Initialization Heuristics for Greedy Bayesian Network Structure Learning

Walter Perez Urcia
and
Denis Deratani Mauá

Instituto de Matemática e Estatística, Universidade de São Paulo, Brazil
wperez@ime.usp.br,denis.maua@usp.br

Abstract. A popular and effective approach for learning Bayesian network structures is to perform a greedy search on the space of variable orderings followed by an exhaustive search over the restricted space of compatible parent sets. Usually, the greedy search is initialized with a randomly sampled order. In this article we develop heuristics for producing informed initial solutions to order-based search motivated by the Feedback Arc Set Problem on data sets without missing values.

Categories and Subject Descriptors: I.2.6 [**Artificial Intelligence**]: Learning

Keywords: Bayesian networks, machine learning, local search

1. INTRODUCTION

Bayesian Networks are space-efficient representations of complex multivariate probability distributions [Jensen 2001]. They are defined by two components: (i) a directed acyclic graph (DAG) encoding the (in)dependence relationships among the variables in the model; and (ii) a collection of local conditional probability distributions of each variable given its parents.

Manually specifying a Bayesian network is a difficult task, and practitioners often resort to "learning" the model from data. A common approach to learning a Bayesian network consists of associating every DAG with a polynomial-time computable score value and searching for structures with high score values [Cooper and Dietterich 1992; Lam and Bacchus 1994; Margaritis 2003; Tessyer and Koller 2005]. The score value of a structure usually rewards structures that assign high probability of observing the data set (i.e., the data likelihood) and penalizes the complexity of the model (i.e., the number of parameters). Some examples are the Bayesian Information Criterion (BIC) [Cover and Thomas 1991], the Minimum Description Length (MDL) [Lam and Bacchus 1994] and the Bayesian Dirichlet score (BD) [Heckerman et al. 1995]. An alternative approach is to learn the DAG by multiple conditional independence hypothesis testing [Spirtes and Meek 1995; ?]. Although both approaches can recover the true DAG (if one exists) given infinite data and computational resources, testing for independence introduces a lot of false positives and it is often followed by a score-based approach [?].

Score-based Bayesian network learning from data is a NP-hard problem [Chickering et al. 2004], even when the in-degree (i.e., maximum number of parents) of the graph is bounded. For this reason, the most common approach is to resort local search methods that find an approximate solution [H. Friedman and Peér 1999; Chickering 2002]. A popular and very effective method for learning Bayesian networks is to perform a local search on the space of topological orderings [Tessyer and

Koller 2005]. The search is usually initialized with an ordering sampled uniformly at random from the space of orderings. This can make the search converge to a poor local optima unless more sophisticate techniques are employed [Elidan et al. 2002], which can add significant computational overhead. An alternative solution is to initialize the search in high-scoring regions.

In this work we design two new heuristics for generating good initial solutions to order-based Bayesian network structure learning. The first heuristic follows the observation that only orderings consistent with a relaxed version of the problem (in which cycles are permitted) can lead to an optimal structure. Although this heuristic biases the search away from regions which are *guaranteed* to be sub-optimal, it generates orderings with equal probability in any other region. Our second heuristic refines the first one by selecting high scoring orderings among the ones that are consistent with the relaxed version solution. We do this by reducing the problem to a variant of the Feedback Arc Set Problem (FASP), which is the problem of transforming a cyclic direct graph into a DAG. Our experiments show that using these new methods improves the quality of order-based local search.

The rest of this paper is structured as follows: we begin in Section 2 explaining greedy search approaches to learning Bayesian networks. Then in Section 3 we describe the new algorithms for generating initial solutions. Section 4 shows the experiments using both approaches and comparing them (in scoring and number of iterations needed) with multiple data sets. Finally, in Section 5 we give some conclusions about the new methods.

2. LEARNING BAYESIAN NETWORKS

In this section, we formally define the score-based approach learning of Bayesian networks, and review some of the most popular techniques for solving the problem.

2.1 Definition of the problem

A Bayesian network specification contains a DAG $G = (V, E)$, where $V = \{X_1, X_2, \ldots, X_n\}$ is the set of (discrete) variables, and a collection of conditional probability distributions $P(X_i \mid Pa_G(X_i))$, $i = 1, \ldots, n$, where $Pa_G(X_i)$ is the set of variables that are parents of X_i in G. This definition shows that the number of numerical parameters (i.e., local conditional probability values) grows exponentially with the number of parents (in-degree) of a node (assuming the values are organized in tables). A Bayesian network induces a joint probability distribution over all the variables through the equation $P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid Pa_G(X_i))$. Hence, Bayesian networks with sparse DAGs succinctly represent joint probability distributions over many variables.

A *scoring function* $sc(G)$ assigns a real-value to any DAG indicating its goodness in representing a given data set.[1] Most scoring functions can be written in the form $sc(G) = F(G) - \varphi(N) \times P(G)$, where N is the number of records in the data set D, $F(G)$ is a data fitness function (i.e., how well the model represents the observed data), $\varphi(N)$ is a non-decreasing function of data size and $P(G)$ measures the model complexity of G. For example, he Bayesian information criterion (BIC) is defined as $BIC(G) = LL(G) - \frac{\log N}{2} size(G)$, where $LL(G) = \sum_{i=1}^{n} \sum_k \sum_j N_{ijk} \log \frac{N_{ijk}}{N_{ij}}$ is the data loglikelihood, $size(G) = \sum_{i=1}^{n} (|\Omega_i| - 1) \prod_{X_j \in Pa(X_i)} |\Omega_j|$ is the "size" of a model with structure G, n is the number of attributes on D, N_{ijk} the number of instances where attribute X_i takes its kth value, and its parents take the jth configuration (for some arbitrary fixed ordering of the configurations of the parents' values), and similarly for N_{ij}, and Ω_i is the set of possible values for the attribute X_i. Most commonly used scoring functions, BIC included, are *decomposable*, meaning that they can be written as a sum of local scoring functions: $sc(G) = \sum_i sc(X_i, Pa(X_i))$. Another property often satisfied by scoring functions is *likelihood equivalence*, which asserts that two structures with same

[1]The dependence of the scoring function on the data set is usually left implicitly, as for most of this explanation we can assume a fixed data set. We assume here that the dataset contains no missing values.

loglikelihood also have the same score [Chickering and Meek 2004]. Likelihood equivalence is justified as a desirable property, since two structures that assign the same loglikelihood to data cannot be distinguished by the data alone. The BIC scoring function satisfies likelihood equivalence.

Given scoring function $sc(G)$, the score-based Bayesian network structure learning problem is to compute the DAG

$$G^* = \arg \max_{G:G \text{ is a DAG}} sc(G) \,. \tag{1}$$

Provided the scoring function is decomposable, we can obtain an upper bound on the value of $sc(G^*)$ by computing $sc(\overline{G})$, where

$$\overline{G} = \arg \sum_i \max_{Pa(X_i)} sc(X_i, Pa(X_i)) \tag{2}$$

is the directed graph where the parents $Pa(X_i)$ of each node X_i are selected so as to maximize the local score $sc(X_i, Pa(X_i))$. We call the parents of a variable in $\overline{G}$ the *best parent set* (for X_i. Note that $\overline{G}$ usually contains cycles, and it is thus not a solution to Equation 1.

2.2 Greedy Search Approaches

Greedy Search is a popular approach used to finding an approximate solution to (1). The method relies on the definition of a neighborhood space among solutions, and on local moves that search for an improving solution in the neighborhood of an incumbent solution. Different neighborhoods and local moves give rise to different methods such as Equivalence-based, Structure-based, and Order-based methods. Algorithm 1 shows a general pseudocode for this approach.

Algorithm 1: Greedy Search

```
1  GreedySearch( Dataset $D$ ) : return a BN $G$
2     $G = Initial\_Solution(X_1, \dots, X_n)$
3     For a number of iterations $K$
4         $best\_neighbor = find\_best\_neighbor(G)$
5         if $score(best\_neighbor) > score(G)$ then
6             $G = best\_neighbor$
7     Return $G$
```

The main idea of the approach is to start with an initial solution (e.g., a randomly generated one), and for a number of iterations K, explore the search space by selecting the best neighbor of the incumbent solution. Additionally, an early stop condition can be added to verify whether the algorithm has reached a local optimum (i.e., if no local move can improve the lower bound). Several methods can be obtained by varying the implementation of lines 2, 4 and 5, which specify how to generate an initial solution, what the search space is and what the scoring function is, respectively.

2.2.1 *Structure-based.* One of earliest approaches to learning Bayesian networks was to perform a greedy search over the space of DAGs, with local moves being the operations of adding, removing or reverting an edge, followed by the verification of acyclicity in the case of edge addition [Cooper and Dietterich 1992; Grzegorczyk and Husmeier 2008]. The initial solution is usually obtained by randomly generating a DAG, using one of the many methods available in the literature [Ide and Cozman 2002; Melançon and Philippe 2004].

2.2.2 *Equivalence-based.* An alternative approach is to search within the class of score-equivalent DAGs. This can be efficiently achieved when the scoring function is likelihood equivalent by using pDAGs, which are graphs that contain both undirected and directed edges (but no directed cycles) with the property that all orientations of a pDAG have the same score. In this case, greedy search operates on the space of pDAGs, and the neighborhood is defined by addition, removal and reversal of edges, just as in structure-based search [Chickering 1996; 2002].

2.2.3 *Order-based.* Order-Based Greedy Search is a popular and effective approach, which is based on the observation that the problem of learning a Bayesian network can be written as

$$G^* = \arg\max_{<} \max_{G \text{ consistent with } <} \sum_{i=1}^{n} sc(X_i, Pa(X_i)) = \arg\max_{<} \sum_{i=1}^{n} \max_{P \subseteq \{X_j < X_i\}} sc(X_i, P), \qquad (3)$$

which means that if an optimal ordering over the variables is known, an optimal DAG can be found by maximizing the local scores independently [Heckerman et al. 1995; H. Friedman and Peér 1999; Tessyer and Koller 2005]. This can be made efficiently if we assume G^* is sparse, which is true for many scoring functions [de Campos and Ji 2011].

Order-Based Search starts with a topological ordering L, and greedily moves to an improving ordering by swapping two adjacent attributes in L if any exists. Algorithm 2 shows a pseudocode for the method. The function *swap* in line 6 swaps the values $L[i]$ and $L[i+1]$ in the order L to obtain a neighbor of the incumbent solution.

Algorithm 2: Order-Based Greedy Search

```
1   OrderBasedGreedySearch( Dataset $D$ ) : **return** a BN
2       $L = Get\_Order(X_1, \ldots, X_n)$
3       For a number of iterations $K$
4           $current\_sol = L$
5           For each $i = 1$ to $n-1$ do
6               $L_i = swap(L, i, i+1)$
7               if $score(L_i) > score(current\_sol)$
8                   $current\_sol = L_i$
9           if $score(current\_sol) > score(L)$ then
10              $L = current\_sol$
11      Return $network(L)$
```

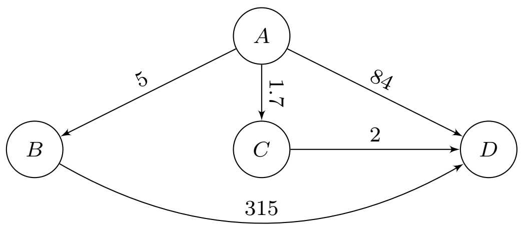
The standard approach to generate initial solutions is to sample a permutation of the attributes uniformly at random by some efficient procedure such as the Fisher-Yates algorithm [Knuth 1998]. While this guarantees a good coverage of the search space when many restarts are performed, it can lead to poor local optima. In the next section, we propose new strategies to informed generation of topological orderings to be used as initial solutions in Order-Based search.

3. GENERATING INFORMED INITIAL SOLUTIONS

As with most local search approaches, the selection of a good initial solution is crucial for avoiding convergence to poor local maxima in Order-Based Learning. Traditionally, this is attempted by randomly generating initial solutions (i.e., a node ordering) in order to cover as much as possible of the search space. In this section, we devise methods that take advantage of the structure of the problem to produce better initial solutions.

3.1 DFS-based solution

We can exploit the information provided by the graph $\overline{G}$ (defined in equation 2) to reduce the space of topological orderings and avoid generating orderings which are guaranteed sub-optimal. Assume the best parent sets are unique, and consider a pair of nodes X_i, X_j in $\overline{G}$ such that X_j is parent of X_i but there is not arc from X_i into X_j. Then, no optimal ordering can have X_i preceding X_j (this can easily be shown by contradiction). Hence, only topological orderings consistent with $\overline{G}$ are potential candidates for optimality, and this number can be much smaller than the full space of orderings. To see this ore clearly, consider Figure 1 which shows a possible graph $\overline{G}$ and the corresponding consistent orderings. As can be noticed we have 14 consistent orderings out of $4! = 24$ possible topological orders. This difference is likely to increase as the number of variables increases.



ABDC	CDBA
ACDB	DABC
ADBC	DACB
ADCB	DBAC
BDAC	DBCA
BDCA	DCAB
CDAB	DCBA

Fig. 1: A an example of a fraph $\overline{G}$ and its consistent topological orderings

Taking into consideration the previous analysis, we propose the following algorithm to generate initial solutions. Take as input the graph $\overline{G}$ and mark all nodes as unvisited. While there is an unvisited node, select an unvisited node X_i uniformly at random and add to the list the nodes visited by a depth-first search (DFS) tree rooted at X_i. Finally, return L, an ordering of the nodes.

3.2 FAS-based solution

The DFS approach can be seen as removing edges from $\overline{G}$ such as to make it a DAG (more specifically, a tree), and then extracting a consistent topological ordering. That approach hence considers that all edges are equally relevant in terms of avoiding poor local maxima. We can estimate the arguably relevance of an edge $X_j \rightarrow X_i$ by

$$W_{ji} = sc(X_i, Pa^*(X_i)) - sc(X_i, Pa^*(X_i) \setminus \{X_j\}), \tag{4}$$

where $Pa^*(X_i)$ denotes the best parent set for X_i (i.e., its parents in $\overline{G}$). The weight W_{ji} represents the cost of removing X_j from the set $Pa^*(X_i)$ and it is always a positive number because $Pa(X_i)$ maximizes the score for X_i. A small value means that the parent X_j is not very relevant to X_i (in that sense), while a large value denotes the opposite. For instance, in the weighted graph $\overline{G}$ in Figure 1 the edge $C \rightarrow D$ is less relevant than the edges $A \rightarrow D$, which in turn is less relevant than the edge $B \rightarrow D$.

The main idea of our second heuristic is to penalize orderings which violate an edge $X_i \rightarrow X_j$ in $\overline{G}$ by their associated cost W_{ij}. We then wish to find a topological ordering of $\overline{G}$ that violates the least cost of edges. Given a directed graph $G = (V, E)$, a set $F \subseteq E$ is called a Feedback Arc Set (FAS) if every (directed) cycle of G contains at least one edge in F. In other words, F is an edge set that if removed makes the graph G acyclic [Demetrescu and Finocchi 2001]. If we assume that the cost of an ordering of $\overline{G}$ is the sum of the weights of the violated (or removed) edges, we can formulate the problem of finding a minimum cost ordering of $\overline{G}$ as a Minimum Cost Feedback Arc Set Problem (min-cost FAS): given the weighted directed graph $\overline{G}$ with weights W_{ij} given by (4), find a FAS F such that

$$F = \min_{G-F \text{ is a DAG}} \sum_{X_i \rightarrow X_j \in E} W_{ij}. \tag{5}$$

Even though the problem is NP-hard, there are efficient and effective approximation algorithms like the one described in Algorithm 3 [Demetrescu and Finocchi 2001].

Algorithm 3: FAS approximation

```
MinimumCostFAS( Graph $G$ ) : Return FAS $F$
    $F$ = empty set
    While there is a cycle $C$ on $G$ do
        $W_{min}$ = lowest weight of all edges in $C$
        For each edge $(u,v) \in C$ do
            $W_{uv} = W_{uv} - W_{min}$
```

```
7                 If $W_{uv} = 0$ add to $F$
8         For each edge in $F$, add it to $G$ if does not build a cycle
9         Return $F$
```

We can now describe our second heuristic for generating initial solutions, based on the minimum cost FAS problem: take the weighted graph $\overline{G}$ with weights W_{ij} as input, and find a min-cost FAS F; Reverse or remove the edges in F from $\overline{G}$ and return a topological order of the obtained graph $\overline{G} - F_G$ (this can be done by performing a DFS starting with root nodes).

4. EXPERIMENTS, RESULTS AND DISCUSSION

In order to evaluate the quality of our approaches, we learned Bayesian networks using Order-based greedy search and different initialization strategies from several data sets commonly used for benchmarking. The names and relevant characteristics of the data sets[2] used are shown in Table I, where the density of a graph is defined as the ratio of the number of edges and the number of nodes. For

Dataset	n (#attributes)	N (#instances)	Density of $\overline{G}$
Census	15	30168	2.85
Letter	17	20000	2.41
Image	20	2310	2.45
Mushroom	23	8124	2.91
Sensors	25	5456	3.00
SteelPlates	28	1941	2.18
Epigenetics	30	72228	1.87
Alarm	37	1000	1.98
Spectf	45	267	1.76
LungCancer	57	27	1.44

Table I: Data sets characteristics

each dataset we performed 1000 runs of Order-Based Greedy Search with a limit of 3 parents ($d = 3$) and 100 iterations ($K = 100$), except for the LungCancer dataset where only 100 runs were performed due to the limited computational resources. We used the BIC score and found the best parent sets for a given ordering by exhaustive search.

We compared our proposed initialization strategies, which we call DFS- and FAS-based, against the standard approach of randomly generating an order (called Random). For each strategy, we compared the best score obtained over all runs (Best score), the average initial score (i.e., the score of the best DAG consistent with the initial ordering), the average best score (i.e., the average of the scores of the local searches) and the average number of iterations that local search took to converge. The results are shown in Table II. The results show that in most of the datasets with less than 25 attributes, the Random strategy finds the highest-scoring networks over all runs, even though it finds worse networks on average. The best initial solutions are found by the FAS-based strategy followed by the DFS-based strategy. For datasets with more than 25 variables, Random is less effective in finding high-scoring networks, except for the LungCancer (which has very little data). These results suggest that more informed approaches to generating initial orderings might be more effective in high dimensionality domains, or when the number of restarts is limited e.g. for computational reasons. The proposed strategies are also more robust, which can be seen by the smaller variance of the average initial and best scores.

The results also suggest that the proposed strategies are more effective than Random in datasets for which the graph $\overline{G}$ is sparser (smaller density), showing that pruning the space of orderings can be

[2]These datasets were extracted from http://urlearning.org/datasets.html

Dataset	Approach	Best Score	Avg. Initial Score	Avg. Best Score	Avg. It.
	Random	**-212186.79**	-213074.18 ± 558.43	-212342.26 ± 174.21	7.26 ± 2.90
Census	DFS-based	-212190.05	-212736.80 ± 379.96	-212339.83 ± 152.26	5.90 ± 2.61
	FAS-based	-212191.64	**-212287.99 ± 92.54**	**-212222.12 ± 70.99**	**3.28 ± 1.67**
	Random	-138652.66	-139774.54 ± 413.74	-139107.13 ± 329.15	6.07 ± 2.50
Letter	DFS-based	-138652.66	-139521.38 ± 396.61	**-138999.84 ± 310.06**	5.75 ± 2.35
	FAS-based	-138652.66	**-139050.43 ± 70.55**	-139039.26 ± 87.97	**2.24 ± 0.96**
	Random	**-12826.08**	-13017.13 ± 44.35	-12924.24 ± 41.39	7.59 ± 2.71
Image	DFS-based	-12829.10	-12999.09 ± 38.56	-12921.13 ± 37.88	7.10 ± 2.47
	FAS-based	-12829.10	**-12930.63 ± 20.83**	**-12882.30 ± 26.43**	**5.05 ± 1.72**
	Random	**-55513.38**	-58450.72 ± 1016.54	-56563.84 ± 616.59	7.59 ± 2.76
Mushroom	DFS-based	**-55513.38**	-58367.11 ± 871.25	-56472.72 ± 546.19	7.75 ± 2.58
	FAS-based	-55574.71	**-56450.49 ± 154.54**	**-56198.66 ± 174.64**	**4.65 ± 1.63**
	Random	**-62062.13**	-63476.33 ± 265.46	-62726.60 ± 251.26	9.22 ± 2.94
Sensors	DFS-based	-62083.21	-63392.60 ± 255.90	-62711.50 ± 257.79	9.65 ± 3.12
	FAS-based	-62074.88	**-62530.26 ± 133.44**	**-62330.94 ± 121.82**	**5.17 ± 2.24**
	Random	-13336.14	-13566.50 ± 65.80	-13429.13 ± 52.14	8.96 ± 3.43
SteelPlates	DFS-based	**-13332.91**	-13572.77 ± 81.12	-13432.30 ± 57.57	9.30 ± 3.38
	FAS-based	-13341.73	**-13485.26 ± 38.27**	**-13397.08 ± 29.53**	**7.77 ± 2.24**
	Random	-56873.76	-57722.30 ± 228.44	-57357.60 ± 222.12	5.89 ± 2.67
Epigenetics	DFS-based	**-56868.87**	**-57615.36 ± 189.17**	**-57308.93 ± 165.18**	6.42 ± 2.47
	FAS-based	**-56868.87**	-57660.09 ± 146.45	-57379.59 ± 148.42	**5.33 ± 2.28**
	Random	-13218.22	-13324.52 ± 30.49	-13245.43 ± 15.63	10.92 ± 3.24
Alarm	DFS-based	**-13217.97**	-13250.72 ± 17.70	-13236.71 ± 12.02	**4.32 ± 2.32**
	FAS-based	-13220.55	**-13249.77 ± 2.57**	**-13233.98 ± 6.19**	6.34 ± 1.74
	Random	-8176.81	-8202.03 ± 5.23	-8189.69 ± 4.65	7.20 ± 2.17
Spectf	DFS-based	**-8172.37**	-8200.04 ± 4.08	-8187.29 ± 4.91	7.86 ± 2.49
	FAS-based	-8172.51	**-8176.98 ± 2.01**	**-8176.07 ± 2.05**	**2.27 ± 1.11**
	Random	**-711.23**	-723.79 ± 2.69	-718.03 ± 2.84	5.46 ± 1.78
LungCancer	DFS-based	-711.36	-720.47 ± 2.51	**-715.29 ± 1.86**	5.02 ± 1.50
	FAS-based	-711.39	**-716.13 ± 0.89**	-715.67 ± 1.19	**2.73 ± 1.79**

Table II: Best score obtained, Average initial score generated, Average best score obtained, Average number of iterations (Avg. It.) using each approach (best values in bold)

effective in those cases. The initial orderings provided by the proposed strategies speed up convergence of the local search, as can be seen by the smaller number of average iterations for those strategies in the Table.

Overall, the new heuristics are able to improve the accuracy of Order-Based Greedy Search with only a small overhead. Although the differences observed in our experiments were small, we expect greater differences in domains of higher dimensionality

5. CONCLUSIONS AND FUTURE WORK

Learning Bayesian networks from data is a notably difficult problem, and practitioners often resort to approximate solutions such as greedy search. The quality of the solutions produced by greedy approaches strongly depends on the initial solution. In this work, we proposed two new heuristics for producing topological orderings to be fed into Order-Based Greedy Bayesian network Structure Search methods. One is based on a Depth-First Search traversal of the (cyclic) graph obtained by greedily selecting the best parents for each variable; the other is based on finding an acyclic subgraph of that same graph by solving a related minimum cost Feedback-Arc Set problem. Experiments with real-world datasets containing from 15 to 57 variables demonstrate that compared to the commonly used strategy of generating initial ordering uniformly at random the proposed heuristics lead to better solutions on average, and increase the convergence of the search with only a small overhead . Although the gains observed in our experiments are small, we expect larger differences for datasets with more variables. A follow-up work should verify this hypothesis.

Our proposed techniques could be adapted to generate initial solutions also for Structure- and Equivalence-based local search methods by returning directed acyclic graphs instead of node orderings. Another extension of this work is to employ the proposed heuristics in branch-and-bound solvers such as [de Campos and Ji 2011] for finding optimal solutions. These ideas are left as future work.

REFERENCES

Chickering, D. M. Learning equivalence classes of Bayesian-network structures. *Conference on Uncertainty in Artificial Intelligence*, 1996.

Chickering, D. M. Learning Equivalence Classes of Bayesian-Network Structures. *Journal of Machine Learning Research*, 2002.

Chickering, D. M., Heckerman, D., and Meek, C. Large-Sample Learning of Bayesian Networks is NP-Hard. *Journal of Machine Learning Research* 5 (1): 1287–1330, 2004.

Chickering, D. M. and Meek, C. Finding Optimal Bayesian Networks. *Journal of Machine Learning Research*, 2004.

Cooper, G. F. and Dietterich, T. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 1992.

Cover, T. M. and Thomas, J. A. *Elements of Information Theory.* Wiley-Interscience, 1991.

de Campos, C. P. and Ji, Q. Efficient Structure Learning of Bayesian Networks using Constraints. *Journal of Machine Learning Research* vol. 12, pp. 663–689, 2011.

Demetrescu, C. and Finocchi, I. Combinatorial Algorithms for Feedback Problems in Directed Graphs. *MURST Project for Young Researchers*, 2001.

Elidan, G., Ninio, M., and Schuurmans, N. F. D. Data Perturbation for Escaping Local Maxima in Learning. *Proceedings of the National Conference on Artificial Intelligence*, 2002.

Grzegorczyk, M. and Husmeier, D. Improving the structure MCMC sampler for Bayesian networks by introducing a new edge reversal move. *Machine Learning*, 2008.

H. Friedman, I. N. and Peér, D. Learning bayesian network structure from massive datasets: The "sparse candidate" algorithm. *Conference on Uncertainty in Artificial Intelligence* (15, 1999.

Heckerman, D., Geiger, D., and Chickering, D. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Journal of Machine Learning Research* 20 (MSR-TR-94-09): 197–243, 1995.

Ide, J. S. and Cozman, F. G. pp. 366–376. In , *Random Generation of Bayesian Networks.* Vol. 2507. Springer Berlin Heidelberg, pp. 366–376, 2002.

Jensen, F. V. *Bayesian Networks and Decision Graphs.* Springer Science and Business Media, 2001.

Knuth. *The Art of Computer Programming 2.* Boston: Adison-Wesley, 1998.

Lam, W. and Bacchus, F. Learning Bayesian Belief Networks. An approach based on the MDL principle. *Computational Intelligence* 10 (4): 31, 1994.

Margaritis, D. Learning Bayesian Network Model Structure from Data, 2003.

Melançon, G. and Philippe, F. Generating connected acyclic digraphs uniformly at random. *Inf. Process. Lett.* 90 (4): 209–213, May, 2004.

Spirtes, P. and Meek, C. Learning Bayesian networks with discrete variables from data. *1st International Conference on Knowledge Discovery and Data Mining*, 1995.

Tessyer, M. and Koller, D. Ordering-Based Search: A Simple and Effective Algorithm for Learning Bayesian Networks. *Conference in Uncertainty in Artificial Intelligence*, 2005.