8-27-2014

Computing Posterior Probabilities of Ancestor Relations in Bayesian Networks

Yetian Chen
Department of Computer Science, Iowa State University, yetianc@iastate.edu

Jin Tian
Department of Computer Science, Iowa State University, jtian@iastate.edu

Computing Posterior Probabilities of Ancestor Relations in Bayesian Networks

Abstract
In this paper we develop a dynamic programming algorithm to compute the exact posterior probabilities of ancestor relations in Bayesian networks. Previous dynamic programming (DP) algorithm by (Parviainen and Koivisto, 2011) evaluates all possible ancestor relations in time $O(n3^n)$ and space $O(3^n)$. However, their algorithm assumes an order-modular prior over DAGs that does not respect Markov equivalence. The resulting posteriors would bias towards DAGs consistent with more linear orders. To adhere to uniform prior, we develop a new DP algorithm that computes the exact posteriors of all possible ancestor relations in time $O(n5^{n-1})$ and space $O(3^n)$.

Keywords
Posterior probabilities, algorithm, ancestor relation, Bayesian networks, dynamic programming

Disciplines
Artificial Intelligence and Robotics | Computer Sciences

Computing Posterior Probabilities of Ancestor Relations in Bayesian Networks

Yetian Chen yetianc@iastate.edu
Jin Tian jtian@iastate.edu
Department of Computer Science
Iowa State University
Ames, IA 50011, USA

Editor:

Abstract

In this paper we develop a dynamic programming algorithm to compute the exact posterior probabilities of ancestor relations in Bayesian networks. Previous dynamic programming (DP) algorithm by (Parviainen and Koivisto, 2011) evaluates all possible ancestor relations in time $O(n3^n)$ and space $O(3^n)$. However, their algorithm assumes an order-modular prior over DAGs that does not respect Markov equivalence. The resulting posteriors would bias towards DAGs consistent with more linear orders. To adhere to uniform prior, we develop a new DP algorithm that computes the exact posteriors of all possible ancestor relations in time $O(n5^{n-1})$ and space $O(3^n)$.

Keywords: Bayesian networks, Posterior probabilities, Ancestor relations, Dynamic programming

1. Introduction

Bayesian networks (BN), representing a set of random variables and their conditional dependencies via directed acyclic graph (DAG), have been widely used for probabilistic inference and causal modeling (Pearl, 2000; Spirtes et al., 2000). In particular, the DAG structure supports specifically causal interpretations. For example, a directed edge represents direct causal relation between two variables; a directed path, composed of consecutively directed edges, represents (indirect) causal relation among two variables. Learning these structures from observational data has been a major challenge.

Traditional model selection approach seeks out a maximum-a-posteriori (MAP) BN G and infers the structures based on this single model. This is problematic because: (1) the assumed "data generating DAG" is unidentifiable from the observational data due to the so-called Markov equivalence of multiple different DAGs (Verma and Pearl, 1990); and (2) other Markov equivalence classes may fit the data almost equally well due to the noises in the data (Friedman and Koller, 2003).

Bayesian approach circumvents the model uncertainty problem by learning the posterior distribution of these structural features (Friedman and Koller, 2003). However, exact computation of these posteriors is hard due to the super-exponentially large DAG space. Therefore, many researches resorted to approximate methods either based on statistical sampling techniques (Madigan et al., 1995; Friedman and Koller, 2003; Eaton and Murphy,

2007; Ellis and Wong, 2008; Grzegorczyk and Husmeier, 2008; Niinimäki et al., 2011; Niinimäki and Koivisto, 2013) or by averaging over top models (Tian et al., 2010; Chen and Tian, 2014). Computing the exact posterior probabilities of structural features is hard, but still tractable in certain cases. Assuming an order-modular prior over DAGs and bounded indegree, a dynamic programming (DP) algorithm can compute the posterior probabilities of modular features, e.g., directed edges, in $O(n2^n)$ time and $O(2^n)$ space (Koivisto and Sood, 2004; Koivisto, 2006). To deal with (harder) non-modular feature, i.e., ancestor relations, an analogous DP algorithm takes $O(n3^n)$ time and $O(3^n)$ space (Parviainen and Koivisto, 2011). As mentioned, these algorithms require special form of structural prior $P(G)$, thus perform summation over order space instead of DAG space. As a result, the computed posteriors would bias towards DAGs consistent with more linear orders and the Markov equivalence is not respected either. To adhere to the uniform prior, Tian and He (2009) developed a novel DP algorithm directly summing over the DAG space. This algorithm is capable of evaluating all directed edges (modular features) in $O(n3^n)$ time and $O(n2^n)$ space.

In this paper we extend Tian and He (2009)'s work and develop a novel algorithm to compute the exact posterior probabilities of ancestor relations (directed path) in Bayesian networks. Unlike the DP algorithm by Parviainen and Koivisto (2011), our algorithm uses uniform prior, thus respects Markov equivalence.

2. Bayesian Learning of Ancestor relations

2.1 Preliminaries

Formally, a Bayesian network is a DAG that encodes a joint probability distribution over a vector of random variables $x = (x_1, ..., x_n)$ with each node of the graph representing a variable in x. For convenience we will typically work on the index set $V = \{1, ..., n\}$ and represent a variable x_i by its index i. The DAG is represented by a vector $G = (Pa_1, ..., Pa_n)$ where each Pa_i is a subset of the index set V and specifies the parents of i in the graph.

Given an observational data D, the joint distribution $P(G, D)$ is composed as

$$P(G, D) = P(G)P(D|G), \tag{1}$$

where $P(G)$ specifies the structure prior, and $P(D|G)$ is the data likelihood.

Assuming global and local parameter independence, and parameter modularity (Cooper and Herskovits, 1992; Friedman and Koller, 2003), the data likelihood $P(D|G)$ can further be decomposed into

$$P(D|G) = \prod_{i \in V} score_i(Pa_i : D), \tag{2}$$

where $score_i(Pa_i : D)$ is the so-called local scores and has a closed-form solution.

Moreover, the structure modularity assumes

$$P(G) = \prod_{i \in V} Q_i(Pa_i), \tag{3}$$

where $Q_i(Pa_i)$ is some function from the subsets of $V \setminus \{i\}$ to the non-negative reals (Friedman and Koller, 2003).

2.2 Algorithm

We say s is an ancestor of t, or t is a descendant of s, if G contains a directed path from s to t, denoted as $s \rightsquigarrow t$. In Bayesian approach, it is more convenient to compute the posterior probability of an ancestor relation $s \rightsquigarrow t$ by

$$P(s \rightsquigarrow t|D) = P(s \rightsquigarrow t, D)/P(D). \tag{4}$$

The joint probability $P(s \rightsquigarrow t, D)$ can be computed by

$$\begin{aligned} P(s \rightsquigarrow t, D) &= \sum_G \delta(s \rightsquigarrow t \in G) P(D|G) P(G) \\ &= \sum_G \delta(s \rightsquigarrow t \in G) \prod_{i \in V} Q_i(Pa_i) score_i(Pa_i : D) \\ &= \sum_G \delta(s \rightsquigarrow t \in G) \prod_{i \in V} B_i(Pa_i), \end{aligned} \tag{5}$$

where for all $Pa_i \subseteq V \setminus \{i\}$ we define

$$B_i(Pa_i) \equiv Q_i(Pa_i) score_i(Pa_i : D), \tag{6}$$

and $\delta(X)$ is the Kronecker delta function, taking value of 1 if X is true and 0 otherwise.

For any T, S such that $T \subseteq S \subseteq V$, let $\mathcal{G}_S(T)$ denote the set of all possible DAGs over S such that $G_S \in \mathcal{G}_S(T)$ if and only if G_S contains a path $s \rightsquigarrow v$ for every $v \in T$, and no path from s to any $v \in S \setminus T$. Define

$$H_s(S, T) \equiv \sum_{G_S \in \mathcal{G}_S(T)} \prod_{i \in S} B_i(Pa_i), \tag{7}$$

where $H_s(S, T) = 0$ if $(s \notin T \wedge T \neq \emptyset)$ or $(s \notin T \wedge s \in S)$, $H_s(S, T) = 1$ if $S = \emptyset \wedge T = \emptyset$. Then we have

Proposition 1

$$P(s \rightsquigarrow t, D) = \sum_{s,t \in T \subseteq V} H_s(V, T). \tag{8}$$

Proof. Let $\mathcal{G}_{s \rightsquigarrow t} = \{G : s \rightsquigarrow t \in G\}$, namely the set of all possible DAGs over V that contains a $s \rightsquigarrow t$. Then we have $\mathcal{G}_{s \rightsquigarrow t} = \cup_{s,t \in T \subseteq V} \mathcal{G}_V(T)$. Further, for any $T_1 \neq T_2$, we have $\mathcal{G}_V(T_1) \cap \mathcal{G}_V(T_2) = \emptyset$. This means $\mathcal{G}_V(T)$ for all $s, t \in T \subseteq V$ form a partition of the set $\mathcal{G}_{s \rightsquigarrow t}$. Thus,

$$\begin{aligned} P(s \rightsquigarrow t, D) &= \sum_G \delta(s \rightsquigarrow t \in G) \prod_{i \in V} B_i(Pa_i) = \sum_{G \in \mathcal{G}_{s \rightsquigarrow t}} \prod_{i \in V} B_i(Pa_i) \\ &= \sum_{s,t \in T \subseteq V} \sum_{G \in \mathcal{G}_V(T)} \prod_{i \in V} B_i(Pa_i) = \sum_{s,t \in T \subseteq V} H_s(V, T). \end{aligned} \tag{9}$$

□

Now the problem is decomposed into computing $H_s(V, T)$ for all T s.t. $s, t \in T \subseteq V$. We show that $H_s(S, T)$ for all $T \subseteq S \subseteq V$ can be computed recursively. We first handle some special cases.

If $T = \emptyset$ and $s \notin S$, $\mathcal{G}_S(\emptyset)$ contains all possible DAGs over S. (Tian and He, 2009) proposed a DP algorithm to sum over $\mathcal{G}_S(\emptyset)$ by exploiting possible sinks of DAGs and inclusion-exclusion principle. Due to **Proposition 2** in (Tian and He, 2009), we have following result,

$$H_s(S, \emptyset) = \sum_{k=1}^{|S|} (-1)^{k+1} \sum_{W \subseteq S, |W|=k} H_s(S \setminus W, \emptyset) \prod_{j \in W} A_j(S \setminus W), \tag{10}$$

where

$$A_j(U) \equiv \sum_{Pa_j \subseteq U} B_j(Pa_j). \tag{11}$$

Now we consider the case where $T \neq \emptyset$ and $s \in T$. Let $\mathcal{G}_S(T, W)$ denote the set of DAGs in $\mathcal{G}_S(T)$ such that all nodes in $W \subseteq S$ are (must be) sinks. Note that $\mathcal{G}_S(T, \emptyset) = \mathcal{G}_S(T)$. For any $W \subseteq S$ and $T \subseteq S$, define

$$F(S, T, W) \equiv \sum_{G_s \in \mathcal{G}_S(T,W)} \prod_{i \in S} B_i(Pa_i). \tag{12}$$

Since every DAG has at least one sink, we have $\mathcal{G}_S(T) = \cup_{j \in S} \mathcal{G}_S(T, \{j\})$. Further, it is clear that $\cap_{j \in W} \mathcal{G}_S(T, \{j\}) = \mathcal{G}_S(T, W)$. Then the summation over $\mathcal{G}_S(T)$ in Eq. (7) can be computed by summing over the DAGs in $\mathcal{G}_S(T, \{j\})$ separately and correcting the overlaps. By weighted inclusion-exclusion principle,

$$\begin{aligned} H_s(S, T) &= \sum_{k=1}^{|S|} (-1)^{k+1} \sum_{W \subseteq S, |W|=k} \sum_{G_s \in \mathcal{G}_S(T,W)} \prod_{i \in S} B_i(Pa_i) \\ &= \sum_{k=1}^{|S|} (-1)^{k+1} \sum_{W \subseteq S, |W|=k} F(S, T, W). \end{aligned} \tag{13}$$

$H_s(S, T)$ and $F(S, T, W)$ can be computed recursively. The central idea is to convert the sum of products in Eq. (12) to product of sums. Several cases should be handled separately. If $T \cap W = \emptyset$, the sum of products in Eq. (12) can be freely decomposed to product of sums for nodes in W and sum over remaining nodes in $S \setminus W$,

$$\begin{aligned} F(S, T, W) &= [\prod_{j \in W} \sum_{Pa_j \subseteq (S \setminus T) \setminus W} B_j(Pa_j)][\sum_{G_S \in \mathcal{G}_{S \setminus W}(T)} \prod_{i \in S \setminus W} B_i(Pa_i)] \\ &= \prod_{j \in W} A_j((S \setminus T) \setminus W) H_s(S \setminus W, T) = \prod_{j \in W} A_j((S \setminus T) \setminus W) H_s(S \setminus W, T \setminus W). \end{aligned} \tag{14}$$

If $T \cap W \neq \emptyset$ and $s \notin T \cap W$, nodes in $W \setminus T$, $T \cap W$, and $S \setminus W$ should be handled separately,

$$
\begin{aligned}
&F(S,T,W) \qquad (15)\\
&= [\prod_{j \in T \cap W} \sum_{\substack{Pa_j \subseteq (S \setminus W) \\ Pa_j \cap (T \setminus W) \neq \emptyset}} B_j(Pa_j)][\prod_{j \in W \setminus T} \sum_{Pa_j \subseteq (S \setminus T) \setminus W} B_j(Pa_j)][\sum_{G_S \in \mathcal{G}_{S \setminus W}(T \setminus W)} \prod_{i \in S \setminus W} B_i(Pa_i)]\\
&= [\prod_{j \in T \cap W} (\sum_{Pa_j \subseteq (S \setminus W)} B_j(Pa_j) - \sum_{Pa_j \subseteq (S \setminus W) \setminus T} B_j(Pa_j))] \prod_{j \in W \setminus T} A_j((S \setminus T) \setminus W) H_s(S \setminus W, T \setminus W)\\
&= [\prod_{j \in T \cap W} (A_j(S \setminus W) - A_j((S \setminus W) \setminus T))] \prod_{j \in W \setminus T} A_j((S \setminus T) \setminus W) H_s(S \setminus W, T \setminus W).
\end{aligned}
$$

If $T \cap W \neq \emptyset$ and $s \in T \cap W$,

$$
\begin{aligned}
F(S,T,W) \; &= [\prod_{j \in W \setminus T} \sum_{Pa_j \subseteq (S \setminus T) \setminus W} B_j(Pa_j)][\sum_{G_S \in \mathcal{G}_{S \setminus (W \setminus T)}(T, T \cap W)} \prod_{i \in S \setminus (W \setminus T)} B_i(Pa_i)] \qquad (16)\\
&= [\prod_{j \in W \setminus T} \sum_{Pa_j \subseteq (S \setminus T) \setminus W} B_j(Pa_j)][\prod_{j \in T \cap W} \sum_{Pa_j \subseteq (S \setminus T) \setminus W} B_j(Pa_j)][\sum_{G_S \in \mathcal{G}_{S \setminus W}(T \setminus W, \emptyset)} \prod_{i \in S \setminus W} B_i(Pa_i)]\\
&= \prod_{j \in W} A_j((S \setminus T) \setminus W) H_s(S \setminus W, T \setminus W).
\end{aligned}
$$

In summary, if $T \cap W = \emptyset$ or $s \in T \cap W$,

$$F(S,T,W) \; = \prod_{j \in W} A_j((S \setminus T) \setminus W) H_s(S \setminus W, T \setminus W), \qquad (17)$$

else if $T \cap W \neq \emptyset$ and $s \notin T \cap W$,

$$
\begin{aligned}
&F(S,T,W)\\
&= [\prod_{j \in T \cap W} (A_j(S \setminus W) - A_j((S \setminus W) \setminus T))] \prod_{j \in W \setminus T} A_j((S \setminus T) \setminus W) H_s(S \setminus W, T \setminus W). \qquad (18)
\end{aligned}
$$

For ease of exposition, define function $\mathcal{A}$ as follows:
If $T \cap W = \emptyset$ or $s \in T \cap W$,

$$\mathcal{A}(S,T,W) \; \equiv \prod_{j \in W} A_j((S \setminus T) \setminus W), \qquad (19)$$

else if $T \cap W \neq \emptyset$ and $s \notin T \cap W$,

$$\mathcal{A}(S,T,W) \; \equiv [\prod_{j \in T \cap W} (A_j(S \setminus W) - A_j((S \setminus W) \setminus T))] \prod_{j \in W \setminus T} A_j((S \setminus T) \setminus W). \qquad (20)$$

Now $F(S,T,W)$ can be neatly written as

$$F(S,T,W) \; = \mathcal{A}(S,T,W) H_s(S \setminus W, T \setminus W) \qquad (21)$$

Then $H_s(S,T)$ can be written as

$$H_s(S,T) \ = \sum_{k=1}^{|S|}(-1)^{k+1} \sum_{W\subseteq S,|W|=k} \mathcal{A}(S,T,W)H_s(S\setminus W, T\setminus W) \tag{22}$$

In summary, we arrive at the following recursive scheme for computing any $H_s(S,T)$.
Algorithm 1

$$\begin{aligned} H_s(S,T) \ &= 1 \ \text{ if } S=\emptyset \wedge T=\emptyset, \\ H_s(S,T) \ &= 0 \ \text{ if } (s\notin T \wedge T\neq\emptyset) \text{ or } (s\notin T \wedge s\in S), \\ H_s(S,T) \ &= \sum_{k=1}^{|S|}(-1)^{k+1} \sum_{W\subseteq S,|W|=k} \mathcal{A}(S,T,W)H_s(S\setminus W, T\setminus W) \ \text{ otherwise.} \end{aligned} \tag{23}$$

□

Note that the third equation in Eqs.(23) has generalized the cases from Eq.(10) and Eq.(22).

2.3 Efficient Computation of $\mathcal{A}(S,T,W)$

To facilitate the computation of $\mathcal{A}(S,T,W)$, we define for any $W\subseteq V, U\subseteq V\setminus W$,

$$AA(U,W) \ \equiv \prod_{j\in W} A_j(U). \tag{24}$$

Due to Tian and He (2009), for a fixed U, we have

$$AA(U,W) \ = A_j(U)AA(U, W\setminus\{j\}) \text{ for any } j\in W. \tag{25}$$

Thus, for a fixed U, $AA(U,W)$ for all $W\subseteq V\setminus U$ can be computed in the manner of dynamic programming in $O(2^{n-|U|})$ time. Then $AA(U,W)$ for all $U\subseteq V$ and all $W\subseteq V\setminus\{U\}$ can be computed in $\sum_{|U|=0}^{n}\binom{n}{|U|}2^{n-|U|} = 3^n$ time. With the precomputation of $AA(U,W)$, $\mathcal{A}(S,T,W)$ can be computed more efficiently, in constant time $O(1)$:
If $T\cap W=\emptyset$ or $s\in T\cap W$,

$$\mathcal{A}(S,T,W) \ = AA((S\setminus T)\setminus W, W), \tag{26}$$

else if $T\cap W\neq\emptyset$ and $s\notin T\cap W$,

$$\begin{aligned} \mathcal{A}(S,T,W) &= \prod_{j\in T\cap W} A_j(S\setminus W) \prod_{j\in W\setminus T} A_j((S\setminus T)\setminus W) - \prod_{j\in W} A_j((S\setminus T)\setminus W) \\ &= AA(S\setminus W, T\cap W)AA((S\setminus T)\setminus W, W\setminus T) - AA((S\setminus T)\setminus W, W). \end{aligned} \tag{27}$$

2.4 Time and Space Complexity

To avoid redundant computations, the recursive computation of $H_s(S,T)$ for all $S\subseteq V$ and $T\subseteq S$ proceeds in the lexicographic order of S and T, with T as the outer loop and S as the inner loop. For example, we start the computation of $H_s(S,\emptyset)$ for all $S\subseteq V$ in

lexicographic order. Then we compute $H_s(S,\{j\})$ for all $\{j\} \subseteq S \subseteq V$ in lexicographic order, so on and so forth and finally we compute $H_s(V,V)$.

Since $H_s(S,T) = 0$ if $(s \notin T \wedge T \neq \emptyset)$ or $(s \notin T \wedge s \in S)$, $H_s(S,T) = 1$ if $S = \emptyset \wedge T = \emptyset$, we don't need to compute $H_s(S,T)$ that belongs to any of these cases.

All $H_s(S,T)$ such that $T = \emptyset$ and $s \notin S$ can be computed by Eq.(10) in $\sum_{|S|=1}^{n-1} \binom{n-1}{|S|} 2^{|S|} = 3^{n-1}$ time.

If $T \neq \emptyset$ and $s \in T$, we compute $H_s(S,T)$ by Eq.(22). Since $H_s(S \setminus W, T \setminus W) = 0$ for any W such that $s \in W$, the actual computation time is $2^{|S|-1} + 2^{|S|-|T|}$. Thus, all $H_s(S,T)$ can be computed in time

$$
\begin{aligned}
&\sum_{|S|=1}^{n} \left\{ \binom{n-1}{|S|-1} \left[\sum_{|T|=1}^{|S|} \binom{|S|-1}{|T|-1} (2^{|S|-1} + 2^{|S|-|T|}) \right] \right\} \\
&= \sum_{|S|=1}^{n} \left\{ \binom{n-1}{|S|-1} \left[2^{|S|-1} \sum_{|T|=1}^{|S|} \binom{|S|-1}{|T|-1} (1 + 2^{-(|T|-1)}) \right] \right\} \\
&= \sum_{|S|=1}^{n} \left\{ \binom{n-1}{|S|-1} \left[2^{|S|-1} \left(2^{|S|-1} + \left(\frac{3}{2}\right)^{|S|-1} \right) \right] \right\} \\
&= \sum_{|S|=1}^{n} \left\{ \binom{n-1}{|S|-1} \left[4^{|S|-1} + 3^{|S|-1} \right] \right\} \\
&= 5^{n-1} + 4^{n-1}.
\end{aligned}
$$

After we obtain $H_s(V,T)$ for all $T \subseteq V$ such that $s,t \in T$, we can compute $P(s \rightsquigarrow t, D)$ by Eq.(8) in $O(2^{n-2})$ time. $P(D)$ can be computed in $O(3^n)$ time using the algorithm presented in Tian and He (2009). Then we obtain $P(s \rightsquigarrow t|D) = P(s \rightsquigarrow t, D)/P(D)$. Thus, the total computation time is $O(5^{n-1} + 4^{n-1} + 3^n) = O(5^{n-1})$. To compute the posterior probabilities for all node pairs s,t, it suffices to repeat the computations for each $s \in V$, as the values $H_s(S,T)$ actually contain the sufficient information regarding all possible descendant nodes t. Thus, the total time for computing all possible ancestor relations $s \rightsquigarrow t$ is $O(n5^{n-1})$.

$B_i(Pa_i)$ for all $i \in V$, $Pa_i \subseteq V \setminus \{i\}$ take $O(n2^{n-1})$ space. $A_j(U)$ for all $j \in V$, $U \subseteq V \setminus \{j\}$ take $O(n2^{n-1})$ space. $H_s(S,T)$ for all $T \subseteq S \subseteq V$ consume $\sum_{|S|=0}^{n} \binom{n}{|S|} 2^{|S|} = O(3^n)$ space, and $AA(U,W)$ for all $U \subseteq V$, $W \subseteq V \setminus U$ consume $\sum_{|U|=0}^{n} \binom{n}{|U|} 2^{n-|U|} = O(3^n)$ space. The total space requirement is therefore $O(3^n + n2^n)$.

3. Conclusions

In this paper, we have developed a new DP algorithm to compute the exact posteriors of all possible ancestor relations in Bayesian networks. Our algorithm adheres to the uniform structure prior so that the Markov equivalence is respected.

References

Yetian Chen and Jin Tian. Finding the k-best equivalence classes of Bayesian network structures for model averaging. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

Gregory F Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9(4):309–347, 1992.

Daniel Eaton and Kevin Murphy. Bayesian structure learning using dynamic programming and MCMC. In *Proceedings of the 23th Conference on Uncertainty in Artificial Intelligence*, 2007.

Byron Ellis and Wing Hung Wong. Learning causal Bayesian network structures from experimental data. *Journal of the American Statistical Association*, 103(482), 2008.

Nir Friedman and Daphne Koller. Being Bayesian about network structure. a Bayesian approach to structure discovery in Bayesian networks. *Machine learning*, 50(1-2):95–125, 2003.

Marco Grzegorczyk and Dirk Husmeier. Improving the structure MCMC sampler for Bayesian networks by introducing a new edge reversal move. *Machine Learning*, 71(2-3): 265–305, 2008.

Mikko Koivisto. Advances in exact Bayesian structure discovery in Bayesian networks. In *Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence*, 2006.

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

David Madigan, Jeremy York, and Denis Allard. Bayesian graphical models for discrete data. *International Statistical Review/Revue Internationale de Statistique*, pages 215–232, 1995.

Teppo Niinimäki and Mikko Koivisto. Annealed importance sampling for structure learning in Bayesian networks. In *23rd International Joint Conference on Artificial Intelligence (IJCAI-13)*, 2013.

Teppo Mikael Niinimäki, Pekka Parviainen, Mikko Koivisto, et al. Partial order MCMC for structure discovery in Bayesian networks. In *Proceedings of the Twenty-Seventh Conference Conference on Uncertainty in Artificial Intelligence (UAI-11)*, 2011.

Pekka Parviainen and Mikko Koivisto. Ancestor relations in the presence of unobserved variables. In *Machine Learning and Knowledge Discovery in Databases*, pages 581–596. 2011.

Judea Pearl. *Causality: models, reasoning and inference*, volume 29. Cambridge Univ Press, 2000.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, prediction, and search*, volume 81. The MIT Press, 2000.

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

Jin Tian, Ru He, and Lavanya Ram. Bayesian model averaging using the k-best Bayesian network structures. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, 2010.

Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, pages 255–270, 1990.