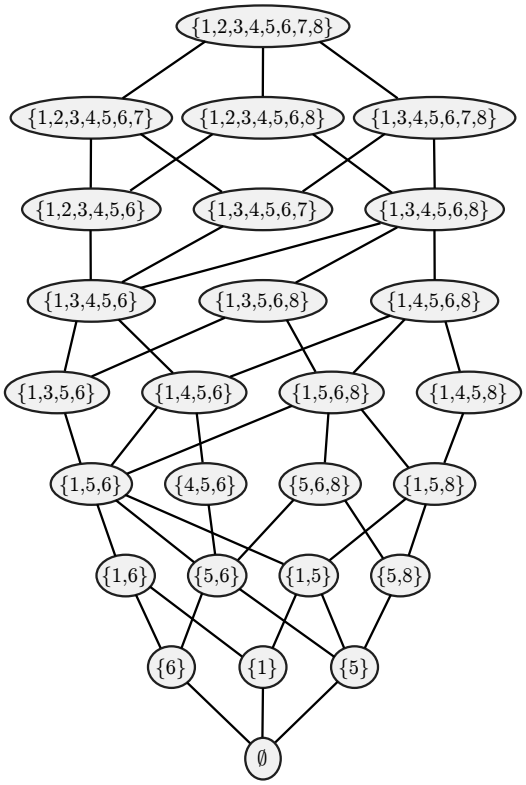



Figure 3: The covering graph of the downset lattice of the partial order shown in Figure 1.

### 2.2 Downsets and Counting Linear Extensions

Let $P$ be a partial order on a set $N$. A subset $Y \subseteq N$ is a *downset* of $P$ if $v \in Y$ and $uv \in P$ imply that $u \in Y$. In the literature downsets are sometimes called also *order ideals* or just *ideals*. We denote the set of downsets by $\mathcal{D}(P)$, or shorter $\mathcal{D}$ when there is no ambiguity about the partial order. The *downset lattice* of $P$ is the set of downsets ordered by inclusion, $(\mathcal{D}, \subseteq)$. While we do not define the notion of lattice here, we note that every lattice is a partially ordered set and thus can be represented by its *covering graph*, that is, $\mathcal{D}$ equipped with the covering relation. An example of a covering graph is shown in Figure 3. Observe that in the covering graph a node $X$ is a parent of another node $Y$ if and only if $X$ is obtained from $Y$ by removing some maximal element of $Y$, that is, an element of

$$\max Y = \left\{ u \in Y : uv \notin P \text{ for all } v \in Y \setminus \{u\} \right\}.$$

(Later we may use the notation $\max Y$ also for a set $Y$ that is not a downset.)

The following result of Habib et al. (2001) guarantees us an efficient access to the downset lattice of a given partial order:

**Theorem 1** *Given a partial order $P$ on an $n$-element set, the covering graph of the downset lattice of $P$ can be constructed in $O(n|\mathcal{D}|)$ time and space.*

**Remark 2** Actually Habib et al. (2001) show a stronger result, namely that the factor $n$ in the time and space complexity can be reduced to the width of the partial order.

As a first use of these concepts we consider the problem of counting the linear extensions of a given partial order $P$ on $N$. Recall that we denote this count by $\ell(P)$. It is immediate that $\ell(P)$ equals the number of paths from the empty set $\emptyset$ to the ground set $N$ in the covering graph of the downset lattice of $P$. Thus, letting $F(\emptyset) = 1$ and, recursively for nonempty $Y \in \mathcal{D}$,

$$F(Y) = \sum_{\substack{v \in Y \\ Y \setminus \{v\} \in \mathcal{D}}} F\big(Y \setminus \{v\}\big)\,,$$

we have that $F(N) = \ell(P)$. This result is a special case of Lemma 18 that will be given in Section 5.2. Because the covering graph provides us with an efficient access to the downsets and their parents in the covering graph, we have the following result:

**Theorem 3** *Given a partial order $P$ on an $n$-element set, the number of linear extensions of $P$ can be computed in $O(n|\mathcal{D}|)$ time and space.*

This result extends to counting the linear extensions of a given DAG $A$. Namely, we observe that $\ell(A)$ equals the number of linear extensions of the partial order $P(A)$ that is obtained by taking the transitive closure of $A$ and adding the self-loop $vv$ for each node $v \in N$. The transitive closure can be computed relatively fast, in $O(n^3)$ time, using the Floyd–Warshall algorithm.

**Corollary 4** *Given a DAG $A$ on an $n$-element set, the number of linear extensions of $A$ can be computed in $O\big(n^3 + n|\mathcal{D}(P(A))|\big)$ time and $O\big(n|\mathcal{D}(P(A))|\big)$ space.*

**Remark 5** In the worst case the number of downsets is $2^n$. We are not aware of any algorithm that would compute the exact number of linear extensions faster than in $O(n2^n)$ time in the worst case. As the problem is #P-complete (Brightwell and Winkler, 1991), there presumably is no polynomial time exact algorithm for the problem. It is possible to *approximate* the count to within any relative error $\varepsilon > 0$, roughly, in $O(\varepsilon^{-2} n^5 \log^3 n)$ time (see Sect. 4 of Bubley and Dyer, 1999). Unfortunately, the large hidden constants and the large degree of the polynomial render the approximation algorithms impractical even for moderate values of $\varepsilon$. It is also known that the linear extensions of $P$ can be enumerated in constant amortized time, which enables counting in $O(n^2 + \ell(P))$ time (Pruesse and Ruskey, 1994; Ono and Nakano, 2005). However, the enumeration approach to counting is not feasible when $\ell(P)$ is large.

### 2.3 Fast Zeta Transforms

Denote by $2^N$ the power set of $N$ and by $\mathbb{R}$ the set of real numbers. For a function $\varphi : 2^N \to \mathbb{R}$ the *zeta transform* of $\varphi$ over the subset lattice $(2^N, \subseteq)$ is the function $\widehat{\varphi} : 2^N \to \mathbb{R}$ defined by

$$\widehat{\varphi}(Y) = \sum_{X \subseteq Y} \varphi(X)\,, \quad \text{for } Y \subseteq N\,. \tag{1}$$

We shall introduce two computational problems that concern the evaluation of the zeta transform in restricted settings where the input function is sparse (has a small support) and also the output function is evaluated only at some subsets. In the *inflating zeta transform* problem we are given a partial order $P$ on $N$, a function $\varphi : \mathcal{D} \to \mathbb{R}$ from the downsets of $P$ to real numbers, and an arbitrary collection $\mathcal{C}$ of subsets of $N$. The task is to compute the zeta transform $\widehat{\varphi}$ restricted to $\mathcal{C}$, that is, to compute $\widehat{\varphi}(Y)$ for every $Y \in \mathcal{C}$. To make the formula (1) applicable, we understand that $\varphi(X) = 0$ for $X \in 2^N \setminus \mathcal{D}$. In the *deflating zeta transform* problem we are given a partial order $P$ on $N$, a collection $\mathcal{C}$ of subsets of $N$, and a function $\varphi : \mathcal{C} \to \mathbb{R}$. The task is to compute the zeta transform $\widehat{\varphi}$ restricted to $\mathcal{D}$. Again, we understand that $\varphi(X) = 0$ for $X \in 2^N \setminus \mathcal{C}$. Clearly, the problems coincide if we take $\mathcal{C} = \mathcal{D}$, and the resulting transform is known as the *zeta transform over the downset lattice*. In these problems we assume that the input function is given as a list of argument–value pairs $(X, \varphi(X))$, where $X$ ranges over the domain of the function (i.e., $\mathcal{C}$ or $\mathcal{D}$).

We begin with the problem of computing a zeta transform over the downset lattice:

**Theorem 6** *Given a partial order $P$ on an $n$-element set and a function $\varphi : \mathcal{D} \to \mathbb{R}$, the zeta transform of $\varphi$ over the downset lattice $(\mathcal{D}, \subseteq)$ can be computed in $O(n|\mathcal{D}|)$ time and space.*

To prove this result we consider the following algorithm. First construct the covering graph of the downset lattice and associate each downset $Y$ with the value $\varphi_0(Y) = \varphi(Y)$. Then find an ordering $v_1, \ldots, v_n$ of the elements of $N$ such that $v_i v_j \in P$ implies $i \leq j$. Next, for each downset $Y$ and for each $i = 1, \ldots, n$, let

$$\varphi_i(Y) = \varphi_{i-1}(Y) + \begin{cases} \varphi_{i-1}(Y \setminus \{v_i\}) & \text{if } v_i \in Y \text{ and } Y \setminus \{v_i\} \in \mathcal{D}\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

Finally return the values $\varphi_n(Y)$ for $Y \in \mathcal{D}$. We can show that the algorithm is correct:

**Lemma 7** *It holds that $\varphi_n(Y) = \widehat{\varphi}(Y)$ for all $Y \in \mathcal{D}$.*

The proof of this lemma and Lemmas 9–11 below are given in the appendix.

To complete the proof of Theorem 6, it remains to analyze the time and space requirements of the algorithm. The time and space requirement of constructing the covering graph are within the budget by Theorem 1. Finding a valid ordering takes $O(n^2)$ time and space using standard algorithms for topological sorting. Finally, for each $Y \in \mathcal{D}$, running the $n$ steps and storing the values $\varphi_i(Y)$ takes $O(n)$ time and space, thus $O(n|\mathcal{D}|)$ in total. Note that the covering graph representation enables constant-time accessing of the downsets $Y \setminus \{v_i\}$ for a given downset $Y$.

Let us then turn to the two more general problems.

**Theorem 8** *The deflating zeta transform problem and the inflating zeta transform problem can be solved in $O(n^2|\mathcal{C}| + n|\mathcal{D}|)$ time and $O(n|\mathcal{C}| + n|\mathcal{D}|)$ space.*

We prove first the claim for the deflating zeta transform problem. We develop our algorithm in two stages so as to split the sum in the zeta transform into two nested sums: the outer sum will be only over the downsets $X$ of $P$, while the inner sum will gather the needed terms for each $X$.

The key concept is the *tail* of a downset $Y$, which we define as the set interval

$$\mathcal{T}_Y = \{X : \max Y \subseteq X \subseteq Y\}\,.$$

The following lemma shows that the tails are pairwise disjoint.

**Lemma 9** *Let $Y$ and $Y'$ be distinct downsets. Then the tails $\mathcal{T}_Y$ and $\mathcal{T}_{Y'}$ are disjoint.*

We also have that each subset of the ground set belongs to some tail:

**Lemma 10** *Let $X \subseteq N$. Let $Y = \{u : uv \in P, v \in X\}$, that is, the downward-closure of $X$. Then $Y \in \mathcal{D}$ and $X \in \mathcal{T}_Y$.*

Lemmas 9 and 10 imply that for any downset $S \in \mathcal{D}$ the tails $\left\{\, \mathcal{T}_Y : Y \in \mathcal{D} \cap 2^S \,\right\}$ partition the power set $2^S$. This allows us to split the zeta transform into two nested summations. Let

$$\beta(Y) = \sum_{X \in \mathcal{T}_Y} \varphi(X)\,, \quad \text{for } Y \in \mathcal{D}\,.$$

Now

$$\widehat{\varphi}(Y) = \widehat{\beta}(Y) = \sum_{\substack{X \subseteq Y \\ X \in \mathcal{D}}} \beta(X)\,, \quad \text{for } Y \in \mathcal{D}\,.$$

Our algorithm computes $\widehat{\varphi}$ in two stages: first, given $\varphi$, it evaluates $\beta$ at all downsets of $P$; second, given $\beta$, it evaluates $\widehat{\varphi}$ at all downsets of $P$. We have already seen how the second stage can be computed within the claimed time and space budget (Theorem 6).

The first stage is computationally relatively straightforward, since each $X \in \mathcal{C}$ contributes to exactly one term $\beta(Y)$. Specifically, we can compute the function $\beta$ by initializing the values $\beta(Y)$ to zero, then considering each $X \in \mathcal{C}$ in turn and incrementing the value $\beta(Y)$ by $\varphi(X)$ for the unique downset $Y$ satisfying $X \in \mathcal{T}_Y$. Using the observation that $Y$ is the downward-closure of $X$, as given by Lemma 10, the set $Y$ can be found in $O(n^2)$ time. This completes the proof of the first claim of Theorem 8.

Consider then the inflating zeta transform problem. We use the same idea as above, however, reversing the order of the two stages. Specifically, the algorithm first computes the zeta transform over the downset lattice $\mathcal{D}$, resulting in the values $\widehat{\varphi}(Y)$ for all $Y \in \mathcal{D}$. Then, the algorithm extends the output function to the domain $\mathcal{C}$ using the observation that the zeta transform is piecewise constant, that is, for any $Z \subseteq N$ we have $\widehat{\varphi}(Z) = \widehat{\varphi}(Y)$ for a unique downset $Y$:

**Lemma 11** *Let $Z \subseteq N$. Then $\mathcal{D} \cap 2^Z = \mathcal{D} \cap 2^Y$, where the set $Y \in \mathcal{D}$ is given by*

$$Y = \{v \in Z : \textit{if } uv \in P, \textit{ then } u \in Z\}\,;$$

*in words, $Y$ consists of all elements of $Z$ whose predecessors also are in $Z$.*

Clearly the set $Y$ can be constructed in $O(n^2)$ time for a given set $Z$. This completes the proof of Theorem 8.

**Remark 12** The zeta transforms studied in this subsection have natural "upward-variants" where the condition $X \subseteq Y$ is replaced by the condition $X \supseteq Y$. The presented results readily apply to these variants, by complementation. Indeed, letting $P$ be a partial order on $N$ and denoting by $\bar{Y}$ the complement $N \setminus Y$ and by $\bar{P}$ the reversed partial order $\{vu : uv \in P\}$, we have the equivalences

$$X \supseteq Y \iff \bar{X} \subseteq \bar{Y} \quad \text{and} \quad Y \in \mathcal{D}(P) \iff \bar{Y} \in \mathcal{D}(\bar{P})\,.$$

Thus an upward-variant can be solved by first complementing the arguments of the input function, then solving the "downward-variants", and finally complementing back the arguments of the output function.

### 2.4 Random Sampling Variants

The above described zeta transform algorithms compute recursively large sums of nonnegative weights $\varphi(Z)$ each corresponding to a subset $Z \subseteq N$. Later we will also need a way to draw independent samples of the subsets $Z \subseteq Y$, proportionally to the weights. Coincidentally, the introduced summation algorithms readily provide us an efficient way to do this: for each sample we only need to stochastically backtrack the recursive steps.

We illustrate this generic approach to extend a summation algorithm into a sampling algorithm by considering the deflating zeta transform problem. Suppose we are given a partial order $P$ on $N$, a collection of $\mathcal{C}$ of subsets of $N$, and a function $\varphi : \mathcal{C} \mapsto \mathbb{R}$. As an additional input we now also assume a downset $Y \in \mathcal{D}$. We consider the problem of generating a random subset $Z \in \mathcal{C} \cap 2^Y$ proportionally to $\varphi(Z)$. We show next that this problem can be solved in $O(n)$ time, provided that the deflating zeta transform has been precomputed and the intermediate results are stored in memory.

In the first stage the algorithm backtracks the zeta transform over the downset lattice, as follows. Let $Y_n = Y$. For $i = n, n-1, \ldots, 1$,

> if $Y_i \setminus \{v_i\} \in \mathcal{D}$, then with probability $\beta_{i-1}(Y_i \setminus \{v_i\}) / \beta_i(Y_i)$ let $Y_{i-1} = Y_i \setminus \{v_i\}$;
> otherwise let $Y_{i-1} = Y_i$.

By the proof of Theorem 6, the resulting set $Y_1$ is a random draw from the subsets in $\mathcal{D} \cap 2^Y$ proportionally to $\beta(Y_1)$. This first stage takes clearly $O(n)$ time.

In the second stage the algorithm generates a random set $X \in \mathcal{T}_{Y_1}$ proportionally to $\varphi(X)$. In this case, the number of alternative options is not constant, and we need an efficient way to sample from the respective discrete probability distribution. A particularly efficient solution to this subproblem is known as the *Alias method* (Walker, 1977; Vose, 1991):

**Theorem 13 (Alias method)** *Given positive numbers $q_1, q_2, \ldots, q_r$, one can in $O(r)$ time construct a data structure, which enables drawing a random $i \in \{1, \ldots, r\}$ proportionally to $q_i$ in constant time.*

We can view the list of the terms $\varphi(X)$, for $X \in \mathcal{T}_{Y_1}$, as an intermediate result, which has been precomputed and stored in memory using the Alias method. Thus the second stage takes only $O(1)$ time.

## 3. Bayesian Learning of Bayesian Networks

In this section we review the Bayesian approach to structure learning in Bayesian networks. Our emphasis will be on the notion of modularity of the various components of the Bayesian model. From the works of Friedman and Koller (2003) and Koivisto and Sood (2004) we also adopt the notion of order-modularity that is central in the partial-order-MCMC method, which we introduce in the next section.

### 3.1 The Bayesian Approach to Structure Learning

We shall consider probabilistic models for $n$ *attributes*, the $v$th attribute taking values in a set $\mathcal{X}_v$. We will also assume the availability of a *data set* $D$ consisting of $m$ records over the attributes. We denote by $D_v^j$ the datum for the $v$th attribute in the $j$th record. We will denote by $N$ the index set $\{1, \ldots, n\}$, and often apply indexing by subsets. For example, if $S \subseteq N$, we write $D_S^j$ for $\{D_v^j : v \in S\}$ and simply $D^j$ for $D_N^j$.

We build a Bayesian network model for the data *a priori*, before seeing the actual data. To this end, we treat each datum $D_v^j$ as a random variable with the state space $\mathcal{X}_v$. We model the random vectors $D^j = (D_1^j, \ldots, D_n^j)$ as independent draws from an $n$-variate distribution $\theta$ which is Markov with respect to some directed acyclic graph $G = (N, A)$, that is,

$$\theta(D^j) = \prod_{v \in N} \theta(D_v^j \,|\, D_{A_v}^j)\,. \tag{2}$$

Recall that $A_v = \{u : uv \in A\}$ denotes the set of parents of node $v$ in $G$. We call the pair $(G, \theta)$ a *Bayesian network*, $G$ its *structure*, and $\theta$ its *parameter*. As our interest will be in settings where the node set $N$ is considered fixed, whereas the arc set $A$ varies, we will identify the structure with the arc set $A$ and drop $G$ from the notation.

Because our interest is in learning the structure from the data, we include the Bayesian network in the model as a random variable. Thus we compose a joint distribution $p(A, \theta, D)$ as the product of a *structure prior* $p(A)$, a *parameter prior* $p(\theta \mid A)$, and the *likelihood* $p(D \,|\, A, \theta) = \prod_j \theta(D^j)$. Once the data $D$ have been observed, our interest is in the *structure posterior* $p(A \,|\, D)$, which, using Bayes's rule, is obtained as the ratio $p(A)p(D \mid A)/p(D)$, where $p(D \,|\, A)$ is the *structure likelihood* and $p(D)$ the *marginal likelihood*. Note that as the parameter $\theta$ is not of direct interest, it is marginalized out. With suitable choices of the parameter prior, the marginalization can be carried out analytically; we will illustrate this below in Example 1.

Various quantities that are central in structure discovery, as well as in prediction of yet unobserved data, can be cast as the posterior expectation

$$\mathbb{E}(f \,|\, D) = \sum_A f(A) p(A \,|\, D) \tag{3}$$

of some function $f$ that associates each structure $A$ with a real number, or more generally, an element of a vector space. For example, if we let $f$ be the indicator function of the structures where $s$ is a parent of $t$, then $\mathbb{E}(f \mid D)$ equals the posterior probability that $st \in A$. For another example, define $f$ in relation to the observed data $D$ and to unobserved data $D'$ as the conditional distribution $p(D' \mid D, A)$. Then $\mathbb{E}(f \mid D)$ equals the *posterior predictive*

*distribution* $p(D' \mid D)$. Note that in this instantiation we allowed the function $f$ depend on the observed data $D$, which is only implicitly enabled in the expression (3). The marginal likelihood $p(D)$ is yet another variant of (3), obtained as the *prior expectation* $\mathbb{E}(f)$ of the function $f(A) = p(D \mid A)$. We will generally refer to the function $f$ of interest as the *feature*.

### 3.2 Modularity and Node Orderings

From a computational point of view, the main challenge is to carry out the summation over all possible structures $A$, either exactly or approximately. As the number of possible structures grows very rapidly in the number of nodes, the hope is in exploiting the properties of both the posterior distribution and the feature. Here, a central role is played by functions that are modular in the sense that they factorize into a product of "local" factors, one term per pair $(v, A_v)$, in concordance with the factorization (2).

We define the notion of modularity so that it applies equally to features, structure prior, and structure likelihood. Let $\varphi$ be a mapping that associates each binary relation $R$ on $N$ with a real number. We say that $\varphi$ is *modular* if for each node $v$ there exists a mapping $\varphi_v$ from the subsets of $N \setminus \{v\}$ to real numbers such that $\varphi(R) = \prod_{v \in N} \varphi_v(R_v)$. We call the functions $\varphi_v$ the *factors* of $\varphi$.

In what follows, the relation $R$ will typically be the arc set of a DAG. For example, the indicator function for a fixed arc $st$ mentioned above is modular, with the factors $f_v$ satisfying $f_v(A_v) = 0$ if $v = t$ and $s \notin A_t$ and $f_v(A_v) = 1$ otherwise.

Modular structure priors can take various specific forms. For example, a simple prior is obtained by assigning each node pair $uv$ a real number $\kappa_{uv} > 0$ and letting the prior $p(A)$ be proportional to $\kappa(A) = \prod_{uv \in A} \kappa_{uv}$. Such prior allows giving separate weight for each arc according to its prior plausibility. The *uniform prior* is a special case where $\kappa_{uv} = 1$ for all node pairs $uv$. Note that proportionality guarantees modularity, since the $n$ factors of the prior can absorb the normalizing constant $c = \sum_A \kappa(A)$, for example, by setting the $v$th factor to $\kappa_v(A_v)c^{-1/n}$. Another example is the prior proposed by Heckerman et al. (1995), that penalizes the distance between $A$ and a user-defined prior DAG $A^0$. This can be obtained by letting $p(A)$ be proportional to $\prod_{v \in N} \kappa_v(A_v)$ where $\kappa_v(A_v) = \delta^{|(A_v \cup A_v^0) \setminus (A_v \cap A_v^0)|}$ and $0 < \delta \leq 1$ is a user-defined constant that determines the penalization strength. Modular structure priors also enable a straightforward way to bound the indegrees of the nodes. Indeed, we often consider the case where $p(A)$ vanishes if there exists a node $v$ for which $|A_v|$ exceeds some fixed *maximum indegree*, denoted by $k$. Note, however, that modular structure priors do not allow similar controlling of the outdegrees of the nodes. For a node $v$ we will call a set $S$ a *potential parent set* if the prior does not vanish when $S$ is the parent set of $v$, that is, $p(A) > 0$ for some structure $A$ with $A_v = S$. Specifically, if $p$ is modular, then $S$ is a potential parent set of $v$ if and only if $p_v(S) > 0$.

The following example describes a frequently used modular structure likelihood (Buntine, 1991; Heckerman et al., 1995):

**Example 1 (Dirichlet–multinomial likelihood)** Suppose that each set $\mathcal{X}_v$ is finite. Consider a fixed structure $A \subseteq N \times N$. For each node $v$ and its parent set $A_v$, let the conditional distribution of $D_v^j$ given $D_{A_v}^j = y$ be categorical with parameters $\theta_{v \cdot y} = \{\theta_{vxy} : x \in \mathcal{X}_v\}$. Define a distribution $\theta$ of $D^j$ as the product of the conditional distributions and observe that $(A, \theta)$ is a Bayesian network. Assign each set of parameters $\theta_{v \cdot y}$ a Dirichlet prior with

parameters $r_{vxy} > 0$, for $x \in \mathcal{X}_v$, and compose the prior $p(\theta \,|\, A)$ by assuming independence of the components. Let $m_{vxy}$ denote the number of data records $j$ where $D^j_v = x$ and $D^j_{A_v} = y$. Then

$$\begin{aligned} p(D \,|\, A) &= \int p(\theta \,|\, A) p(D \,|\, A, \theta) \mathrm{d}\theta \\ &= \prod_{v \in N} \prod_{y \in \mathcal{X}_{A_v}} \frac{\Gamma(r_{v \cdot y})}{\Gamma(r_{v \cdot y} + m_{v \cdot y})} \prod_{x \in \mathcal{X}_v} \frac{\Gamma(r_{vxy} + m_{vxy})}{\Gamma(r_{vxy})}, \end{aligned}$$

where $r_{v \cdot y}$ and $m_{v \cdot y}$ are the sums of the numbers $r_{vxy}$ and $m_{vxy}$, respectively, and $\Gamma$ is the gamma function. In an important special case, each parameter $r_{vxy}$ is set to a so-called *equivalent sample size* $\alpha \geq 0$ divided by the number of joint configurations of $x$ and $y$ (i.e., by $|X_v|$ if $A_v$ is empty and by $|\mathcal{X}_v||\mathcal{X}_{A_v}|$ otherwise).

By a *modular model* we will refer to a Bayesian model, where both the structure likelihood and the structure prior are modular. We will assume that such a model is represented in terms of the factors $\lambda_v$ and $\kappa_v$ of a likelihood $\lambda$ and an unnormalized prior $\kappa$, respectively. In our empirical study we have used the following simple model:

**Example 2 (uniform Dirichlet–multinomial model, UDM)** In this modular model, the structure prior is specified by letting $\kappa_v(A_v) = 1$ if $|A_v| \leq k$, and $\kappa_v(A_v) = 0$ otherwise. The likelihood is the Dirichlet–multinomial likelihood with the equivalent sample size parameter set to 1 (see Example 1). The maximum indegree $k$ is a parameter of the model.

For a demonstration of the computational benefit of modularity, let us consider the probability (density) of the data $D$ conditioning on the constraint that the unknown DAG $A$ is compatible with a given linear order $L$ on the nodes. We may express this compatibility constraint by writing simply $L$, and hence the quantity of interest by $p(D \,|\, L)$. Write first

$$p(D \,|\, L) = \sum_A p(A, D \,|\, L) = \frac{\sum_{A \subseteq L} p(A) p(D \,|\, A)}{\sum_{A \subseteq L} p(A)} = \frac{\sum_{A \subseteq L} \kappa(A) \lambda(A)}{\sum_{A \subseteq L} \kappa(A)}.$$

Here the last equality holds because the normalizing constants of the prior cancel out. Then we use the key implication of the constraint $A \subseteq L$, namely, that the set of possible structures $A$ decomposes into a Cartesian product of the sets of potential parent sets $A_v$ for each node $v$ (Buntine, 1991; Cooper and Herskovits, 1992):

$$p(D \,|\, L) = \prod_{v \in N} \sum_{A_v \subseteq L_v} \kappa_v(A_v) \lambda_v(A_v) \Bigg/ \prod_{v \in N} \sum_{A_v \subseteq L_v} \kappa_v(A_v). \tag{4}$$

This factorization enables independent processing of the parent sets for each node, which amounts to significant computational savings compared to exhaustive enumeration of all possible structures. A similar factorization can be derived for the conditional posterior expectation $\mathbb{E}(f \,|\, D, L)$ of a modular feature $f$.

Motivated by the savings, Friedman and Koller (2003) addressed the unconstrained setting where no node ordering is given. They proposed averaging $\mathbb{E}(f \,|\, D, L)$ over a sample

of linear orders drawn from a distribution that is proportional to the marginal likelihood $p(D \mid L)$, as we will describe in the next subsection. For *exact* averaging over all linear orders, Koivisto and Sood (2004) gave exponential-time dynamic programming algorithm. We will obtain that algorithm as a special case of the algorithm we give in Section 5.

When the modularity is exploited as described above, the actual joint model of the data and structures does not remain modular. This is essentially because some DAGs are consistent with fewer linear orders than other DAGs. We will discuss this issue further at the end of the next subsection. To characterize the model, for which the node-ordering based methods work correctly, we follow Koivisto and Sood (2004) and call a model *order-modular* if the likelihood function is modular and the structure prior $p(A)$ is proportional to $\kappa(A) \sum_{L \supseteq A} \mu(L)$, for some modular functions $\kappa$ and $\mu$. Thus an order-modular model can be interpreted as a "modular" joint model for the structure $A$ and the linear order $L$. Note, that if $\mu(L) > 0$ for all linear orders $L$, then the support of such a prior contains the same DAGs as the support of the corresponding modular prior determined by $\kappa$. We will also use the terms *order-modular prior* and *order-modular posterior* in an obvious manner.

**Example 3 (order-modular UDM)** In this model, the structure likelihood and the factors $\kappa_v$ are as in a modular UDM (see Example 2), and the maximum indegree $k$ is a parameter of the model. The difference to the modular UDM is that we specify instead an order-modular prior by letting $\mu(L) = 1$ for all linear orders $L$ on $N$. Thus the prior $p(A)$ is proportional to $\kappa(A)\ell(A)$.

### 3.3 Sampling-based Approximations

We next review the basic sampling-based approaches for structure learning in Bayesian networks. For a broader introduction to the subject in the machine learning context, we refer to the survey by Andrieu et al. (2003).

Importance sampling methods provide us with a generic approach to approximate the expectation $\mathbb{E}(f \mid D)$ by a sample average

$$\frac{1}{T} \sum_{t=1}^{T} \frac{f(A^t) p(A^t \mid D)}{q(A^t)}, \qquad A^1, A^2, \ldots, A^T \sim q(A), \tag{5}$$

where $q(A)$ is some appropriate *sampling distribution*. It would be desirable that (i) $q(A)$ is as close to the function $|f(A^t)p(A^t \mid D)|$ as possible, up to a multiplicative constant, and (ii) that the samples $A^t$ are independent. In order to compute the average we also need, for any given sample $A^t$, (iii) an access to the value $q(A^t)$ either exactly or up to a small error. If the function $q$ can be evaluated only up to a constant factor, then one typically uses the *self-normalized importance sampling estimate*, which is obtained by dividing the sample average (5) by the sample average of $1/q(A^t)$. In practice, it is possible to simultaneously satisfy only some of the desiderata (i–iii).

The *structure-MCMC* method of Madigan and York (1995), in particular, draws the samples by simulating a Markov chain whose stationary distribution is $p(A \mid D)$. Thus, while the sampling distribution tends to $p(A \mid D)$ in the limit, after a finitely many simulation steps there is no guarantee about the quality of the sampling distribution. Furthermore, since the samples are dependent, their number has to be relatively large to obtain the efficiency that

could be obtained with a smaller number of independent samples. Finally, the computation of $q(A^t)$ is avoided by assuming it to be a good approximation to $p(A^t \,|\, D)$ and thus canceling in the estimator. The performance of structure-MCMC depends crucially on the mixing rate of the Markov chain, that is, how fast the chain "forgets" its initial state so that the draws will be (approximately) from $p(A \,|\, D)$. The main shortcoming of structure-MCMC is that the chain may get easily trapped at small regions near local maxima of the posterior.

The *order-MCMC* method of Friedman and Koller (2003), which we already mentioned in the previous subsection, aims at better performance by sampling node orderings from a distribution that is proportional to $p(D \mid L)$. Compared to the space of DAGs, the space of node orderings is not only significantly smaller but it also smoothens the sampling distribution, because for any $L$ the marginal likelihood $p(D \,|\, L)$ is a sum over exponentially many DAGs. The resulting estimator becomes

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}(f \,|\, D, L^t)\,, \qquad L^1, L^2, \ldots, L^T \sim p'(L \,|\, D)\,,$$

where $p'(L \,|\, D) \propto p(D \,|\, L) p'(L)$ is obtained via $p(D \,|\, L)$ by re-modelling the $n!$ possible node ordering constraints $L$ as mutually exclusive events, assigned with a uniform prior, $p'(L)$. In cases where exact computation of the expectation $\mathbb{E}(f \,|\, D, L^t)$ is not feasible, it can, in turn, be approximated by a single evaluation at a sampled DAG,

$$f(A^t)\,, \qquad A^t \sim p(A \,|\, D, L^t)\,,$$

or, more accurately, by an average over several independent samples from $p(A \mid D, L^t)$. Importantly, the latter sampling step, if needed, is again computationally easy thanks to the fixed node ordering. Thus the difficulty of sampling only concerns the sampling of node orderings. Friedman and Koller (2003) showed that simple local changes, namely swaps of two nodes, yield a Markov chain that mixes relatively fast in their sets of experiments. We omit a more detailed description of the order-MCMC method here, as the method is obtained as a special instantiation of the method we will introduce in Section 4.

The innocent-looking treatment of node ordering constraints as mutually exclusive events, however, introduces a "bias" to the estimator. Indeed, it is easy to see that the samples $A^t$ will be generated from the distribution

$$\sum_{L \supseteq A} p'(L \,|\, D) p(A \,|\, D, L) \propto p(A \,|\, D) \ell(A)\,.$$

In other words, compared to sampling from the true posterior $p(A \,|\, D)$, sampling via node orderings favors DAGs that are compatible with larger numbers of linear orders. But also a different interpretation is valid: the DAGs are sampled from the correct posterior under a modified (order-modular) model where the original, modular structure prior $p(A)$ is replaced by the order-modular prior that is proportional to $p(A)\ell(A)$ (as in Example 3). Oftentimes, the modularity of the structure prior is preferred, as it can express priors that are uniform over all DAGs (subject to a maximum indegree constraint). That being said, Friedman and Koller (2003) give supportive arguments also for the other viewpoint.

## 4. Sampling Partial Orders

Our key idea is to extend the order-MCMC of Friedman and Koller (2003) by replacing the state space of node orderings by an appropriately defined space of partial orders on the nodes. Throughout this section we will denote (perhaps counter to the reader's anticipation) by $\pi'$ the modular posterior distribution and by $\pi$ its "biased" order-modular counterpart that arises due to treating node orderings as mutually exclusive events. Our goal is to perform Bayesian inference under the modular posterior $\pi'$. Because our approach is to sample from the order-modular posterior $\pi$, we obtain, as a by-product, also an inference method for order-modular models. When needed, we will refer to the underlying modular model by $p'$ and to the induced order-modular model by $p$.

### 4.1 Sampling Spaces of Partial Orders

We will consider sampling from a state space that consists of partial orders on the node set $N$. We will denote the state space by $\mathcal{P}$. The idea is that the states in $\mathcal{P}$ partition the set of all linear orders on $N$. To this end, we require $\mathcal{P}$ to be an *exact cover* on $N$, that is, every linear order on $N$ must be an extension of exactly one partial order $P$ in $\mathcal{P}$. Examples 4 and 5 below illustrate the concept and also introduce the notion of a bucket order, which is central in our implementation of the proposed methods.

**Example 4 (bucket orders)** A partial order $B$ is a *bucket order* if the ground set admits a partition into pairwise disjoint sets, $B_1, B_2, \ldots, B_h$, called the *buckets*, such that $uv \in B$ if and only if $u = v$ or $u \in B_i$ and $v \in B_j$ for some $i < j$. Intuitively, the order of elements in different buckets is determined by the buckets' order, while within each bucket the elements are incomparable. We say that the bucket order is of *type* $(b_1, b_2, \ldots, b_h)$, when $b_i$ is the size of $B_i$. We call the bucket order a *balanced bucket order* with a *maximum bucket size* $b$ if $b = b_1 = b_2 = \cdots = b_{h-1} \geq b_h$. Furthermore, we call two bucket orders *reorderings* of each other if they have the same ground set and they are of the same type. It is immediate that the set (equivalence class) of reorderings of a bucket order $P$ constitute an exact cover on their common ground set (Koivisto and Parviainen, 2010). For a later reference, we also note that the number of downsets of the bucket order is $1 + \sum_i (2^{b_i} - 1)$.

The next example gives a straightforward extension of bucket orders.

**Example 5 (parallel bucket orders)** A partial order $P$ is a parallel composition of bucket orders, or *parallel bucket order* for short, if $P$ can be partitioned into $r$ bucket orders $B^1, B^2, \ldots, B^r$ on disjoint ground sets. We call two parallel bucket orders $P$ and $Q$ *reorderings* of each other if their bucket orders can be labelled as $P^1, P^2, \ldots, P^r$ and $Q^1, Q^2, \ldots, Q^r$ such that each $P^j$ is a reordering of $Q^j$. It is known that the set (equivalence class) of reorderings of a parallel bucket order $P$ is an exact cover on their common ground set (Koivisto and Parviainen, 2010). Compared to bucket orders, parallel bucket orders make it possible to obtain better time–space tradeoffs in the context of exact structure discovery. However, our preliminary calculations (Niinimäki et al., 2011) show that parallel bucket orders are unlikely to yield substantive advantages in the sampling context.

Because the methods we shall consider are based on local moves in the sampling space, we equip the sampling space with a neighborhood structure. Formally, a neighborhood

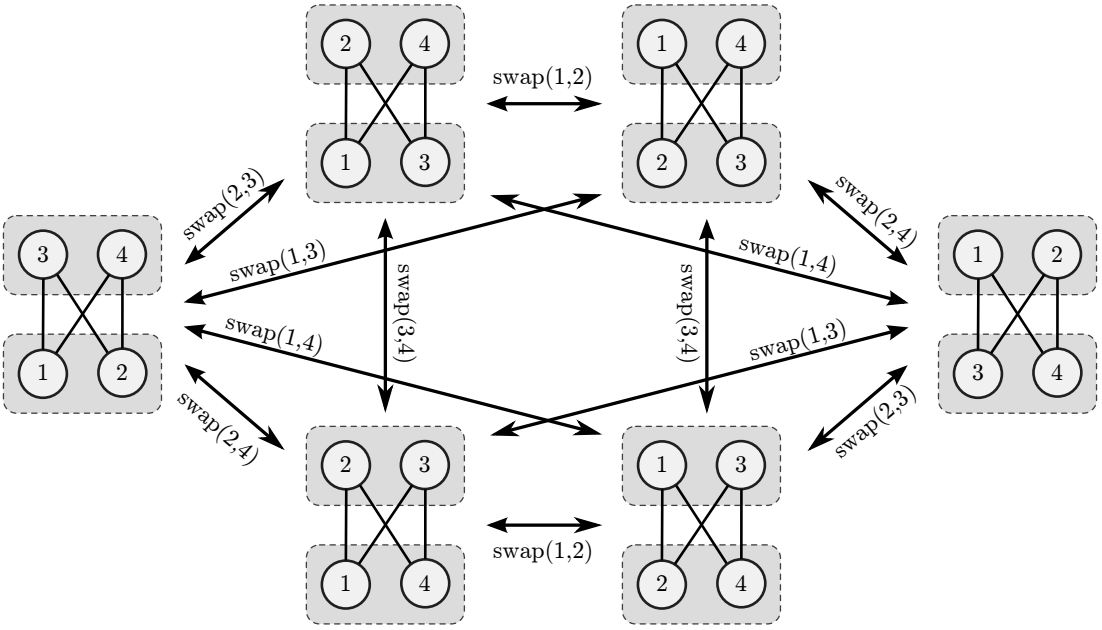



Figure 4: All reorderings of a bucket order of type $(2, 2)$ on the node set $\{1, 2, 3, 4\}$. Adjacency in the swap neighborhood is indicated by arrows labeled by the corresponding swap operation. Each bucket order is visualized using a Hasse diagram, with rectangles indicating individual buckets.

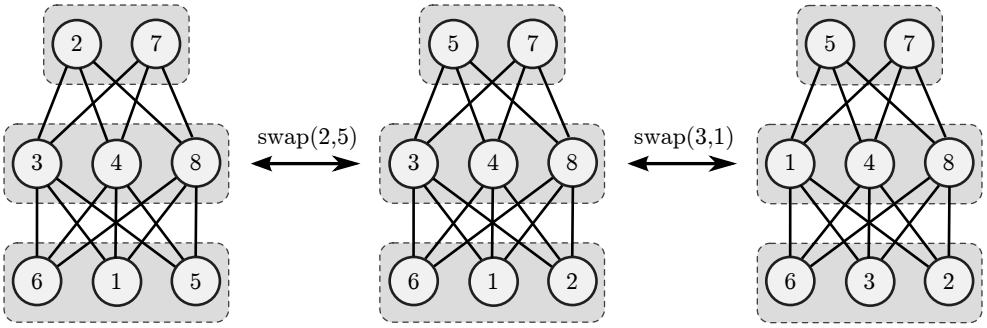



Figure 5: Three adjacent states in the space of reorderings of a bucket order of type $(3, 3, 2)$.

structure on $\mathcal{P}$ is just a graph on $\mathcal{P}$, the adjacency relation specifying the neighborhood relation. Here, we do not make an attempt to give any specific neighborhood structure that would be appropriate for an arbitrary exact cover $\mathcal{P}$. Instead, we continue with an example:

**Example 6 (swap neighborhood)** Let $P$ be a (parallel) bucket order on $N$, and let $\mathcal{P}$ the set of reorderings of $P$. The *swap neighborhood* on $\mathcal{P}$ is a graph in which the vertex set consists of the reorderings $\mathcal{P}$ and two members $Q, R \in \mathcal{P}$ are adjacent if $Q$ is obtained from

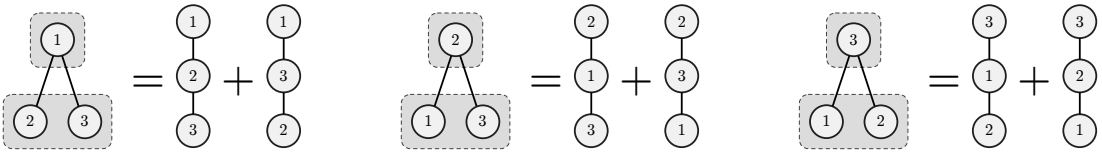


Figure 6: There are in total six linear orders on the node set $\{1, 2, 3\}$ and three reorderings of a bucket order of type (2,1). As shown in the figure, the probability of each bucket order is the sums of the probabilities of its linear extensions. Observe, how the bucket orders form an exact cover and hence partition the set of linear orders into three disjoint subsets.

$R$ by *swapping* two nodes $s, t \in N$, more formally:

$$uv \in R \iff \sigma(u)\sigma(v) \in Q\,,$$

where $\sigma$ is the transposition $N \to N$ that swaps $s$ and $t$. Clearly, the swap neighborhood is connected. See Figures 4 and 5 for an illustration of bucket orders and swap neighborhoods.

For each partial order $P$ in an exact cover $\mathcal{P}$, the posterior probability $\pi(P)$ is obtained simply as the sum of the posterior probabilities $\pi(L)$ of all linear extensions $L$ of $P$. This is illustrated in Figure 6. Note that the choice of the state space $\mathcal{P}$ does not affect the posterior $\pi$ and consequently leaves the bias untouched. The correction of the bias will be taken care of separately, as shown in the next subsection.

### 4.2 Partial-Order-MCMC and Bias Correction

The basic *partial-order-MCMC* method has three steps. It first samples states from $\mathcal{P}$ along a Markov chain whose stationary distribution is $\pi$. Then it draws a DAG from each sampled partial order. Finally, it estimates the posterior expectation of the feature in interest by taking a weighted average of the samples. In more detail, these steps are as follows:

1. *Sample partial orders along a Markov chain using the Metropolis–Hastings algorithm.* Start from a random partial order $P^1 \in \mathcal{P}$. To move from state $P^t$ to the next state, first draw a candidate state $P^\star$ from a proposal distribution $q(P^\star \,|\, P^t)$. Then accept the candidate with probability

$$\min\left\{1, \frac{\pi(P^\star)q(P^t \,|\, P^\star)}{\pi(P^t)q(P^\star \,|\, P^t)}\right\},$$

   and let $P^{t+1} = P^\star$; otherwise let $P^{t+1} = P^t$. This produces a sample of partial orders $P^1, P^2, \ldots, P^T$. Each move constitutes one *iteration* of the algorithm.

2. *Sample DAGs from the sampled partial orders.* For each sampled partial order $P^t$, draw a DAG $A^t$ compatible with $P^t$ from the conditional posterior distribution $\pi(A^t \,|\, P^t)$. This produces a sample of DAGs $A^1, A^2, \ldots, A^T$.

3. *Estimate the expected value of the feature by a weighted sample average.* Return

$$f_{\text{MCMC}} = \sum_t \frac{f(A^t)}{\ell(A^t)} \Bigg/ \sum_t \frac{1}{\ell(A^t)}$$

as an estimate of $\mathbb{E}_{\pi'}(f)$. Recall that $\ell(A^t)$ is the number of linear extensions of $A^t$.

Several standard MCMC techniques can be applied to enhance the method in practice. Specifically, it is computationally advantageous to thin the set of samples $P^t$ by keeping only, say, every 100th sample. Also, it is often a good idea to start collecting the samples only after a burn-in period that may cover, say, as much as 50% of the total allocated running time. On the other hand, we can compensate these savings in step 2 of the method by drawing *multiple*, independent DAGs per sampled partial order $P^t$, which can improve considerably the estimate obtained in step 3. We will consider these techniques in more detail in the context of our empirical study in Section 6.

It is worth noting that access to the exact posterior probabilities $\pi(P^t)$ are not needed in step 1. It suffices that we can evaluate a function $g$ that is proportional to $\pi$. An appropriate function is given by

$$g(P^t) = \sum_{L \supseteq P^t} \sum_{A \subseteq L} p'(A, D)\,. \tag{6}$$

We will see later that the modularity of the model $p'$ enables fast computation of $g(P^t)$.

We can show that under mild conditions on the proposal distribution $q$, the Markov chain constructed in step 1 converges to the target distribution $\pi(P)$, and consequently, the estimate $f_{\text{MCMC}}$ tends to $\mathbb{E}_{\pi'}(f)$ as the number of samples $T$ grows. In the next paragraphs we justify these two claims.

For the first claim it suffices to show that the chain is irreducible, that is, from any state $P$ the chain can reach any other state $P^\star$ with some number of moves (with a positive probability). In principle, this condition would be easily satisfied by a proposal distribution $q(P^\star \mid P)$ whose support is the entire $\mathcal{P}$ for every $P$. From a practical point of view, however, it is essential for good mixing of the chain to have a proposal distribution that concentrates the proposes locally to a few *neighbors* of the current state $P$. In that case it is crucial to show that the induced neighborhood graph on $\mathcal{P}$ is strongly connected, thus implying irreducibility of the chain. In the order-MCMC method, connectedness was obtained because any two node orderings can be reached from each other by some number of swaps of two nodes. It is easy to see that this proposal distribution applies to partial orders as well, only noting that some swaps of nodes may result in partial orders that are outside the fixed state space $\mathcal{P}$ and must thus be rejected (or avoid proposing). Rather than pursuing the issue in full generality, we extend the swap proposal for the sampling space of parallel bucket orders:

**Example 7 (swap proposal for parallel bucket orders)** Consider the set of reorderings $\mathcal{P}$ and the swap-neighborhood described in Example 6. For any $P \in \mathcal{P}$, let the conditional proposal distribution $q(P^\star \mid P)$ be uniform over the neighbors $P^\star$ of $P$ (and vanish otherwise). Note that it is possible to sample directly from this conditional distribution by drawing two nodes $s, t$ that belong to the same part but different buckets in the partition of $N$, uniformly at random, and proposing the swap of $s$ and $t$.

We then turn to the second claim that the estimate tends to the posterior expectation of the feature. We investigate the behavior of the estimate $f_{\text{MCMC}}$. By the properties of the Markov chain we may assume that the $P^1, P^2, \ldots, P^T$ are an ergodic sample from $\pi(P)$, that is, any sample average tends to the corresponding expected value as $T$ grows. Consequently, the $A^1, A^2, \ldots, A^T$ are an ergodic sample from the biased posterior

$$\pi(A) = \sum_{P \in \mathcal{P}} \pi(P)\, \pi(A \,|\, P) = \sum_{P \in \mathcal{P}} \sum_{L \supseteq P} \pi(P, L, A) = \sum_{L} \pi(L, A) \propto \pi'(A)\ell(A)\,.$$

Here the last equation holds because $\mathcal{P}$ is an exact cover on $N$; and the last proportionality holds because $\pi(L, A)$ vanishes if $L$ is not an extension of $A$, and is otherwise proportional to $\pi'(A)$. The ergodicity now guarantees that the average of the $1/\ell(A^t)$ tends to the expectation $\mathbb{E}_\pi(1/\ell(A^t)) = 1/c$, where

$$c = \sum_{A} \pi'(A)\ell(A)\,, \tag{7}$$

and that the average of the $f(A^t)/\ell(A^t)$ tends to the ratio $\mathbb{E}_{\pi'}(f)/c$. This implies that $f_{\text{MCMC}}$ approaches $\mathbb{E}_{\pi'}(f)$ as the number of samples $T$ grows.

The computational complexity of partial-order-MCMC is determined, in addition to the number of samples $T$, by the complexity of the problems solved for each sampled partial order $P^t$ and DAG $A^t$. These problems—of which analysis we postpone to Section 5—are:

**(a)** *Unnormalized posterior:* Compute $g(P^t)$ for a given $P^t$.

**(b)** *Sample DAGs:* Draw a DAG $A^t$ from $\pi(A^t \,|\, P^t)$ for a given partial order $P^t$ .

**(c)** *Number of linear extensions:* Compute $\ell(A^t)$ for a given DAG $A^t$.

We will see that problems (a) and (b) can be solved in time that scales, roughly, as $C + |\mathcal{D}|$, where $C$ is the total number of potential parent sets and $|\mathcal{D}|$ is the number of downsets of the partial order $P^t$.

The problem (c) of counting the linear extensions was already discussed in Section 2, Corollary 4. We note that if the target model is order-modular, then there is no need to compute the terms $\ell(A^t)$, as no bias correction is needed. Moreover, for an order-modular model also the DAG sampling step can be avoided if the interest is in a modular feature $f$. Namely, then the estimate is obtained simply as an average of the conditional expectations $\mathbb{E}_\pi(f \,|\, P^t)$, which gives us one more problem to solve:

**(a')** *Expectation of a modular feature:* Compute $\mathbb{E}_\pi(f \,|\, P^t)$ for a given partial order $P^t$.

We will see in Section 5 that this problem can be computed in almost the same way as the unnormalized posterior probability of $P^t$, which justifies the label (a').

### 4.3 Metropolis-coupled Markov Chain Monte Carlo (MC$^3$)

Tempering techniques can enhance mixing of the chain and, as a byproduct, they also offer a good estimator for the marginal likelihood $p(D)$. Here we consider one such technique, *Metropolis-coupled MCMC* (MC$^3$) (Geyer, 1991). In MC$^3$, several Markov chains, indexed

by $0, 1, \ldots, K$, are simulated in parallel, each chain $i$ having its own stationary distribution $\pi_i$. The idea is to take $\pi_0$ as a "hot" distribution, for example, the uniform distribution, and then let the $\pi_i$ be increasingly "cooler" and closer approximations of the posterior $\pi$, putting finally $\pi_K = \pi$. Usually, powering schemes of the form

$$\pi_i \propto \pi^{\beta_i}, \quad 0 \leq \beta_0 < \beta_1 < \cdots < \beta_K = 1$$

are used. For instance, Geyer and Thompson (1995) suggest harmonic stepping, $\beta_i = 1/(K+1-i)$; in our experiments we have used linear stepping, $\beta_i = i/K$.

In addition to running the chains in parallel, every now and then we propose a swap of the states $P_i$ and $P_j$ of two randomly chosen chains $i$ and $j = i + 1$. The proposal is accepted with probability

$$\min\left\{1, \frac{\pi_i(P_j)\pi_j(P_i)}{\pi_i(P_i)\pi_j(P_j)}\right\}.$$

We note that each $\pi_i$ needs to be known only up to some constant factor, that is, it suffices that we can efficiently evaluate a function $g_i$ that is proportional to $\pi_i$. By using samples from the coolest chain only, an estimate of the expectation $\mathbb{E}_{\pi'}(f)$, which we denote by $f_{\text{MC3}}$, is obtained by following steps 2 and 3 of the partial-order-MCMC method. In Section 4.5 we will show that, by using samples from all chains, we can also get good estimates of the marginal likelihood $p'(D)$. This technique is a straightforward extension of the technique for estimating $p(D)$.

### 4.4 Annealed Importance Sampling (AIS)

AIS produces independent samples of partial orders $P^1, P^2, \ldots, P^T$ and associated importance weights $w^1, w^2, \ldots, w^T$. Like in MC$^3$, a sequence of distributions $\pi_0, \pi_1, \ldots, \pi_K$ is introduced, such that sampling from $\pi_0$ is easy, and as $i$ increases, the distributions $\pi_i$ provide gradually improving approximations to the posterior distribution $\pi$, until finally $\pi_K$ equals $\pi$. In our experiments we have used the same scheme as for MC$^3$, however, with a much larger value of $K$. For each $\pi_i$ we assume the availability of a corresponding function $g_i$ that is proportional to $\pi_i$ and that can be evaluated fast at any given point.

To sample $P^t$, we first sample a sequence of partial orders $P_0, P_1, \ldots, P_{K-1}$ along a Markov chain, starting from $\pi_0$ and moving according to suitably defined transition kernels $\tau_i$, as follows:

Generate $P_0$ from $\pi_0$.
Generate $P_1$ from $P_0$ using $\tau_1$.
$\vdots$
Generate $P_{K-1}$ from $P_{K-2}$ using $\tau_{K-1}$.

The transition kernels $\tau_i$ are constructed by a simple Metropolis–Hastings move: At state $P_{i-1}$ a candidate state $P_\star$ is drawn from a proposal distribution $q(P_\star \,|\, P_{i-1})$; the candidate is accepted as the state $P_i$ with probability

$$\min\left\{1, \frac{g_i(P_\star)q(P_{i-1} \,|\, P_\star)}{g_i(P_{i-1})q(P_\star \,|\, P_{i-1})}\right\},$$

and otherwise $P_i$ is set to $P_{i-1}$. It follows that the transition kernel $\tau_i$ leaves $\pi_i$ invariant. Finally, we set $P^t = P_{K-1}$ and assign the importance weight as

$$w^t = \frac{g_1(P_0)}{g_0(P_0)} \frac{g_2(P_1)}{g_1(P_1)} \cdots \frac{g_K(P_{K-1})}{g_{K-1}(P_{K-1})} .$$

We then generate a DAG $A^t$ from each $P^t$ as in step 2 of the partial-order-MCMC method. An estimate of $\mathbb{E}_{\pi'}(f)$ is given by

$$f_{\text{AIS}} = \sum_t \frac{w^t f(A^t)}{\ell(A^t)} \Bigg/ \sum_t \frac{w^t}{\ell(A^t)} . \tag{8}$$

To see that this self-normalized importance sampling estimate is consistent, we examine separately the expected values of the numerator and the denominator, and show that their ratio equals $\mathbb{E}_{\pi'}(f)$. To this end, consider a fixed $t$ and any function $h$ of partial orders. Denote by $q$ the joint sampling distribution of the partial orders $P_0^t, P_1^t, \ldots, P_K^t$ and the DAG $A^t$. The general result of Neal (2001) implies the following: Let $\rho$ denote the ratio of the normalizing constants of $g_K$ and $g_0$. Then

$$\mathbb{E}_q\big(w^t h(P_K^t)\big) = \rho \cdot \mathbb{E}_\pi\big(h(P_K^t)\big) \quad \text{and} \quad \mathbb{E}_q\big(w^t\big) = \rho .$$

Using the fact that $\mathbb{E}_q\big(w^t\, h'(A^t)\big) = \mathbb{E}_q\big(w^t\, \mathbb{E}_q(h'(A^t) \,|\, P_0^t, \ldots, P_K^t)\big) = \mathbb{E}_q\big(w^t\, \mathbb{E}_\pi(h'(A^t) \,|\, P_K^t)\big)$ and applying the above result with a particular choice of the functions $h$ and $h'$ yields

$$\mathbb{E}_q\left(w^t \cdot \frac{f(A^t)}{\ell(A^t)}\right) = \mathbb{E}_q\left(w^t \cdot \mathbb{E}_\pi\left(\frac{f(A^t)}{\ell(A^t)} \,\middle|\, P_K^t\right)\right) = \rho \cdot \mathbb{E}_\pi\left(\frac{f(A^t)}{\ell(A^t)}\right) = \frac{\rho}{c} \cdot \mathbb{E}_{\pi'}\big(f(A^t)\big) ,$$

where $c$ is, as given before in (7), the normalizing constant of $\pi'(A)\ell(A)$. From this we also see that the expected value of each term in the denominator in (8) equals $\rho/c$. Thus the ratio of the expectations is $\mathbb{E}_{\pi'}(f)$ as desired.

### 4.5 Estimating the Marginal Likelihood

We now turn to the estimation of the marginal likelihood $p'(D)$ using samples produced by either MC$^3$ or AIS. We will view $p'(D)$ as the normalizing constant of the function $g'(A) = p'(A, D)$, and denote the constant by $c'$ for short. We will estimate $c'$ indirectly, by estimating a ratio $\rho' = c'/c_0$, where $c_0$ is another normalizing constant that we can compute exactly. In fact, $c_0$ will be the normalizing constant $g_0/\pi_0$, and in general, we will denote by $c_i$ the normalizing constant $g_i/\pi_i$.

With a sample generated by MC$^3$, our estimate for the marginal likelihood is obtained, in essence, as a product of estimates of the ratios $c_{i+1}/c_i$, as given by

$$\rho'_{\text{MC3}} = \prod_{i=0}^{K-1} \left(\frac{1}{T} \sum_t \frac{g_{i+1}\big(P_i^t\big)}{g_i\big(P_i^t\big)}\right) \left(\frac{1}{T} \sum_t \frac{1}{\ell(A^t)}\right) .$$

To see that the estimate is asymptotically unbiased, observe first that

$$\mathbb{E}_{\pi_i}\left(\frac{g_{i+1}\big(P_i^t\big)}{g_i\big(P_i^t\big)}\right) = \frac{c_{i+1}}{c_i} \quad \text{and} \quad \mathbb{E}_\pi\left(\frac{1}{\ell(A^t)}\right) = \frac{c'}{c_K} .$$

Here the former equation is easy to verify. For the latter we recall that $\mathbb{E}_\pi\big(1/\ell(A^t)\big) = 1/c$ and write the normalizing constant of $g_K = g$ using (6) as

$$c_K = \sum_P g(P) = \sum_L \sum_{A \subseteq L} p'(A, D) = p'(D) \sum_A \pi'(A)\ell(A) = c'c\,.$$

Now, if the estimates were independent and, moreover, the samples $P_i^t$ were exactly from $\pi_i$, then $\rho'_{\mathrm{MC3}}$ would be an unbiased estimate of the marginal likelihood. While neither condition is satisfied in our case, the ergodicity of the chains guarantees that each estimate, and thereby their product, is asymptotically unbiased.

With a sample generated by AIS, our estimate for the marginal likelihood is

$$\rho'_{\mathrm{AIS}} = \frac{1}{T} \sum_t \frac{w^t}{\ell(A^t)}\,.$$

It is not difficult to see that this estimate is unbiased. Namely, we have already seen that the expected value of this estimate is $\rho/c$, where $\rho = c_K/c_0$. Because we just showed that $1/c = c'/c_K$, we obtain $\rho/c = c'/c_0 = \rho'$, as desired.

The AIS-based estimate has two main advantages over the MC$^3$-based estimate. One is that we can use a fairly large number of steps $K$ in AIS, which renders the estimate more accurate. In MC$^3$ we have to use a much smaller $K$ to reserve time for simulating each chain a large number of steps. A smaller $K$ is expected to yield less accurate estimates. The other advantage of the AIS-based estimate stems from the unbiasedness and independence of the samples. Indeed, these two properties allow us to compute high-confidence *lower bounds* for the marginal likelihood. We will make use the following elementary theorem; for variations and earlier uses in other contexts, we refer to the works of Gomes et al. (2007) and Gogate and Dechter (2011).

**Theorem 14 (lower bound)** *Let $Z_1, Z_2, \ldots, Z_s$ be independent nonnegative random variables with mean $\mu$. Let $0 < \delta < 1$. Then, with probability at least $1 - \delta$, we have*

$$\delta^{1/s} \min\{Z_1, Z_2, \ldots, Z_s\} \leq \mu\,.$$

**Proof** By Markov's inequality, $Z_i > \delta^{-1/s}\mu$ with probability at most $\delta^{1/s}$, for each $i$. Taking the product gives that $\min\{Z_1, Z_2, \ldots, Z_s\} > \delta^{-1/s}\mu$ with probability at most $\delta$. To complete the proof, multiply both sides by $\delta^{1/s}$ and consider the complement event. ■

We apply this result by dividing our $T$ samples into $s$ bins of equal size and letting $Z_i$ be the estimate of the marginal likelihood based on the samples in the $i$th bin. There is a tradeoff in choosing a good value of $s$. Namely, to obtain good individual estimates $Z_i$, we would like to set $s$ as small as possible. On the other hand, we would like to use a large $s$ in order to have a *slack* factor $\delta^{1/s}$ as close to 1 as possible. The following examples show two different ways to address this tradeoff.

**Example 8 (slack-2 lower bound)** Put $\delta = 2^{-5} = 0.03125$ and $s = 5$. Then $\delta^{1/s} = 1/2$.

**Example 9 (square-root lower-bounding scheme)** Put $\delta = 2^{-5}$ and $s = \lfloor \sqrt{T} \rfloor$ for $T$ samples. Then $\delta^{1/s}$ grows with $T$, being $2^{-1}$, $2^{-1/2}$, $2^{-1/4}$ at $T = 25, 100, 400$, respectively.

## 5. Per-Sample Computations

In the previous section we encountered a number of computational problems associated with each sampled partial order and DAG (see the end of Section 4.2). In this section we give algorithms to solve those problems. We begin by formulating the computational problems and stating the main results in Section 5.1. The proofs are given in Sections 5.2–5.4. Finally, in Section 5.5 we discuss the possibility to reduce the time and space requirements in certain special cases that are relevant for the present applications.

### 5.1 Problems and Results

We shall derive solutions to the computational problems (a), (b), and (a') of Section 4.2 as specific instantiations of slightly more abstract problems concerning modular functions. Recall that the problem (c) was already discussed in Section 2.

We abstract the core algorithmic problem underlying problems (a) and (a') as what we call the *DAG-extensions* (DAGE) problem, defined as follows. As input we are given a modular function $\varphi$ that associates each DAG on $N$ with a real number. We assume that each factor $\varphi_v$, for $v \in N$, is given explicitly as a list of argument–value pairs $(X, \varphi_v(X))$ where $X$ runs through some collection $\mathcal{C}_v$ of subsets of $N \setminus \{v\}$. We further assume that the factor vanishes outside this collection. As input we are also given a partial order $P$ on $N$. Our task is to compute the value $\varphi(P)$ defined by

$$\varphi(P) = \sum_{L \supseteq P} \sum_{A \subseteq L} \varphi(A) \,.$$

We denote by $C$ the sum of the sizes $|\mathcal{C}_v|$, and by $\mathcal{D}$ the set of downsets of $P$.

Problems (a) and (a') reduce to the DAGE problem: We obtain the unnormalized posterior probability $g(P)$ as $\varphi(P)$ by letting $\varphi(A) = \kappa(A)\lambda(A)$. The collections $\mathcal{C}_v$ consists of the potential parent sets of node $v$. Similarly, we obtain the expectation $\mathbb{E}_\pi(f \,|\, P)$ of a modular feature $f$ as a ratio $\varphi^f(P)/g(P)$ by letting $\varphi^f(A) = f(A)\kappa(A)\lambda(A)$.

In the next subsection we prove:

**Theorem 15 (DAG-extensions)** *Given a partial order $P$ on $N$ and a modular function $\varphi$ over $N$, we can compute $\varphi(P)$ in $O\big(n^2(C + |\mathcal{D}|)\big)$ time and $O\big(n(C + |\mathcal{D}|)\big)$ space.*

To address problem (b), we define the *DAG sampling* problem as follows. Our input is as in the DAGE problem, except that we are also given a number $T$. Our task is to sample $T$ independent DAGs from a distribution that is proportional to $\varphi(A)\ell(A \cup P)$. Observe that $\varphi(P)$ can be written as a sum of $\varphi(A)\ell(A \cup P)$ over all DAGs $A$ on $N$. Problem (b) reduces to the DAG sampling problem in an obvious manner.

In Section 5.3 we prove:

**Theorem 16 (DAG sampling)** *Given a partial order $P$ on $N$, a modular nonnegative function $\varphi$ over $N$, and a number $T > 0$, we can draw $T$ independent DAGs $A$ on $N$ proportionally to $\varphi(A)\ell(A \cup P)$ in $O\big(n^2(C + |\mathcal{D}| + T)\big)$ time and $O\big(nC + n^2|\mathcal{D}|\big)$ space.*

We also consider the following variant of the DAGE problem, which we call the *arc probabilities* problem. Our input is as in the DAGE problem. For a pair of nodes $s, t \in N$,

define $\varphi^{st}$ as the modular function obtained from $\varphi$ by setting each factor as

$$\varphi_v^{st}(A_v) \equiv \begin{cases} 0 & \text{if } t = v \text{ and } s \notin A_v\,, \\ \varphi_v(A_v) & \text{otherwise}\,. \end{cases}$$

Our task is to compute the values $\varphi^{st}(P)$ for all node pairs $st$. This problem models the task of computing the posterior probabilities of all arcs: we see that $\varphi^{st}(P)/g(P)$ equals $\mathbb{E}_\pi(f)$ when we set $\varphi(A) = \kappa(A)\lambda(A)$, and $f(A) = 1$ if $st \in A$ and $f(A) = 0$ otherwise.

In Section 5.4 we prove:

**Theorem 17 (arc probabilities)** *Given a partial order $P$ on $N$ and a modular function $\varphi$ over $N$, we can compute the values $\varphi^{st}(P)$ for every pair of two nodes $s, t \in N$ simultaneously in $O\big(n^2(C + |\mathcal{D}|)\big)$ time and $O\big(n(C + |\mathcal{D}|)\big)$ space.*

### 5.2 Proof of Theorem 15

We prove Theorem 15 by giving an algorithm that evaluates $\varphi(P)$ in the claimed time and space. The algorithm consists of two phases, which stem from the sum–product expression

$$\varphi(P) = \sum_{L \supseteq P} \prod_v \alpha_v(L_v)\,, \quad \text{where} \quad \alpha_v(L_v) = \sum_{A_v \subseteq L_v} \varphi_v(A_v)\,. \tag{9}$$

This expression is obtained by applying the same decomposition that was used to obtain factorization (4). In the first phase of the algorithm, we compute the values $\alpha_v(L_v)$ for each $v \in N$ and every relevant subset $L_v \subseteq N \setminus \{v\}$—we will see that only the downsets of $P$ can be relevant. In the second phase, we compute the sum over all linear extensions of $P$ by dynamic programming across the downsets of $P$, now assuming efficient access to each (precomputed) value $\alpha_v(L_v)$.

Let us first consider the first phase for a fixed node $v$. We observe that the problem of computing the values $\alpha_v(Y)$ for all downsets $v \notin Y \in \mathcal{D}$ reduces trivially to the deflating zeta transform problem, and can thus, by Theorem 8, be solved in $O(n^2|\mathcal{C}_v| + n|\mathcal{D}|)$ time and $O(n(|\mathcal{C}_v| + |\mathcal{D}|))$ space. Summing the time bounds over the $n$ nodes yields a time bound of $O(n^2(C + |\mathcal{D}|)$ in total. Because the working space can be reused for different nodes $v$, the space requirement is dominated by the input and the output size, which is $O(n(C + |\mathcal{D}|)$ in total.

Consider then the second phase. Define the function $F$ from $\mathcal{D}$ to real numbers by letting $F(\emptyset) = 1$ and for nonempty $Y \in \mathcal{D}$ recursively:

$$F(Y) = \sum_{\substack{v \in Y \\ Y \setminus \{v\} \in \mathcal{D}}} \alpha_v\big(Y \setminus \{v\}\big)\, F\big(Y \setminus \{v\}\big)\,. \tag{10}$$

Lemma 18 below shows that $F(N) = \varphi(P)$. Thus $\varphi(P)$ can be evaluated in $O(n|\mathcal{D}|)$ time and space using the covering graph of the downset lattice (Theorem 1).

**Lemma 18** *We have $F(N) = \varphi(P)$.*

**Proof** For any subset $Y \subseteq N$ denote by $P[Y]$ the induced partial order $\{xy \in P : x, y \in Y\}$. We show by induction on the size of $Y$ that

$$F(Y) = \sum_{L \supseteq P[Y]} \prod_{v \in Y} \alpha_v(L_v)\,,$$

where the sum is over all linear extensions of $P[Y]$.

For the base case, consider an arbitrary singleton $Y = \{v\} \in \mathcal{D}$. From the definition we get that $F(Y) = \alpha_v(\emptyset)F(\emptyset) = \alpha_v(\emptyset)$. Likewise, the induction claim evaluates to $F(Y) = \alpha_v(\emptyset)$, as $P[Y] = \{vv\}$.

For the induction step, let $\emptyset \neq Y \in \mathcal{D}$. We write the induction claim as

$$\begin{aligned}\sum_{L \supseteq P[Y]} \prod_{v \in Y} \alpha_v(L_v) &= \sum_{u \in \max Y} \alpha_u\big(Y \setminus \{u\}\big) \sum_{L' \supseteq P[Y \setminus \{u\}]} \prod_{v \in Y \setminus \{u\}} \alpha_v(L'_v) \\ &= \sum_{\substack{u \in Y \\ Y \setminus \{u\} \in \mathcal{D}}} \alpha_u\big(Y \setminus \{u\}\big)\, F\big(Y \setminus \{u\}\big)\,,\end{aligned}$$

which equals $F(Y)$ by the recursive definition (10). ■

### 5.3 Proof of Theorem 16

Consider the following algorithm. First solve the corresponding instance of the DAGE problem as described in the proof of Theorem 15. Store the intermediate results in the way described in Section 2.4. This takes $O\big(n^2(C + |\mathcal{D}|)\big)$ time and $O\big(nC + n^2|\mathcal{D}|\big)$ space. (Now we do not reuse the space.)

Then, to generate one of the $T$ independent samples, do the following:

1. Generate a random linear extension $L \supseteq P$ proportionally to $\prod_v \alpha_v(L_v)$ by stochastically backtracking the recurrence (10). Using the Alias method (Theorem 13) this takes only $O(n)$ time and space, since there are $n$ recursive steps.

2. For each node $v \in N$, generate a random parent set $A_v \in \mathcal{C}_v \cap 2^{L_v}$ by stochastically backtracking the deflating zeta transform as described in Section 2.4. This takes $O(n^2)$ time and space by the arguments given in Section 2.4.

3. Output the obtained DAG $A$.

Thus the algorithm has the claimed complexity in total.

### 5.4 Proof of Theorem 17

We will derive an algorithm that solves the problem in the claimed time and space. The algorithm extends the forward–backward algorithm of Koivisto (2006) to accommodate the partial order constraint.

Let the functions $\alpha_v$, for each node $v \in N$, be as defined in the proof of Theorem 15. Define the "forward" function $F : \mathcal{D} \to \mathbb{R}$ as in definition (10), that is, by letting $F(\emptyset) = 1$

and, recursively,

$$F(Y) \;=\; \sum_{\substack{v\in Y\\ Y\setminus\{v\}\in\mathcal{D}}} F\big(Y\setminus\{v\}\big)\,\alpha_v\big(Y\setminus\{v\}\big)\,, \quad \text{for } \emptyset\subset Y\in\mathcal{D}\,.$$

Likewise, define the "backward" function $B:\mathcal{D}\to\mathbb{R}$ by letting $B(N)=1$ and, recursively,

$$B(Y) \;=\; \sum_{\substack{v\in N\setminus Y\\ Y\cup\{v\}\in\mathcal{D}}} \alpha_v\big(Y\big)\,B\big(Y\cup\{v\}\big)\,, \quad \text{for } N\supset Y\in\mathcal{D}\,.$$

Furthermore, for each node $t\in N$, let

$$\gamma_t(A_t) = \sum_{\substack{Y\in\mathcal{D}\\ Y\supseteq A_t}} F\big(Y\big)\,B\big(Y\cup\{t\}\big)\,, \quad \text{for } A_t\in\mathcal{C}_t\,.$$

**Lemma 19** *It holds that*

$$\varphi^{st}(P) \;=\; \sum_{s\in A_t\in\mathcal{C}_t} \varphi_t(A_t)\,\gamma_t(A_t)\,. \tag{11}$$

**Proof** Consider a fixed pair of nodes $s,t\in N$. Starting from (9), write

$$\varphi^{st}(P) \;=\; \sum_{L\supseteq P}\left(\sum_{A_t\subseteq L_t}\varphi_t^{st}(A_t)\right)\prod_{v\neq t}\alpha_v(L_v) \;=\; \sum_{A_t\subseteq N\setminus\{t\}}\varphi_t^{st}(A_t)\,\overbrace{\sum_{\substack{L\supseteq P\\ L_t\supseteq A_t}}\prod_{v\neq t}\alpha_v(L_v)}^{\text{Denote by }\gamma'_t(A_t).}\,.$$

Note, that $\varphi_t^{st}(A_t)$ vanishes unless $s\in A_t$ and otherwise equals $\varphi_t(A_t)$, which in turn vanishes unless $A_t\in\mathcal{C}_t$. Thus, it remains to show that the just-introduced function $\gamma'_t$ equals $\gamma_t$. To see this, we split the sum over $L\supseteq P$ and $L_t\supseteq A_t$ into two nested sums that first iterate over $L_t\supseteq A_t$ such that $L_t\in\mathcal{D}$ and then over $L_v$ for $v\neq t$. Furthermore, once $L_t$ is fixed in the outer sum, then the inner sum must have $L_v\subset L_t$ for $v\in L_t$ and $L_v\supseteq L_t\cup\{t\}$ for $v\in N\setminus(L_t\cup\{t\})$. The inner sum can thus be split into two independent sums, as follows:

$$\gamma'_t(A_t) = \sum_{\substack{L_t\supseteq A_t\\ L_t\in\mathcal{D}}}\left(\sum_{L'\supseteq P[L_t]}\prod_{v\in L_t}\alpha_v(L'_v)\right)\left(\sum_{L'\supseteq P[N\setminus L_{t+}]}\prod_{v\in N\setminus L_{t+}}\alpha_v(L'_v\cup L_{t+})\right),$$

where $L_{t+}$ is a shorthand for $L_t\cup\{t\}$. To complete the proof, we have to show that

$$F(L_t) = \sum_{L'\supseteq P[L_t]}\prod_{v\in L_t}\alpha_v(L'_v)$$

and

$$B(L_{t+}) = \sum_{L'\supseteq P[N\setminus L_{t+}]}\prod_{v\in N\setminus L_{t+}}\alpha_v(L'_v\cup L_{t+})\,.$$

The first equation follows directly from the proof of Lemma 18. The proof for the second equation is analogous, and is thus not repeated here. ■

We arrive at the following algorithm:

**Algorithm** AllArcs
Input: partial order $P$ on $N$ and $\varphi_v(A_v)$ for $(v, A_v) \in N \times \mathcal{C}_v$.
Output: $\varphi^{st}(P)$ for all pairs $s, t \in N$.

1. Compute $\alpha_v(Y)$ for all $(v, Y) \in N \times \mathcal{D}$.
2. Compute $F(Y)$ and $B(Y)$ for all $Y \in \mathcal{D}$.
3. For each $t \in N$:
   (a) Compute $\gamma_t(A_t)$ for all $A_t \in \mathcal{C}_t$.
   (b) For each $s \in N \setminus \{t\}$:
   Compute $\varphi^{st}(P)$ using (11).

The complexity of the first two steps is clearly within the claimed budget—these steps are essentially the same as in our algorithm for the DAGE problem.

Step 3a is the upward-variant of the inflating zeta transform problem and can thus, by Theorem 8 and Remark 12, be solved in $O(n^2|\mathcal{C}_t| + n|\mathcal{D}|)$ time, for each $t$. This gives $O(n^2(C + |\mathcal{D}|))$ time in total. The space requirement is, again, clearly within the claimed budget, since the same space can be reused for different nodes $t$.

Step 3b takes only $O(n|\mathcal{C}_t|)$ time for each $t$, thus $O(nC)$ in total. The additional space requirement is negligible.

### 5.5 Special Cases: Regular Parent Set Collections and Bucket Orders

We have formulated our results in a very general setting where (i) the collections $\mathcal{C}_v$ of potential parent sets can be arbitrary for each node $v$, and (ii) the partial order $P$ can be arbitrary. In our bounds for the time and space requirements we have paid a relatively high cost for this generality: in many cases we obtained a running time bound of $O(n^2(C+|\mathcal{D}|))$.

It appears that these bounds can be reduced significantly if we restrict (i') the parent set collections to all sets of size at most some maximum indegree $k$, and (ii') the partial orders to bucket orders. This restricted setting was, in fact, considered already by Koivisto and Sood (2004, Theorem 12), who showed that (using our terminology) the DAGE problem can be solved in $O(C+n2^b)$ time, when the maximum bucket size is $b$. We note that this bound hides a factor that is linear in $k$. For the term that depends on the number of downsets, the improvement is thus from about $n^2(n/b)2^b$ to $n2^b$, assuming a balanced bucket order (see Example 4). We have observed that in this restricted setting similar improved bounds can be obtained also for the DAG sampling problem and for the arc probabilities problem (we omit details).

## 6. Experimental Results

We have implemented the proposed partial-order-MCMC method, including the extensions based on $MC^3$ and AIS, for the special case of bucket orders (see Examples 4–7).[2] We will refer to these three variants of the methods simply as MCMC, $MC^3$, and AIS. This section reports experimental results on a selection of data sets of different characteristics. Details of the data sets and the employed Bayesian models are given in Section 6.1. Implementation details of the computational methods are given in Section 6.2.

We aim to answer four main questions: Does sampling partial orders provide us with a significant advantage over sampling linear orders? How accurate is AIS as compared to $MC^3$? Does the bias correction approach (i.e., scaling by the number of linear extensions) work in practice? How well can we estimate and lower bound the marginal likelihood of the model? We address these questions in Sections 6.3–6.6, respectively.

### 6.1 Data Sets, Model Parameters, and Features of Interest

Table 1 lists the data sets used in our experiments. The *Flare*, *German*, *Mushroom*, and *Spambase* data sets are obtained from the UCI Machine Learning Repository (Lichman, 2013). The *Alarm* data set was generated from the Alarm network (Beinlich et al., 1989). Of the *Mushroom* data set we used both the whole data set and a subsample consisting of 1000 randomly selected records of the data set. We will refer to these two versions as *Mushroom-8124* and *Mushroom-1000*. The data set with the fewest attributes, *Flare*, was used only for examining the performance of the bias correction method of Ellis and Wong (2008).

For each data set we employed the modular and the order-modular uniform Dirichlet–multinomial model described in Examples 2 and 3. In these models we set the maximum indegree parameter $k$ to 4 for all data sets, except for *Spambase*, for which we set the value to 3 in order to keep the per-sample computations feasible.

We focus on the estimation of the arc posterior probabilities and the marginal likelihood of the model. For comparison purposes, we also computed exact values of these quantities on the *Flare*, *German*, and *Mushroom* data sets using the algorithms of Koivisto and Sood (2004; 2006) and Tian and He (2009).

### 6.2 Implementation Details

We made the following implementation choices in the MCMC method:

*Sampling space.* The sampling space was set to the balanced bucket orders of maximum bucket size $b$ (see Example 4). Separately for each data, we set the parameter $b$ to a value as large as possible, subject to the condition that its impact to the running time is no more than about 2 times the impact of the terms that do not depend on $b$. Table 1 shows the obtained values. Note that linear orders correspond to the special case of $b = 1$.

*Proposal distribution.* We employed swap proposals, as described in Example 7.

2. The program BEANDisco, written in C++, is publicly available at `www.cs.helsinki.fi/u/tzniinim/BEANDisco/`.

| Name | $n$ | $m$ | $k$ | $b$ | CPU time | Number of iterations: Linear orders | Number of iterations: Bucket orders |
|---|---|---|---|---|---|---|---|
| *Flare* | 13 | 1066 | 4 | – | 1 d | $6.1 \times 10^8$ | – |
| *German* | 20 | 1000 | 4 | 7 | 4 d | $2.4 \times 10^8$ | $1.0 \times 10^8$ |
| *Mushroom* | 22 | 8124 | 4 | 7 | 4 d | $1.9 \times 10^8$ | $9.6 \times 10^7$ |
| *Spambase* | 58 | 4601 | 3 | 9 | 4 d | $1.5 \times 10^7$ | $9.6 \times 10^6$ |
| *Alarm* | 37 | 1000 | 4 | 10 | 4 d | $1.0 \times 10^7$ | $6.3 \times 10^6$ |

Table 1: Data sets and basic parameters used in the experiments. Abbreviations: number of attributes $n$, number of data records $m$, maximum indegree $k$, maximum bucket size $b$.

*Burn-in iterations.* Always 50% of the samples were treated as burn-in samples that were not included in the estimates of the quantities of interest.

*Thinning.* We included only every 1024th of the visited states in the final sample.

*Number of DAG samples.* Per sampled partial order (after thinning), we draw as many independent DAGs as was possible within 25% of the time needed for sampling the partial order (i.e., 1024 partial orders due to thinning). In order to ensure that the varying per-DAG processing time does not cause any bias, the DAGs were drawn in two phases: First, 10% of the time budget is used to get an estimate $T'$ for the number of DAGs that can be drawn within the remaining time budget. Then, exactly $T'$ DAGs are drawn for the use of the algorithm.

*Running time.* Per configuration we allowed a total running time of 4 days (excluding the additional time used to sample the DAGs), except for the *Flare* data set, for which we only ran some of configurations and at the maximum of 1 day. Table 1 shows the approximate total number of iterations made, both for linear orders and bucket orders.

*Independent runs.* We ran each configuration 7 times, starting from states drawn independently and uniformly at random.

In addition to the above choices, we made the following additional choices in the MC$^3$ and AIS methods:

*Tempering scheme.* We used the linear stepping scheme. For MC$^3$ we varied the number of temperature levels $K$ in $\{3, 15, 63\}$, and the thinning factor was reduced to $1024/(K+1)$ correspondingly. For AIS we set the number of levels proportionally to the data size, $K = K'mn$ where the factor $K'$ varied in $\{1/4, 1, 4\}$. These values were found by preliminary experiments (results not shown).

*Number of chain swap proposals.* For MC$^3$ swaps of adjacent chains were proposed $1000 \times K$ times every time before moving all the chains one step. Here the rationale is that chain

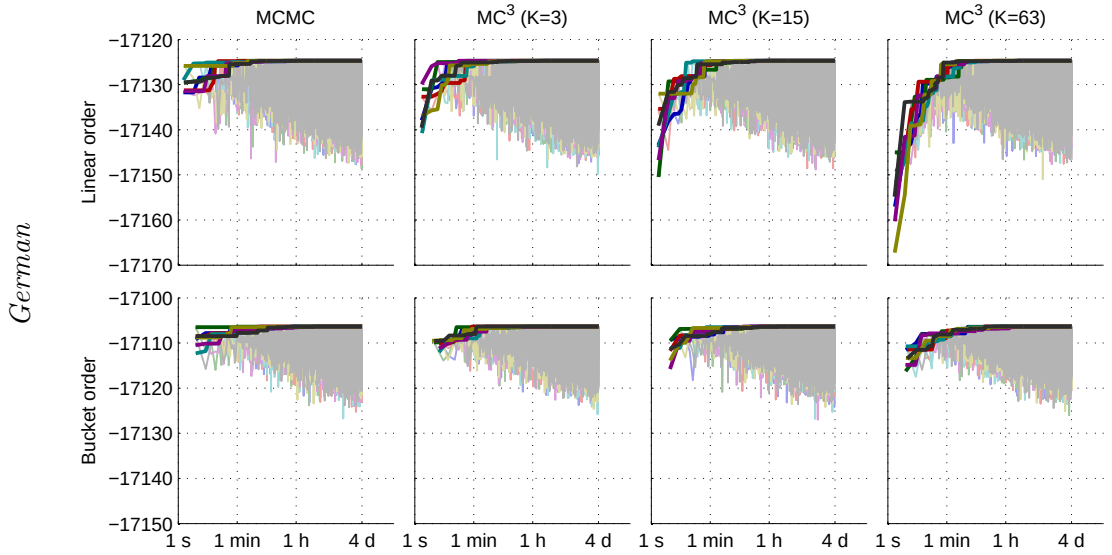



Figure 7: Mixing and convergence of MCMC and MC$^3$ on the *German* data set, for linear orders (top) and bucket orders (bottom). Each panel shows the traces of 7 independent runs (thin pale lines), that is, the natural logarithm of the unnormalized posterior probability of the visited state (y-axis) as a function of the time elapsed (x-axis). The cumulative maximum of these values are also shown for all the 7 runs (thick dark lines). If mixing is good, then all 7 runs should quickly converge to approximately same posterior probability levels. This seems to be the case in all eight panels. Note that posterior probabilities of linear orders and bucket orders are not directly comparable.

swaps are computationally cheap and improve mixing considerably especially when $K$ is large.

*Number of iterations along the coolest chain.* For AIS we ran $K/4$ iterations along the coolest chain. Here the rationale is that collecting a large number of samples from the coolest chain is relatively cheap in comparison to the long annealing schedule. Note that thinning concerns only these $K/4$ iterations.

We will mainly examine the methods' performance as functions of running time. For visualization purposes we did a second round of thinning by an additional factor of 10. Note however that this additional thinning does not affect the estimates, which are based on the full set of samples obtained after the first round of thinning.

### 6.3 Advantage of Partial Orders

We first compared the effect of the sampling space—whether linear orders or bucket orders—to the mixing rate and convergence speed of the Markov chains. We did this for the basic MCMC as well as for MC$^3$ with varying number of temperature levels $K$ (Figures 7–9). We found that on the *German* and *Alarm* data sets the two sampling spaces perform about

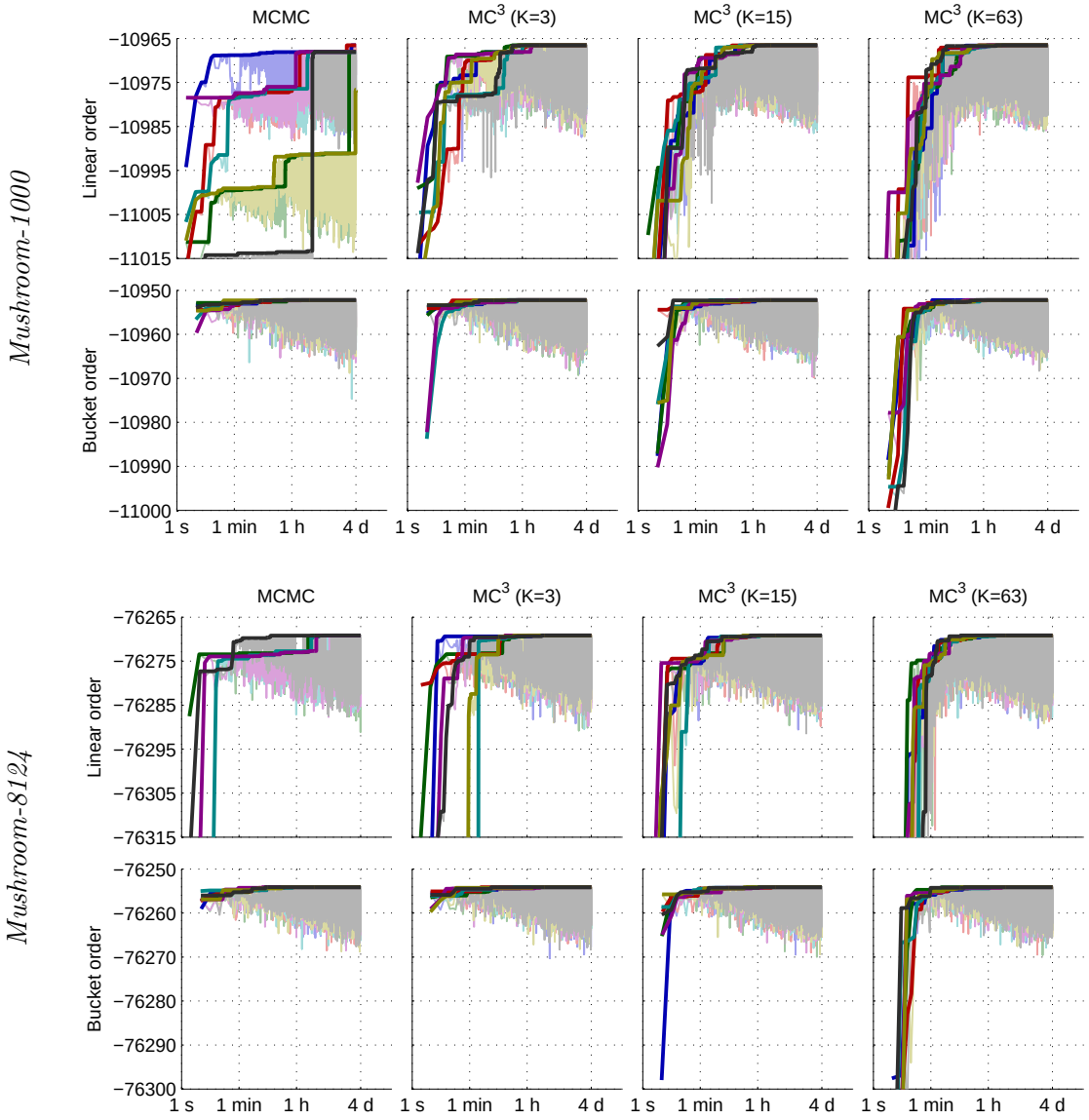



Figure 8: Mixing and convergence of MCMC and MC$^3$ on the *Mushroom* data sets. See the caption of Figure 7 for further descriptions. The top left panels of both data sets are examples of bad mixing. Note that MCMC with linear orders fails completely on the *Mushroom-8124* data set for 3 out of the 7 runs, and consequently the traces do not achieve the visible range of the y-axis.

equally well. On both these data sets, the chains seem to converge within a couple of minutes. We also observe that that tempering is not particularly beneficial. The results confirm that linear orders can perform very well on some data sets. The harder data sets, *Mushroom* and *Spambase*, on the other hand, separate the two sampling spaces: the performance of bucket orders is superior to linear orders. While the difference is particularly

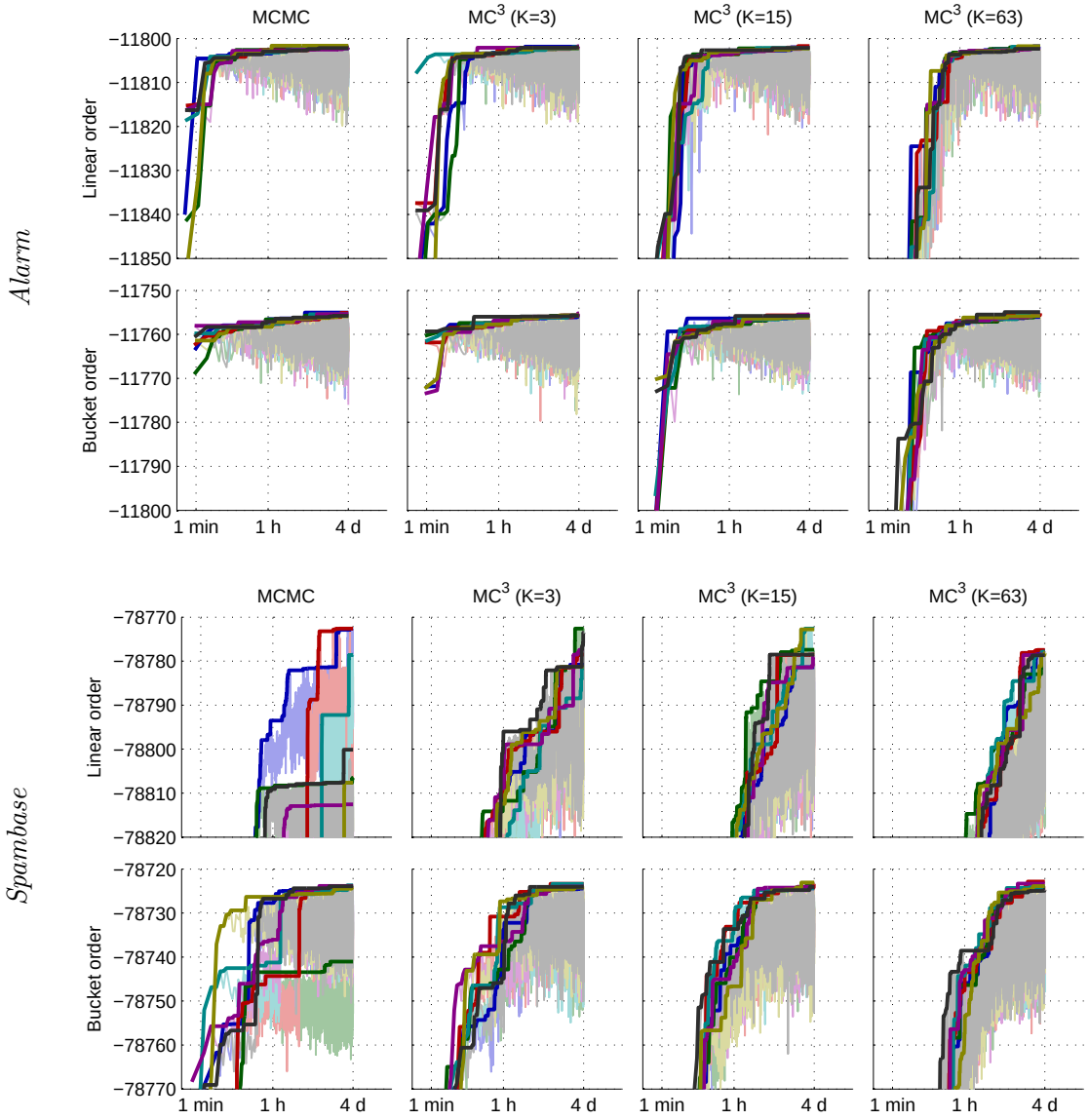



Figure 9: Mixing and convergence of MCMC and MC$^3$ on the *Alarm* and *Spambase* data sets. See the caption of Figure 7 for further descriptions.

large for the basic MCMC method, the difference remains significant for the MC$^3$ variants: On *Mushroom*, all MC$^3$ runs seem to converge, but the convergence is quicker when using bucket orders. On *Spambase*, it is not clear if any of the linear-order-based runs managed to converge within the given time budged, while all the bucket-order-based MC$^3$ runs appear to converge.

Next we investigated how the differences in mixing and convergence rates translate to differences in the accuracy of the arc posterior probability estimates under the order-modular model (i.e., without bias correction). For each pair of nodes, we measure the

accuracy by the standard deviation of the estimates in the 7 independent runs. When the exact values were available we also gauged the accuracy by the median of the 7 absolute errors. In a worst-case spirit, we report the respective the *largest standard deviation* and the *largest median error*, which we obtain by taking the maximum over the node pairs. We found that, qualitatively, the results follow closely the mixing and convergence behavior of the methods (Figure 10). Specifically, on the *German* and *Alarm* data sets all the methods perform about equally well, whereas on the *Mushroom* data sets the estimates we obtain with bucket orders are significantly more accurate than the estimates we obtain with linear orders, the difference being about one order of magnitude. On the *Spambase* data set none of the methods performs particularly well, which can be probably explained by the insufficiency of the allocated running time for achieving proper convergence. As the exact values are not available for *Spambase*, the small empirical standard deviations might be just due to similar yet insufficient convergence of the 7 runs. On the other hand, we also observe that, in general, the largest standard deviation reflects very well the largest median error.

Based on these results with different number of temperature levels $K$ for MC$^3$, we fixed $K = 15$ for presenting the remaining results in the next subsections. This value of $K$ appears to make a good compromise between a large number of iterations and fast mixing.

### 6.4 Accuracy of AIS

To study to performance of AIS, we compared the obtained arc posterior probability estimates to those obtained with MC$^3$, now with $K = 15$ only (Figure 11, left; Figure 12, left). We found that on the easiest data set, *German*, the methods perform almost identically, regardless of the value of the $K'$ parameter. This holds also on the *Mushroom* data sets, provided that $K'$ is large enough (1 or 4). Furthermore, we observe that bucket orders are superior to linear orders also in the case of AIS. On the *Alarm* data set AIS is slightly behind MC$^3$ even when sampling bucket orders, and on the *Spambase* data set the difference of the methods is larger.

Based on these results with different values of $K'$, we fixed $K' = 1$ for presenting the remaining results in the next subsections. This value of $K'$ appears to make a good compromise between a long annealing schedule and a relatively large number of independent samples.

### 6.5 Efficiency of Bias Correction

So far we have only discussed the results obtained under the order-modular model. We next turn to the results under the modular model, which we obtain by applying the bias-corrected estimators. The results are presented graphically in Figures 11–13.

First we studied the subproblem of counting the linear extensions of a given DAG. We generated random DAGs for a varying number of nodes $n$, setting the maximum indegree to either 3 or 6, and then ran the exact dynamic programming algorithm (from Section 2.2). We found that while both the time and the space complexity of the algorithm grow exponentially in $n$, the computations are feasible as long as $n$ is at most about 40 for sparse DAGs, or at most about 60 for dense DAGs (Figure 13a). For example, the linear extensions of a 49-node DAG of maximum indegree 6 can typically be counted within a couple of seconds.

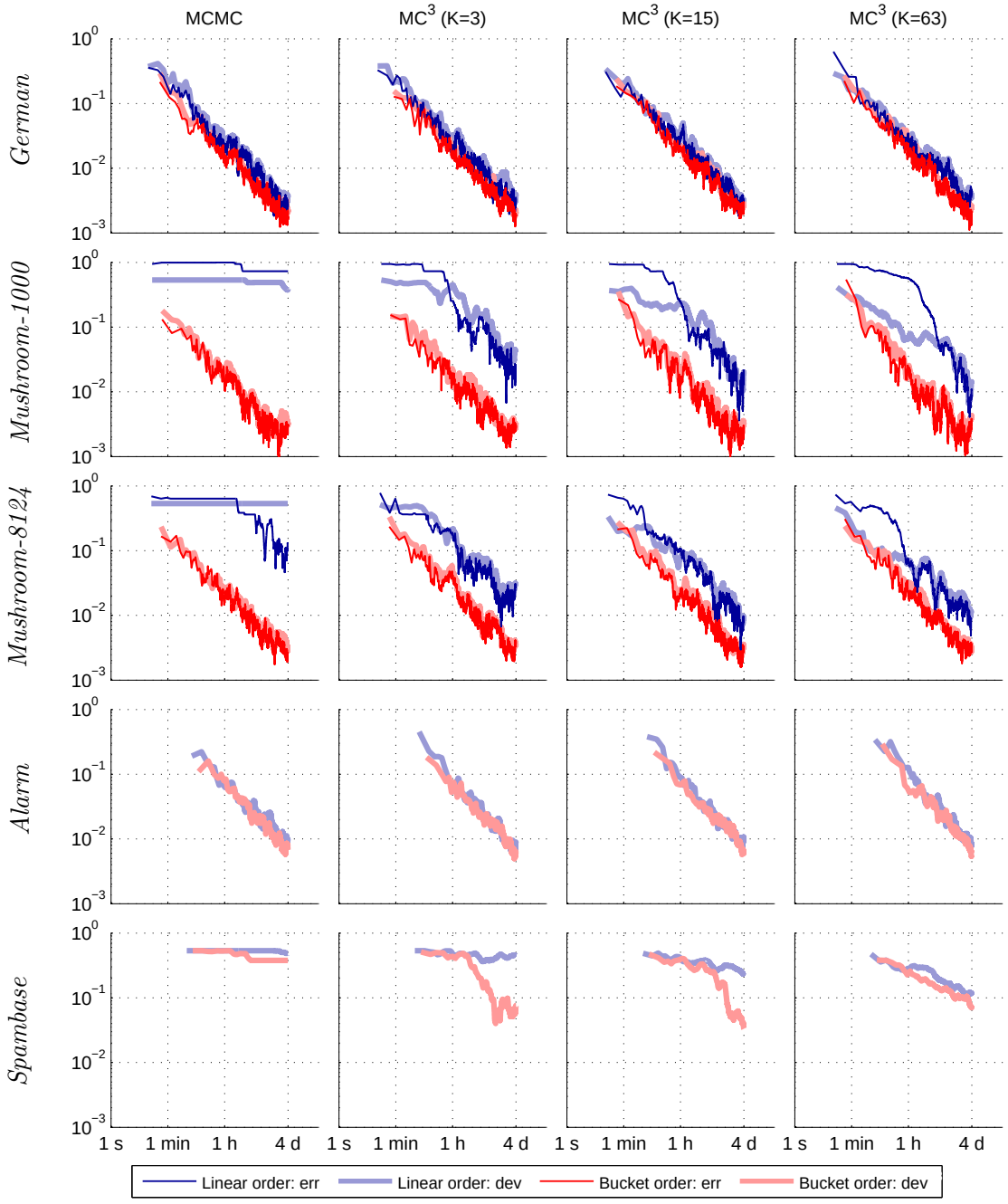



Figure 10: The accuracy of the arc posterior probability estimates under an order-modular model, for MCMC and MC$^3$. For each method the largest median error (err) and the largest standard deviation (dev) are shown as a function of the time elapsed.

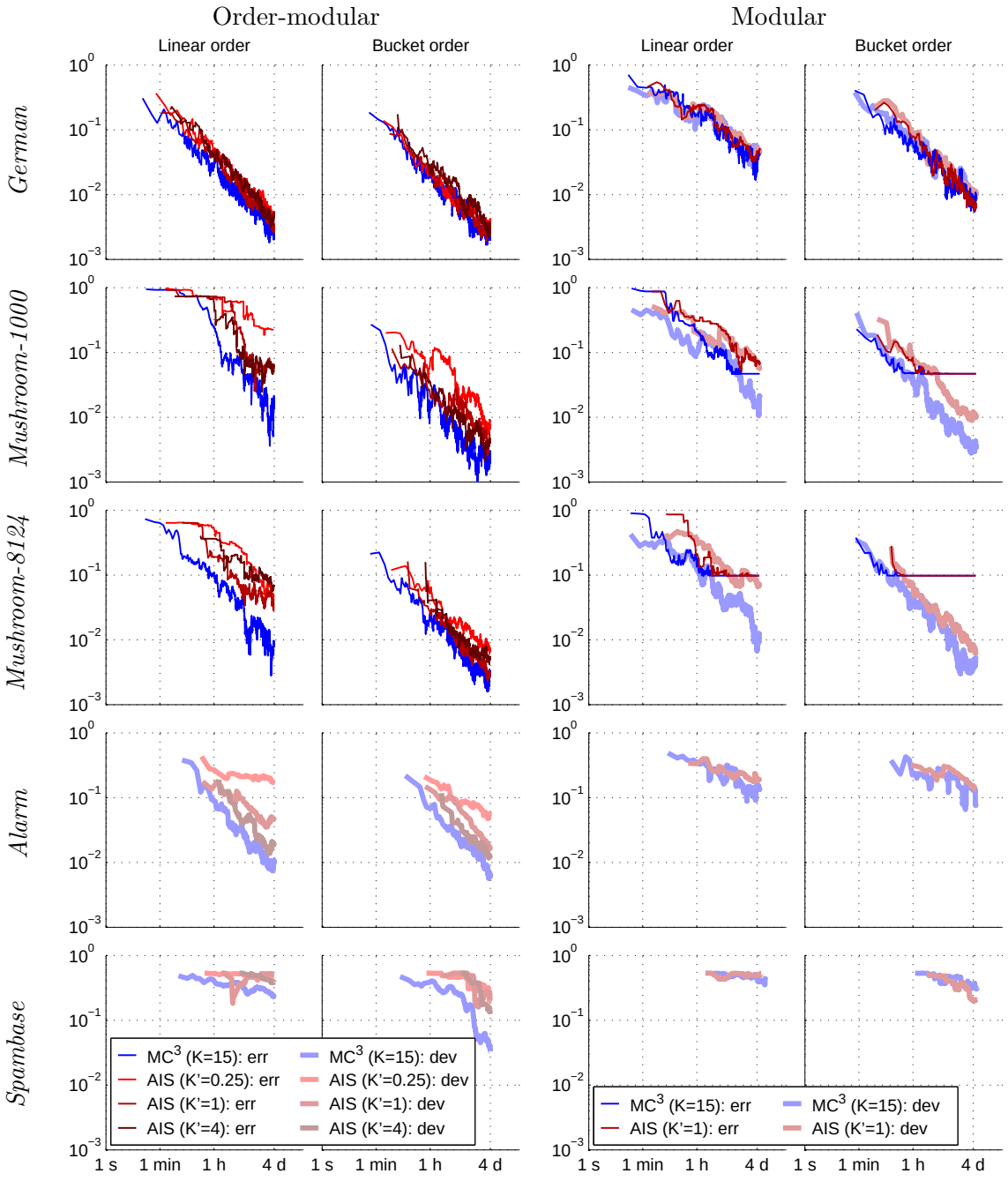



Figure 11: The accuracy of the arc posterior probability estimates under an order-modular model (left) and a modular model (right), for AIS and MC$^3$. For clarity, only the largest median errors are shown for the *German* and *Mushroom* data sets.

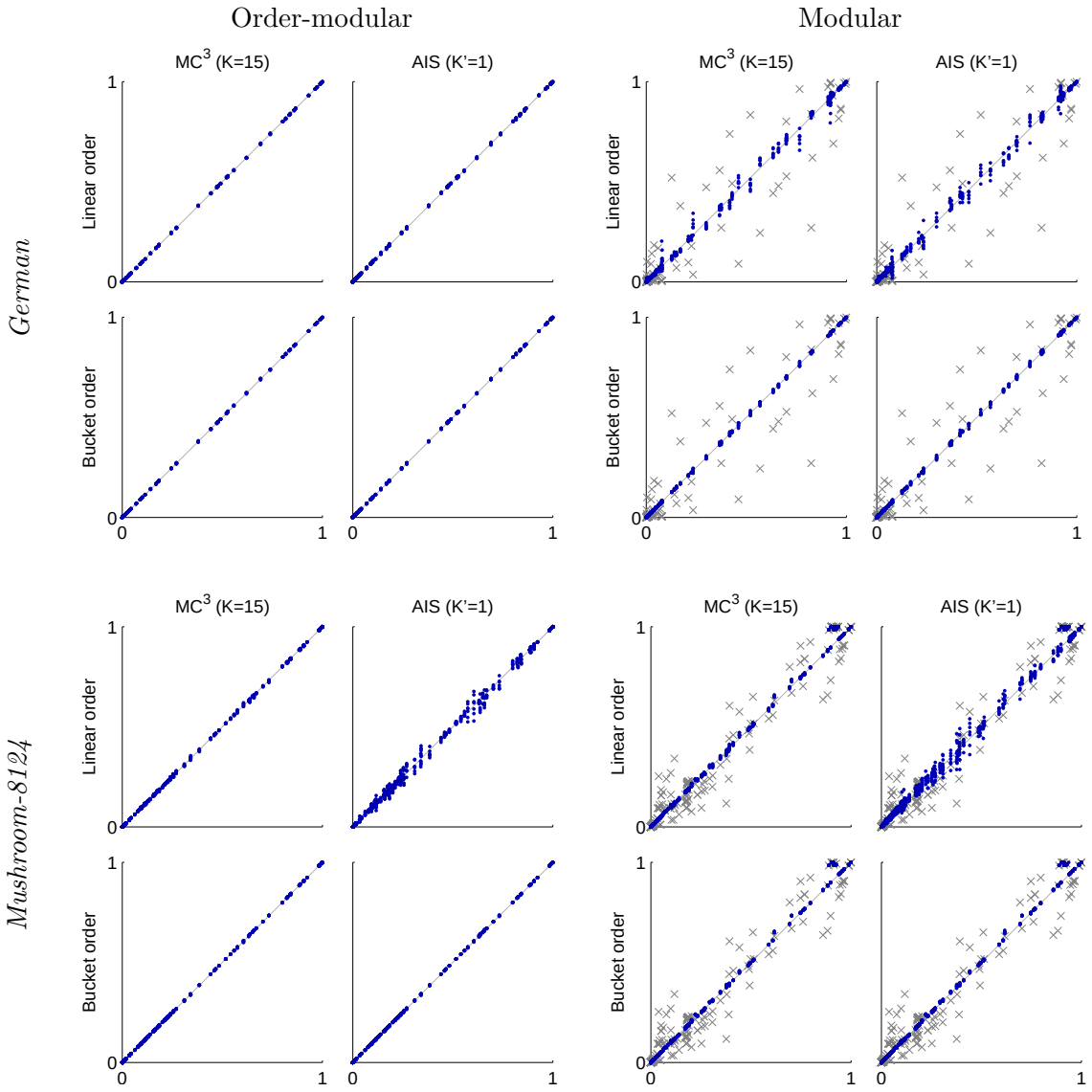
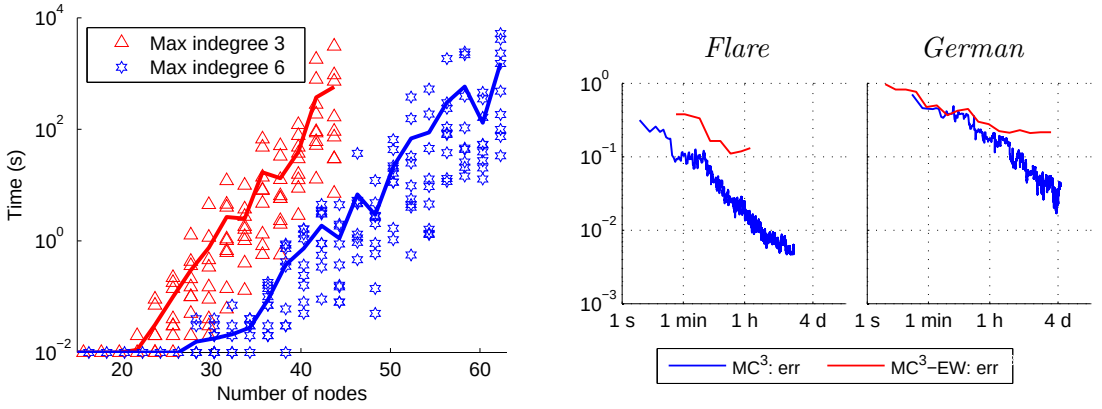



Figure 12: The arc posterior probability estimates for 7 independent runs (y-axis) plotted against the exact values (x-axis), under an order-modular and modular model. For the modular model also the biased values (exact under the order-modular model) are plotted against the unbiased exact values (gray ×).

However, the simple dynamic programming algorithm does not exploit sparsity well and so the performance degrades for maximum indegree 3.

Then we compared our bias correction method to that of Ellis and Wong (2008), which we refer to as the *EW method* in the sequel. The EW method works as follows. First it draws a sample of node orderings from an approximate posterior distribution, like the order-MCMC method of Friedman and Koller (2003). Then, from each sampled node ordering, it




(a) The speed of counting linear extensions.

(b) The accuracy of bias correction methods.

Figure 13: (a) The runtime required to count the linear extensions of 9 random DAGs with a varying number of nodes and a varying maximum indegree. Runtimes less than 0.01 seconds are rounded up to 0.01. The mean of the runtimes is shown by a solid line. The available memory was limited to 16 GB. No runtime estimates are shown when the memory requirement exceeded the limit for at least one of the 9 DAGs. (b) The accuracy of the proposed bias correction method (MC$^3$) compared to the EW method with $\epsilon = 0.05$ (MC$^3$-EW). Both methods are based on sampling linear orders by MC$^3$ under an order-modular model. The largest median error of the arc posterior probability estimates over 7 independent runs is shown as a function of the time elapsed. The runs of the EW method terminated as soon as one of the 7 runs ran out the 20 GB of memory allocated for storing the DAGs. For the EW method the estimates were computed at doubling sample sizes $2, 4, 8, \ldots$, to make sliding burn-in periods computationally feasible.

generates a number of independent DAGs from the conditional posterior distribution until the total posterior mass of the unique DAGs obtained is at least $1 - \epsilon$, where $\epsilon > 0$ is a parameter of the method. Finally, the DAGs so obtained for each node ordering are merged into a single set, duplicates are removed, and each DAG is assigned an importance weight proportional to the posterior probability of the DAG.

The results of the comparison on the *Flare* and *German* data sets are shown in Figure 13b. On these data sets the EW method turned out to be inferior to the proposed method. On the other data sets (*Mushroom-1000*, *Mushroom-8124*, *Alarm*, *Spambase*), the EW method either ran out of memory immediately or was able to process only a couple of node ordering samples, yielding poor estimates (the largest media error close to 1; results not shown).

Finally we compared the arc posterior probability estimates under the two models, order-modular and modular (Figure 11, right). The results confirm the expectation that the estimates are less accurate under the modular model. The weaker performance is due to the additional importance sampling step needed for bias correction, which reduces the effective sample size. Otherwise, the earlier conclusions hold also under the modular model:

bucket orders outperform linear orders and $MC^3$ is slightly superior to AIS. However, the bias correction method also brings a new feature: on the *Mushroom* data sets the largest median error stops decreasing at some point, stagnating at an error of about 0.10 (while the largest standard deviation continues decreasing). This phenomenon is due to a few node pairs for which the arc posterior probability is very close to 1 under the order-modular model, but around 0.90 under the modular model. The culprits can be located in Figure 12 (right) as a cluster of points in the upper-right corner of the panels. Note that for the other node pairs the arc posterior probability estimates are very accurate.

### 6.6 Estimating the Marginal Likelihood

For estimating the marginal likelihood AIS clearly outperforms $MC^3$, as expected (Figure 14). The relatively small number of chains (i.e., temperature levels) in $MC^3$ leads to a large fluctuation in the estimates within a single run and between independent runs. AIS, on the other hand, benefits from the long annealing scheme and produces very accurate estimates on the *German* and *Mushroom* data sets. On the *Alarm* data set the variance of the estimates is larger for both methods. Yet the estimates of AIS seem to converge well. The *Spambase* data set is, again, the hardest instance for both method, the results suggesting insufficient convergence of the samplers.

We also observed that when the estimates of AIS are close to the exact value, then also the high-confidence lower bounds are very good. Indeed, on the *German* and *Mushroom* data sets the lower bounds obtained with the square-root lower-bounding scheme are within an absolute error of about 0.4 or less in the logarithmic scale, hence within a relative error of about $e^{0.4} - 1 \approx 0.4$ or less. Whether the model is order-modular or modular affects the accuracy of the marginal likelihood estimates considerably on the *German* data set but very little on the *Mushroom* data sets.

## 7. Conclusions and Future Work

We have investigated a sampling-based approach to Bayesian structure learning in Bayesian networks. Our work has been inspired to a large extent by the order-MCMC method of Friedman and Koller (2003), in which the sampling space consists of all possible linear orders of the nodes. Our partial-order-MCMC methods advance the methodology in four dimensions:

1. *Smoother sampling space.* The space of partial orders is smaller still than the space of linear orders. Also, the posterior distribution tends to be smoother, because the posterior probability of each partial order is obtained by summing the posterior probabilities of its linear extensions. Our empirical results agree with these expectations, and show instances where partial-order-MCMC performs significantly better than order-MCMC.

2. *Arbitrary structure priors.* We proposed a method to correct the bias that arises because the samples are drawn (approximately) from an order-modular rather than a modular posterior distribution. The correction amounts to a simple scaling term per sampled DAG. The term only depends on the number of linear extensions of the DAG, and as we showed, it can be computed sufficiently fast when the number of

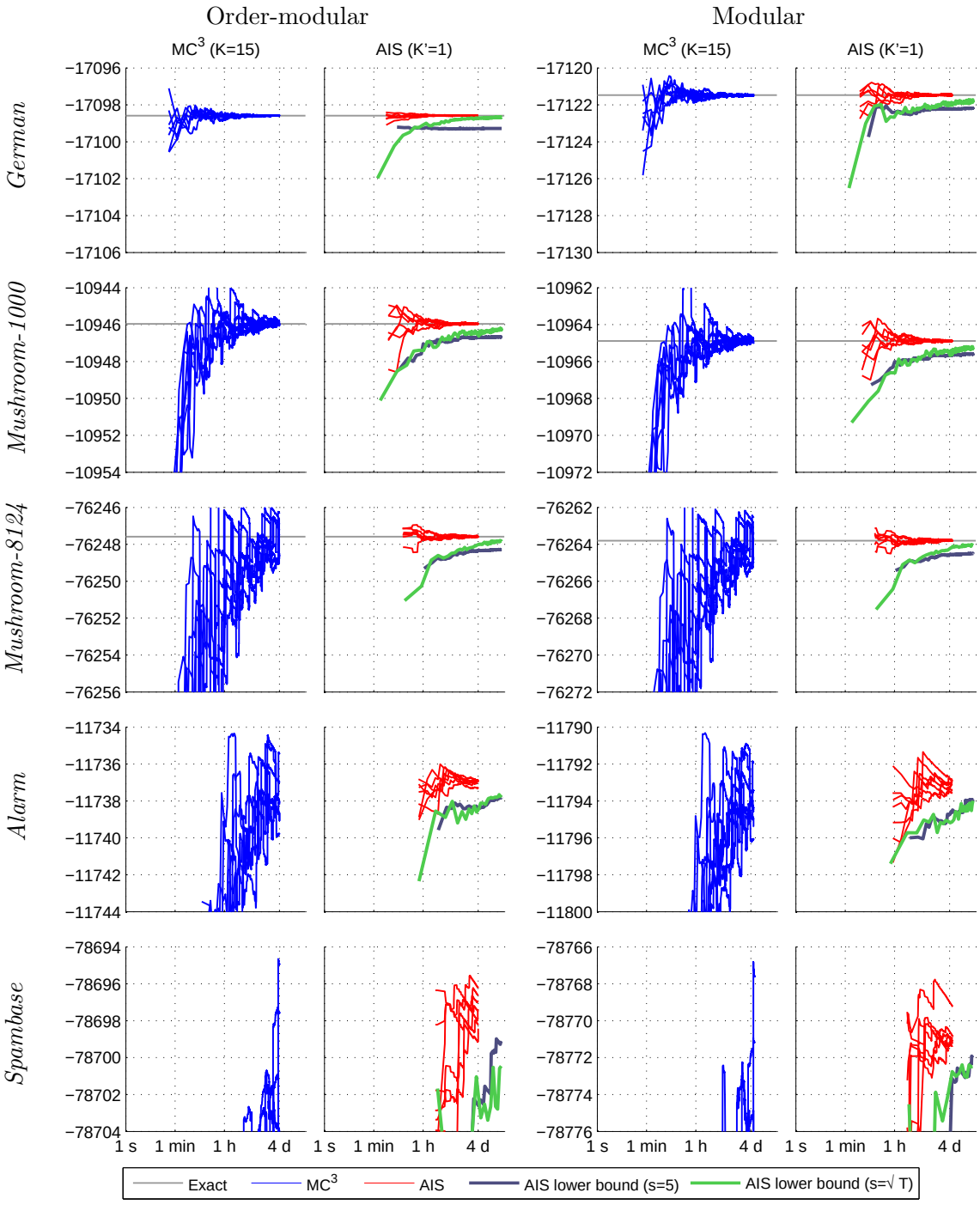



Figure 14: The marginal likelihood of the model estimated by 7 independent runs of AIS and MC$^3$. For AIS, shown are also lower bound estimates obtained with all the samples pooled together (thus extending the total running). The lower bound schemes are as described in Example 8 and 9. On the y-axis is the natural logarithm of the estimate or exact value.

nodes is moderate, say, at most 40. Our empirical results confirmed that, in general, the correction works well also in practice—some rare cases where the correction fails are discussed below.

For convenience, we focused on the special case where the correct model is the modular counterpart of an order-modular model. In principle, it is straightforward to extend the estimators to accommodate an arbitrary model, as long as the corresponding posterior probability function (a) can be efficiently evaluated for any given DAG (up to a normalizing constant) and (b) has a support that is contained in the support of the order-modular sampling distribution. The statistical efficiency of the estimator may, however, deteriorate if the true distribution is far from the sampling distribution.

3. *Efficient parallel computation.* We observed that the annealed importance sampling method (AIS) is easy to run in parallel, since the method is designed to produce independent samples. We compared AIS empirically to another tempering method, Metropolis-coupled MCMC ($MC^3$), and found that AIS-based estimates are slightly less accurate for arc posterior probabilities but significantly more accurate for the marginal likelihood of the model.

4. *Quality guarantees.* We also observed that AIS allows us to compute high-confidence lower bounds for the marginal likelihood. We showed that the lower bounds are very good, within a factor of about 1.2 on the data sets for which exact values were available. Admitted, lower bounds on the marginal likelihood is just a small step toward accuracy guarantees more generally.

These advancements (1–4) upon the order-MCMC method come essentially "for free" concerning both computational and statistical efficiency. Indeed, sampling partial orders is not more expensive than sampling linear orders, provided that the partial orders are sufficiently thin (i.e., the number of downsets is small). Namely, the bottleneck in both methods is the need to visit a large number of potential parent sets of the nodes. Likewise, while counting the linear extensions of a sampled DAG can be computationally demanding, the computational cost is compensated by the relatively small number of sampled DAGs (after thinning) as compared to the total number of partial order samples (before thinning).

The proposed methods also have shortcomings that call for further investigations. First, the complexity of the per-sample computations grows rapidly with the number of nodes $n$. In particular, the complexity of computing the unnormalized posterior probability of a partial order does not scale well with the indegree $k$, and the computations become impractical when, say, $n \geq 60$ and $k \geq 4$. For example, on the *Spambase* data set ($n = 58$, $k = 3$) we observed that the large number of potential parent sets rendered the allocated running time of 4 days insufficient for proper convergence of the sampling methods. To significantly expedite these computations, a plausible idea is to resort to approximations instead of exact computation. In the sampling space of *linear* node orders, the approach has proved successful, provided that the number of data records is large (Friedman and Koller, 2003; Niinimäki and Koivisto, 2013a). In contrast, the problem of counting the linear extensions of a sampled DAG seems to get harder for sparser DAGs. However, we believe this is rather a feature of our simple algorithm—our preliminary results with a more sophisticated algorithm suggest that sparsity can actually be turned into a computational advantage.

Second, the theoretical accuracy guarantees of the proposed methods are quite modest. For example, currently the methods do not produce good lower or upper bounds for the arc posterior probabilities. Thus the user has to resort to monitoring empirical variance and related statistics that do not always generalize well beyond the obtained sample. We saw a warning example of that on the *Mushroom* data set, where the bias-corrected estimates had small variance over 7 independent runs, but where the estimates were biased and the bias did not decrease as the number of samples grew. While this failure concerned only a few node pairs (the estimates being very accurate for the rest), it shows that sometimes the proposed bias correction method is not efficient. It may happen that all the sampled DAGs contain the arc of interest, because the biased, order-modular posterior probability of the arc is close to 1, and yet the unbiased, modular posterior probability of the arc can be much below 1, say, 0.90. In such cases the present bias correction approach may fail. It is an intriguing open question how situations of this kind can be treated or circumvented.

## Acknowledgments

The authors would like to thank Kustaa Kangas for valuable discussions about the problem of counting linear extensions, and Tianqi Li for her contributions to preliminary experiments on the bias correction heuristic of Ellis and Wong (2008). This research was funded in part by the Academy of Finland, Grants 276864 "Supple Exponential Algorithms" and 251170 "Finnish Centre of Excellence in Computational Inference Research COIN".

## Appendix A. Proofs

**Lemma 7** *It holds that $\varphi_n(Y) = \widehat{\varphi}(Y)$ for all $Y \in \mathcal{D}$.*

**Proof** For a set $X$ write $X_i$ for $X \cap \{v_i\}$, and for sets $X, Y$ write $X \subseteq_i Y$ as a shorthand for $X \subseteq Y$ and $X_j = Y_j$ for $j > i$. Let $Y \in \mathcal{D}$. We prove by induction on $i$ that

$$\varphi_i(Y) = \sum_{\substack{X \in \mathcal{D} \\ X \subseteq_i Y}} \varphi(X)\,.$$

The claim then follows, for the condition $X \subseteq_n Y$ is equivalent to $X \subseteq Y$. Note also that the base case of $i = 0$ clearly holds, since $X \subseteq_0 Y$ is equivalent to $X = Y$.

Let then $i > 0$. Suppose first that $v_i \notin Y$ or $Y \setminus \{v_i\} \notin \mathcal{D}$. Then $\varphi_i(Y) = \varphi_{i-1}(Y)$. If $v_i \notin Y$, then the conditions $X \subseteq_i Y$ and $X \subseteq_{i-1} Y$ are equivalent, and by the induction hypothesis the claim follows. Assume then that $v_i \in Y$ and $Y \setminus \{v_i\} \notin \mathcal{D}$. By the induction hypothesis it suffices to show that, if $X \in \mathcal{D}$, then the conditions $X \subseteq_i Y$ and $X \subseteq_{i-1} Y$ are equivalent. Because the latter condition implies the former, it remains to consider the other direction. To this end, suppose the contrary, $X \subseteq_i Y$ but $X_i \neq Y_i$. Thus we must have $v_i \notin X$. However, the assumption $Y \setminus \{v_i\} \notin \mathcal{D}$ and the fact that $Y \in \mathcal{D}$ imply that there is an element $v_j \in Y \setminus \{v_i\}$ such that $v_i v_j \in P$, which, together with the ordering of the elements imply that $i < j$. Therefore, as $X \subseteq_i Y$, we have $v_j \in X$, which contradicts our assumption that $X \in \mathcal{D}$.

Suppose then that $v_i \in Y$ and $Y \setminus \{v_i\} \in \mathcal{D}$. Then $\varphi_i(Y) = \varphi_{i-1}(Y) + \varphi_{i-1}(Y \setminus \{v_i\})$. Let $X \in \mathcal{D}$. Denote the conditions of interest by

$$\begin{aligned} E &= X \subseteq_i Y\,, \\ E_1 &= X \subseteq_{i-1} Y\,, \\ E_2 &= X \subseteq_{i-1} Y \setminus \{v_i\}\,. \end{aligned}$$

By the recurrence and the induction hypothesis, it suffices to show that $E$ holds if and only if exactly one the conditions $E_1$ and $E_2$ holds. Clearly, if either $E_1$ or $E_2$ holds, so does $E$. Suppose therefore that $E$ holds. Now, either $X_i = \{v_i\} = Y_i$, in which case $E_1$ holds (but $E_2$ does not); or $X_i = \emptyset$, in which case $E_2$ holds (but $E_1$ does not). ■

**Lemma 9** *Let $Y$ and $Y'$ be distinct downsets. Then the tails $\mathcal{T}_Y$ and $\mathcal{T}_{Y'}$ are disjoint.*

**Proof** Suppose the contrary that there exists a $X \in \mathcal{T}_Y \cap \mathcal{T}_{Y'}$. Since $Y$ and $Y'$ are distinct, by symmetry we may assume $Y \setminus Y'$ contains an element $w$. Thus $w \notin X$, because $X \subseteq Y'$. Since $\max Y \subseteq X$, we have $w \notin \max Y$. On the other hand, from the definition of $\max Y$ together with transitivity and acyclicity of $P$ it follows that, for every $u \in Y \setminus \max Y$ there exists a $v \in \max Y$ such that $uv \in P$. In particular, there exists a $v \in \max Y$ such that $wv \in P$. Since $w \notin Y'$ and $Y'$ is in $\mathcal{D}$, it follows from the definition of a downset that $v \notin Y'$. But this is a contradiction, because we had $v \in \max Y$ and $\max Y \subseteq X \subseteq Y'$, which imply $v \in Y'$. ■

**Lemma 10** *Let $X \subseteq N$. Let $Y = \{u : uv \in P, v \in X\}$, that is, the downward-closure of $X$. Then $Y \in \mathcal{D}$ and $X \in \mathcal{T}_Y$.*

**Proof** To see that $Y$ is a downset, suppose the contrary that there exist $v \in Y$ and $uv \in P$ such that $u \notin Y$. By the definition of $Y$, there must be $vw \in P$ such that $w \in X$. But since $P$ is transitive, $uw \in P$ and thus $u \in Y$ which is a contradiction.

Consider then the second claim, that $\max Y \subseteq X \subseteq Y$. The second inclusion follows from the reflexivity of $P$. For the first inclusion, note that that $\max Y = \max X$, since $Y$ only contains elements $u$ such that $uv \in P$ for some $v \in X$.

■

**Lemma 11** *Let $Z \subseteq N$. Then $\mathcal{D} \cap 2^Z = \mathcal{D} \cap 2^Y$, where the set $Y \in \mathcal{D}$ is given by*

$$Y = \{v \in Z : \textit{if } uv \in P, \textit{ then } u \in Z\}\,;$$

*in words, $Y$ consists of all elements of $Z$ whose predecessors also are in $Z$.*

**Proof** We show first that $Y \in \mathcal{D}$. Suppose the contrary that $v \in Y$ and $uv \in P$ but $u \notin Y$. By the definition of $Y$ we have $u \in Z$. Since $u \notin Y$, there is a $wu \in P$ such that $w \notin Z$. However, as $P$ is transitive, we have $wv \in P$ and thus $w \in Z$ which is a contradiction.

To complete the proof, we show that $\mathcal{D} \cap 2^Z = \mathcal{D} \cap 2^Y$. Clearly $\mathcal{D} \cap 2^Z \supseteq \mathcal{D} \cap 2^Y$. For the other direction, we consider an arbitrary $X \in \mathcal{D} \cap 2^Z$ and show that $X \subseteq Y$. To this end, consider any $v \in X$. Now, if $uv \in P$, then $u \in X$ (because $X$ is a downset)

and consequently $u \in Z$ (because $X \subseteq Z$). Thus, by the definition of $Y$ we get that $v \in Y$. ■

## References


Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. Introduction to MCMC for machine learning. *Machine Learning*, 50:5–43, 2003.

Mark Bartlett and James Cussens. Advances in Bayesian network learning using integer programming. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 182–191, 2013.

Alexis J. Battle, Martin Jonikas, Peter Walter, Jonathan Weissman, and Daphne Koller. Automated identification of pathways from quantitative genetic interaction data. *Molecular Systems Biology*, 6:379–391, 2010.

Ingo Beinlich, Henri J. Suermondt, R. Martin Chavez, and Gregory F. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the 2nd European Conference on Artificial Intelligence in Medicine (AIME)*, pages 247–256, 1989.

Graham Brightwell and Peter Winkler. Counting linear extensions. *Order*, 8:225–242, 1991.

Russ Bubley and Martin E. Dyer. Faster random generation of linear extensions. *Discrete Mathematics*, 201:81–88, 1999.

Wray Buntine. Theory refinement on Bayesian networks. In *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 52–60, 1991.

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

Jukka Corander, Magnus Ekdahl, and Timo Koski. Parallel interacting MCMC for learning of topologies of graphical models. *Data Mining and Knowledge Discovery*, 17:431–456, 2008.

Daniel Eaton and Kevin Murphy. Bayesian structure learning using dynamic programming and MCMC. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 101–108, 2007.

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

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.

Charles J. Geyer. Markov chain Monte Carlo maximum likelihood. In *Proceedings of the 23rd Symposium on the Interface*, pages 156–163, 1991.

Charles J. Geyer and Elizabeth A. Thompson. Annealing Markov chain Monte Carlo with application to ancestral inference. *Journal of the American Statistical Association*, 90: 909–920, 1995.

Vibhav Gogate and Rina Dechter. Sampling-based lower bounds for counting queries. *Intelligenza Artificiale*, 5:171–188, 2011.

Vibhav Gogate, Bozhena Bidyuk, and Rina Dechter. Studies in lower bounding probabilities of evidence using the Markov inequality. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 141–148, 2007.

Carla P. Gomes, Joerg Hoffmann, Ashish Sabharwal, and Bart Selman. From sampling to model counting. In *Proceedings of the 20th international joint conference on Artifical intelligence (IJCAI)*, pages 2293–2299, 2007.

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.

Michel Habib, Raoul Medina, Lhouari Nourine, and George Steiner. Efficient algorithms on distributive lattices. *Discrete Applied Mathematics*, 110:169–187, 2001.

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

Mikko Koivisto. Advances in exact Bayesian structure discovery in Bayesian networks. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 241–248, 2006.

Mikko Koivisto and Pekka Parviainen. A space–time tradeoff for permutation problems. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 484–492, 2010.

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

Moshe Lichman. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, 2013. URL `http://archive.ics.uci.edu/ml`.

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

Radford Neal. Annealed importance sampling. *Statistics and Computing*, 11:125–139, 2001.

Teppo Niinimäki and Mikko Koivisto. Treedy: A heuristic for counting and sampling subsets. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 469–477, 2013a.

Teppo Niinimäki and Mikko Koivisto. Annealed importance sampling for structure learning in Bayesian networks. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1579–1585, 2013b.

Teppo Niinimäki, Pekka Parviainen, and Mikko Koivisto. Partial order MCMC for structure discovery in Bayesian networks. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 557–565, 2011.

Akimitsu Ono and Shin-Ichi Nakano. Constant time generation of linear extensions. In *Fundamentals of Computation Theory*, pages 445–453, 2005.

Sascha Ott, Seiya Imoto, and Satoru Miyano. Finding optimal models for small gene networks. In *Pacific Symposium on Biocomputing*, pages 557–567, 2004.

Pekka Parviainen and Mikko Koivisto. Bayesian structure discovery in Bayesian networks with less space. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistic (AISTATS)*, pages 589–596, 2010.

Pekka Parviainen and Mikko Koivisto. Finding optimal bayesian networks using precedence constraints. *Journal of Machine Learning Research*, 14:1387–1415, 2013.

Gara Pruesse and Frank Ruskey. Generating linear extensions fast. *SIAM Journal on Computing*, 23:373–386, 1994.

Tomi Silander and Petri Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 445–452, 2006.

Ajit Singh and Andrew Moore. Finding optimal Bayesian networks by dynamic programming. Technical report, Carnegie Mellon University, 2005.

Jin Tian and Ru He. Computing posterior probabilities of structural features in Bayesian networks. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 538–547, 2009.

Michael Vose. A linear algorithm for generating random numbers with a given distribution. *IEEE Transactions on Software Engineering*, 17:972–975, 1991.

Alastair Walker. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software*, 3:253–256, 1977.

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