Journal of Machine Learning Research 5 (2004) 549-573 Submitted 12/03; Revised 4/04; Published 5/04

# Exact Bayesian Structure Discovery in Bayesian Networks

**Mikko Koivisto** MIKKO.KOIVISTO@CS.HELSINKI.FI
*HIIT Basic Research Unit*
*Department of Computer Science*
*University of Helsinki*
*FIN-00014 Helsinki, Finland*

**Kismat Sood** KISMAT.SOOD@KTL.FI
*National Public Health Institute*
*Department of Molecular Medicine*
*FIN-00251 Helsinki, Finland*

**Editor:** David Maxwell Chickering

## Abstract

Learning a Bayesian network structure from data is a well-motivated but computationally hard task. We present an algorithm that computes the exact posterior probability of a subnetwork, e.g., a directed edge; a modified version of the algorithm finds one of the most probable network structures. This algorithm runs in time $O(n2^n + n^{k+1}C(m))$, where $n$ is the number of network variables, $k$ is a constant maximum in-degree, and $C(m)$ is the cost of computing a single local marginal conditional likelihood for $m$ data instances. This is the first algorithm with less than super-exponential complexity with respect to $n$. Exact computation allows us to tackle complex cases where existing Monte Carlo methods and local search procedures potentially fail. We show that also in domains with a large number of variables, exact computation is feasible, given suitable a priori restrictions on the structures; combining exact and inexact methods is also possible. We demonstrate the applicability of the presented algorithm on four synthetic data sets with 17, 22, 37, and 100 variables.

**Keywords:** complex interactions, dynamic programming, layering, structure learning

## 1. Introduction

Structure discovery in Bayesian networks has attracted a great deal of research over the last decade. A Bayesian network specifies a joint probability distribution of a set of random variables in a structured fashion. A key component in this model is the network structure, a directed acyclic graph on the variables, encoding a set of conditional independence assertions. Learning unknown dependencies from data is motivated by a broad collection of applications in prediction and inference (Heckerman et al., 1995b).

Bayesian methods for structure learning concern the posterior distribution of network structures. Different applications require different types of posterior summaries—which are usually hard to compute. For example, when interest is in the prediction of future observations, one ought to integrate over the posterior distribution in the manner of model averaging. When the interest is purely inferential, then one may search for structures, or local structural features, that are highly probable. Since the seminal works of Buntine (1991) and Cooper and Herskovits (1992) numerous algorithms have been presented for these structure learning tasks. However, because the number of possible structures grows super-exponentially with respect to the number of network variables, exact computations are often found infeasible. Indeed, it is known that finding an optimal Bayesian network structure is NP-hard even when the maximum in-degree is bounded by a constant greater than one (Chickering et al., 1995). Consequently, much of the research has focused on inexact methods.

To find a good or an optimal network structure, various generic heuristics, like stochastic local search and genetic algorithms, have been used (see, e.g., Heckerman et al., 1995a; Larrañaga et al., 1996). These methods can be also extended to find equivalence classes of network structures (for recent advances, see Chickering, 2002; Acid and de Campos, 2003; Castelo and Kočka, 2003). A central problem in all these algorithms is that one cannot guarantee the quality of the output. Also, the time requirement, though often practical, may be difficult to estimate beforehand.

Discovering high-probable structural features has not been studied so extensively. Madigan and York (1995) propose a Markov chain Monte Carlo (MCMC) method in the space of network structures. Friedman and Koller (2003) design a more efficient MCMC procedure in the space of variable orders. These algorithms output approximate posterior probabilities of structural features. The approximation quality is not guaranteed in finite runs.

Exact algorithms for structure learning have been presented for very restricted classes of Bayesian networks only. The algorithm by Cooper and Herskovits (1992) is polynomial in the number of variables, but a consistent ordering of the variables is assumed as an input. Chow and Liu (1968) give an efficient algorithm for learning tree structures (i.e., maximum in-degree is one). However, the most efficient exact algorithms, thus far, that allow for arbitrary network structures have super-exponential time complexity (Cooper and Herskovits, 1992; Friedman and Koller, 2003). Consequently, they can be applied only when the number of variables is very small (say, at most 10). Interestingly, however, Chickering (2002) shows that under certain monotonicity assumptions, a greedy search method will find a so called inclusion optimal network structure when the data size approaches infinity—but the time requirement can be exponential.

In this paper, we propose a novel *exact* algorithm for structure discovery in Bayesian networks of a moderate size (say, 25 variables or less). In fact, we consider two versions of the algorithm: one for computing posterior probabilities of structural features; another for finding an optimal structure. We also present a rigorous complexity analysis of the algorithm. This work is motivated by three serial observations summarized below.

First, we can expect that existing methods fail in cases where the posterior “landscape” of network structures is highly complex, including multiple “modes”. This is the case especially when the underlying dependencies show no marginal signals. That is, the edges cannot be detected based on pairwise correlations (which may be all zero). Then it is necessary to consider all possible local dependency structures. This motivates the efforts to extend the scope of exact computation.

Second, it turns out that there are essentially two parallel contributions to the complexity of exact computation. One is due to consideration of all possible local dependency structures in light of the data. This part is unavoidable and may, in practice, actually dominate the overall complexity. Additively to this, another contribution is due to exploration of all graph structures. Although this seems to take more than polynomial time, it turns out that this part does not depend on the size of the data nor the complexity of local models. Thus, for a sufficiently small number of network variables, we really can afford exact exploration through all network structures.

The third observation is that the existing exact algorithms (Cooper and Herskovits, 1992; Friedman and Koller, 2003) can be improved significantly. In particular, we present the first algorithm that averages (or alternatively maximizes) over all network structures in less than super-exponential time.

The remainder of this paper is organized as follows. In Section 2, we recall the ingredients of Bayesian networks and the task of structure discovery. Following the work of Buntine (1991), Cooper and Herskovits (1992), and Friedman and Koller (2003) we, in Section 3, describe the idea of conditioning by orders. Based on this, Section 4 presents a novel algorithm for averaging over all network structures. In Section 5, we modify this algorithm to handle the maximization problem. Possible extensions for large networks are sketched in Section 6. In Section 7, we provide experimental results on four synthetic data sets. Finally, Section 8 concludes with a brief discussion.

## 2. Preliminaries

We define a Bayesian network as a probability model for a vector $x = (x_1, \ldots, x_n)$ of random variables. Throughout this paper, we work on the index set $V = \{1, \ldots, n\}$ rather than on the set of the random variables. If $S = \{i_1, \ldots, i_r\}$ is a subset of $V$ with $i_1 < \cdots < i_r$, we let $x_S$ denote the vector $(x_{i_1}, \ldots, x_{i_r})$. A *graph structure* of a Bayesian network specifies conditional independencies among the variables. We represent the graph structure as a vector $G = (G_1, \ldots, G_n)$, where each $G_i$ is a subset of $V$ and specifies the *parents* $x_{G_i}$ of $x_i$. Only *acyclic graphs* are valid, so that given a graph structure, the probability of $x$ is composed by *local conditional distributions* $p(x_i \mid x_{G_i}, \theta)$ as

$$p(x \mid G, \theta) = \prod_{i=1}^{n} p(x_i \mid x_{G_i}, \theta)\,. \tag{1}$$

Usually local distributions are of some common parametric form, such as Bernoulli or linear Gaussian. Therefore we here include the *parameters* $\theta$ explicitly in the conditional part. A *Bayesian network* is specified by a graph structure and a collection of associated conditional distributions.

Bayesian networks can be used to model multiple vectors $x[1], \ldots, x[m]$, henceforth called *data*. Throughout this paper we assume that the data is *complete*, i.e., there are no missing (unobserved) values. To incorporate the idea of learning from data, the vectors are judged to be *exchangeable*[1] (but not independent) so that the probability of the data, given the graph structure, can be expressed as

$$p(x[1], \ldots, x[m] \mid G) = \int_{\Theta} p(\theta \mid G) \prod_{t=1}^{m} p(x[t] \mid G, \theta)\, d\theta\,. \tag{2}$$

Here each term $p(x[t] \mid G, \theta)$ decomposes as in (1). Thus, when modeling multiple vectors in this way, a collection of Bayesian networks parametrized by $\theta \in \Theta$ is considered and a *prior* distribution $p(\theta \mid G)$ is assigned to the parameters. A natural application for (2) is the prediction of future events based on past observations.

Henceforth, we denote all the data briefly by $x$. Accordingly, for $i \in V$ we let $x_i$ denote the vector $(x_i[1], \ldots, x_i[m])$.

When the graph structure is not known, it is subject to learning. After introducing a prior on the graph structures, we can write

$$p(x) = \sum_{G} p(G)\, p(x \mid G)\,.$$

1. The data is judged to be part of an infinite exchangeable sequence.

This gives the distribution of data which can be used, e.g., in prediction tasks. Unfortunately, averaging over all graph structures is notoriously hard, as the number of possible graphs is found to be at least $2^{\binom{n}{2}}$ and thus super-exponential in the number of variables. Also, bounding the in-degree, i.e., the number of parents, per each variable by a constant $k$ does not help much, a lower bound still being at least $2^{kn\log n}$ (for large enough $n$).

Sometimes, the interest is in the graph structure as such. Then it is appropriate to consider the posterior distribution of the graph structures, which by Bayes rule is given by

$$p(G \mid x) = \frac{p(G)\, p(x \mid G)}{p(x)}\,.$$

Various computational methods have been dedicated to the problem of finding a plausible graph structure. Unfortunately, finding a structure that maximizes the posterior probability,

$$\hat{G} \in \arg\max_{G} p(G \mid x)\,,$$

is known to be NP-hard in general (Chickering et al., 1995). But there are also statistical limits for this approach. Namely, searching for a single structure may not be most relevant, since exponentially many graph structures can be almost equally probable in light of modest amount of data.

An alternative approach is to compute local summaries of the posterior distribution of the graph structures. In restricted forms this idea appears already in the works of Buntine (1991) and Cooper and Herskovits (1992) but is significantly further developed by Friedman and Koller (2003). Friedman and Koller consider several types of local structural features, such as edges and Markov-blankets. More precisely, if $f$ is the indicator of a structural feature, we are interested in the posterior

$$p(f \mid x) = \sum_{G} p(G \mid x) f(G)\,.$$

Here the indicator $f$ is supposed to take value 1 if the feature is present and 0 otherwise. By turning to a clever decomposition of the space of possible graph structures, Friedman and Koller find an efficient MCMC method to estimate the above sum. We next review some key parts of their approach.

## 3. Conditioning on Orders

There is an efficient way to sum over an exponential number of graph structures that are consistent with a fixed order of variables. The key insight of Friedman and Koller (2003) is to use this result, originally due to Buntine (1991), to average over graph structures: they integrate over possible orders by MCMC. We next review the idea of conditioning on orders; MCMC methods will be only briefly mentioned in Section 6.

We define an *order* of variables as a total order on the index set $V$. We represent an order $\prec$ as a vector $(U_1,\ldots,U_n)$, where $U_i$ gives the predecessors of $i$ in the order, i.e.,

$$U_i = \{j \in V : j \prec i\}\,.$$

We say that a graph structure $(G_1,\ldots,G_n)$ is *consistent* with an order $(U_1,\ldots,U_n)$, denoted $G \subseteq \prec$, if $G_i \subseteq U_i$ for all $i$. Thus, the structures that are consistent with a fixed order form a subset of the set of directed acyclic graphs. Note that for different orders, these subsets overlap. Actually, from this we get a fairly tight upper bound $n!\,2^{\binom{n}{2}}$ for the number of acyclic graphs. Asymptotically this is $o((2+\varepsilon)^{\binom{n}{2}})$ for any fixed $\varepsilon > 0$, and gives thus a much tighter characterization than the usual bounds $3^{\binom{n}{2}}$ or $2^{\Theta(n^2)}$ (see, e.g., Friedman and Koller, 2003).

Friedman and Koller (2003) observe that, from the computational point of view, it is advantageous to treat different variable orders as mutually exclusive events. While this is somewhat unnatural, since the corresponding sets of consistent graphs are overlapping, this approach is mathematically valid. Thus, in what follows, a graph structure alone does not determine whether an order is present or not. Therefore, we augment the prior of graph structures to a joint prior on orders and graphs. We also assume some fairly standard modularity assumptions stated below; related definitions are given, e.g., by Cooper and Herskovits (1992) and Friedman and Koller (2003).

**Definition 1** *We say that a Bayesian network model $p$ is* modular over $\prec, G, \theta$ and $x$, *or simply* order-modular *or* modular, *if the following properties hold:*

(M1) *If $G$ is consistent with an order $\prec$, then*

$$p(\prec, G) = c\prod_{i=1}^{n} q_i(U_i)\, q_i'(G_i)\,,$$

*where $q_i$ and $q_i'$ are probability distributions on the subsets of $V - \{i\}$ for each $i$, and $c$ is a normalization constant. Otherwise, if $\prec$ is not a total order or if $G$ is not consistent with $\prec$, then $p(\prec, G) = 0$.*

(M2) *Given a structure $G$, the parameters $\theta$ decompose into $(\theta_{1,G_1}, \ldots, \theta_{n,G_n})$ such that*

$$p(\theta \mid G) = \prod_{i=1}^{n} p(\theta_{i,G_i} \mid G_i)\,,$$

*and $p(x_i \mid x_{G_i}, \theta) = p(x_i \mid x_{G_i}, \theta_{i,G_i})$ for all $i$.*

We note that while the above described augmentation is convenient, the semantics of the variable order becomes somewhat strange, thus making the elicitation of the prior distribution potentially difficult. However, one can avoid introducing a joint prior if one agrees with the resulting marginal prior distribution, $p(G)$, on graph structures. In that case the role of orders and associated probability distributions is technical. It is important to note that, in general, we do not have $p(G_i) = q_i'(G_i)$. Namely the prior $p(G_i)$ favors sets $G_i$ that are small and thus consistent with many orders. Yet, it is easily seen that we have the following simple transform when conditioning by an order.

**Proposition 2** *If $p$ is modular and $G$ is a graph structure consistent with an order $\prec$, then*

$$p(G \mid \prec) = \prod_{i=1}^{n} p(G_i \mid U_i)\,,$$

*where $p(G_i \mid U_i) = q_i'(G_i) / \sum_{G_i' \subseteq U_i} q_i'(G_i')$.*

We give two examples to elucidate the relationship between orders and graphs. First, let each $q_i$ and $q'_i$ be uniform. Then it is not difficult to conclude that $p(G_i \mid U_i) = 2^{-|U_i|}$ and that $p(\prec) = 1/n!$. However, we note that, with this choice, the distribution of the number of parents is not uniform. In our second example $q_i$ is again uniform, however, we set $q'_i(G_i)$ to be proportional to $1/\binom{n-1}{|G_i|}$ for parents $G_i$ with cardinality at most the maximum in-degree. This assignment is natural when different cardinalities of parents are judged to be uniformly distributed. Note, however, that averaging over orders renders the marginal distribution of the cardinality $|G_i|$ slightly biased from uniform, since small cardinalities are favored as they are consistent with more orders.

While Proposition 2 above essentially follows from property (M1) of Definition 1, another appealing consequence of modularity is due to property (M2). Namely, the distribution of the data remains factorized when the parameters $\theta$ are marginalized out. A more precise statement is given below; the proof is simple and standard, and therefore omitted.

**Proposition 3** *If p is modular, x is complete, and G is a graph structure, then*

$$p(x \mid G) = \prod_{i=1}^{n} p(x_i \mid x_{G_i}),$$

*where* $p(x_i \mid x_{G_i}) = \int p(\theta_{i,G_i} \mid G_i)\, p(x_i \mid x_{G_i}, \theta_{i,G_i})\, d\theta_{i,G_i}$.

We proceed to consider probabilities of local features. Here we restrict our attention to modular features.

**Definition 4** *A mapping $f$ from graph structures onto $\{0,1\}$ is called* modular *if $f(G) = \prod_{i=1}^{n} f_i(G_i)$ where each $f_i$ is a mapping from the subsets of $V - \{i\}$ onto $\{0,1\}$.*

For example, the indicator of a directed edge between two nodes is clearly modular. Also, the constant functions 1 and 0 are trivially modular. More generally, the indicator of any subgraph (a directed acyclic graph on a subset of $V$) is modular.

The key observation is that the summation over graph structures decomposes into a product of "local summations" (Buntine, 1991; Friedman and Koller, 2003).

**Theorem 5** *If $p$ and $f$ are modular, x is complete, and $\prec = (U_1, \ldots, U_n)$ is a variable order, then*

$$p(x, f \mid \prec) = \prod_{i \in V} \sum_{G_i \subseteq U_i} p(G_i \mid U_i)\, p(x_i \mid x_{G_i})\, f_i(G_i).$$

**Proof** Using first the marginalization and chain rules of probability, and then Proposition 2, Proposition 3, and Definition 4 to the three terms, respectively, we get

$$\begin{aligned} p(x, f \mid \prec) &= \sum_{G} p(G \mid \prec)\, p(x \mid G, \prec)\, p(f \mid x, G, \prec) \\ &= \sum_{G_1 \subseteq U_1} \cdots \sum_{G_n \subseteq U_n} \prod_{i=1}^{n} p(G_i \mid U_i)\, p(x_i \mid x_{G_i})\, f_i(G_i), \end{aligned}$$

which factorizes into the desired product. ■

The posterior of the feature is obtained via Bayes rule:

$$p(f \mid x, \prec) = p(x, f \mid \prec) / p(x \mid \prec).$$

Note that the probability of the data, $p(x \mid \prec)$ can be represented as $p(x, f' \mid \prec)$ where feature $f'$ is the constant function 1.

Finally, once having the conditional probabilities for the feature, the unconditional posterior is obtained as

$$p(f \mid x) = \sum_{\prec} p(\prec \mid x)\, p(f \mid x, \prec)\,. \tag{3}$$

Here $\prec$ runs through all orders on the set $V$. There are $n!$ different orders, which is still super-exponential with respect to $n$. Yet, this is much less than the number of all possible graph structures.

Friedman and Koller (2003) propose an MCMC method to estimate sum (3) by drawing a sample of orders from the posterior $p(\prec \mid x) \propto p(\prec)\, p(x \mid \prec)$. They argue that this approach is more efficient than MCMC directly in the space of graph structures (Madigan and York, 1995).

It is important to note that, despite of the closed form expression of Theorem 5, the computations, given an order, may be quite expensive. Namely, one has to consider all possible sets of parents for each variable. For a fixed maximum number of parents, $k$, roughly $n^{k+1}$ terms $p(x_i \mid x_{G_i})$, henceforth called *local conditional marginals*, need to be computed. Depending on the data size, on the functional forms of the local conditional distributions, and on the value $k$, this may contribute significantly to the total computational complexity. Henceforth we suppose that any local term $p(x_i \mid x_{G_i})$ can be computed in time $O(C(m))$, where $C$ is a function of data size. For example, when a local conditional distributions is taken from the exponential family with appropriate conjugate priors, then the associated $C(m)$ is linear in $m$. For more structured models, e.g., decision graphs (Chickering et al., 1997), the complexity of computing local conditional marginals can be significantly greater, yet often linear in $m$.

## 4. Summation by Dynamic Programming

We next show how the summation over orders can be carried out in roughly $n2^n$ operations, which grows much slower than $n!$. This improved requirement should be contrasted with the lower bound $n^{k+1}$ (with a potentially large constant factor). For moderate $n$ (say, $n < 20$) and relatively large $k$ (say, $k = 5$), the total complexity may be dominated by the polynomial term. Yet, the computations are feasible on modern computers.

We consider a summation that is slightly different from (3). We write

$$p(f \mid x) = p(f, x)/p(x)\,, \tag{4}$$

and continue by considering evaluation of

$$p(f, x) = \sum_{\prec} p(\prec)\, p(f, x \mid \prec)\,. \tag{5}$$

Note that $p(x)$ is of the same form.

In order to facilitate the forthcoming development, we define for each $i \in V$ a function $\alpha_i$ as follows. Let $i$ be an element of $V$ and let $S$ be a subset of $V$ that does not contain $i$. We define

$$\alpha_i(S) = \sum_{G_i \subseteq S} q_i'(G_i)\, p(x_i \mid x_{G_i})\, f_i(G_i)\,. \tag{6}$$

In essence, the function $\alpha_i$ gives the contribution of the $i$th local component ($x_i$ and its unknown parents) to sum (5) above; notice the similarity to the terms in Theorem 5. The functions $\alpha_i$ serve

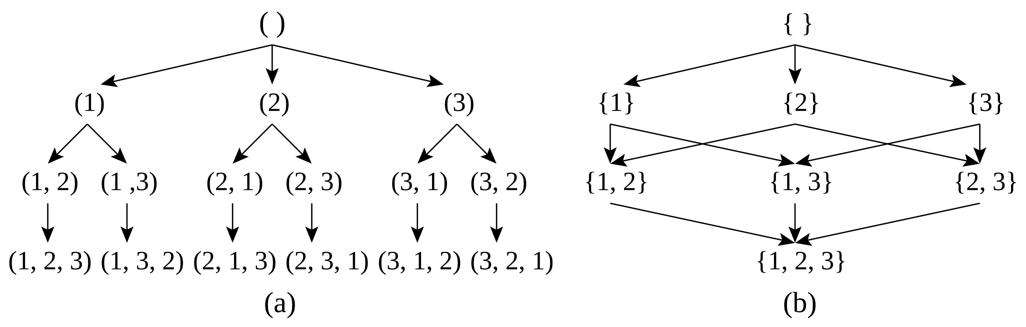


Figure 1: Illustrations of (a) the permutation tree and (b) the subset lattice of $\{1,2,3\}$. The nodes of the permutation tree are labeled by the corresponding paths from the root. (The labels of the edges are not shown but can be easily deduced.)

as a way to split the problem of evaluating sum (5) into two steps. The first is to compute these functions given a feature $f$ and a data set $x$. The second task is to compute sum (5), given the functions $\alpha_i$. We first consider the latter problem assuming that the functions $\alpha_i$ are precomputed; we will later come back to the former problem.

It turns out that the sum over orders is advantageous to compute in a manner of variable elimination. Here variables refer to the $n$ elements $\sigma_1,\ldots,\sigma_n \in V$, where $\sigma_j$ is the $j$th element in the order. A key observation is that when considering the parents of variable $x_{\sigma_j}$ it is sufficient to know the unordered set $\{\sigma_1,\ldots,\sigma_{j-1}\}$ of the possible parents. In other words, the order of the possible parents is irrelevant.

One way to view this reduction from ordered sets to unordered sets is to consider a *permutation tree*. A permutation tree of $V=\{1,\ldots,n\}$ is a rooted (directed) tree with $n$ levels. Any node at the $h$th level has $n-h$ children labeled by distinct elements from $V$, so that the labels on any (directed) path are all distinct. Thus, a path from the root to a leaf corresponds to a unique permutation on $V$ (see Figure 1(a)). Evaluation of the sum over orders is carried out by a propagation algorithm, where each node obtains a value by summing up the values of its parents, each multiplied by a quantity that depends on the associated path (details will be given soon). However, apart from the last label of the path, only the unordered set of the labels matters. This means that computations over different paths can be merged. Graphically, the permutation tree collapses to a *subset lattice*. A subset lattice of $V$ is a graph, where nodes corresponds to subsets of $V$ and there is an edge from $A$ to $B$ if and only if $B = A\cup\{i\}$ for some $i \notin A$ (see Figure 1(b)).

We summarize the above discussion more formally:

**Proposition 6** *If $p$ and $f$ are modular and $x$ is complete, then*

$$p(f,x) = c\,g(V)\,,$$

*where $c$ is the normalizing constant and $g$ is defined for all subsets $S$ of $V$ recursively by*

$$g(S) = \sum_{i\in S} q_i(S-\{i\})\,\alpha_i(S-\{i\})\,g(S-\{i\}) \qquad (7)$$

*with boundary* $g(\emptyset) = 1$.

**Proof** By simple induction we get that

$$g(V) = \sum_{\sigma_1,\ldots,\sigma_n} \prod_{j=1}^{n} q_i(S_j)\, \alpha_{\sigma_j}(S_j)\,,$$

where $(\sigma_1,\ldots,\sigma_n)$ runs through all permutations on $V$, and $S_j = \{\sigma_1,\ldots,\sigma_{j-1}\}$. Thus, each $(\sigma_1,\ldots,\sigma_n)$ is a realization of an order $\prec = (U_1,\ldots,U_n)$ with $\sigma_1 \prec \sigma_2 \prec \cdots \prec \sigma_n$ and $U_{\sigma_j} = \{\sigma_1,\ldots,\sigma_{j-1}\} = S_j$. By the modularity of $p$, we have

$$c \prod_{i=1}^{n} q_i(U_i)\, q_i'(G_i) = p(\prec)\, p(G \mid \prec)\,.$$

Thus, by (5), Proposition 2, and Theorem 5, we have $c\,g(V) = p(x, f)$. ■

This result gives us a relatively efficient way to sum over orders in the manner of dynamic programming (for a related algorithm, see Bellman, 1962). Provided that each value of the functions $q_i, \alpha_i$, and $g$ can be accessed in a constant (amortized) time, we have an $O(n2^n)$ time algorithm for computing the posterior $p(f \mid x)$. Note that the constant $c$ cancels out in (4).

We now take a step back and consider the computation of the functions $\alpha_i$. Since the representation of each function $\alpha_i$ already takes $2^{n-1}$ numbers, the best we can hope is to find an algorithm that runs in time $O(2^n)$. We next show how to achieve this optimal time complexity.[2]

Consider a fixed $i \in V$. It is convenient to define a new function, $\beta_i$, by

$$\beta_i(G_i) = q_i'(G_i)\, p(x_i \mid x_{G_i})\, f_i(G_i) \qquad \text{for all } G_i \subseteq V - \{i\}\,. \tag{8}$$

Hence, by the definition of $\alpha_i$ we have

$$\alpha_i(S) = \sum_{T \subseteq S} \beta_i(T) \qquad \text{for all } S \subseteq V - \{i\}\,.$$

The operator that in this way maps a function of subsets of $V - \{i\}$ to another such function is known as the *Möbius transform* (on the subset lattice of $V - \{i\}$). We say that $\alpha_i$ is the Möbius transform of $\beta_i$.

How fast can we evaluate the Möbius transform? In the following discussion we assume that the values of $\beta_i$ are precomputed and can be accessed in a constant (amortized) time. The straightforward approach is to compute separately for each subset $S$ the summation over its subsets. We see that one step takes $O(2^{|S|})$ time, and hence, the total time complexity is $O(3^n)$. To reduce this bound, we notice that $\beta_i(T)$ vanishes for all $T$ with more than $k$ elements. This yields the complexity $O(n^k 2^n)$. An alternative, and more efficient approach is to use the fast Möbius transform algorithm (see, e.g., Kennes and Smets, 1991). It takes advantage of the overlapping parts of the $2^n$ summations and runs in time $O(n2^n)$. However, it does not exploit the in-degree bound $k$. We next show that under this additional constraint we can still slightly reduce the time complexity, to the desired $O(2^n)$.

2. This bound is optimal up to a constant factor that may depend on the maximum in-degree $k$ which is assumed to be constant.

```
FAST-TRUNCATED-MÖBIUS-TRANSFORM($h_0$, $k$)
1  for $j \leftarrow 1$ to $n$ do
2      for each $S \subseteq N$ with $|S \cap \{j+1,\ldots,n\}| \leq k$ do
3          $h_j(S) \leftarrow 0$
4          if $|S \cap \{j,\ldots,n\}| \leq k$ then
5              $h_j(S) \leftarrow h_{j-1}(S)$
6          if $j \in S$ then
7              $h_j(S) \leftarrow h_j(S) + h_{j-1}(S - \{j\})$
8  return $h_n$
```

Figure 2: An algorithm for evaluating the truncated Möbius transform on the subset lattice of $N = \{1,\ldots,n\}$. Input $h_0$ is the function to be transformed, and $k$ is a number between 1 and $n$.

We extend the notion of Möbius transform by defining a truncated Möbius transform as a Möbius transform where the summation over subsets is restricted to subsets with at most $k$ elements, where $k$ is an additional input parameter. Figure 2 describes a general algorithm for computing truncated Möbius transforms on the subset lattice of $N = \{1,\ldots,n\}$. (When we apply this algorithm to compute a function $\alpha_i$, we replace $n$ by $n-1$.) The algorithm mainly follows the steps of the standard fast Möbius transform algorithm (see, e.g., Kennes and Smets, 1991) and splits the transform into $n$ smaller transforms, each being a summation over the two subsets of a singleton $\{j\} \subseteq N$. These $n$ transforms operate one after another, the $j$th transform to the result of the $(j-1)$th transform. This procedure can be also viewed as a variable elimination algorithm, where for each $\{j\}$ "its subset" is treated as a variable taking two values. In the fast truncated Möbius transform algorithm, described in Figure 2, this standard algorithm is modified so that each of the $n$ transforms is evaluated only at as few subsets of $N$ as needed. While the last transform has to be evaluated at all $2^n$ subsets, a polynomial number of evaluations is sufficient for the first transforms. Details of this discussion are given in the proof of the following result.

**Proposition 7** *Algorithm* FAST-TRUNCATED-MÖBIUS-TRANSFORM *(Figure 2) with input* $(h_0, k)$ *runs in time* $O(2^n)$ *for a constant k and computes the function* $h_n$*, given by*

$$h_n(S) = \sum_{T \subseteq S : |T| \leq k} h_0(T) \qquad \text{for all } S \subseteq N.$$

**Proof** We show by induction on $j$ that $h_j(S)$ for $S \subseteq N$ with $|S \cap \{j+1,\ldots,n\}| \leq k$, computed at the $j$th iteration, is given by

$$h_j(S) = \sum_{T_1 \subseteq S_1} \cdots \sum_{T_j \subseteq S_j} 1(|T_{1,j} \cup S_{j+1,n}| \leq k)\, h_0(T_{1,j} \cup S_{j+1,n}), \tag{9}$$

where we denote $S_r = S \cap \{r\}$ and $S_{r,r'} = S_r \cup S_{r+1} \cup \cdots \cup S_{r'} = S \cap \{r, r+1, \ldots, r'\}$ (similarly for $T_r$ and $T_{r,r'}$).

In the case $j = 0$ expression (9) trivially holds. We proceed by letting $j > 0$. Let $S \subseteq N$ with $|S \cap \{j+1,\ldots,n\}| \leq k$. We separate two cases, $S_j = \emptyset$ and $S_j = \{j\}$, and show that expression (9) holds in both cases.

First consider the case $S_j = \emptyset$. By the algorithm, after the $j$th step, we have

$$h_j(S) = h_{j-1}(S)\,.$$

Using the induction assumption (9) for $h_{j-1}$, it is easy to verify that $h_j(S)$ can be expressed as in the induction claim (9) since $T_j = S_j = \emptyset$.

Then consider the remaining case, $S_j = \{j\}$. After the $j$th step, we have

$$h_j(S) = 1(|S \cap \{j,\ldots,n\}| \le k)\, h_{j-1}(S) + h_{j-1}(S - \{j\})\,.$$

The first term in the sum corresponds to the case $T_j = S_j = \{j\}$ in the summation (9). By the induction assumption (9) for $h_{j-1}$, it is sufficient to verify that

$$1(|S \cap \{j,\ldots,n\}| \le k)\, 1(|T_{1,j-1} \cup S_{j,n}| \le k) = 1(|T_{1,j} \cup S_{j+1,n}| \le k)\,,$$

and that $T_{1,j-1} \cup S_{j,n} = T_{1,j} \cup S_{j+1,n}$. The latter is obvious since $T_j = S_j$. The former identity follows since $S \cap \{j,\ldots,n\} = S_{j,n}$. Note that the condition $|S \cap \{j,\ldots,n\}| \le k$ ensures that $h_j$ has been evaluated at $S$ in the previous iteration.

The second term in the sum, $h_{j-1}(S - \{j\})$, corresponds to the case $T_j = \emptyset$. In this case, $|T_{1,j} \cup S_{j+1,n}| \le k$ holds since we have assumed that $|S \cap \{j+1,\ldots,n\}| \le k$. The argument of $h_0$ remains invariant because in expression (9) for $h_{j-1}(S - \{j\})$, we have $T_j = S_j = \emptyset$.

We have now shown that (9) holds for all $j = 1,\ldots,n$. Particularly, in the case $j = n$ we get the claimed transformation. This completes the first part of the proof.

We next show that the algorithm runs in time $O(2^n)$. For the steps $j = n-k+1,\ldots,n$ the required total number of operations is proportional to

$$\sum_{j=n-k+1}^{n} 2^j 2^{n-j} = k2^n = O(2^n)\,.$$

For the steps $j = 1,\ldots,n-k$ we get a bound

$$\sum_{j=1}^{n-k} 2^j \sum_{r=0}^{k} \binom{n-j}{r} \le \sum_{j=1}^{n-k} 2^j (n-j)^k \le 2^n \sum_{j=0}^{\infty} (1/2)^j j^k = O(2^n)\,,$$

since the infinite sum converges to a finite limit for a fixed $k$. Thus, the total running time is $O(2^n)$. This completes the second part of the proof. ■

It is possible to extend the above complexity result for $k$ that is not a constant. In particular, for $k = n$ the presented algorithm obviously computes (ordinary) Möbius transform in time $O(n2^n)$. For general $k$, however, the presented complexity analysis is too rough to give a tight bound. We conjecture that the time complexity is $O(k2^n)$, but proving this would require a more detailed analysis beyond the scope of this discussion.

We finally summarize. By the fast truncated Möbius transform, the functions $\alpha_i$ for $i = 1,\ldots,n$ can be computed in total time $O(n2^n)$. This assumes that the functions $\beta_i$ for $i = 1,\ldots,n$ are precomputed, which obviously can be done in $O(n^{k+1}C(m))$ time. Hence, we get our main result.

**Theorem 8** *Let $p$ and $f$ be modular and let $x$ be complete with $m$ records. Assume that any local conditional marginal can be computed in time $O(C(m))$. Then for a constant in-degree bound $k$, the probability $p(f \mid x)$ can be evaluated in time $O(n2^n + n^{k+1}C(m))$.*

## 5. Finding One of the Most Probable Structures

We now turn to the problem of finding a single best graph structure. Since the algorithms of the previous section essentially exploit the distributive law of a sum-product semiring, it is not surprising that only slight modifications are needed for a max-product semiring. As various aspects of general semiring algorithms have been studied extensively in the last decade (e.g., Stearns and Hunt III, 1996; Lauritzen and Jensen, 1997; Dechter, 1999; Aji and McEliece, 2000), we here omit algorithmic details and focus on certain key points that are central from the modeling point of view, though algorithmically rather irrelevant.

We first observe that finding a maximizing pair

$$(\prec^*, G^*) \in \arg\max_{\prec, G} p(\prec, G \mid x)$$

would be rather straightforward based on certain modification in the summation algorithms described in the previous section. However, we are interested in finding a graph $\hat{G}$ that maximizes the marginal posterior $p(G \mid x)$. Note that this target function favors structures that are consistent with many (a priori probable) orders. It seems that we now have a somewhat harder task as we need to average over orders while maximizing over graphs.

Fortunately, there is a satisfactory solution to the computational problem stated above: we slightly change the problem. We specify a modular prior distribution directly on graph structures without augmenting the probability space by orders as was done in Definition 1. Actually, this has been a more common way to specify a prior on Bayesian network structures (Cooper and Herskovits, 1992; Heckerman et al., 1995a) than coupling with orders. The corresponding modularity property is as follows; related definitions are given, e.g., by Cooper and Herskovits (1992) and Friedman and Koller (2003).

**Definition 9** *We say that a Bayesian network model $p$ is* modular over $G, \theta$ and $x$, *or simply* graph-modular, *if (M2) of Definition 1 and (M1') below hold.*

(M1') *If G is acyclic, i.e., consistent with some order, then*

$$p(G) = c' \prod_{i=1}^{n} q_i''(G_i)\,,$$

*where each $q_i''$ is a probability distribution over the subsets of $V - \{i\}$. Otherwise, if G is cyclic, then $p(G) = 0$. Here $c'$ is a normalization constant.*

We note that property (M1) in Definition 1 and property (M1') in Definition 9 imply slightly different forms of the structure prior $p(G)$. In particular, property (M1) is *not* a special case of property (M1'). Specifically, orders are now not treated as disjoint events, but instead, if a graph structure is consistent with two orders, then both are present. Thus, under (M1'), the probabilities $p(\prec)$ of different orders $\prec$, though well-defined, do not sum up to one. For a similar discussion, see Friedman and Koller (2003). In the maximization task we are considering, the advantage of (M1') over (M1) is that the former allows us to search for most probable graph structures in the joint space of orders and structures.

**Proposition 10** *Let $p$ be graph-modular and $x$ complete. Let*

$$(\prec^*, G^*) \in \arg\max_{\prec, G} \left\{ \prod_{i=1}^{n} q_i''(G_i)\, p(x_i \mid x_{G_i}) \right\},$$

*where $\prec$ runs through all total orders and $G$ structures that are consistent with $\prec$. Then $G^*$ maximizes the posterior $p(G \mid x)$.*

**Proof** The product to be maximized is seen to be equal to $p(G \mid x)$ up to a constant factor. Since the joint space of orders and graph structures includes every directed acyclic graph, it also includes a graph that maximizes the posterior $p(G \mid x)$. ■

This simple observation gives us a way to find a graph structure that maximizes the posterior probability. Namely, it is rather straightforward to modify the summation algorithms given in the previous section to compute maximizing arguments instead; details are omitted. We have the following counterpart of Theorem 8.

**Theorem 11** *Let $p$ be graph-modular and let $x$ be complete with $m$ records. Assume that any local conditional marginal can be computed in time $O(C(m))$. Then for a constant in-degree bound $k$, a graph structure that maximizes the posterior probability can be found in time $O(n2^n + n^{k+1}C(m))$.*

We note that the idea of searching for an optimal order as a means for learning Bayesian networks is not new: Larrañaga et al. (1996) observe that this task resembles the Traveling Salesman Problem (TSP) and design a genetic algorithm for searching for good orders. However, they do not mention the exact dynamic programming solution that resembles the algorithm for the TSP due to Bellman (1962).

## 6. Discovering Larger Networks

We next sketch how the presented exact methods can be applied in cases where the number of variables is large, say $n > 30$. One possibility is to force additional restrictions on the space of structures. Another possibility is to resort to inexact techniques, such as MCMC and local search procedures. Though we in this section mainly consider the summation problem, modifications for the maximization problem are also possible in the same manner as discussed in Section 5.

### 6.1 Restrictions on Orders

It is fortunate that in some cases, where the number of variables is large, the modeler may have a strong prior on the graph structures. For example, some variables cannot have parents while some others cannot have children. Such knowledge may arise naturally when the direction of the edges are interpreted as the direction of causality (Heckerman et al., 1999; Pearl, 2000).

For a more general treatment, let $\{V_1, \ldots, V_\ell\}$ be a partition of the set $V = \{1, \ldots, n\}$. Recall that a total order on $V$ corresponds to a permutation on $V$. Denote the set of all permutations on a set $S$ by $\mathcal{S}(S)$. Now assume that prior knowledge justifies the restriction to the permutations in the Cartesian product

$$\mathcal{S}(V_1) \times \cdots \times \mathcal{S}(V_\ell) \subseteq \mathcal{S}(V).$$

```
FEATURE-PROBABILITY-IN-LAYERED-NETWORK($V_1, \ldots, V_\ell$, $q$, $\beta$)
1   $g(\emptyset) \leftarrow 1$
2   for $h \leftarrow 1$ to $\ell$ do
3       for each $i \in V_h$ do
4           compute $\beta'_i$ according to (10)
5           $\alpha'_i \leftarrow$FAST-TRUNCATED-MÖBIUS-TRANSFORM($\beta'_i$, $k$)
6       for each nonempty $S \subseteq V_h$, in increasing order do
7           $g(S) \leftarrow \sum_{i \in S} q_i(S - \{i\})\, \alpha'_i(S - \{i\})\, g(S - \{i\})$
8   return $\prod_{h=1}^{\ell} g(V_h)$
```

Figure 3: An algorithm for computing the joint probability $p(f,x)$ up to a normalizing constant in a layered network.

That is, an edge from $i \in V_s$ to $j \in V_t$ is allowed only if $s \leq t$. This restricts us to a space of "layered" networks. On one extreme we get the set of all orders ($\ell = 1$), and on the other, we get a single order ($\ell = n$). A layering is a property induced by the prior on orders and graph structures. We say that a layering and a model $p$ are *compatible* if the prior probability of any graph structure that violates the layering vanishes.

Fixing a layering structure may dramatically reduce the computational complexity of evaluating feature probabilities. We observe that it is sufficient to consider permutations on different layers $V_h$ separately. Thus, the sum over orders on $V$ factorizes into a product of sums of orders on each layer $V_h$. Perhaps a less immediate fact is that also the summations over different sets of parents can be handled efficiently. This is because the elements of $V_1 \cup \cdots \cup V_{h-1}$ are always possible parents of an $i \in V_h$ regardless of the order on $V_h$. To take advantage of this observation, we modify the mappings $\beta_i$ defined in (8). For all subsets $T$ of $V_h - \{i\}$ of size at most $k$ we define

$$\beta'_i(T) = \sum_W \beta_i(T \cup W)\,, \tag{10}$$

where $W$ runs through all subsets of the union $V_1 \cup \cdots \cup V_{h-1}$. Recall that $\beta_i(T \cup W) = 0$ when the size $|T \cup W|$ is larger than $k$.

Figure 3 displays an algorithm that exploits a layered network decomposition. The input consists of a partition $\{V_1, \ldots, V_\ell\}$ of $\{1, \ldots, n\}$, a modular prior on orders specified by $q = (q_1, \ldots, q_n)$, and a precomputed function $\beta = (\beta_1, \ldots, \beta_n)$ as defined in (8). Note that this $\beta$ depends on the feature $f$. The algorithm outputs the proportional probability $p(f,x)/c$, where $c$ is the normalizing constant (independent of $f$ and $x$). Recall that this value is sufficient when computing posterior probabilities of features.

**Theorem 12** *Let $p$ and $f$ be modular and let $x$ be complete with $m$ records. Assume $p$ is compatible with a layering $\{V_1, \ldots, V_\ell\}$. Further assume that any local conditional marginal can be computed in time $O(C(m))$. Then for a constant in-degree bound $k$, the probability $p(f \mid x)$ can be evaluated in time $O(n2^{n_*} + n^{k+1}C(m))$, where $n_*$ is the maximum of $|V_1|, \ldots, |V_\ell|$.*

**Proof** We first show that Algorithm FEATURE-PROBABILITY-IN-LAYERED-NETWORK, given in Figure 3, is correct, and then we analyze its time requirement.

Let $i$ be an element of $V_h$ and $S$ a subset of $V_h - \{i\}$. By the definitions of the functions $\alpha_i$, $\beta_i$, and $\beta'_i$, we have, after line 5,

$$\alpha'_i(S) = \sum_{T \subseteq S} \beta'_i(T) = \sum_{(T \cup W) \subseteq S \cup V_1 \cup \cdots \cup V_{h-1}} \beta_i(T \cup W) = \alpha_i(S \cup V_1 \cup \cdots \cup V_{h-1})\,.$$

From this it is obvious that

$$\prod_{h=1}^{\ell} g(V_h) = \sum_{\sigma_1,\ldots,\sigma_n} \prod_{j=1}^{n} q_i(S_j)\, \alpha_{\sigma_j}(S_j)\,,$$

where $(\sigma_1,\ldots,\sigma_n)$ runs through all permutations in the restricted set $\mathcal{S}(V_1) \times \cdots \times \mathcal{S}(V_\ell)$, and $S_j = \{\sigma_1,\ldots,\sigma_{j-1}\}$. Hence, by the arguments given in the proof of Proposition 6, we obtain $p(f,x) = c\prod_{h=1}^{\ell} g(V_h)$, as required.

We then analyze the complexity of the algorithm. Fix a level $h \in \{1,\ldots,\ell\}$. Denote $V' = V_1 \cup \cdots \cup V_h$. Line 4 can be performed in time $O(|V'|^k)$, since the summations (10) for different $T$ include each subset $(T \cup W) \subseteq V'$ of size at most $k$ just once. Line 5 can be performed in time $O(2^{|V_h|})$. Thus, the overall complexity of the for-loop starting in line 3 is $O(|V_h|2^{|V_h|} + |V_h||V'|^k)$. The time complexity of lines 6–7 is clearly $O(|V_h|2^{|V_h|})$.

Summing the complexities for $h = 1,\ldots,\ell$ gives the total time requirement of $O(n2^n + n^{k+1})$. The algorithm assumes that functions $\beta_i$ are precomputed; this can be done in time $O(n^{k+1}C(m))$. Hence, combining these two bounds gives the claimed time complexity. ■

Sometimes domain knowledge may allow for further reduction in the problem complexity. As an example, consider the special case, where we know a priori that some variables cannot have parents while some others cannot have children. This can be expressed by a partition $\{V_1, V_2, V_3\}$ with three layers. We notice that, in this particular case, the orders on $V_1$ and $V_3$ are irrelevant, which reduces the computational complexity to $O(n2^{|V_2|} + n^{k+1}C(m))$. Thus, exact structure discovery remains practical even in cases where $V_1$ and $V_3$ are large.

### 6.2 Combining with Inexact Techniques

When prior knowledge cannot be used to restrict the space of graph structures, one has to resort to inexact methods. Currently the most efficient general method is perhaps the MCMC algorithm by Friedman and Koller (2003). It draws a sample from the posterior distribution of orders and estimates feature probabilities by sample averages. It can be fairly easily modified to search for most probable graphs, e.g., by using simulated annealing techniques. Moreover, as noted by Friedman and Koller, MCMC can be naturally extended to deal with data sets where some of the data is missing.

Based on the concept of layering, introduced in Section 6.1, we propose an extension of Friedman and Koller's method. Instead of sampling total orders on $V$, it might be advantageous to sample partitions $\{V_1,\ldots,V_\ell\}$, where the number of subsets $\ell$ is fixed and the subsets are of an equal size $r = n/\ell$. This reduces the size of the sample space by the factor $(r!)^{n/r}$. Still, for relatively small $r$, say $r = 10$, the computational cost per sampled partition is relatively low.

It is known that reducing the state space—sometimes called Rao-Blackwellization (Gelfand and Smith, 1990)—is generally a good idea to boost sampling methods, provided that the resulting computational overhead is relatively small. Namely, merging states may result in a smoother, and thus easier, “landscape” with fewer local maxima (with respect to the neighborhood induced by the algorithm). Friedman and Koller (2003) showed that orders are preferred over graphs. We believe that, likewise, layerings are preferred over orders. However, validating this conjecture requires a dedicated study which is beyond the scope of this paper.

## 7. Experimental Results

We have implemented the algorithms described in this paper. Our implementation is written in the C++ programming language. The experiments to be described next were run under Linux on an ordinary desktop PC with a 2.4GHz Pentium processor and 1.0GB of memory.

The objectives of these experiments are threefold. First, we generate and analyze data sets that, we believe, might be hard to analyze by inexact methods. For comparison, some results for a small sample of existing heuristic algorithms are also presented. Second, we illustrate the summation and maximization tasks in structure discovery from the methodological point of view. This includes exemplification of the layering method described in the previous section. Third, we demonstrate that exact computations are feasible for relatively large networks. This includes measurements of exponential and polynomial contributions to the overall time requirement.

### 7.1 Data Sets

We have tested the summation and maximization algorithms on a small selection of data sets. The data sets contain discrete variables only and no values are missing. The *Zoo* data set is available from the UCI Machine Learning Repository (Blake and Merz, 1998, the data set contributed by Richard Forsyth). It contains 17 variables and 101 records. The *Alarm* data set built by (Herskovits, 1991) contains 37 variables and 20000 records, generated from the Alarm Monitoring System (Beinlich et al., 1989). In our experiments we used the first 100, 500, and 2500 records from the original *Alarm* data set.

We also generated two new data sets. Both employ the binary parity function. The *Parity1* data set contains 22 binary variables and 2500 records. We generated the data from a Bayesian network with structure as in Figure 4(a). The maximum in-degree is 4. The local distributions for each variable $x_i$ given its parents $x_{G_i}$ were specified as follows. For each variable we associate a fixed noise rate $\varepsilon_i \in [0,1]$. For each record $x[t]$ we set $x_i[t] = \mathrm{xor}(x_{G_i}[t])$ with probability $1-\varepsilon_i$ and $x_i[t] = 1-\mathrm{xor}(x_{G_i}[t])$ with probability $\varepsilon_i$. Here the parity function $\mathrm{xor}(z_1,\ldots,z_h)$ returns 1 if the sum $z_1+\cdots+z_h$ is odd and 0 if it is even. The idea behind using the parity function here is, of course, that no subset of the correct set of parent variables should give any hint about the state of the child. In structure learning, incremental construction of the parent set should be unstable if not impossible, thus requiring an exhaustive search. However, following this idea fully would require that value configurations of the parents of a node are uniformly distributed and that no node has just one parent. These conditions are not met in *Parity1* which makes learning somewhat easier—thus serving as a more realistic example. The *Parity2* data set extends this idea to 100 variables with a specific layering topology. We arranged 100 variables into two layers, one with 78 and the other with 22 variables, satisfying the following two constraints: (i) there are no edges within the first layer (78 variables); (ii) the edges between the layers are directed from the first layer to the second layer. Here we used the maximum in-degree 3. From the specified Bayesian network 2500 records

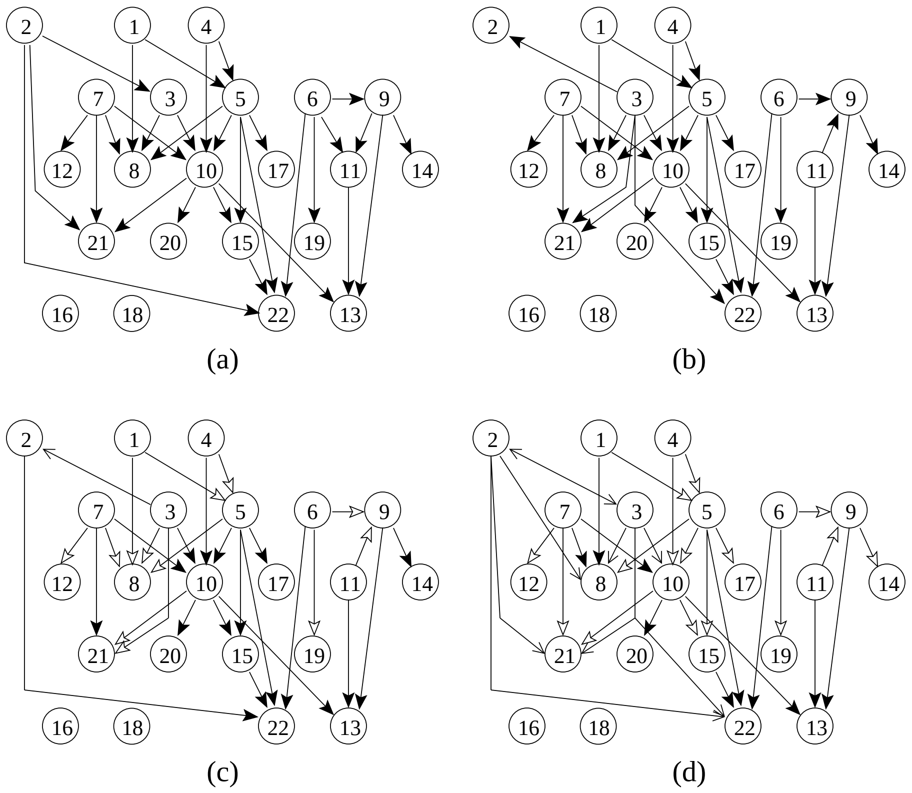


Figure 4: Discovering network structure for *Parity1*. (a) The correct structure. (b) A structure that maximizes the posterior probability with 500 records. Posterior probabilities of edges with (c) 2500, and (d) 500 records. In (c) and (d) arrow heads ▲, $\triangle$, and $\wedge$ correspond to intervals $[0.99, 1.00]$, $[0.67, 0.99)$, and $[0.33, 0.67)$, respectively.

were generated. In our experiments we also used prefixes of sizes 100 and 500 of the *Parity1* and *Parity2* data sets.

## 7.2 Model Specifications for Learning

In our experiments we used a simple generic Bayesian network model for structure discovery on the selected data sets. We used different values of the maximum in-degree for different data sets. For *Zoo*, *Alarm*, *Parity1*, and *Parity2* the values were 6, 4, 5, and 3, respectively. Note that in the *Alarm* network (Figure 5(a)) the in-degree of node 27 is 4. We set $q_i(U_i)$ to be uniform and $q'_i(G_i)$ to be proportional to $1/\binom{n-1}{|G_i|}$ (for $|G_i| \leq k$). In the case of maximization we used the same distribution on parents, that is, $q''_i(G_i) = q'_i(G_i)$. Since all variables are discrete, the local conditional distributions are multinomial. For the multinomial parameters we assigned a Dirichlet prior with hyperparameters set to 1. Note that a Dirichlet prior yields a closed-form solution to the local conditional marginal distributions. Hence, the complexity $C(m)$ of computing such a marginal is linear in the data size $m$ (Cooper and Herskovits, 1992).

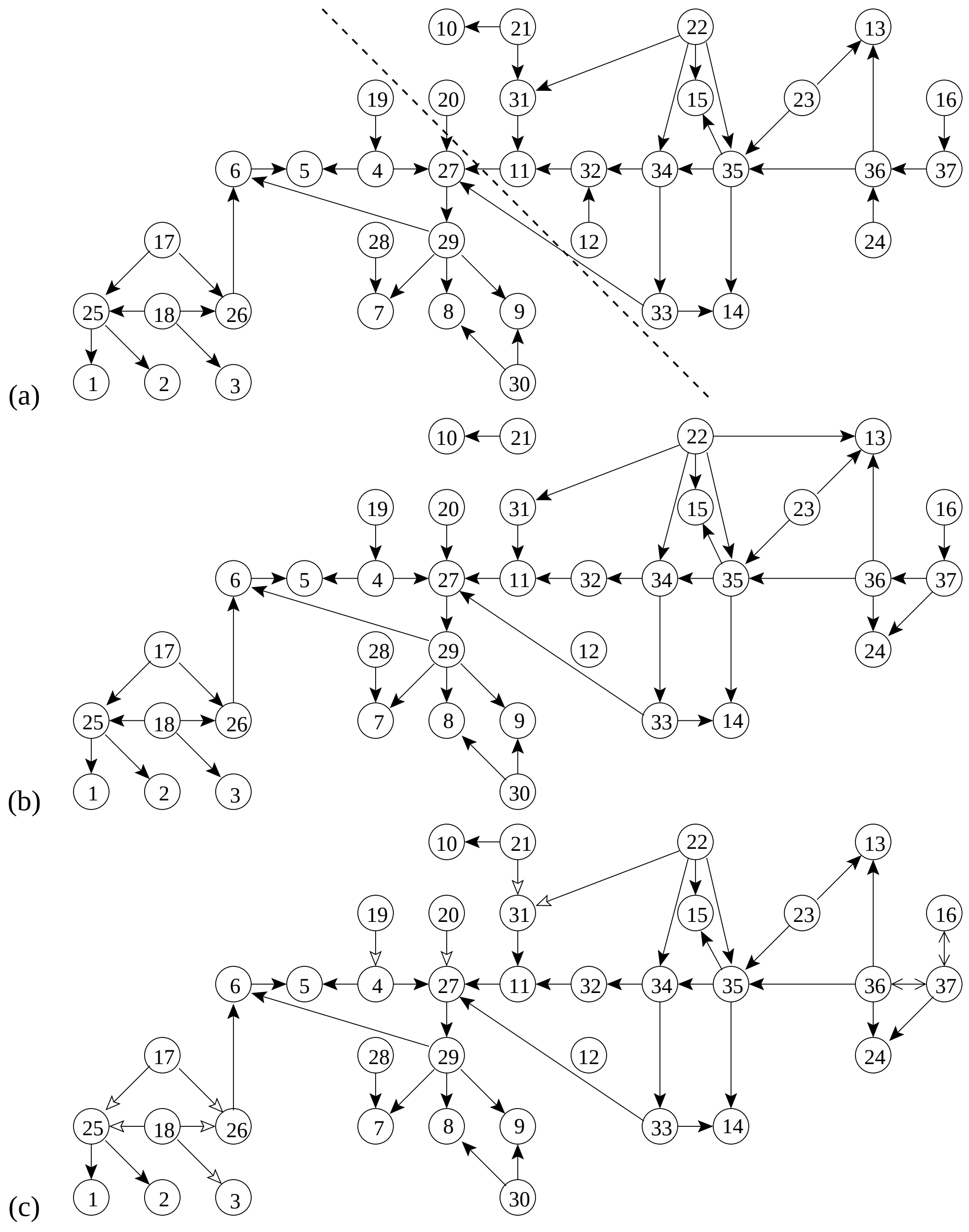


Figure 5: Discovering network structure for *Alarm*. (a) The correct structure. A dashed line separates two layers. For 2500 records, (b) a structure that maximizes the posterior probability, and (c) edge probabilities. Arrow heads ▲, △, and ∧ correspond to intervals $[0.99, 1.00]$, $[0.67, 0.99)$, and $[0.33, 0.67)$, respectively.

For the larger data sets, *Alarm* and *Parity2*, we resorted to a priori layering, as unconstrained exact computation is infeasible. For the *Alarm* data set, we divided the 37 variables into two layers of sizes 18 and 19 as illustrated in Figure 5(a). For the *Parity2* data set, we used the “correct” two layers of sizes 78 and 22.

## 7.3 Results on Structure Discovery

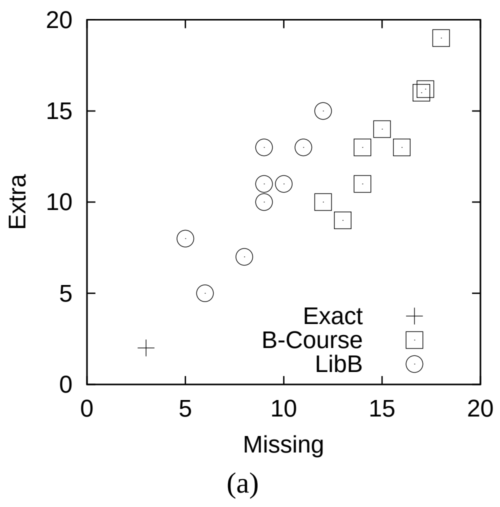
For each data set we computed a single network structure that maximizes the posterior probability. We also computed posterior probabilities of edges for every pair of variables. That is, although the presented methodology supports computation of the posterior probabilities of arbitrary subgraphs, here we only consider directed edges. Here we show results for the data sets *Parity1* and *Alarm* only. For the *Zoo* data set we are not aware of any correct structure, and for the *Parity2* data set the results are similar to the results for the *Parity1* data set.

Figure 4 illustrates possible end results of structure learning on the *Parity1* data sets of different sizes. We see that with the data size of 500 records the best structure found (Figure 4(b)) is almost identical with the underlying correct network (Figure 4(a)). However, the edge between $x_2$ and $x_3$ is reversed and $x_3$ has stolen the children of $x_2$. This can be explained by a strong correlation of $x_2$ and $x_3$. We also see that the underlying pattern of $x_6$, $x_9$, and $x_{11}$ is just partly discovered. With 2500 records there is some improvement: the correct direction between $x_2$ and $x_3$ is identified, $x_{22}$ is correctly detected as a child of $x_2$, and the incorrect edge from $x_3$ to $x_{22}$ is removed (results not shown). This expected learning phenomenon can be observed in finer detail from the posterior probabilities of edges with the data sizes 500 and 2500 (Figure 4(c–d)). A small surprise is that with 2500 records the correct direction between $x_2$ and $x_3$ is less probable than the incorrect one.

Likewise, Figure 5 displays results of structure learning on the *Alarm* data sets of different sizes. We see that with 2500 records almost all correct edges were assigned a high probability and only few edges are added or are missing (meaning low posterior probabilities). An example of a missing edge is the one between the variables 12 and 37. That this edge is not supported by the data is in consistence with earlier studies on the *Alarm* data (e.g., Cooper and Herskovits, 1992).

To investigate whether exact search for an optimal structure can really outperform heuristic search methods, we ran two freely available programs on the *Parity1* data set with 2500 records and on its extended version with 10000 records. B-Course[3] (Myllymäki et al., 2002) is a web-based tool for Bayesian network modeling. Among other features it implements a search algorithm for finding plausible network structures. Because the program runs on a web server, the time of a single trial is limited to 15 minutes (in real time). The user cannot change the parameters of the algorithm which combines greedy and stochastic search heuristics. LibB[4] is a Bayesian network toolbox developed by Nir Friedman and Gal Elidan. It provides various search algorithms based on greedy and stochastic heuristics. Based on preliminary experiments we selected a greedy stochastic hill-climbing algorithm with 20 random restarts so that the time of a single run was about six minutes for the data set with 2500 records. (This is quite fair as our exact algorithm takes about nine minutes.) Since the algorithms implemented in B-Course and LibB are stochastic, we run both programs nine times. Each learned network structure was scored by comparing it against the correct structure. We counted the number of missing edges and the number of extra edges. A reversed edge does not contribute to these error counts if and only if the graph remains "locally equivalent", that is, the correct graph is equivalent[5] to a graph with the reversed edge. It is clear from the results, summarized in Figure 6, that finding an optimal structure is difficult for the tested heuristic methods. For these heuristic methods the number of errors is typically about four times greater than for the exact algorithm. That LibB performs better than B-Course might be explained by the fact that we

3. B-Course is available at http://b-course.cs.helsinki.fi/.
4. LibB is available at http://www.cs.huji.ac.il/labs/compbio/LibB/.
5. Two graphs are equivalent if they have the same skeleton (undirected structure) and the same sets of collapsing edges (also called v-structures).



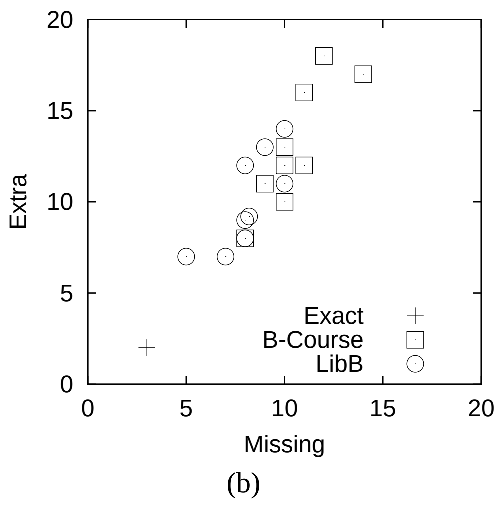


Figure 6: The number of missing and extra edges in graphs found by the exact algorithm and two heuristic methods with nine independent runs each. Structures were learned from the *Parity1* data set with (a) 2500 and (b) 10000 records. For visualization, coincident points are slightly perturbed. The correct structure has 31 edges.

made LibB to use many random restarts, whereas B-Course, perhaps, iterates over a single search chain; however, we do not know the actual search algorithm used by B-Course. Since the number of nodes, 22, is not very large and the optimization landscape is likely to be complex, using several random restarts may be a better strategy. Finally, it is worth noting that we did not set any in-degree bound for the heuristic methods. However, we compensated this by using the bound 5 for the exact algorithm, while the actual maximum in-degree is 4.

### 7.4 The Speed of Exact Structure Discovery

We also measured the speed of the implementation. To get a rather detailed picture of how the total time requirement is composed, separate measurements were carried out for a number of subroutines in the summation and maximization tasks. Table 1 summarizes the speed measurements over different data sets.

In the case of summation, Table 1 reports the time used for computing the functions $\alpha$, $\beta$, and $g$ as defined in Equations (6), (8), and (7), respectively. We denote these three time requirements by $T_\alpha$, $T_\beta$, and $T_g$. Recall that for $T_\alpha$ and $T_g$ we have an exponential bound $O(n2^n)$ whereas for $T_\beta$ we have a polynomial bound $O(n^{k+1}C(m))$. When two layers were used (instead of one) the measurement was done for the function $\alpha'$ expressed in the algorithm in Figure 3 (though reported as $T_\alpha$ for convenience).

In the case of maximization, Table 1 displays the counterparts of these quantities. A difference, however, is that for the latter two quantities we only report their sum. This is because we implemented the maximization method so that it first finds an optimal order and then, based on the order, it finds an optimal graph. This reduces the memory requirement with the expense that we have to

| | | | | Summation Time (sec.) | | | | Maximization Time (sec.) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Name | $n$ | $k$ | $m$ | $T_\alpha$ | $T_\beta$ | $T_g$ | Total | $T_\alpha$ | $T_{\beta+g}$ | Total |
| *Zoo* | 17 | 6 | 101 | 14 | 30 | 1 | 45 | 10 | 36 | 46 |
| *Parity1* | 22 | 5 | 100 | 536 | 17 | 30 | 583 | 379 | 35 | 414 |
| | | | 500 | 531 | 36 | 30 | 597 | 377 | 57 | 434 |
| | | | 2500 | 519 | 132 | 30 | 681 | 378 | 175 | 553 |
| *Alarm* | $19+18$ | 4 | 100 | 66 | 28 | 4 | 98 | 47 | 42 | 89 |
| | | | 500 | 67 | 73 | 4 | 144 | 47 | 104 | 151 |
| | | | 2500 | 63 | 264 | 4 | 331 | 47 | 429 | 476 |
| *Parity2* | $78+22$ | 3 | 100 | 345 | 235 | 32 | 612 | 253 | 448 | 701 |
| | | | 500 | 330 | 1042 | 31 | 1403 | 253 | 1925 | 2178 |
| | | | 2500 | 320 | 5517 | 31 | 5868 | 252 | 10070 | 10322 |

Table 1: The speed of summation and maximization on selected data sets.

recompute some values of the functions $\beta$. That said, in this case the polynomial and exponential contribution are not easily told apart.

The experimental results are in good agreement with the presented asymptotic bounds. We see that the total time requirements are about the same for the summation and maximization tasks. Yet, $T_\alpha$ is consistently less for maximization than for summation. This is because of certain special numeric routines used for adding small numbers. That this difference is not preserved in the total time requirement can be explained by the fact that in the maximization some quantities are computed twice. We also observe that the relationship of the data size $m$ and $T_\beta$ is close to linear, as expected. (The slight distortion from linearity can be explained by the fact that when processing first hundreds records some expensive memory allocation routines are used.) Note also that $T_\alpha$ is invariant with respect to $m$.

Regarding the number of seconds needed for computing the probability of a feature, or, for finding an optimal network structure, it is clear that none of the combinations of model and data parameters used in our experiments lies close to the boundaries of practical tractability: the longest time is about three hours, for finding an optimal structure for the *Parity2* data set with 2500 records. Yet, via the polynomial contribution, the time requirement is highly sensitive to the maximum in-degree $k$. For example, replacing $k = 3$ by $k = 4$ for the *Parity2* data set would result in about 25-fold increase in the time requirement.[6] Also, what is not explicit in the reported time requirements is that feature probabilities are usually computed for a large number of features; in our case, for every directed edge between two variables. Another important fact is that the space requirement of the implemented method is exponential: roughly $n2^n$ floating point values need to be stored. This limits the use of the current implementation to at most 25 variables.

## 8. Concluding Remarks

We have presented a novel algorithm for learning Bayesian network structures from data. Our algorithm computes the exact posterior probability of a queried local subnetwork, e.g., a directed edge between two nodes. A modified version of the algorithm finds a global network structure

6. The increase is from $\binom{100}{3}$ to $\binom{100}{4}$ rather than from $100^3$ to $100^4$ (as hinted by the asymptotic expression).

which maximizes the posterior probability. A major advantage of this method is that it explores all possible structures, still running “only” in an exponential time with respect to the number of network variables. We can expect that this feature makes it possible to successfully analyze cases where inexact methods may fail. Although we provided some bits of evidence for this hypothesis, more convincing validation would require a dedicated comparison study, not pursued in this paper. Here we, instead, reported a set of experiments to illustrate the presented methods and to investigate the actual speed of the implemented algorithms.

Our experiments demonstrate the applicability of the method. For moderate-size networks the running times were found to be feasible. The results also highlight the crucial role of the maximum in-degree and the size of the data (i.e., the polynomial contribution) in the overall complexity. We also showed that structure discovery on large networks is practical if one can judge a suitable layering constraint on the space of network structures. Another main conclusion is that the actual bottleneck in the current implementation is the space requirement, not the speed. For example, a simple extrapolation (not shown) based on the speed measurements (shown in Table 1) reveals that a posterior maximizing structure for the Alarm data (37 variables, 2500 examples, maximum in-degree 4) could be found in about three days, if enough memory is present.

Although the presented algorithm shows promise as an exact method for structure discovery in complex Bayesian networks, there remain many important open problems. From the practical point of view, reducing the space complexity is a question of great importance. An open problem is whether the space requirement can be significantly reduced with little or no computational overhead. Interesting, yet rather theoretic, is the question whether there exists an algorithm faster than the one presented in this paper (asymptotically with respect to the number of variables). Since the maximization and summation problems resemble much the TSP problem and the computation of matrix permanents, respectively—for which no faster algorithms is known—we believe that the answer is negative.

There are many directions in which the presented methods can be extended. A practically important issue is handling missing observations; in this paper we only considered the case of complete data. The full Bayesian solution involves integration over the missing data. Unfortunately, it seems that exact computation is not feasible in general, unless the number of missing values is very small (less than 10, depending on the number of possible values of the variables). MCMC methods provide a practical way to overcome this problem. It is worth noting, however, that the exact summation algorithm readily applies to (posterior) prediction of the values of some variables, given data on the rest of the variables. A different issue is relaxing the structure discovery by allowing for undirected edges. In particular, instead of finding a directed graph that maximizes the posterior probability, one might be interested in searching for an equivalence class of such optimal graphs, conveniently expressed by a partially directed acyclic graph (PDAG); see, e.g., a recent paper by Castelo and Kočka (2003) and references therein. We note that an optimal PDAG can be found indirectly by first finding an optimal directed graph and then turning to its equivalence class. However, it is not clear how the presented algorithms could be used for computing feature probabilities on PDAGs, or whether the techniques used in our work can be extended to handle PDAGs in some more direct way.

Finally, we think that the structural constraints exploited by the presented exact algorithms deserve discussion. The assumptions of order-modularity and graph-modularity are built on related previous work (Cooper and Herskovits, 1992; Heckerman et al., 1995a; Friedman and Koller, 2003). While the former essentially factorizes the conditional prior distribution $p(G \mid \prec)$ of structures given a variable order, the latter factorizes the unconditional prior $p(G)$. These assumptions are not only vital for efficient computation, but also offer a convenient and often suitable form for expressing prior beliefs. Nevertheless, modular priors can be criticized as the posterior distribution typically will not remain modular. A novelty in the work of Friedman and Koller (2003) was the introduction of a prior distribution $p(\prec)$ on orders; we took a small step further assuming that this distribution factorizes. Accordingly, the modeler should be able to think of a "true order", which may be difficult if not unjustified. However, treating the order as a technical artifact without any interpretation removes the problem. Then the remaining problem is to justify the resulting form of the structure prior $p(G)$. How well this form in practice matches the modeler's prior is likely to be case-dependent.

In addition to the modularity assumption, we required that the number of parents of any variable is bounded by a constant. This is a convenient way to reduce the size of the search space. Unfortunately, only seldom is a small hard bound justified by background knowledge. However, under certain regularity assumptions one can argue that no variable should have more than about $\log_2 m$ parents, where $m$ is the data size (Bouckaert, 1994). This is because sufficient amount of evidence for many parents is needed to compensate the cost of model complexity. A similar result is due to Höffgen (1993) who shows that about $2^k$ data records are sufficient (maybe also needed) to learn Boolean networks, where each variable has at most $k$ parents. These results—though rather obvious when ignoring the issues of accuracy and confidence—recognize that if the modeler's background knowledge is vague, then a moderate in-degree bound is justified. On the other hand, it is worth noting that the algorithms presented in this paper work even if no in-degree bound is fixed. It is not difficult to see that then the time complexity is $O(n2^nC(m,n))$, where $C(m,n)$ is the time needed for computing a single local conditional marginal for $m$ data records over at most $n$ variables.

## Acknowledgments

The authors thank Heikki Mannila for useful discussions and the anonymous reviewers for comments that helped to improve the manuscript. This work was partly supported and inspired by the Morgam (www.ktl.fi/morgam) and GenomeEUTwin (www.genomeutwin.org) projects.

## References

S. Acid and L. de Campos. Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, 18:445–490, 2003.

S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

I. Beinlich, G. Suermondt, R. Chavez, and G. F. Cooper. The ALARM monitoring system. In J. Hunter, editor, *Proceedings of the Second European Conference on Artificial Intelligence and Medicine*, pages 247–256. Springer-Verlag, Berlin, 1989.

R. Bellman. Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM*, 9:61–63, 1962.

C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998.

R. R. Bouckaert. Properties of Bayesian belief network learning algorithms. In R. Lopez de Mantaras and D. Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 102–109, Seattle, WA, 1994. Morgan Kaufmann, San Francisco, CA.

W. Buntine. Theory refinement on Bayesian networks. In B. D'Ambrosio, P. Smets, and P. Bonissone, editors, *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60, Los Angeles, CA, 1991. Morgan Kaufmann, San Mateo, CA.

R. Castelo and T. Kočka. On inclusion-driven learning of Bayesian networks. *Journal of Machine Learning Research*, 4:527–574, 2003.

D. M. Chickering. Optimal structure indentification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

D. M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks: Search methods and experimental results. In *Proceedings of the Fifth Conference on Artificial Intelligence and Statistics,* pages 112–128. Society for Artificial Intelligence and Statistics, Ft. Lauderdale, 1995.

D. M. Chickering, D. Heckerman, and C. Meek. A Bayesian approach to learning Bayesian networks with local structure. In D. Geiger and P. Shenoy, editors, *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 80–89, Providence, RI, 1997. Morgan Kaufmann, San Francisco, CA.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

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

R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113 (1-2):41–85, 1999.

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

A. E. Gelfand and A. F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85:398–409, 1990.

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

D. Heckerman, A. Mamdani, and M. P. Wellman. Real-world applications of Bayesian networks. *Communications of the ACM*, 38(3):24–30, 1995b.

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

E. Herskovits. *Computer-Based Probabilistic Network Construction*. PhD thesis, Medical Information Sciences, Stanford University, 1991.

K.-U. Höffgen. Learning and robust learning of product distributions. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 77–83, Santa Cruz, CA, USA, 1993. ACM Press.

R. Kennes and P. Smets. Computational aspects of the Möbius transformation. In P. B. Bonissone, M. Henrion, L. N. Kanal, and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence 6*, pages 401–416. North-Holland, Amsterdam, 1991.

P. Larrañaga, C. Kuijpers, R. Murga, and Y. Yurramendi. Learning Bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(4):487–493, 1996.

S. L. Lauritzen and F. V. Jensen. Local computation with valuations from a commutative semigroup. *Annals of Mathematics and Artificial Intelligence*, 21(1):51–69, 1997.

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

P. Myllymäki, T. Silander, H. Tirri, and P. Uronen. B-Course: A web-based tool for Bayesian and causal data analysis. *International Journal on Artificial Intelligence Tools*, 11(3):369–387, 2002.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, Cambridge, 2000.

R. E. Stearns and H. B. Hunt III. An algebraic model for combinatorial problems. *SIAM Journal on Computing*, 25(2):448–476, 1996.