An Order-based Algorithm for Learning Structure of Bayesian Networks

Shahab Behjati SHBEHJATI@IPM.IR
School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran

Hamid Beigy BEIGY@SHARIF.EDU
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

Abstract

In this paper, we study the problem learning structure of Bayesian networks from data. The problem of Bayesian networks structure learning (BNSL) takes a dataset as input and produces a directed acyclic graph (DAG) as the output. This problem is known to be NP-hard which is commonly solved using the heuristic methods. There are generally three main approaches to the BNSL problem: score-based , constraint-based and hybrid learning. We propose a new simple and fast algorithm for addressing BNSL problem. The proposed hybrid algorithm is based on a partial ordering learned from data. We reduce the super-exponential search space of structures to the smaller ordering space of nodes. We evaluate the proposed algorithm using some standard benchmark datasets and compare the results with those of some state-of-the-art algorithms. Finally, we show that our algorithm is competitive with recent algorithms.

Keywords: Bayesian network; structure learning; score-based algorithms; constraint-based algorithm.

1. Introduction

Bayesian networks (BNs) have a broad range of applications in various areas of artificial intelligence. BNs are graphical representations of multivariate joint probability distributions and have been widely used in various data-mining tasks to produce causal networks. A Bayesian network includes two components: structure and parameters. A Bayesian network structure is a directed acyclic graph whose nodes represent the random variables $X_1, \ldots, X_n$ in the problem domain and whose edges correspond to the direct probabilistic dependencies. A Bayesian network structure G encodes a set of conditional independence assumptions: each variable is conditionally independent of its non-descendants given its parents. With the above semantics, a Bayesian network structure provides a compact representation for joint distributions and supports efficient algorithms for handling probabilistic queries.

Generally, there are three approaches to the BNSL problem: score-based, constraint-based and hybrid methods. Constraint-based methods use statistical testings such as χ^2 to identify conditional independence relations from the data and construct a Bayesian network structure that best fits those independence relations.

Score-based learning evaluates the quality of Bayesian network structures using a scoring function and selects the one that has the best score. The scoring function computes a measure of goodness of fitting a network to a dataset (Heckerman et al., 1995). Score-based learning algorithms formulate the learning problem as a combinatorial optimization problem. They work well for datasets with few variables, but may fail to find optimal solutions for large datasets (Yuan et al., 2013).

Hybrid learning aims to integrate the advantages of constraint-based and score-based approaches and uses a combination of them for solving the BNSL problem (Perrier et al., 2008; Tsamardinos et al., 2006; Gasse et al., 2014; Gámez et al., 2011) . One popular strategy is to use constraint-based learning to create a skeleton graph and then use score-based learning to find a high-scoring network structure that is a subgraph of the skeleton.

This paper presents a new algorithm for learning Bayesian networks based on a partial order of nodes that learned from data. A novel contribution of our hybrid method is using the best parent set for each variable to aid the learning algorithm in inferring a partial order of nodes. We reduce the super-exponential search space of structures to the smaller ordering space of nodes. We evaluate the proposed algorithm using some standard benchmark datasets and compare the results with the results obtained from some state of the art algorithms.

The rest of the paper is organized as follows. In Section 2, we review the structure learning problem and the necessary background. In Section 3, we review some search algorithms based on a given node ordering. Then, and in Section 4 we introduce the proposed algorithm. Finally, in Section 5 we give experiment results. We conclude with case studies and discussion in Section 6.

2. Structure Learning

We are given a data set $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$ of N instances on the n categorial random variable $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$. The goal is to find a DAG $G = (V, E)$ that best fits $\mathcal{D}$ that means that encodes the conditional independencies present in the data $\mathcal{D}$. We assume all variables are discrete and there is no missing values in data $\mathcal{D}$. Next, we give a brief review of the three main for the BNSL problem.

2.1 Constraint-based learning methods

These methods typically use statistical tests to identify conditional independence relations from the data and build a Bayesian network structure that best fits those independence relations (Pearl, 2014). Therefore, by using a set of statistical tests, we determine which random variables are conditionally independent of each other.The algorithms that belong to this category include: the PC (Spirtes and Glymour, 1991), the Max-Min Parents and Children (MMPC) (Tsamardinos et al., 2003a), the Incremental Association Markov Boundary (IAMB) (Tsamardinos et al., 2003b), the Incremental IAPC Association Parents and Children (IAPC) (De Morais and Aussem, 2010) and the Hybrid Parents and Children (HPC) (De Morais and Aussem, 2010). Almost, all constraint-based structure learning algorithms have two phases. A constraint-based learning algorithm in the first phase learns the skeleton of the DAG, and then, in the second phase, arc directions are established.

2.2 Score-based learning methods

Score-based learning methods evaluate the quality of Bayesian network structures using a scoring function and select the one with the best score (Cooper and Herskovits, 1992; Heckerman et al., 1995). A scoring function is a metric that measures the goodness of fitting a network structure to data set $\mathcal{D}$. These methods formulate the learning problem as a combinatorial optimization problem. A score-based algorithm can be formulated as follows, given a dataset $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$, search for a DAG G^* such that

$$G^* = \arg\max_{G \in \mathcal{G}_n} score(G)$$

where $\mathcal{G}_n$ is the family of all the DAGs defined on $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$.

A score function is usually defined as a measure of fitness between the structure graph and the data. There are so many scoring functions that can be used to measure the quality of a network structure. These functions can be classified into two main categories: Bayesian, and information-theoretic scoring functions.

The local search learning algorithms can be more efficient if the scoring function being used has the property of decomposability. A scoring function s is decomposable if the score assigned to each network decomposed over the network structure in such a way that can be expressed as a sum of local scores that depends only on each node and its parents. That means,

$$score(G) = \sum_{i=1}^{n} score(X_i|\Pi_i) \tag{1}$$

2.3 Hybrid learning algorithms

We now summarize the pros and cons of both score-based and constraint-based approaches. Bayesian and MDL scoring functions are well-defined regardless of the data generating distribution P, or the size of the data set $\mathcal{D}$. In that sense, the score-based approach is theoretically well-suited in every situation, and in practice, it is known to produce better structures than the constraint-based approach. Its major drawback, however, is an exponential time complexity with respect to the number of variables considered, which makes it rather prohibitive for high-dimensional datasets (Gasse, 2017).

On the other hand, constraint-based approaches require two major assumptions: 1) the independence model $I(p)$ is faithful to a DAG; and 2) the conditional independence (CI) tests performed on $\mathcal{D}$ accurately reflect $I(p)$. Both of these assumptions are problematic. Firstly, the DAG-faithfulness assumption forbids many simple kinds of interactions between the variables of interest, such as deterministic relationships which violate the intersection property. In practice, such relationships are frequent in many systems, making the DAG-faithfulness assumption rather unrealistic. Secondly, even when I(p) is faithful to a DAG, it may very well be that the independence model extracted empirically with CI tests, $I(\mathcal{D})$, is not. As a result, constraint-based methods are known to be quite unstable, and are prone to cascading effects where a single early error on in the building process can result in very a different DAG structure. This is particularly true during the edge-orientation step. Still, constraint-based methods are in practice rather fast, as their computational complexity relates closely to the maximum in-degree of any node in the DAG, regardless of the total size of the graph (Dash and Druzdzel, 1999),(Gasse, 2017).

Hybrid methods integrate the advantages of constraint-based methods and score-based methods for solving the structure learning problem (Dash and Druzdzel, 1999). Several hybrid methods have been proposed recently such as (Gámez et al., 2011) which in the neighbourhood is dynamically constrained to speed the search. The Min–Max Hill Climbing (MMHC) algorithm (Tsamardinos et al., 2006) is one of the most used algorithms for BN structure learning capable of dealing with high dimensional data in a reasonable time. MMHC start first with a skeleton that is learned using the constraint-based MMPC algorithm, then a high-scoring DAG is found using a greedy search within the restricted search space of the skeleton, enhanced with a TABU list as in (Friedman et al., 1999) to escape local maxima.

3. Ordering-Based Search Algorithms

The search for the network structure can be carried out in the space of DAGs, the space of equivalence classes and the space of orderings among the domain variables (Alonso-Barba et al., 2011). We know the search space of Bayesian network structures is super exponential in the number of nodes:

$$f(n) = \sum_{i=1}^{n} (-1)^{i+1} \frac{n!}{(n-i)!n!} 2^{i(n-i)} f(n-1)$$

Fortunately, for a given ordering $\prec$ on the nodes of a Bayesian network, the problem of finding the best-scoring networks consistent with respect to $\prec$ is not NP-hard. Indeed, if the in-degree of nodes are bounded to k, then the best-scoring network can be learned in time $O(n^k)$. However, determining a true ordering of nodes is a very difficult problem. A very simple and easy-to-implement method for addressing this problem is given by (Teyssier and Koller, 2005). For a given ordering $\prec$ defined as the possible parent sets for each node X_i with an upper bound k on the number of parents per node is given as:

$$\mathcal{U}_{i,\prec} = \{U : U \prec X_i, |U| \leq k\},$$

where all nodes in U precede X_i in $\prec$. Then, the optimal parent set for each node X_i is:

$$\Pi_{i,\prec} = \arg \max_{U \in \mathcal{U}_{i,\prec}} score(X_i, U)$$

The possible parent sets for the all nodes can be computed in time $O(n^{k+1})$. We can find the optimal network by searching the optimal ordering, where the score of an ordering is the score of the best network consistent with it. The search operator over the orderings space is called neighbor-swapping operator:

$$(X_{i_1}, \ldots, X_{i_j}, X_{i_{j+1}}, \ldots) \mapsto (X_{i_1}, \ldots, X_{i_{j+1}}, X_{i_j}, \ldots)$$

The search is done by considering all $(n-1)$ candidate successors of the current ordering. The algorithm than compares the delta-scores of the successor orderings obtained by these swaps i.e., the difference between their scores and the current one, and takes the one that gives the highest delta-score. The TABU list is used to prevent the algorithm from reversing a swap that was executed recently in the search. The process is continued until a local maximum is reached (Teyssier and Koller, 2005). This search is performed using a greedy hill-climbing with random restarts and a TABU list.

The Ordering-based Max-Relevance and Min-Redundancy Greedy (OMRMRG) is an Ordering-based Max Relevance and Min Redundancy Greedy algorithm (Liu et al., 2007). OMRMRG presents an ordering-based greedy search method with a greedy pruning procedure, applies Max-Relevance and Min-Redundancy feature selection method, and proposes Local Bayesian Increment function according to BIC score function. The OMRMRG algorithm is divided into two parts. The first part is to learn Bayesian network given an ordering on the variables. The second part is to learn Bayesian network without the constraint of an ordering on the variables.

At the first part, MRMRG algorithm initializes the current parents set Π_i of the variable X_i to NULL, and then adds the variables one by one, which acquire the maximal value for Local Bayesian Increment (LBI) function, into the parents set Π_i from $Pre_i - \Pi_i$, until the result of LBI function is no more than 0. Repeating the above steps for every variable, we can obtain an approximately optimal Bayesian network. Pre_i denotes the set of variables that precede X_i and Π_i denotes the current

parents set of the variable X_i. At the second part, the OMRMRG Algorithm apply the neighbor-swapping operator and iterative greedy search method over the space of orderings in OMRMRG algorithm. In each iteration, OMRMRG algorithm firstly performs the search by considering all $(n-1)$ candidate successors of the current ordering, then compares the scores of the current ordering and the successor orderings, finally takes the one that gives the highest score. The process is continued until a local maximum is reached. OMRMRG algorithm restarts a new iteration by selecting a new current ordering at random, and try the process MAX times to avoid local maximum. In comparison to other ordering-based greedy learning algorithms such as (Teyssier and Koller, 2005), given an ordering on the variables, OMRMRG algorithm replaces traditional greedy BN learning algorithms with the procedure MRMRG() in order to learn more accurately and efficiently on limited datasets. Furthermore, OMRMRG algorithm uses a greedy pruning procedure based on Max Relevance and Min Redundancy technology, which pre-selects a more accurate set of candidate parents for each variable X_i than the current candidate parents set selection methods described in (Friedman et al., 1999) on limited datasets (Liu et al., 2007). Another most popular ordering-based search algorithm is K2 (Cooper and Herskovits, 1992) that we will explain it in the next subsection. There are some algorithms for search in the space of equivalence classes. The most important of these is GES (greedy equivalence search) (Chickering, 2002a). However, original GES which takes greedy strategy into account may easily fall into local optimization trap because of the empty initial structure. Recently, some improvements of GES method are proposed (Alonso-Barba et al., 2011, 2013; Zhang et al., 2013; Chickering and Meek, 2015).

3.1 k2 Algorithm

The K2 algorithm (Cooper and Herskovits, 1992) is a one of the most important ordering-based search algorithms that learns the BN structure. This greedy algorithm uses a prior ordering of nodes as input for reducing the complexity of the search space.

Let r_i be the maximum number of possible values for variable X_i. also r be the defined by $r = \max_{1 \leq i \leq n} r_i$ and $N = |\mathcal{D}|$. Therefore, the time complexity of K2 algorithm is $O(mn^2k^2r)$. In worst case , $k = n-1$ and its time is $O(Nn^4r)$.

A novel algorithm that obtains a node ordering from data has been proposed by (Chen et al., 2008). By using this ordering as input to the K2 algorithm the structure of the BN is learned. This algorithm aims to identify information for the correct ordering of nodes and runs in three phases. First, it finds an undirected network (UDN) using mutual information (MI) and interdependence tests. Second, the UDN is refined using d-separation and the conditional independence test to eliminate possible false edges and add true edges that may have been missed in the original UDN. Finally, orientations are assigned for each edge using interdependency tests and Bayesian scoring metrics. The MI between two random variables X and Y denoted $I(X:Y)$ is defined as follows:

$$I(X:Y) = H(X) - H(X|Y) \tag{2}$$

where $H(X)$ and $H(Y)$ are the entropy of random variables X and Y respectively, and $H(X|H)$ is the conditional entropy of random variable X given Y. The entropy and conditional entropy are defined as follows:

$$H(X) = -\sum_{i=1}^{n} P(X_i) log P(X_i) \tag{3}$$

$$H(X|Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(X = x_i, Y = y_j) \times logP(X = x_i|Y = y_j) \tag{4}$$

To construct the UDN, we first compute MI for all nodes, and then determine the maximum MI (MMI) for each node X_i denoted by $MMI(X_i)$. If either one of the following conditions is satisfied, an edge (X_i, X_j) will be added between variables X_i and X_j:

$$I(X_i : X_j) \geq \alpha MMI(X_i) \text{ or } I(X_i : X_j) \geq \alpha MMI(X_j) \tag{5}$$

where $0.5 \leq \alpha \leq 1$. The parameter α determines the number of connections at this stage. If α is close to 1.0, there is a high probability of several true edges being rejected at this stage of the algorithm. If α is close to 0.0, this could lead to the inclusion of several wrong edges at this stage. For all the structure-learning problems in this study, the parameter α is set to 0.9 and is found to include most of the true edges while allowing the addition of only a few wrong edges into the network structure.(Chen et al., 2008)

Let N and n be the number of samples and the number of variables, respectively. Let r be the maximum number of possible values for any variable. The time complexity of this algorithm is $O(n^4) + O(Nn^2)$, which is polynomial with respect to the number of nodes and linear with respect to the number of samples.

4. Learning structure via learning the order of nodes

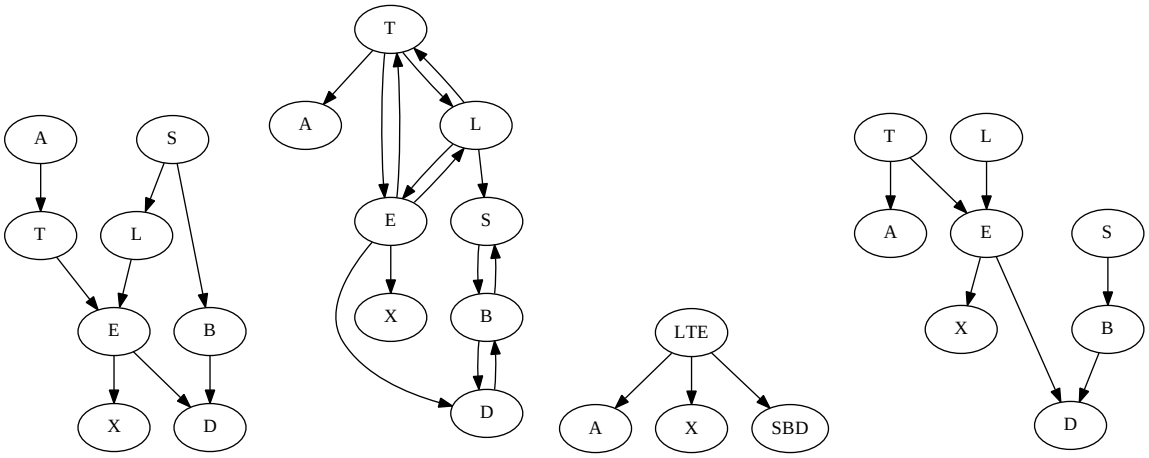
As we have seen, all the ordering-based search methods use a predefined order given on nodes. But there exist no information about how this order has been made. In this paper, we show how one can obtain a partial ordering on nodes given data. First, we find the optimal parent sets(those with maximum scores) for each variable X_i by ignoring the acyclic constraints. This method was described by (Yuan et al., 2013) to find optimal parents for a variable X_i out of a candidate set. For each variable, there exist a parent graph. For n variables, there are 2^n nodes in the order graph, and n parent graphs with 2^{n-1} parent nodes each. In total, $n2^{n-1}$ parent scores need to be computed. As the number of variables increases, computing and storing the order and parent graphs quickly becomes infeasible (Yuan et al., 2013). However, the following theorem (de Campos and Ji, 2010; Teyssier and Koller, 2005) helps us to prune a significant portion of the candidate parent sets for each node.

Theorem 1 *Let **U** and **S** be two candidate parent sets for X , **U**$\subset$ **S** and Score(X,**U**)$\leq$ Score(X,**S**). Then **S** is not an optimal parent set of X for any candidate set.*

We now explain the workings of our algorithm, which relies on three concepts, namely, strongly connected components, strongly connected components graph and topological sorting.

A strongly connected component of a directed graph $G = (V, E)$ is a maximal set of vertices $C \subseteq V$ such that for every pair of vertices u and v in C, vertices u and v are reachable from each other.

Suppose that G has strongly connected components $C_1, C_2, \ldots, C_k$. The vertex set V^{SCC} is $\{v_1, v_2, \ldots, v_m\}$, and it contains a vertex v_i for each strongly connected component C_i of G. There is an edge $(v_i, v_j) \in E^{SCC}$ if G contains a directed edge (X, Y) for some $X \in C_i$ and some $Y \in C_j$



(a) Real BN ASIA (b) Initial Bayesian network ASIA (c) The SCC graph (d) Final learned BN ASIA

Figure 1: Real BN ASIA, initial BN ASIA,the SCC graph and the final learned BN

. The key property is that the component graph $G^{SCC} = (V^{SCC}, E^{SCC})$ is a DAG (Cormen et al., 2009) and therefore has a topological order.

Furthermore, a topological order (sort) of a DAG $G = (V, E)$ is a linear ordering of all its vertices such that if G contains an edge (u, v), then u appears before v in the ordering. Topological sorting for a graph is not possible when the graph is not a DAG. We can view a topological sort of a graph as an ordering of its vertices along a horizontal line so that all directed edges go from left to right (Cormen et al., 2009).

By using the Sparse Parent Graph Algorithm in (Yuan et al., 2013) and setting k as the maximum parent size for each node, we use pre-computed best parent sets for each node. We then construct structure graph $G = (V, E)$ from the possible parent sets for each node. This directed graph may not be a DAG. In the next step, the algorithm computes the strongly connected components of graph $G = (V, E)$ and form the component graph $G^{SCC} = (V^{SCC}, E^{SCC})$. Then we construct the component graph $G^{SCC} = (V^{SCC}, E^{SCC})$ in linear time $O(V + E)$ as follows. We obtain a topological sort $(v_{i_1}, v_{i_2}, \ldots, v_{i_m})$ of the component graph $G^{SCC} = (V^{SCC}, E^{SCC})$. For each node $v_{i_j} \in V^{SCC}$ where $j = 1, 2, \ldots, m$, if $|v_{i_j}| \geq 2$, then we take a random permutation on all its variables. This give us a ordering on all nodes of graph $G = (V, E)$. Finally, using this ordering and given ordering-based search algorithm K2 in (Chen et al., 2008), we can find a high-scoring network structure effectively. The pseudocode of the our algorithm are shown in Algorithm 1.

In the next subsection, we illustrate the workings of our algorithm through an example.

4.1 An illustrative example

The ASIA network is a small BN, which is used for a fictional medical example. The network asks whether a patient has tuberculosis, lung cancer, or bronchitis, and consists of eight nodes and eight edges. Each random variable is discrete and can take two states. Figure 1a shows the true structure for the ASIA BN.

Algorithm 1: Learning the structure via learning the order of the nodes

Input : Variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ and dataset $\mathcal{D}$ on $\mathcal{X}$.
Output: A Bayesian network structure G

for $X_i \leftarrow 1$ **to** n **do**
  $POP[i] \leftarrow$ Potential-Optimal-ParentSets(X_i,$\mathcal{D}$)
end
for $X_i \leftarrow 1$ **to** n **do**
  Create node X_i;
  $\mathcal{U}_i = \arg \max_{S \in POP[i]} score(S)$;
  for $X_j \in \mathcal{U}_i$ **do**
    add directed edge (X_i, X_j) to the E;
  end
end
Construct $G^{scc} = (V^{scc}, E^{scc})$ from the graph $G = V, E)$;
Let $C_1, \ldots, C_k$ be the strongly connected components graph $G = (V, E)$. Therefore, $V^{scc} = \{v_1, \ldots, v_k\}$;
Make a topological sort on graph G^{scc} and call it $O^{G_{scc}}$;
for $v_i \in O^{G_{scc}}$ **do**
  if $|v_i| > 1$ **then**
    consider a random order $O'[i]$ on nodes in v_i
  end
  else
    set $O'[i]$ as $\{v_i\}$
  end
end
Construct Total order O on all vertices of graph G By concatenating of O' order elements;
$G^* \leftarrow$ Ordering-Based-Search($\mathcal{X}$,$\mathcal{D}$, O);
return G^*

The BIC scores for potential parent sets with max parent size $k = 2$ for each node are given in Table 1. The initial graph G is shown in Figure 1b. The strongly connected components graph $G^{scc} = (V^{scc}, E^{scc})$ graph of $G = (V, E)$ is shown in Figure 1c.

A topological sort of graph Figure 1c is $(L, T, E), A, X, (S, B, D)$. By expanding it, we obtain ordering L, T, E, A, X, S, B, D on vertices graph G. The final graph which is learned using this node ordering and K2 algorithm is shown in Figure 1d.

5. Experiments

In this section, we present the experimental results obtained from our algorithm and compare with three others algorithms on five BNs taken from the Bayesian network repository included in the **bnlearn** R package[1]. We selected one *small* size network(20 nodes), three *medium* size networks(20-50 nodes) and one *large* size networks(50-100 nodes). Table 2 shows detailed information about

1. http://www.bnlearn.com/bnrepository/

Table 1: Potential parent sets for BN ASIA

NODE	BEST POTENTIAL PARENT SET
A	{T}:-117.247665
S	{L,B}:-3155.790283
T	{L,E}:-53.389854
L	{T,E}:-44.909386
E	{T,L}:-17.034386
B	{S,D}:-2498.392334
X	{D}:-1701.340454
D	{B,E}:-2336.587158

each network consist: number of the variables, edges and parameters. For each network we independently sampled one dataset containing 10000 instances.

Table 2: Datasets used

Dataset	# of variables	#of edges	# of parameters
ASIA	8	8	18
INSURANCE	27	52	984
MILDEW	35	46	540150
ALARM	37	46	509
HAILFINDER	56	66	2656

5.1 Results

We compare our algorithm against the state-of-the-art algorithm (Bartlett and Cussens, 2013) referred to as Gobnilp[2], the greedy hill-climbing search (Gámez et al., 2011) and the Greedy Equivalence Search Algorithm (GES) (Chickering, 2002b).

use both greedy hill-climbing search (Gámez et al., 2011) and exact learning algorithm (Bartlett and Cussens, 2013) referred to as Gobnilp[3].

To compare the three searches, we recorded the best scoring network found by each one. We report, for each dataset, the score of the maximum reached by each algorithm. The results of our experiments is given in Table 3. In Table 3 we have reported, for each dataset, the score of the maximum reached by each algorithm.

We use the quality of the network (BDe score) as performance measure. The use of CPU times as measure has the disadvantage of being dependent on the actual implementation and hardware. However we include them here as an indication. For each network, 3 datasets were randomly sampled in size of 10000. The performance metrics of an algorithm were the averages over those 3 datasets for each network. All experiments are conducted on 2.6 GHz Intel Core i7 PC with 8 GB of RAM. We have used Tetrad 4[4] software to run the GES algorithm.

2. https://www.cs.york.ac.uk/aig/sw/gobnilp/
3. https://www.cs.york.ac.uk/aig/sw/gobnilp/
4. http://www.phil.cmu.edu/projects/tetrad/

The results are shown in Table 3. We can see that our algorithm outperforms greedy hill-climbing and GES algorithm in terms of precision. Moreover, our algorithm, which we implemented in Python, is the fastest among all algorithms except greedy hill-climbing algorithm.

Table 3: Comparison of results

Score Results

Dataset	Exact	hill-climbing	GES algorithm	Our algorithm
ASIA	-22629.67	-22629.67	-22629.67	-22834.30
INSURANCE	-132563.50	-134017.90	-133761.41	-132831.82
MILDEW	-462641.22	-503568.30	-475561.78	-455478.60
ALARM	-106251.4	-107176.3	-109140.07	-106899.43
HAILFINDER	-472322.31	-498918.2	$-54.041.12$	-484455.63

Time Results

Dataset	Exact	hill-climbing	GES algorithm	Our algorithm
ASIA	$2.60s$	$0.02s$	$0.059s$	$0.10s$
INSURANCE	$3.51s$	$0.27s$	$4.236s$	$1.21s$
MILDEW	$14.33s$	$0.31s$	$5.10s$	$1.27s$
ALARM	$20.22s$	$0.39s$	$6.663s$	$1.14s$
HAILFINDER	$67.40s$	$0.69s$	$56.729s$	$2.18s$

6. Conclusion

We described a simple and fast algorithm for learning Bayesian networks from data. Our method has two stage. First, we learn a partial order of nodes from data. Second, we learn the structure using this node ordering. We evaluated our algorithm on a variety of commonly used benchmark datasets against current state-of-the-art algorithms. In most cases, we showed that our algorithms outperformed existing methods by running faster, and finding better solutions more quickly. In comparison, our algorithm is very fast and gives reasonable results and is scalable to large networks. Our main contribution is obtaining a partial ordering on nodes from given data. Improving the scalability of optimal structure learning algorithms has many practical applications.

In future, we will work on both other orderings and also we will work on finding out components with the almost the same size.

ACKNOWLEDGEMENTS

The authors would like to thank James Cussens and Brandon Malone for useful comments for providing us using GOBNILP and URLearning softwares respectively.

References

J. I. Alonso-Barba, J. M. Puerta, et al. Structural learning of bayesian networks using local algorithms based on the space of orderings. *Soft Computing*, 15(10):1881–1895, 2011.

J. I. Alonso-Barba, J. A. Gámez, J. M. Puerta, et al. Scaling up the greedy equivalence search algorithm by constraining the search space of equivalence classes. *International Journal of Ap-*

proximate Reasoning, 54(4):429–451, 2013.

M. Bartlett and J. Cussens. Advances in bayesian network learning using integer programming. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 182–191, 2013.

X.-W. Chen, G. Anantha, and X. Lin. Improving bayesian network structure learning with mutual information-based node ordering in the k2 algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):628–640, 2008.

D. M. Chickering. Learning equivalence classes of bayesian-network structures. *Journal of machine learning research*, 2(Feb):445–498, 2002a.

D. M. Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002b.

D. M. Chickering and C. Meek. Selective greedy equivalence search: Finding optimal bayesian networks using a polynomial number of score evaluations. *arXiv preprint arXiv:1506.02113*, 2015.

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

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

D. Dash and M. J. Druzdzel. A hybrid anytime algorithm for the construction of causal models from sparse data. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 142–149, 1999.

C. P. de Campos and Q. Ji. Properties of bayesian dirichlet scores to learn bayesian network structures. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 431–436, 2010.

S. R. De Morais and A. Aussem. An efficient and scalable algorithm for local bayesian network structure discovery. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 164–179, 2010.

N. Friedman, I. Nachman, and D. Peér. Learning bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 206–215, 1999.

J. A. Gámez, J. L. Mateo, and J. M. Puerta. Learning bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery*, 22(1):106–148, 2011.

M. Gasse. *Probabilistic Graphical Model Structure Learning: Application to Multi-Label Classification*. PhD thesis, Université Lyon 1-Claude Bernard, 2017.

M. Gasse, A. Aussem, and H. Elghazel. A hybrid algorithm for bayesian network structure learning with application to multi-label learning. *Expert Systems with Applications*, 41(15):6755–6772, 2014.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

F. Liu, F. Tian, and Q. Zhu. A novel ordering-based greedy bayesian network learning algorithm on limited data. In *Australasian Joint Conference on Artificial Intelligence*, pages 80–89, 2007.

J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 2014.

E. Perrier, S. Imoto, and S. Miyano. Finding optimal bayesian network given a super-structure. *Journal of Machine Learning Research*, 9:2251–2286, Oct. 2008.

P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social science computer review*, 9(1):62–72, 1991.

M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence*, pages 584–590, 2005.

I. Tsamardinos, C. F. Aliferis, and A. Statnikov. Time and sample efficient discovery of markov blankets and direct causal relations. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 673–678, 2003a.

I. Tsamardinos, C. F. Aliferis, A. R. Statnikov, and E. Statnikov. Algorithms for large scale markov blanket discovery. In *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference*, volume 2, pages 376–380, 2003b.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The Max-Min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

C. Yuan, B. Malone, et al. Learning optimal bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research*, 48:23–65, 2013.

Y. Zhang, W. Zhang, and Y. Xie. Improved heuristic equivalent search algorithm based on maximal information coefficient for bayesian network structure learning. *Neurocomputing*, 117:186–195, 2013.