Exact Bayesian Learning of Ancestor Relations in Bayesian Networks

Yetian Chen **Lingjian Meng** **Jin Tian**
Department of Computer Science, Iowa State University, Ames, IA 50011, USA

Abstract

Ancestor relations in Bayesian networks (BNs) encode long-range causal relations among random variables. In this paper, we develop dynamic programming (DP) algorithms to compute the exact posterior probabilities of ancestor relations in Bayesian networks. Previous 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 the uniform prior, we develop a new DP algorithm that computes the exact posteriors of all possible ancestor relations in time $O(n^2 5^{n-1})$ and space $O(3^n)$. We also discuss the extension of our algorithm to computing the posteriors of $s \rightsquigarrow p \rightsquigarrow t$ relations, i.e., a directed path from s to t via p. We apply our algorithm to a biological data set for discovering protein signaling pathways.

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 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) devel-

Appearing in Proceedings of the 18^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2015, San Diego, CA, USA. JMLR: W&CP volume 38.

oped a novel DP algorithm directly summing over the DAG space. Their 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 the standard structure-modular prior, thus respects uniform prior and Markov equivalence.

2 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 $G = (Pa_1^G, ..., Pa_n^G)$ where each Pa_i^G is a subset of the index set $V - \{i\}$ and specifies the parents of i in the graph G.

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.

With standard assumptions on the parameter priors (Dirichlet prior for multinomial random variables, Wishart prior for Gaussian random variables) including global and local parameter independence and parameter modularity (Cooper and Herskovits, 1992; Friedman and Koller, 2003), the data likelihood $P(D|G)$ can be decomposed into

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

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

Moreover, the *structure modularity* assumes

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

where $Q_i(Pa_i^G)$ is some function from the subsets of $V - \{i\}$ to the non-negative reals. For ease of exposition, we define, for any $i \in V$ and $Pa_i^G \subseteq V - \{i\}$

$$B_i(Pa_i^G) \equiv Q_i(Pa_i^G) score_i(Pa_i^G : D). \tag{4}$$

Then

$$P(G, D) = \prod_{i \in V} B_i(Pa_i^G). \tag{5}$$

3 PREVIOUS APPROACHES

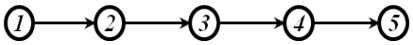
Let f be a structural feature represented by an indicator function such that $f(G)$ is 1 if the feature is present in G and 0 otherwise. In Bayesian approach, we are interested in computing the posterior $P(f|D)$ of the feature, which can be obtained by computing the joint probability $P(f, D)$ as

$$P(f, D) = \sum_G f(G)P(G, D). \tag{6}$$

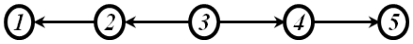
The summation is intractable in practice since the number of all possible DAGs is $O(n!\, 2^{n(n-1)/2})$. Thus, much research has proposed to work on the order space (Friedman and Koller, 2003; Koivisto and Sood, 2004; Koivisto, 2006; Parviainen and Koivisto, 2011). Formally, an order $\prec$ is a linear order $(L_1, ..., L_n)$ on the index set V, where L_i specifies the predecessors of i in the order, i.e., $L_i = \{j : j \prec i\}$. We say that a structure $G = (Pa_1, ..., Pa_n)$ is consistent with an order $\prec$, denoted by $G \in \prec$, if $Pa_i \subseteq L_i$ for all i. Then we can compute

$$P(f, D) = \sum_{\prec} P(\prec) \sum_{G \in \prec} f(G)P(D|G)P(G|\prec). \tag{7}$$

It turns out with such treatment, the computation can be more efficient and convenient. Indeed, it has been shown that the posteriors of all possible ancestor relations can be evaluated in time $O(n3^n)$ and space $O(3^n)$ using this order-based summation scheme (Parviainen and Koivisto, 2011).



(a) A chain DAG



(b) A tree DAG

Figure 1: Two Markov equivalent DAGs

This treatment is problematic because it treats different variable orders as mutually exclusive events. However, the corresponding sets of consistent DAGs are overlapping. If we introduce a uniform prior $P(\prec)$ and a uniform $P(G|\prec)$, the resulting prior $P(G)$ is not uniform. It weights the DAGs by the number of linear extensions. For example, an empty network without any edge is consistent with $n!$ linear orders, while a chain network (see Figure 1(a)) is consistent with only one linear order. The resulting posteriors will bias towards DAGs consistent with more linear orders. For the same reason, two Markov equivalent DAGs (Pearl, 2000) may receive unequal priors. For example, the two DAGs shown in Figure 1 are Markov equivalent. However, the tree DAG in Figure 1(b) will be weighted

6 times as the chain DAG in Figure 1(c). Thus, the Markov equivalence is not respected.

In this paper, we will develop a novel algorithm for Bayesian learning of ancestor relations that directly performs summation over the DAG space by exploiting sinks. Our algorithm allows the uniform prior $P(G)$ and respects the Markov equivalence.

4 BAYESIAN LEARNING OF ANCESTOR RELATIONS

4.1 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 $s \rightsquigarrow t$. The posterior probability of an ancestor relation $s \rightsquigarrow t$ is evaluated by

$$P(s \rightsquigarrow t|D) = P(s \rightsquigarrow t, D)/P(D). \tag{8}$$

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

$$P(s \rightsquigarrow t, D) = \sum_{G \in \mathcal{G}_{s \rightsquigarrow t}} P(G, D) = \sum_{G \in \mathcal{G}_{s \rightsquigarrow t}} \prod_{i \in V} B_i(Pa_i^G), \tag{9}$$

where $\mathcal{G}_{s \rightsquigarrow t} \equiv \{G : s \rightsquigarrow t \in G\}$, namely the set of all possible DAGs over V that contain a $s \rightsquigarrow t$.

For any $S \subseteq V$, let G_S denote a DAG over S. For any $v \in S$, let $Pa_v^{G_S}$ be the parent set of v in G_S, and $de_{G_S}(v) \equiv \{u | u \leftarrow \cdots \leftarrow v \text{ in } G_S \text{ or } u = v\}$ be the set of all descendants of v (including v) in G_S. For any T, S such that $s \in T \subseteq S \subseteq V$, let $\mathcal{G}_s(S, T)$ denote the set of all possible DAGs over S such that T are the set of all descendants of s in G_S. That is, $G_S \in \mathcal{G}_s(S, T)$ if and only if $de_{G_S}(s) = T$. We define, for any $s \in T \subseteq S \subseteq V$,

$$H_s(S, T) \equiv \sum_{G_S \in \mathcal{G}_s(S,T)} \prod_{i \in S} B_i(Pa_i^{G_S}). \tag{10}$$

Then we have

Lemma 1

$$P(s \rightsquigarrow t, D) = \sum_{T: \{s,t\} \subseteq T \subseteq V} H_s(V, T). \tag{11}$$

Proof. We have $\mathcal{G}_{s \rightsquigarrow t} = \cup_{T: \{s,t\} \subseteq T \subseteq V} \mathcal{G}_s(V, T)$. Further, for any $T_1 \neq T_2$, we have $\mathcal{G}_s(V, T_1) \cap \mathcal{G}_s(V, T_2) = \emptyset$. This means $\mathcal{G}_s(V, T)$ for all T such that $s, t \in T \subseteq V$ form a partition of the set $\mathcal{G}_{s \rightsquigarrow t}$. Thus,

$$\begin{aligned} P(s \rightsquigarrow t, D) &= \sum_{G \in \mathcal{G}_{s \rightsquigarrow t}} \prod_{i \in V} B_i(Pa_i^G) \\ &= \sum_{T: \{s,t\} \subseteq T \subseteq V} \sum_{G \in \mathcal{G}_s(V,T)} \prod_{i \in V} B_i(Pa_i^G) \\ &= \sum_{T: \{s,t\} \subseteq T \subseteq V} H_s(V, T). \end{aligned} \tag{12}$$

□

Now the problem is decomposed into computing $H_s(V, T)$ for all T s.t. $\{s, t\} \subseteq T \subseteq V$. We show that $H_s(S, T)$ for all T, S such that $\{s\} \subseteq T \subseteq S \subseteq V$ can be computed recursively. We immediately noticed that these $H_s(S, T)$'s can be divided into two cases: $T = \{s\}$ and $T \neq \{s\}$ (or $T - \{s\} \neq \emptyset$).

Case 1: $T = \{s\}$.

In this case, s is sink in G_S (see Figure 2). For any $S \subseteq V$, let $\mathcal{G}(S)$ denote the set of all possible DAGs over S. Then we have

$$\begin{aligned} H_s(S, \{s\}) &= [\sum_{Pa_s \subseteq S - \{s\}} B_s(Pa_s)] \\ &[\sum_{G_{S-\{s\}} \in \mathcal{G}(S-\{s\})} \prod_{i \in S - \{s\}} B_i(Pa_i^{G_{S-\{s\}}})]. \end{aligned} \tag{13}$$

For any $S \subseteq V$, define function

$$H(S) \equiv \sum_{G_S \in \mathcal{G}(S)} \prod_{i \in S} B_i(Pa_i), \tag{14}$$

and for each $i \in V$ and all $U \subseteq V - \{i\}$, define

$$A_i(U) \equiv \sum_{Pa_i \subseteq U} B_i(Pa_i). \tag{15}$$

The function A_i is known as the zeta transform of B_i which can be computed by the so-called fast zeta transform algorithm in time $O(n2^n)$ (Koivisto and Sood, 2004). Now we can write Eq. (13) as

$$H_s(S, \{s\}) = A_s(S - \{s\}) H(S - \{s\}). \tag{16}$$

Tian and He (2009) proposed a DP algorithm to sum over $\mathcal{G}(S)$ by exploiting possible sinks of DAGs and inclusion-exclusion principle. Due to Proposition 2 in (Tian and He, 2009), we have that $H(S)$ can be computed recursively by the following

$$H(S) = \sum_{k=1}^{|S|} (-1)^{k+1} \sum_{W \subseteq S, |W| = k} H(S - W) \prod_{j \in W} A_j(S - W), \tag{17}$$

with the base case $H(\emptyset) = 1$. $H(S)$ for all $S \subseteq V$ can be computed with time $O(n3^{n-1})$ and space $O(n2^n)$ (Tian and He, 2009).

Case 2: $T \neq \{s\}$ **(or** $T - \{s\} \neq \emptyset$**).**

For any $W \subseteq S$, let $\mathcal{G}_s(S, T, W)$ denote the set of DAGs in $\mathcal{G}_s(S, T)$ such that all nodes in W are (must be) sinks.[1] We first note that for any W such that

[1] W may not include all the sinks in G_S. Some nodes in $S - W$ could be sinks.

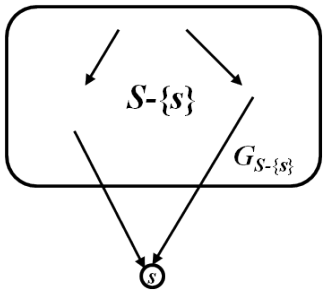
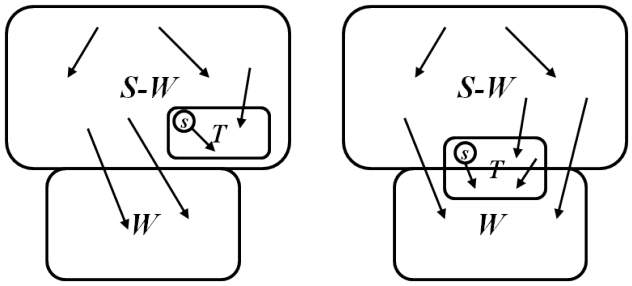


Figure 2: Case 1: $T = \{s\}$.



(a) $T \cap W = \emptyset$ (b) $T \cap W \neq \emptyset$

Figure 3: Two sub-cases when computing $F_s(S, T, W)$.

$s \in W$, $\mathcal{G}_s(S, T, W) = \emptyset$. This trivially holds because $T - \{s\} \neq \emptyset$ implies that s must have other descendants besides itself thus s cannot be a sink in G_S. Note that $\mathcal{G}_s(S, T, \emptyset) = \mathcal{G}_s(S, T)$. For any $W \subseteq S - \{s\}$, we define

$$F_s(S, T, W) \equiv \sum_{G_S \in \mathcal{G}_s(S,T,W)} \prod_{i \in S} B_i(Pa_i^{G_S}). \quad (18)$$

Since every DAG has at least one sink, we have $\mathcal{G}_s(S, T) = \cup_{j \in S-\{s\}} \mathcal{G}_s(S, T, \{j\})$. Further, it is clear that $\cap_{j \in W} \mathcal{G}_s(S, T, \{j\}) = \mathcal{G}_s(S, T, W)$. Then the summation over $\mathcal{G}_s(S, T)$ in Eq. (10) can be computed by summing over the DAGs in $\mathcal{G}_s(S, T, \{j\})$ separately and correcting the overlaps. By weighted inclusion-exclusion principle,

$$H_s(S, T) = \quad (19)$$

$$\sum_{k=1}^{|S|-1} (-1)^{k+1} \sum_{W \subseteq S-\{s\}, |W|=k} \sum_{G_S \in \mathcal{G}_s(S,T,W)} \prod_{i \in S} B_i(Pa_i^{G_S})$$

$$= \sum_{k=1}^{|S|-1} (-1)^{k+1} \sum_{W \subseteq S-\{s\}, |W|=k} F_s(S, T, W).$$

Next we show that $F_s(S, T, W)$ and $H_s(S, T)$ can be computed recursively. The central idea is to convert the sum of products in Eq. (18) to product of sums. That is, we will consider the summation over W and the summation over $S - W$ separately. Since any node in W must be a sink in G_S, it can only select parents from $S - W$. There are two sub-cases.

Sub-case 1: $T \cap W = \emptyset$.

If $T \cap W = \emptyset$, the sum of products in Eq. (18) can be freely decomposed to product of sums for nodes in W and sum over remaining nodes in $S - W$. As showed in Figure 3(a), any node in W can only select parents from $S - W - T$. For nodes in $S - W$, we have summation over $\mathcal{G}_s(S - W, T)$. Then we have

$$\begin{aligned} F_s(S, T, W) &= [\prod_{j \in W} \sum_{Pa_j \subseteq (S-T-W)} B_j(Pa_j)] \\ &[\sum_{G_{S-W} \in \mathcal{G}_s(S-W,T)} \prod_{i \in S-W} B_i(Pa_i^{G_{S-W}})] \\ &= \prod_{j \in W} A_j(S - T - W) H_s(S - W, T) \\ &= \prod_{j \in W} A_j(S - T - W) H_s(S - W, T - W) \\ &\text{(because } T - W = T \text{ in this case).} \end{aligned} \quad (20)$$

Sub-case 2: $T \cap W \neq \emptyset$.

In this case, nodes in $W - T$, $T \cap W$, and $S - W$ should be handled separately (see Figure 3(b)). Nodes in $W - T$ can only select parents from $S - W - T$. Any node in $T \cap W$ can select parents from $S - W$. In addition, at least one node from $T - W$ must be included in its parent set to guarantee that it is a descendant of s. For nodes in $S - W$, we have summation over $\mathcal{G}_s(S - W, T - W)$. Then we have

$$\begin{aligned} F_s(S, T, W) &= [\prod_{j \in T \cap W} \sum_{\substack{Pa_j \subseteq (S-W) \\ Pa_j \cap (T-W) \neq \emptyset}} B_j(Pa_j)] \quad (21) \\ &[\prod_{j \in W-T} \sum_{\substack{Pa_j \subseteq \\ (S-T-W)}} B_j(Pa_j)][\sum_{\substack{G_{S-W} \in \\ \mathcal{G}_s(S-W,T-W)}} \prod_{i \in S-W} B_i(Pa_i)] \\ &= \{\prod_{j \in T \cap W} [\sum_{Pa_j \subseteq (S-W)} B_j(Pa_j) - \sum_{\substack{Pa_j \subseteq \\ (S-W-T)}} B_j(Pa_j)]\} \\ &\prod_{j \in W-T} A_j(S - T - W) H_s(S - W, T - W) \\ &= \{\prod_{j \in T \cap W} [A_j(S - W) - A_j(S - W - T)]\} \\ &\prod_{j \in W-T} A_j(S - T - W) H_s(S - W, T - W). \end{aligned}$$

For ease of exposition, define function $\mathcal{A}_s$ as follows:

$$\mathcal{A}_s(S,T,W) \quad (22)$$
$$\equiv \begin{cases} \prod_{j\in W} A_j(S-T-W) & \text{if } T\cap W=\emptyset \\ \{\prod_{j\in T\cap W}[A_j(S-W)-A_j(S-W-T)]\} & \\ \prod_{j\in W-T} A_j(S-T-W) & \text{if } T\cap W\neq\emptyset \end{cases}$$

Now combining Sub-case 1 and 2, $F_s(S,T,W)$ can be neatly written as

$$F_s(S,T,W) = \mathcal{A}_s(S,T,W)H_s(S-W,T-W) \quad (23)$$

Plugging Eq. (23) into Eq. (19), we obtain

$$H_s(S,T) = \sum_{k=1}^{|S|-1}(-1)^{k+1}\sum_{\substack{W\subseteq S-\{s\}\\|W|=k}}\mathcal{A}_s(S,T,W)H_s(S-W,T-W) \quad (24)$$

In summary, we arrive at the following recursive scheme for computing $H_s(S,T)$ for any $\{s\}\subseteq T\subseteq S\subseteq V$.

Theorem 1

(1) For all S such that $s\in S\subseteq V$,

$$H_s(S,\{s\}) = A_s(S-\{s\})H(S-\{s\}),$$

where for all $S\subseteq V-\{s\}$,

$$H(S) = \sum_{k=1}^{|S|}(-1)^{k+1}\sum_{W\subseteq S,|W|=k}H(S-W)\prod_{j\in W}A_j(S-W),$$

with the base case $H(\emptyset)=1$.

(2) For all T,S such that $\{s\}\subset T\subseteq S\subseteq V$,

$$H_s(S,T) = \sum_{k=1}^{|S|-1}(-1)^{k+1}\sum_{\substack{W\subseteq S-\{s\}\\|W|=k}}\mathcal{A}_s(S,T,W)H_s(S-W,T-W).$$

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

We note that there are repeated computation of $\prod_{j\in W}A_j(U)$ in the phase of computing $H(S)$ and in the phase of computing $H_s(S,T)$. To facilitate the computation of $\prod_{j\in W}A_j(U)$ and $\mathcal{A}_s(S,T,W)$, we define for any $W\subseteq V$, $U\subseteq V-W$,

$$AA(U,W) \equiv \prod_{j\in W}A_j(U). \quad (25)$$

Then for a fixed U, we have

$$AA(U,W) = A_j(U)AA(U,W-\{j\}) \text{ for any } j\in W. \quad (26)$$

Thus, for a fixed U, $AA(U,W)$ for all $W\subseteq V-U$ 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 $W\subseteq V-U$ can be computed in $\sum_{|U|=0}^{n}\binom{n}{|U|}2^{n-|U|}=O(3^n)$ time. With the precomputation of $AA(U,W)$, $\mathcal{A}_s(S,T,W)$ for any T, S such that $\{s\}\subset T\subseteq S\subseteq V$ can be computed more efficiently:

If $T\cap W=\emptyset$,

$$\mathcal{A}_s(S,T,W) = \prod_{j\in W}A_j(S-T-W) = AA(S-T-W,W), \quad (27)$$

else if $T\cap W\neq\emptyset$,

$$\mathcal{A}_s(S,T,W) = \{\prod_{j\in T\cap W}[A_j(S-W)-A_j(S-W-T)]\} \prod_{j\in W-T}A_j(S-T-W) = \{\prod_{j\in T\cap W}[AA(S-W,\{j\})-AA(S-W-T,\{j\})]\} AA(S-T-W,W-T). \quad (28)$$

In summary, we have

$$\mathcal{A}_s(S,T,W) \equiv \quad (29)$$
$$\begin{cases} AA(S-T-W,W) & \text{if } T\cap W=\emptyset \\ \{\prod_{j\in T\cap W}[AA(S-W,\{j\})-AA(S-W-T,\{j\})]\} & \\ AA(S-T-W,W-T) & \text{if } T\cap W\neq\emptyset \end{cases}.$$

4.3 Overall Algorithm

Finally we summarize the results in Section 4.1 and 4.2 and outline the algorithm for computing posterior probability of any ancestor relation $s\rightsquigarrow t$.

Algorithm 1 Computing the posterior probability of an ancestor relation $s\rightsquigarrow t$.

(a) For all $i\in V$, $Pa_i\subseteq V-\{i\}$, compute $B_i(Pa_i)$. Time complexity $O(n2^{n-1})$.

(b) For all $i\in V$, $U\subseteq V-\{i\}$, compute $A_i(U)$. Time complexity $O(n2^{n-1})$.

(c) For all $U\subseteq V$, $W\subseteq V-U$, compute $AA(U,W)$. Time complexity $O(3^n)$.

(d) For all $S \subseteq V$, compute $H(S)$ in the lexicographic order of S. Time complexity $O(3^{n-1})$.

(e) For all $S \subseteq V$ s.t. $s \in S$, compute $H_s(S, \{s\})$. Time complexity $O(2^{n-1})$.

(f) For all T, S such that $\{s\} \subset T \subseteq S \subseteq V$, compute $H_s(S, T)$ 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 $H_s(S, \{s, i\})$ for each $i \in V - \{s\}$ and all S such that $\{s, i\} \subseteq S \subseteq V$ in the lexicographic order of S. Then we compute $H_s(S, \{s, i, j\})$ for each $\{i, j\} \subseteq V - \{s\}$ and all S such that $\{s, i, j\} \subseteq S \subseteq V$ in the lexicographic order of S, so on and so forth and finally we compute $H_s(V, V)$.

(g) Compute $P(s \rightsquigarrow t, D)$ by $P(s \rightsquigarrow t, D) = \sum_{T:\{s,t\}\subseteq T\subseteq V} H_s(V, T)$ and output $P(s \rightsquigarrow t|D) = P(s \rightsquigarrow t, D)/H(V)$. [2] Time complexity $O(2^{n-2})$.

It is worth mentioning that the posterior probability of any ancestor relation can only be interpreted with regard to its prior probability. This prior probability can also be computed by **Algorithm 1** with all local scores $score_i(Pa_i^G : D)$ set to 1.

4.4 Time and Space Complexity

The computing times for steps (a) to (e) and step (g) have been given already. The overall computing time is actually dominated by step (f).

For any T, S s.t. $\{s\} \subset T \subseteq S \subseteq V$, we compute $H_s(S, T)$ in $O(|S|\cdot 2^{|S|-1})$ time (any $\mathcal{A}_s(S, T, W)$ can be computed on the fly in time $O(|S|)$). Thus, all $H_s(S, T)$ can be computed in time

$$\sum_{|S|=2}^{n} \left\{ \binom{n-1}{|S|-1} \left[\sum_{|T|=2}^{|S|} \binom{|S|-1}{|T|-1} |S|\cdot 2^{|S|-1} \right] \right\}$$
$$= \sum_{|S|=2}^{n} \left\{ \binom{n-1}{|S|-1} \left[|S|\cdot 2^{|S|-1} \sum_{|T|=2}^{|S|} \binom{|S|-1}{|T|-1} \right] \right\}$$
$$= \sum_{|S|=2}^{n} \left[\binom{n-1}{|S|-1} |S|\cdot 2^{|S|-1} \cdot 2^{|S|-1} \right]$$
$$= \sum_{|S|=2}^{n} \left[\binom{n-1}{|S|-1} |S|\cdot 4^{|S|-1} \right] = n5^{n-1}.$$

Thus, the total computation time is $O(n5^{n-1} + 3^n + n2^{n-1}) = O(n5^{n-1})$.

To compute the posterior probabilities for all node pairs s, t, it suffices to repeat the computation step (e) and step (f) for each $s \in V$, and for a given s to repeat step (g) for each t. Thus, the total time for computing all possible ancestor relations $s \rightsquigarrow t$ is $O(n^2 5^{n-1})$.

$B_i(Pa_i)$ for all $i \in V$, $Pa_i \subseteq V - \{i\}$ take $O(n2^{n-1})$ space. $A_j(U)$ for all $j \in V$, $U \subseteq V - \{j\}$ take $O(n2^{n-1})$ space. $AA(U, W)$ for all $U \subseteq V$, $W \subseteq V - U$ consume $\sum_{|U|=0}^{n} \binom{n}{|U|} 2^{n-|U|} = O(3^n)$ space. $H(S)$ for all $S \subseteq V$ take $O(2^n)$ space. $H_s(S, T)$ for all $\{s\} \subseteq T \subseteq S \subseteq V$ consume $\sum_{|S|=1}^{n} \binom{n-1}{|S|-1} 2^{|S|-1} = O(3^{n-1})$ space. Since each step in **Algorithm 1** relies only on the previous step. We need only store relevant scores in the memory. That is, after we compute all $A_i(U)$ scores, we can immediately delete all $B_i(Pa_i)$ scores; after computing all $AA(U, W)$ scores, we delete $A_i(U)$ scores. When computing step (f), we need only keep $AA(U, W)$ and $H_s(S, T)$ scores in memory. In such way, we can use the memory more efficiently. The space requirement is therefore $O(3^n + n2^n)$.

In summary, we have the following theorem.

Theorem 2 *The posterior probability for any ancestor relation $s \rightsquigarrow t$ can be computed in $O(n5^{n-1})$ time and $O(3^n + n2^n)$ space. The posterior probabilities for all $n(n-1)$ possible ancestor relations can be computed in $O(n^2 5^{n-1})$ time and $O(3^n + n2^n)$ space.*

4.5 Exact Learning of $s \rightsquigarrow p \rightsquigarrow t$ Relations

It turns out that the techniques for computing $s \rightsquigarrow t$ relations can be extended to compute the posteriors of $s \rightsquigarrow p \rightsquigarrow t$ relations, i.e., a directed path from s to t via p. For example, biologists are interested in whether the influence of a gene on a downstream gene is regulated by some intermediate gene or factor. Learning this type of structural features is therefore of great interests. Due to the space limit, we only present our conclusion here in **Theorem 3**. The algorithm and proofs are included in the supplemental material.

Theorem 3 *The posterior probability of any $s \rightsquigarrow p \rightsquigarrow t$ relation can be computed in $O(n7^{n-2})$ time and $O(4^{n-2} + 3^n)$ space.*

5 EXPERIMENTS

We have implemented **Algorithm 1** in C++. We used BDe score for $score_i(Pa_i : D)$ with equivalent sample size being 1 (Heckerman et al., 1995). We applied uniform structure prior $P(G)$ by setting all $Q_i(Pa_i)$'s to be 1.[3] All experiments were done on a Linux desktop PC with 3.33 GHz Intel Core2 Duo CPU and 4 GB memory.

[2] It has been shown by Tian and He (2009) that $P(D) = H(V)$.

[3] Note that a constant $P(G)$ could be canceled out in computing $P(f|D) = P(f, D)/P(D)$ (see Eq. (6)).

5.1 Running Times

We first examine the running times of our algorithm on several data sets from the UCI Machine Learning Repository. The results are presented in Table 1, where n is the number of variables, m is the sample size of each data set, $T(B)$ records the time for computing all $B_i(Pa_i)$'s, i.e., the local scores, and $T(total)$ is the total time for evaluating all $n(n-1)$ ancestor relations. We clearly see that the running times are reflective of the exponential dependence on n with a base around 5. This is consistent with **Theorem 2**.

Table 1: Running Times (in seconds)

Data Sets	n	m	$T(B)$	$T(total)$
Weather	5	14	3e-4	7e-4
Iris	5	150	6e-4	6e-4
Asia	8	500	0.02	0.2
Tic-Tac-Toe	10	958	0.6	6.5
CYTO	11	5400	8.4	32
Wine-11	11	178	0.8	76
Wine-12	12	178	1.8	411
Wine-13	13	178	4.6	2331
Wine	14	178	11.6	12856

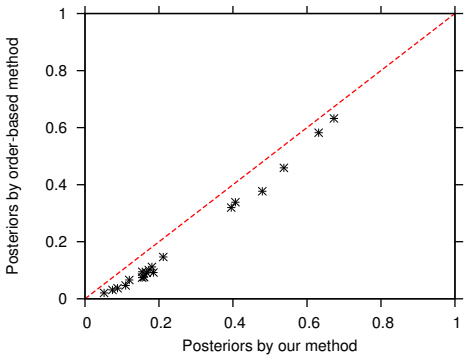
5.2 Comparison of Posteriors

The order-based approach by Parviainen and Koivisto (2011) and our approach differ in the structure prior $P(G)$ assigned to DAGs. The order-based approach places a non-uniform prior over DAGs, favoring DAGs that are consistent with more linear orders, while our approach adheres to the uniform prior. Here we compare the posteriors computed by the two approaches on four different data sets in Figure 4.

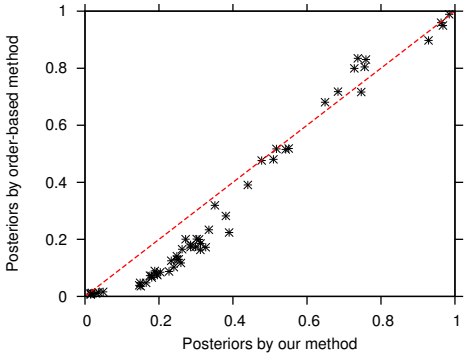
We can see that the posteriors computed by the two approaches differ in most of the cases, demonstrating the non-negligible effect of priors on the computation results. Further, we observed that the order-based approach often underestimates the posteriors (see Figure 4(a)(c)). This can be understood by noticing that DAGs consistent with more linear orders usually have simpler structures, for example, fewer edges or fewer directed paths than those DAGs consistent with fewer linear orders. Since DAGs consistent with fewer linear orders receive less weights in the order-based approach, the ancestor relations implied by these DAGs are undercounted.

5.3 Knowledge Discovery

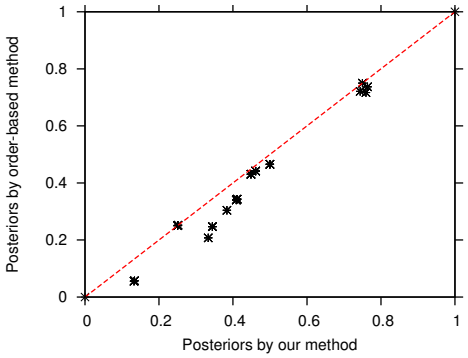
Finally, we applied our algorithm to a biological data set (CYTO) which consists of flow cytometry measurements of $n = 11$ phosphorylated proteins and phospholipids under 7 different interventions and 2 unperturbed conditions. 600 measurements are taken in each condition yielding a total dataset of $m = 5400$ samples. The data have been discretized into 3 states,



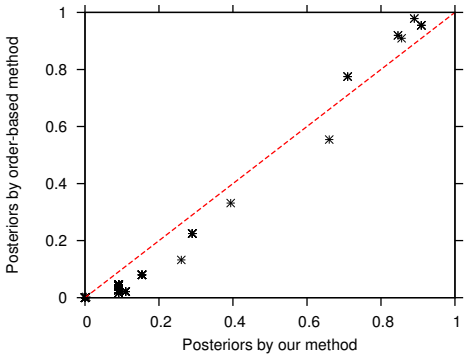
(a) Weather ($n = 5$, $m = 14$)



(b) Asia ($n = 8$, $m = 500$)



(c) Tic ($n = 10$, $m = 958$)



(d) Wine ($n = 12$, $m = 178$)

Figure 4: Scatter plots that compare posteriors of ancestor relations computed by our algorithm and by order-based algorithm.

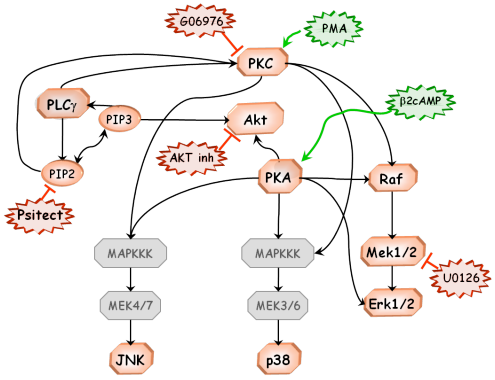


Figure 5: Classical model of the CYTO data set. Modified from Sachs et al (2005). The proteins of interest are in highlighted red rectangles, i.e., PKC, PLCγ, PIP2, PIP3, Akt, PKA, JNK, p38, Raf, Mek, Erk. Ovals with serrated edges represent various interventions (green=activators, red=inhibitors).

Table 2: Ancestor relations learned for CYTO data set

Source	Sinks
Raf	**Mek**, PLCγ, PIP2, PIP3, **Erk**, Akt, PKA, p38, JNK
Mek	PLCγ, PIP2, PIP3, **Erk**, Akt, PKA, p38, JNK
PLCγ	**PIP2**, **PIP3**
PIP2	**PIP3**
Erk	Akt
PKA	PLCγ, PIP2, PIP3, **Erk**, **Akt**, **p38**, **JNK**
PKC	**Raf**, **Mek**, PLCγ, PIP2, PIP3, **Erk**, Akt, PKA, **p38**, **JNK**
JNK	PLCγ, PIP2, PIP3, p38

representing low, medium and high activity according to Sachs et al (2005). Figure 5 shows a currently accepted consensus network.

We then used our algorithm to compute the posteriors of all 110 possible ancestor relations. We modified the local likelihood scores $B_i(Pa_i)$ to take into account the interventional nature of the data as in (Tian and Pearl, 2001). Our results shows that among all 110 possible ancestor relations, 42 ancestor relations have posteriors greater than 0.95, while all other ancestor relations have posteriors less than 0.03.[4] Table 2 tabulates the 42 most probable ancestor relations. Proteins are made bold if the corresponding ancestor relations also exist in the classical model. The learning results exhibit a high false positive rate if the classical model is really the true model. This may be because that the ancestor relation learning is very sensitive to the local errors. For example, one flipped edge can lead to a large number of ancestor relation errors. However, the classical model is not necessarily the true model. As our results showed, we had high certainty on the presence or absence of each of the ancestor relations. Thus, it is possible that these ancestor relations suggest potential protein signaling pathways that have yet to be discovered.

We also compared our direct learning of ancestor relation to the deduction of (important) ancestor relations from the edge posteriors. We used the algorithm by Tian and He (2009), which requires $O(n3^n)$ time and $O(n2^n)$ space, to compute the posteriors of all 110 possible *edges*. We then constructed a network that consisted of edges whose posteriors were greater than 0.5 and inferred the ancestor relations from this network. We observed that the set of (important) ancestor relations deduced from the most likely edges were exactly the same as those predicted by the direct learning. This suggests that the two approaches do not differ significantly in predicting the most significant ancestor relations, at least on this CYTO data set. More systematic evaluation will be needed to confirm this observation. Moreover, we observed that computing all edge posteriors took about 9 seconds, much faster than computing the posteriors of ancestor relations (32 seconds, see Table 1). However, direct learning of ancestor relation outputs the ancestor posterior probabilities while the network constructed from most likely edges does not provide such information.

6 SUMMARY

In this paper, we have developed a new DP algorithm to compute the exact posteriors of all possible ancestor relations in Bayesian networks. Compared to previous order-based algorithm, our algorithm respects the uniform structure prior and the Markov equivalence. Experimental comparison showed the order-based approach tends to underestimate the posteriors. We have also applied our algorithm to a biological data set to discover signaling pathways. This demonstrated our algorithm in the task of knowledge discovery. Further, we have developed an algorithm to compute the exact posterior of any $s \rightsquigarrow p \rightsquigarrow t$ relation, i.e., a directed path from s to t via p.

One major limitation of the exact algorithms proposed here (and in previous work) is their exponential complexities, which prevent their practical use for large networks. To circumvent the limitation, approximate methods such as MCMC sampling are commonly used. One potential application of the exact algorithms given in this paper is to assess the approximate quality of approaches such as MCMC sampling.

Acknowledgements

We thank the anonymous reviewers for valuable comments.

[4]Note that the prior of an ancestor relation is 0.45 for $n = 11$ with the uniform structure prior.

References

[1] 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.

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

[3] 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.

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

[5] 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.

[6] 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.

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

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

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

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

[11] 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.

[12] 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.

[13] 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.

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

[15] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005.

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

[17] 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.

[18] Jin Tian and Judea Pearl. Causal discovery from changes. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 512–521. Morgan Kaufmann Publishers Inc., 2001.

[19] 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.

[20] 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.