Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Finding the k-Best Equivalence Classes of Bayesian Network Structures for Model Averaging

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

Abstract

In this paper we develop an algorithm to find the k-best equivalence classes of Bayesian networks. Our algorithm is capable of finding much more best DAGs than the previous algorithm that directly finds the k-best DAGs (Tian, He, and Ram 2010). We demonstrate our algorithm in the task of Bayesian model averaging. Empirical results show that our algorithm significantly outperforms the k-best DAG algorithm in both time and space to achieve the same quality of approximation. Our algorithm goes beyond the maximum-a-posteriori (MAP) model by listing the most likely network structures and their relative likelihood and therefore has important applications in causal structure discovery.

Introduction

Directed graphical models, i.e., Bayesian networks (BN), have been widely used in various tasks for probabilistic inference and causal modeling (Pearl 2000; Spirtes, Glymour, and Scheines 2000). One major challenge in these tasks is to learn the structure of the model from data. Model selection approach seeks out a BN structure G that maximizes certain score metric, e.g., the posterior probability $P(G|D)$ given observed data D, and subsequent inference proceeds conditionally on this single model. This maximum-a-posteriori (MAP) structure, however, may provide inadequate summary in cases where a large number of distinct DAGs are equally probable. This often happens in domains where the amount of data is small relative to the size of the model (Friedman and Koller 2003). In this situation, if what we are interested in is learning model structure or causal relationships between variables, the MAP model may give unwarranted conclusions.

Bayesian model averaging (BMA) provides a principled solution to model-uncertainty problem by integrating all possible models weighted by their respective posterior probabilities. Formally, for any hypothesis of interest h, we compute the posterior probability of h as $P(h|D) = \sum_G P(h|G,D)P(G|D)$. Then we can draw conclusions on h based on $P(h|D)$. For example, if we are interested in some structural feature f (e.g., $f(G) = 1$ if such feature exists in DAG G, $f(G) = 0$ otherwise), we compute posterior probability of f as $P(f|D) = \sum_G f(G)P(G|D)$.

However, the exact computation of these posteriors requires summation over all possible DAGs, the number of which is super-exponential $O(n!\,2^{n(n-1)/2})$ with respect to the number of nodes n. Thus, full Bayesian model averaging is often intractable in practice. Although the recently developed dynamic programming (DP) aglorithms successfully reduced the computation to exponential time and space, they have certain limitations. For example, the algorithms in (Koivisto and Sood 2004; Koivisto 2006; Tian and He 2009) can estimate modular features such as arcs in exponential time and space, but fail in cases where non-modular features such as paths (ancestor relations) are concerned. To deal with non-modular features such as ancestor relations, the fastest algorithm takes $O(n3^n)$ time (Parviainen and Koivisto 2011).

Instead of doing full Bayesian model averaging, there are several approaches developed to approximate the exhaustive enumeration. The central idea is to select a representative set of DAGs $\mathcal{G}$, and estimate the posterior as $P(h|D) \approx \sum_{G\in\mathcal{G}} P(h|G,D)P(G|D)/\sum_{G\in\mathcal{G}} P(G|D)$. Among these approaches are a group of methods based on Markov Chain Monte Carlo (MCMC) technique, which provides a principled way to sample DAGs from their posterior distribution $P(G|D)$ (Madigan, York, and Allard 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). However, MCMC-based methods suffer from the problem of no guarantee on the approximation quality in finite runs (the Markov chains may not mix and converge in finite runs).

Another approach proposes to construct $\mathcal{G}$ with a set of high-scoring DAGs. In particular, (Tian, He, and Ram 2010) studied the idea of using the k-best DAGs for model averaging. The estimation accuracy could be monotonically improved by spending more time to compute for larger k, and the model averaging over these k-best models achieved good accuracy in structural discovery. As they showed experimentally, one main advantage of constructing k-best models over sampling is that MCMC method exhibited a non-negligible variability across different runs because of the randomness nature of MCMC, while the k-best method always gave consistent estimation due to its deterministic nature.

One issue with the k-best DAG algorithm (we will call it *kBestDAG*) is that the best DAGs found actually coalesce into a fraction k of Markov equivalence classes, where the DAGs within each class represent the same set of conditional independence assertions and determine the same statistical model. It is therefore desirable if we are able to directly find the k-best equivalence classes of Bayesian networks.

Searching in the equivalence class (EC) space has been studied in (Madigan et al. 1996; Chickering 2002a; 2002b; Castelo and Kocka 2003). The potential advantages of using the EC space instead of DAG space include: (1) The cardinality of EC space is smaller than DAG space; (2) Searching in the EC space improves the efficiency of search because moves within the same EC can be avoided. The first advantage does not alleviate substantially the learning complexity as showed in (Gillispie and Perlman 2001) that the ratio of the number of DAGs to the number of equivalence classes reaches an asymptote around 3.7 with as few as ten nodes. Searching in the EC space may also suffer from overhead due to compulsory additional operations, e,g., converting DAGs to its equivalence class partial DAG representation and vice versa (Chickering 2002b).

In this paper, we developed an algorithm called *kBestEC* to find the k-best Markov equivalence classes of Bayesian nework structures. *kBestEC* has slightly greater time complexity than *kBestDAG*, but has the same space complexity as *kBestDAG*. Since space complexity is the bottleneck of the algorithms with current computer, *kBestEC* is capable of finding much more best DAGs than *kBestDAG*. We tested *kBestEC* on the task of BMA to compute the posterior probabilities of edge structure features on several data sets from the UCI Machine Learning Repository as well as synthetic data sets. Our experiments showed that *kBestEC* significantly outperformed the *kBestDAG* algorithm in both time and space usages to achieve the same quality of approximation. Thus, in the problem of searching for the k-best DAGs, benefits from working in EC space significantly outweigh the overhead.

Our algorithm provides a useful tool for reseachers interested in learning model structures or discovering causal structures. For example, biologists are interested in recovering gene regulation networks from data. Recovering the MAP network alone often does not give the full picture. There may exist a number of equally probable DAGs (or equivalence classes) with distinct structures when the amount of data is small relative to the size of the model. For example, in our experiments (shown in Table 1, and Figure S2 in supplemental materials (Chen and Tian 2014)), there exist 10 equivalence classes having the same best score for Tic dataset; the MAP network of Vote dataset is just 2.4 times more likely than the 10th-best equivalence class. Our algorithm should be a very useful tool for understanding model structures in these situations by listing the most likely models and their relative likelihood.

Preliminaries

Formally, a Bayesian network is a DAG that encodes a joint probability distribution over a set of random variables $V = \{v_1, ..., v_n\}$ with each node of the graph representing a variable in V. In this paper we will use nodes and variables interchangeably. And we let $Pa_v^G \subseteq V \setminus \{v\}$ denote the parent set of any node v in G. In the problem of learning BNs from a data set D, which consists of complete $i.i.d$ samples from the joint distribution $P(V)$, we seek a BN that best explains the data D, evaluated by some scoring function, e.g., $\ln P(G, D)$. In this paper, we assume decomposable score such that

$$score(G : D) = \sum\nolimits_{v \in V} score_v(Pa_v^G : D), \qquad (1)$$

where $score(G : D)$ will be written as $score(G)$ for short in the following discussion.

Now we give a few of definitions and theorems that describe the semantics and properties of BNs. .

Definition 1 (Verma and Pearl 1990) *Two DAGs are equivalent if they represent the same set of conditional independence assertions among the set of variables.*

Definition 2 *A v-structure in a DAG G is an ordered triple of nodes (u, v, w) such that G contains the directed edges $u \to v$ and $w \to v$ and u and w are not adjacent in G.*

Theorem 1 (Verma and Pearl 1990) *Two DAGs G and G' are equivalent if and only if they have the same skeletons and the same v-structures.*

Definition 3 [*Score equivalence*]
Let $score(G)$ be some scoring function that is decomposable. We say that it satisfies score equivalence if for all equivalent DAGs G and G' we have $score(G) = score(G')$ for any data set D.

Score equivalence is the nature of several common scoring functions such as MDL, BDe and BIC. As a result, the set of equivalent DAGs are indistinguishable by these scoring functions. Thus, our goal is to find "a best", instead of "the best". However, finding a best BN is NP-hard (Chickering, Geiger, and Heckerman 1995). Recently, a family of DP algorithms have been developed to find the optimal BN in time $O(n2^n)$ and space $O(2^n)$ (Singh and Moore 2005; Silander and Myllymäki 2006). The central idea exploits the fact that a DAG must have a sink s. Considering any $s \in V$, the best DAG over V with s as sink can be constructed by piecing together the best DAG $G^*_{V\setminus\{s\}}$ over $V \setminus \{s\}$ and the best parent set $Pa^*_s \subseteq V \setminus \{s\}$ assuming $G^*_{V\setminus\{s\}}$ is already known. Then we choose the best sink s that optimizes this construction. Applying this idea to all $W \subseteq V$ results in a DP algorithm that finds the best DAG for all 2^n possible W recursively.

Later, (Tian, He, and Ram 2010) generalized the algorithm to recursively find the k-best DAGs and proposed to make inference by averaging over these DAGs. Instead of considering a single best DAG, their algorithm maintains a list of k-best DAGs for each node set $W \subseteq V$. However, these k-best DAGs are redundant in the sense that they coalesce into only a fraction k of equivalence classes and from one DAG we can efficiently infer other members in its class. Thus, it is desirable if we are able to directly find the k-best

equivalence classes. In next section, we present such an algorithm.

Finding the k-best Equivalence Classes of Bayesian Networks

The following definitions will be useful in the development of our algorithm.

Definition 4 [*Score for sub-graph* $G_W, W \subseteq V$]
For any decomposable score, define $score(G_W) = \sum_{v \in W} score_v(Pa_v^{G_W})$ *for any DAG* G_W *over any node set* $W \subseteq V$*, where* $Pa_v^{G_W}$ *is the parent set of* v *in* G_W.

Definition 5 [*Graph growth operator* $\oplus$]
For any G_W, $v \in V \setminus W$, $Pa_v \subseteq W$, *define* $G_{W \cup \{v\}} = G_W \oplus Pa_v$ *as an operation growing* G_W *to* $G_{W \cup \{v\}}$ *s.t.* $G_{W \cup \{v\}}$ *contains all edges in* G_W *and the directed edges from* Pa_v *to* v.

Proposition 1 *For any decomposable score function that satisfies score equivalence, we have* $score(G_W) = score(G'_W)$ *if* G_W *and* G'_W *are equivalent over node set* $W \subseteq V$.

The proof of **Proposition 1** is given in supplemental materials (Chen and Tian 2014). **Proposition 1** says the score equivalence actually holds for DAGs over any subset $W \subseteq V$. This property allows us to recursively construct top equivalence classes over all $W \subseteq V$.

Now we outline the algorithm for finding the k-best equivalence classes given in **Algorithm 1**. It has three logical steps:

1. Compute the family scores $Score_v(Pa_v)$ for all $n2^{n-1}$ (v, Pa_v) pairs (lines 1–3);
2. Find the k-best parent sets in candidate set C for all $C \subseteq V \setminus \{v\}$ for all $v \in V$ (lines 4–6);
3. Recursively find the k-best equivalence classes over all node sets $W \subseteq V$ (in lexicographic order) (lines 7–13).

The first two steps follow naturally from those steps in (Silander and Myllymäki 2006) and (Tian, He, and Ram 2010) and we will use their algorithms.

We will use the idea of DP to find the k-best equivalence classes recursively for all $W \subseteq V$, while the *kBestDAG* algorithm in (Tian, He, and Ram 2010) finds the k-best DAGs recursively. However working in the equivalence class space requires more careful treatment. It is not immediately clear that the idea of exploiting sink will work in the equivalence class space.

For a node set $W \subseteq V$, let EC_W^i, $i \in \{1, ..., k\}$ denote the top k equivalence classes over W. For each EC_W^i, we use a DAG over W, denoted as G_W^i, to represent the whole equivalence class.[1] For each $W \subseteq V$, we keep track of k DAGs, $G_W^1, ..., G_W^k$, each of them comes from one of the top k equivalence classes. Now assume we have identified such k-best ECs $G_{W\setminus\{s\}}^1, ..., G_{W\setminus\{s\}}^k$ for all $s \in W$. Finding the k-best ECs $G_W^1, ..., G_W^k$ for W takes two sub-steps:

3a. For each $s \in W$, identify the k-best ECs $G_{W,s}^1, ..., G_{W,s}^k$ over W with s as a sink (line 10 in **Algorithm 1**).

3b. Let $G_W^1, ..., G_W^k$ be the k-best nonequivalent DAGs among $\cup_{s \in W}\{k$-best ECs $G_{W,s}^1, ..., G_{W,s}^k$ over W with s as a sink$\}$ (line 11 in **Algorithm 1**).

In 3a, to find the k-best ECs $G_{W,s}^1, ..., G_{W,s}^k$, let $bestPa_s(C, j)$ denote the j-th best parent set for node s in the set of candidate parents C. Define function $value_{W,s}(i, j)$ by

$$value_{W,s}(i,j) = score(G^i_{W\setminus\{s\}}) + score_s(bestPa_s(W \setminus \{s\}, j)). \quad (2)$$

Algorithm 1 Finding the k-best Equivalence Classes

```
1: for all v ∈ V do
2:     Compute score_v(Pa_v) for all Pa_v ⊆ V \ {v}.
3: end for
4: for all v ∈ V do
5:     Find the k-best parent sets {bestPa_v(C, i),
       i = 1, ..., k} in parent candidate set C for all
       C ⊆ V \ {v} recursively.
6: end for
7: for all W ⊆ V in lexicographic order do
8:     A priority queue bestDAGs(W) with size limit of
       k, initialized to ∅. The elements in bestDAGs(W)
       is denoted by G^i_W, i ∈ {1, ..., k}.
9:     for all s ∈ W do
10:        Find the k-best G^1_{W,s}, ..., G^k_{W,s} with s as a sink
           from {G^i_{W\{s}} ⊕ bestPa_s(W \ {s}, j) :
           i = 1, ..., k, j = 1, ..., k}.
11:        For all i ∈ {1, ..., k}, insert G^i_{W,s} into queue
           bestDAGs(W) if
           score(G^i_{W,s}) > min{score(G^i_W), i = 1, ..., k}
           and G^i_{W,s} is not equivalent to any DAG in
           bestDAGs(W).
12:    end for
13: end for
14: return bestDAGs(V)
```

We can find the k-best scores among $\{value_{W,s}(i,j) : i, j \in \{1, ..., k\}\}$ by performing a best-first search with root node $(1, 1)$ and children of (i, j) being $(i+1, j)$ and $(i, j+1)$, as suggested in (Tian, He, and Ram 2010). Let $G_{W,s}^1, ..., G_{W,s}^k$ denote the k DAGs $G^i_{W\setminus\{s\}} \oplus bestPa_s(W \setminus \{s\}, j)$ corresponding to the k-best scores. Now do they represent the k-best ECs? In other words, can some of these DAGs be equivalent to each other, or are there other DAGs having better scores than these DAGs? One concern is that in

[1] An alternative way to represent a EC is called completed partially DAG (CPDAG), consisting of a directed edge for every compelled edge and an undirected edge for every reversible edge in the EC (Chickering 2002a). We choose DAG over CPDAG because: (1) encoding a DAG is space more efficient than encoding a CPDAG, which makes significant difference when we have to keep $k2^n$ networks in memory; (2) growing a DAG using $\oplus$ results in a valid DAG while growing CPDAG using $\oplus$ results in a PDAG which need be converted to a CPDAG with extra effort.

constructing $G^1_{W,s}, ..., G^k_{W,s}$ we only use one representative DAG $G^i_{W\setminus\{s\}}$ from its corresponding EC. Is it safe to ignore other DAGs equivalent to $G^i_{W\setminus\{s\}}$? The following theorem guarantees that $G^1_{W,s}, ..., G^k_{W,s}$ indeed represent the k-best ECs.

Theorem 2 *The k DAGs corresponding to the k-best scores output by the best-first search represent the k-best ECs over W with s as a sink.*

Proof. We first prove that these k DAGs are mutually nonequivalent. For any $G^p_{W,s}$ and $G^q_{W,s}$, $p, q \in \{1, ..., k\}$, $p \neq q$, we have two different cases.

Case 1: $G^p_{W,s}$ and $G^q_{W,s}$ are constructed from $G^i_{W\setminus\{s\}}$ and $G^j_{W\setminus\{s\}}$ respectively. $G^1_{W\setminus\{s\}}$,..., $G^k_{W\setminus\{s\}}$ over $W\setminus\{s\}$ are nonequivalent. This implies that any two of them, say, $G^i_{W\setminus\{s\}}$ and $G^j_{W\setminus\{s\}}$, are either have different skeletons or have the same skeleton but different v-structures. Since adding parents Pa_s for s changes neither the skeleton nor any v-structures in $G^i_{W\setminus\{s\}}$ and $G^j_{W\setminus\{s\}}$, $G^p_{W,s}$ and $G^q_{W,s}$ must either have different skeletons or have the same skeleton but different v-structures. Therefore, $G^p_{W,s}$ and $G^q_{W,s}$ are not equivalent.

Case 2: $G^p_{W,s}$ and $G^q_{W,s}$ are constructed from the same $G^i_{W\setminus\{s\}}$ but with different parent sets for s. Since two different parent sets for s have different nodes, the sets of edges respectively added to $G^i_{W\setminus\{s\}}$ to construct $G^p_{W,s}$ and $G^q_{W,s}$ are different. As a result, $G^p_{W,s}$ and $G^q_{W,s}$ have different skeletons. Therefore, they are not equivalent.

Now we prove that the output $G^1_{W,s}, ..., G^k_{W,s}$ are the k-best over W with s as a sink. All we have to show is that for each equivalence class $EC^i_{W\setminus\{s\}}$, it is safe to keep just one DAG $G^i_{W\setminus\{s\}}$ while discarding others. That is, using another member $G'^i_{W\setminus\{s\}}$, we are unable to construct a DAG $G'_{W,s} = G'^i_{W\setminus\{s\}} \oplus Pa_s$ such that $score(G'_{W,s}) > score(G^k_{W,s})$ and it is nonequivalent to any of $G^1_{W,s}, ..., G^k_{W,s}$. Assume we can construct such $G'_{W,s}$, then we can construct an equivalent DAG by $G_{W,s} = G^i_{W\setminus\{s\}} \oplus Pa_s$. By **Proposition 1**, $score(G_{W,s}) = score(G'_{W,s}) > score(G^k_{W,s})$. Best-first search guarantees that this $G_{W,s}$ is in the list of $G^1_{W,s}, ..., G^k_{W,s}$. This contradicts the assumption that $G'_{W,s}$ is nonequivalent to any of $G^1_{W,s}, ..., G^k_{W,s}$.
Thus, **Theorem 2** holds. □

After 3a, we have the k-best ECs over W with s as a sink for each $s \in W$. [2] In 3b, we identify $G^1_W, ..., G^k_W$ as the k-best DAGs from $\cup_{s\in W}\{k$-best $G^1_{W,s}, ..., G^k_{W,s}$ over W with s as a sink$\}$ that are mutually nonequivalent. For this purpose, we need explicitly check the equivalence of two DAGs if they are constructed from distinct sink s, s'. We first compare the scores. If the scores are not equal, two DAGs are nonequivalent. Otherwise, we need check whether they are equivalent. The algorithm for checking the equivalence of two DAGs is ommited here, yet included in the supplemental materials.

Theorem 3 *The k DAGs $G^1_V, ..., G^k_V$ output by Algorithm 1 represent the k-best ECs over V.*

Proof. For each $W \subseteq V$, $\{G_W$ over $W\} = \cup_{s\in W}\{G_W$ over W with s as a sink$\}$, therefore the k-best nonequivalent DAGs over W are the k-best among $\cup_{s\in W}\{k$-best ECs $G^1_{W,s}, ..., G^k_{W,s}$ over W with s as a sink$\}$. Thus, for each $W \subseteq V$, $G^1_W, ..., G^k_W$ obtained from Step 3b represent the k-best ECs over W. By induction, $G^1_V, ..., G^k_V$ output from Algorithm 1 represent the k-best ECs over V. □

Algorithm 2 $EnumEquivalentDAGs(G_V)$

```
list ← {G_V}
REV ← FindReversibleEdges(G_V)
for each subset CE ⊆ REV do
    Construct a new G'_V by reversing edges CE in G_V
    if CHECKACYCLICITY(G'_V)= true then
        flag ←true
        for each v participating in some edge of CE do
            if CheckVStruc(v, G_V, G'_V) = false then
                flag ←false and break
            end if
        end for
        if flag = true then
            list.add(G'_V)
        end if
    end if
end for
return list
```

Now we give a theoretical discussion on the run-time and space complexity of the algorithm. Step 1 takes $O(n2^{n-1})$ time and $O(2^{n-1})$ space. Step 2 takes $O(k \log k(n-1)2^{n-2})$ time and $O(k2^{n-1})$ space in the worst case. Doing a best-frst search to find the k-best elements from space $\{(i,j) : i,j \in \{1, ..., k\}\}$ takes $O(k \log k)$ time. Checking the equivalence of two DAGs has a worst-case run-time of $O(|W|d^2_W)$, where d_W is the maximum size of parents in G_W and G'_W. Thus, the worst-case run-time for step 3 is $\sum^n_{|W|=1}\binom{n}{|W|}|W|(k \log k + k|W|d^2_W) = O(n2^{n-1}k(\log k + \frac{nd^2}{2}))$, where d is the maximum size of the parents computed in Step 2. [3] The worst space complexity is $O(k2^n)$ since we have to memorize no more than k DAGs for each $W \subseteq V$. [4]

For the same k, step 3 for finding the k-best ECs is $\frac{\log k + nd^2/2}{\log k + nd/2}$ times slower in the worst case than step 3 in

[2] There may be less than k such DAGs for W when $|W|$ is small, but the number reaches k very rapidly as W grows.

[3] Checking whether two DAGs G_W and G'_W are the same has a run-time of $O(|W|d_W)$. Therefore the run-time for *kBestDAG* algorithm is $O(n2^{n-1}k(\log k + \frac{nd}{2}))$.

[4] We say worst space because for small W's, there may exist less than k equivalence classes.

kBestDAG for finding the k-best DAGs. Thus, *kBestEC* has slightly larger time complexity than *kBestDAG*. Both algorithms have the same space requirement.

Bayesian Model Averaging Using the k-best Equivalence Classes

We have presented an algorithm to obtain the k-best DAGs $G_V^1, ..., G_V^k$ representing the k-best equivalence classes $EC_V^1, ..., EC_V^k$. One application of our algorithm is to compute the posterior of hypothesis of interests with BMA. If the application is to evaluate class-invariant structural features such as Markov blanket or to predict new observations, the problem can generally be formulated as computing the posterior of the hypothesis h by

$$\hat{P}(h|D) = \frac{\sum_{i=1}^{k} w_i P(h|G_V^i, D) P(G_V^i, D)}{\sum_{i=1}^{k} w_i P(G_V^i, D)}, \quad (3)$$

where w_i is a weight we assign to each equivalence class EC_V^i. For example, if we want to treat each equivalence class as a single statistical model (Madigan et al. 1996; Castelo and Kocka 2003), we simply set $w_i = 1$. If we'd like model averaging over original DAG space, we set $w_i = |EC_V^i|$, i.e, the number of DAGs in equivalence class EC_V^i.

If the application is to evaluate structural features such as an arrow $u \to v$ or a path $u \rightsquigarrow v$ that is not necessarily class-invariant, we have to enumerate the DAGs in each equivalence class in order to compute the posterior

$$\hat{P}(h|D) = \frac{\sum_{i=1}^{k} P(G_V^i, D) \sum_{G \in EC_V^i} P(h|G, D)}{\sum_{i=1}^{k} |EC_V^i| P(G_V^i, D)}. \quad (4)$$

Algorithm 2 sketches an algorithm to enumerate all DAGs in an equivalence class and to compute $|EC_V^i|$ in the mean time. Given a DAG G_V, we first determine the set of reversible edges, i.e., their directions vary among the equivalent DAGs (line 2). (Chickering 1995) provided a $O(|E_{G_V}|)$ algorithm to find all compelled edges, i.e, their directions are invariant among the DAGs in an EC. We slightly modified this algorithm so that it outputs the set of reversible edges REV in G_V. All possible DAGs equivalent to G_V can be enumerated by reversing all possible edge combinations in REV. If the generated "DAG" passes the test of acyclicity and v-structures, it is a DAG equivalent to G_V. The overall algorithm takes $O((|V|+|E_{G_V}|+|E_{G_V}|^2)2^{|REV|})$ in the worst case. Note here we implemented a straightforward algorithm for enumerating all DAGs in an EC. Its run-time is negligible compared with the time for finding the k-best ECs due to the fact that the number of DAGs in an EC is pretty small.

Experiments

We implemented **Algorithm 1** in C++ [5]. To see how it performs, we consider the problem of computing the posteriors for all $n(n-1)$ possible directed edges using Eq. (4) by enumerating all DAGs in each EC. We used BDe score (Heckerman and Chickering 1995) for $score_i(Pa_i)$ with a uniform structure prior $P(G)$ and equivalent sample size 1. We compare the performances of our *kBestEC* algorithm with the *kBestDAG* algorithm, in terms of run-time, memory usage and quality of approximation. For approximation quality, we define cumulative posterior probability density of the set of DAGs in $\mathcal{G}$ used to perform model averaging by

$$\Delta = \sum\nolimits_{G \in \mathcal{G}} P(G|D) = \frac{\sum_{G \in \mathcal{G}} P(G, D)}{P(D)}. \quad (5)$$

We used the algorithm in (Tian and He 2009) to compute the exact $P(D)$ value. Note that $\Delta \leq 1$ and the larger of Δ, the closer of the estimation to the full Bayesian model averaging. In practice, it is often reasonable to make predictions using a collection of the best models discarding other models that predict the data far less well, even though the large amount of models with very small posteriors may contribute substantially to the sum such that Δ is much smaller than 1 (Madigan and Raftery 1994). Therefore, we introduce another measure for the quality of estimation. We define the relative ratio of the posterior probability of the MAP structure G_{MAP} over the posterior of the worst structure G_{MIN} in the k-best ECs or DAGs by

$$\lambda = \frac{P(G_{MAP}|D)}{P(G_{MIN}|D)} = \frac{P(G_{MAP}, D)}{P(G_{MIN}, D)}. \quad (6)$$

Note that both Δ and λ measures were used in (Tian, He, and Ram 2010).

kBestEC v.s. *kBestDAG*

We tested both algorithms on datasets from the UCI Machine Learning Repository as well as several synthetic datasets. All experiments were performed on a desktop with 2.4 GHz Intel Duo CPU and 4 GB of memory. The results are presented in Table 1. Besides k, Δ and λ, we list the number of variables n, sample size m, combined run-time T_{pn} for finding the k-best parent sets and finding the k-best ECs (or DAGs) (lines 4–14 in **Algorithm 1**), combined run-time T_e for enumerating DAGs (**Algorithm 2**) (0 for *kBestDAG* algorithm) and computing the posteriors, overall run-time T, total number of DAGs stored in memory $|\mathcal{G}_M|$, memory usage M (in MB), number of DAGs covered by the k-best ECs $|\mathcal{G}_k|$, and the average $\frac{|DAG|}{|EC|}$ ratio $\frac{|\mathcal{G}_k|}{k}$. All run-times are measured in seconds.

Our first observation is that, for all datasets, the running time T_e spent in enumerating all DAGs in k ECs is insignificant compared to the time for finding the k-best parent sets and ECs T_{pn} and the total time T. The total running time is dominated either by computing the local scores or by finding the k-best parent sets and the k-best ECs.

For the same k, BMA over k-best ECs has significantly better approximation quality than BMA over the k-best DAGs (see Δ values). This is straightforward since k ECs cover more than k DAGs and absorb more posterior probability density. $|\mathcal{G}_k|$ records the number of DAGs covered by the k-best ECs. Further, we see that *kBestEC* did spend

[5] *kBestEC* is available at http://www.cs.iastate.edu/~jtian/Software/AAAI-14-yetian/KBestEC.htm

Table 1: Experimental Results

				kBestEC									***kBestDAG***						
Data	n	m	k	T_{pn}	T_e	T	$\lvert\mathcal{G}_M\rvert$	M	Δ	λ	$\lvert\mathcal{G}_k\rvert$	$\frac{\lvert\mathcal{G}_k\rvert}{k}$	T_{pn}	T_e	T	$\lvert\mathcal{G}_M\rvert$	M	Δ	λ
Asia	8	500	1	0.008	0.001	7.17	256	0.05	0.011	1	3	3	0.005	0	7.11	256	0.05	0.0036	1
			3										0.008	0	7.12	750	0.12	0.011	1
			10	0.06	0.01	7.20	2255	0.35	0.064	4.5	43	4.3	0.02	0.01	7.13	2283	0.36	0.022	2.3
			43										0.1	0.01	7.22	8502	1.31	0.064	4.5
			100	0.65	0.04	7.81	16981	2.61	0.225	17.2	467	4.67	0.27	0.02	7.4	17793	2.74	0.101	6.9
			467										3.1	0.07	10.3	77614	11.9	0.225	17.2
			1000	11.8	0.4	19.3	106631	16.5	0.525	129	4694	4.69	10.9	0.13	18.1	132503	20.4	0.316	28.2
			4694										209	0.65	217	476045	73.6	0.525	129
			1e+4	528	3.8	539	875329	135	0.805	1602	44864	4.49	887	1.28	896	969503	150	0.628	270
Tic	10	958	1	0.03	0.01	7.79	1024	0.16	0.059	1	7	7	0.04	0.01	7.81	1024	0.16	0.0084	1
			7										0.1	0.01	7.81	6922	1.06	0.059	1
			10	0.43	0.01	8.16	9777	1.50	0.563	1	67	6.7	0.06	0.01	7.9	9826	1.51	0.084	1
			67										1.44	0.01	9.17	59952	9.17	0.563	1
			100	5.18	0.07	13.1	86213	13.2	0.759	1005	673	6.73	2.4	0.02	10.3	87936	13.4	0.694	3.6
			673										42.6	0.07	51	545247	83.3	0.759	1005
			1000	102	0.73	111	677869	104	0.759	5.1e+7	7604	7.6	85.3	0.2	93.3	753873	115	0.759	2.2e+4
			7604										4226	0.8	4237	4967225	759	0.759	5.1e+7
Syn-1	15	100	1	1.2	0.02	18.2	32768	5.01	1.69e-5	1	4	4	0.8	0.01	18.2	32768	5.01	4.23e-6	1
			4										3.0	0.01	20	130919	20	1.69e-5	1
			10	26.2	0.06	43.2	326696	49.9	3.34e-4	1.9	114	11.4	10.4	0.01	27.5	326801	49.9	4.14e-5	1.1
			100	497	0.1	514	3224431	492	1.65e-3	4.4	1084	10.8	321	0.02	338	3230906	493	3.03e-4	1.9
			114										390	0.02	407	3681594	562	3.34e-4	1.9
Syn-2	15	200	1	0.96	0.04	24.7	32768	5.01	1.65e-3	1	13	13	0.77	0.01	24.5	32768	5.01	1.27e-4	1
			10	26.8	0.3	50.8	326696	49.9	0.0129	2.5	185	18.5	10.3	0.01	34.0	326801	49.9	1.27e-3	1
			13										15.2	0.01	39.0	424742	64.8	1.65e-3	1
			100	512	2.28	538	3224431	492	0.0483	10.3	1808	18.1	331	0.01	355	3230906	493	0.0081	1.9
			185										811	0.03	836	5967226	911	0.0129	2.5
Vote	17	435	1	6.21	0.09	172	131072	20.0	0.0125	1	3	3	3.8	0.01	172	131072	20.0	0.0042	1
			3										10.5	0.01	177	393180	60	0.0125	1
			10	122	1	289	1309470	200	0.0871	2.4	30	3	51	0.01	218	1309606	200	0.0376	1.3
			30										270	0.01	437	3924566	599	0.0871	2.4
			100	2684	5.34	2865	13031570	1988	0.302	10.8	318	3.18	1946	0.02	2150	13041226	1990	0.172	4.3

more time for the same k as it requires extra overhead to respect equivalence. Both algorithms consume almost the same memory, which is consistent with the theory. An interesting observation is that *kBestEC* sometimes used slightly less memory than *kBestDAG* (see Asia $k = 1000$, $k = 1e4$, Tic $k = 1000$). This can be explained by comparing $|\mathcal{G}_M|$, the total number of DAGs stored in memory. *kBestEC* has smaller $|\mathcal{G}_M|$ than *kBestDAG*. This is because for small $W \subseteq V$, we usually have less than k distinct DAGs, and much less than k ECs to be stored. The effect is additive and in some cases causes big saving in both memory and time. For example, in case of Asia $k = 1e4$, $|\mathcal{G}_M|$ is significantly smaller for *kBestEC* than that for *kBestDAG*, such that *kBestEC* ($T = 539$ seconds) even ran faster than *kBestDAG* ($T = 896$ seconds).

Now we compare the two algorithms under the assumption that the same quality of approximation is achieved, i.e., they find the same number of DAGs, and therefore achieving the same Δ values. In order to achieve the same Δ as using k-best ECs, we have to run *kBestDAG* for a larger $k' = |\mathcal{G}_k|$ (the number of DAGs in the k-best ECs). With the same Δ, we observed that *kBestDAG* required significantly more time and memory. This is consistent with theoretical prediction of time ratio $\frac{k(\log k + nd^2/2)}{k'(\log k' + nd/2)}$ and space ratio $\frac{k}{k'}$. And for some Δ that *kBestEC* could easily achieve with the available resource, *kBestDAG* failed. In particular, for Syn-2 dataset, BMA over the top 100 ECs is equivalent to a BMA over the top 1808 DAGs. The former used only 492 MB memory, while the latter requires about 9 GB by estimation. Thus, *kBestEC* significantly outperformed *kBestDAG* in space and time usage to achieve the same quality of approximation.

A systematic comparison on T_{pn} of two algorithms when they find the same number of DAGs is presented in Figure 1. It plots the ratio $\frac{T_{pn,kBestDAG}(k')}{T_{pn,kBestEC}(k)}$ for $k' = |\mathcal{G}_k|$, against $\log_2 k$ for all five data sets. A red dashed horizontal line is drawn for where the ratio is 1. The figure clearly shows that $kBestEC$ is more efficient than $kBestDAG$ in finding the same number of DAGs.

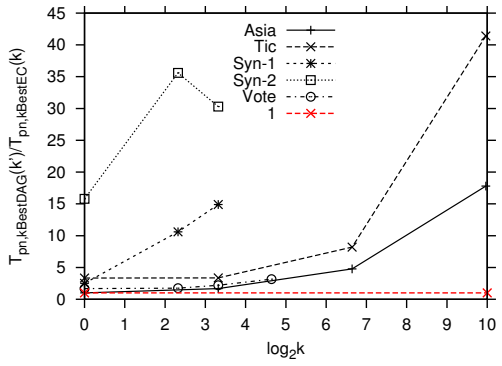

Figure 1: Comparison of run-times of two algorithms to achieve the same Δ values.

Structural Discovery

One important application of our algorithm is in (causal) structural discovery. To evaluate the performance of the algorithm, we randomly generated several network structures over 15 variables and used both *kBestDAG* and *kBestEC* to estimate the posteriors of all 210 edges. We also compare them to posteriors computed from exact method (Tian and

He 2009). The detailed results are presented in Figure S1 in the supplemental materials (Chen and Tian 2014). It shows the accuracy for model averaging over the k-best ECs is significantly better than that over the k-best DAGs as expected.

Another observation concerns about the reliability of using MAP model for structural inference. We first examine the λ value (Table 1). For Tic data set, the top 10 ECs are all equally probable. For data set Syn-1, the MAP equivalence class is only 1.9 times more probable than the 10-th best equivalence class, and only 4.4 times more probable than the 100-th best equivalence class. Similar results can be observed on Syn-2 and Vote data sets. This reflects that in many cases there are a significant number of distinct models explaining the data equally well and using MAP model for structure inference or causal reasoning is not reliable. Our algorithm will be a handy tool in understanding model structures in this kind of situation. A detailed comparison of the top 10 ECs for Tic data set is presented in Figure S2 in the supplemental materials (Chen and Tian 2014). It shows these 10 ECs agree only on one edge and disagree on other edges (even the skeleton). Further, most of the edges have probability below 0.5, indicating the high uncertainty on the network structure.

Conclusions and Future Work

In this paper we developed an algorithm to find the k-best equivalence classes of Bayesian networks. It is the first approach to our knowledge for finding the k-best equivalence classes. We show that our algorithm is significantly more efficient than the previous algorithm that directly finds the k-best DAGs (Tian, He, and Ram 2010). Our algorithm has applications in BMA and causal structure discovery.

Both *kBestDAG* and *kBestEC* are based on the DP algorithm. Recently, alternative approaches to finding the optimal BN have been proposed and shown being competitive or faster than the DP algorithm. These approaches include A* search (Yuan, Malone, and Wu 2011; Yuan and Malone 2012; Malone and Yuan 2012; 2013) and Integer Linear Programming (ILP) (Jaakkola et al. 2010; Cussens 2011; Bartlett and Cussens 2013). The A* search based algorithm URLearning formulates the learning problem as a shortest path finding problem and employs A* search algorithm to explore the search space. A potential future work is to explore the feasibility of generalizing the A* based algorithm to find the k-best DAGs or ECs. ILP based algorithm GOBNILP casts the structure learning problem as a linear program and solves it using the SCIP framework (Cussens 2011). In such setting, it is possible to rule out specific BNs with linear constraints. This allows GOBNILP to iteratively find the top k BNs in deccreasing order of score (Bartlett and Cussens 2013). Thus, another future work is to compare *kBestDAG*, *kBestEC* with GOBNILP in finding the k-best BNs.

Acknowledgments

We thank the anonymous reviewers for valuable comments.

References

Bartlett, M., and Cussens, J. 2013. Advances in Bayesian network learning using integer programming. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI-13)*, 182–191.

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

Chen, Y., and Tian, J. 2014. Supplemental materials to this paper. http://www.cs.iastate.edu/~jtian/papers/AAAI-14-Yetian-Supplement.pdf.

Chickering, D. M.; Geiger, D.; and Heckerman, D. 1995. Learning Bayesian networks: Search methods and experimental results. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, 112–128.

Chickering, D. M. 1995. A transformational characterization of equivalent Bayesian network structures. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, 87–98.

Chickering, D. M. 2002a. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research* 2:445–498.

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

Cussens, J. 2011. Bayesian network learning with cutting planes. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI-11):*, 153–160.

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

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

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

Gillispie, S. B., and Perlman, M. D. 2001. Enumerating markov equivalence classes of acyclic digraph dels. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, 171–177.

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

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

Jaakkola, T.; Sontag, D.; Globerson, A.; and Meila, M. 2010. Learning Bayesian network structure using lp relaxations. In *International Conference on Artificial Intelligence and Statistics*, 358–365.

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

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

Madigan, D., and Raftery, A. E. 1994. Model selection and accounting for model uncertainty in graphical models using occam's window. *Journal of the American Statistical Association* 89(428):1535–1546.

Madigan, D.; Andersson, S.; Perlman, M.; and Volinsky, C. 1996. Bayesian model averaging and model selection for markov equivalence classes of acyclic digraphs. In *Communications in Statistics: Theory and Methods*, 2493–2519.

Madigan, D.; York, J.; and Allard, D. 1995. Bayesian graphical models for discrete data. *International Statistical Review/Revue Internationale de Statistique* 215–232.

Malone, B., and Yuan, C. 2012. A parallel, anytime, bounded error algorithm for exact Bayesian network structure learning. In *Proceedings of the Sixth European Workshop on Probabilistic Graphical Models (PGM-12)*.

Malone, B., and Yuan, C. 2013. Evaluating anytime algorithms for learning optimal Bayesian networks. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI-13)*.

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

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

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

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

Silander, T., and Myllymäki, P. 2006. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the 22th Conference on Uncertainty in Artificial Intelligence*, 445–452.

Singh, A. P., and Moore, A. W. 2005. Finding optimal Bayesian networks by dynamic programming. Technical report, CMU-CALD-05-106, Carnegie Mellon University.

Spirtes, P.; Glymour, C.; and Scheines, R. 2000. *Causation, prediction, and search*, volume 81. The MIT Press.

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

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

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

Yuan, C., and Malone, B. 2012. An improved admissible heuristic for learning optimal Bayesian networks. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI-12)*.

Yuan, C.; Malone, B.; and Wu, X. 2011. Learning optimal Bayesian networks using a* search. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, 2186–2191.