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 multivariate probability distributions over random variables [Jensen 2001]. They are defined by two components: (i) a directed acyclic graph (DAG), where the nodes are the variables and the edges encode the (in)dependence relationships among the variables; and (ii) a collection of local conditional probability distributions of each variable given its parents. More formally, 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

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid Pa_G(X_i)). \tag{1}$$

Hence, Bayesian networks with sparse DAGs succinctly represent joint probability distributions over many variables.

A common approach to learning Bayesian networks from a given data set is score-based. These methods such as K2 algorithm [Cooper and Dietterich 1992], consists of associating every graph structure with a polynomial-time computable score value and searching for structures with high score values [Margaritis 2003]. However, there is another approach to solve this problem that is a constraint-based one where the methods learn conditional independence statements among the variables using statistical tests on the data set. With these statements the methods build an undirected structure and then determine the edge's orientation to obtain a Bayesian network [Spirtes and Meek 1995], but these are not the focus in this work, so we will not to these any further

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].

Score-based Bayesian network structure 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 to solve the problem is to use local search methods such as searching over the space of DAGs [H. Friedman and Peér 1999], searching over Markov equivalence classes [Chickering 2002], and searching over the space of topological orders [Tessyer and Koller 2005]. Local search methods are well-known to be vulnerable to poor local maxima unless a strategy for avoiding low score regions is used. One such strategy is the use of non-greedy heuristics that allow moving for lower value solutions during search in order to escape local maxima [Elidan et al. 2002]. Another strategy is to use *good initialization heuristics* that attempt to start the search in regions of high local maxima. Most often, a simple uniformed random generation of initial solutions is adopted.

In this article we propose new heuristics for generating good initial solutions to be fed into local search methods for Bayesian network structure learning. We focus on order-based greedy methods on data sets which are state of the art. Also, one of the heuristics is motivated by solutions 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 article is structured as follows: we begin in Section 2 explaining the Greedy Search approach to learning Bayesian networks. 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 define the score-based approach learning of Bayesian networks, and review some of its most popular techniques.

2.1 Definition of the problem

Given a data set D and a scoring function $sc(G)$[1], the Bayesian network structure learning problem is to select a score-maximizing DAG, that is to find

$$G^* = \arg \max_{G:G \text{ is a DAG}} sc(G) \,. \tag{2}$$

Most scoring functions can be rewritten in the form

$$sc(G) = F(G) - \varphi(N) \times P(G), \tag{3}$$

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

[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.

$P(G)$ measures the model complexity of G. One example is the Bayesian information criterion (BIC) 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: $sc(G) = \sum_i sc(X_i, Pa(X_i))$. Provided the scoring function is decomposable, we can obtain an upper bound on the value of $sc(G^*)$ by computing $sc(\overline{G})$, where

$$\overline{G} = \arg \sum_i \max_{Pa(X_i)} sc(X_i, Pa(X_i)) \,. \tag{4}$$

is the directed graph such that each $Pa(X_i)$ is the parent set that maximizes the score $sc(X_i, Pa(X_i))$, also called *best parent set*. Note that $\overline{G}$ usually contains cycles, and it is thus not a solution to Equation 2. Another property often satisfied by scoring functions is *likelihood equivalence*, which asserts that two structures with same 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.

2.2 Greedy Search Approaches

Greedy Search is a popular approach used to finding an approximate solution. 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       $G = Initial\_Solution(X_1, \ldots, X_n)$
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 random 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

$$G^* = \arg\max_{<} \max_{G \text{ consistent with } <} \sum_{i=1}^{n} sc(X_i, Pa(X_i)) = \arg\max_{<} \sum_{i=1}^{n} \max_{P \subseteq \{X_j < X_i\}} sc(X_i, P), \qquad (5)$$

which means that if an optimal ordering over the attributes is known, an optimal DAG can be found by maximizing the local scores independently [Tessyer and Koller 2005]. This can be made efficiently if we assume G^* is sparse and this can be shown to be true for many scoring functions [de Campos and Ji 2011].

The method 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 $network(L)$
```

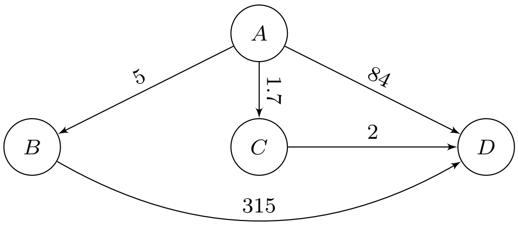
The standard approach to generate initial solutions is to sample a permutation of the attributes uniformly at random by some efficient procedure such as the Fisher-Yates algorithm [Knuth 1998]. While this guarantees a good coverage of the search space when many restarts are performed, it can lead to 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 that avoids poor local maxima is crucial for finding good solutions. Traditionally, this is attempted by randomly generating initial solutions (DAGs, pDAGs or orderings) in order to cover as much as possible of the space. In this section, we devise methods that take advantage of the structure of the problem to produce better initial solutions. Although we focus on Order-Based search, our methods could be used for any of the other greedy approaches discussed in Section 2 doing some modifications that will be explained in Section 5.

3.1 DFS-based solution

The structure of a problem instance provided by the graph $\overline{G}$ (defined in equation 4) can be used to prune the space of topological ordering by avoiding generating ordering which are guaranteed sub-optimal as follows. This graph tell us some information about if it is good that a variable $X_j \in Pa(X_i)$ should be before X_i in an order, because it has as relationships the best parent set for each X_i. This means that if a variable $X_j \notin Pa(X_i)$ for some variable X_i, it does not maximize the score for that variable and could be better to appear after X_i.
For instance, Figure 1b shows the possible orders consistent with the relationships in the graph $\overline{G}$ (Figure 1a). As can be noticed we have 14 possible orders given $\overline{G}$, but $4! = 24$ possible orders using a random approach. So we reduce the search space and also improve the possibilities that getting a good initial order. Also, the difference could be greater in problems with more variables.



(a)

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

(b)

Fig. 1: (a) Graph $\overline{G}$. (b) Possible orders for graph $\overline{G}$

Taking into consideration the previous analysis, we propose the following algorithm to generate initial solutions. Having as input the graph $\overline{G}$ and starting with an empty list L, choose a node X_k not visited until that moment and perform a depth-first search (DFS) using X_k as root. Every node visited during the DFS is added to list L. Repeat these steps while exists some node not visited, then return L.

3.2 FAS-based solution

The DFS approach can be seen as removing edges from $\overline{G}$ such as to make it a DAG, and then extract a consistent topological ordering. A weakness with that approach is that it uses $\overline{G}$ and generate orders consistent with $\overline{G}$ using a random node as root, but does not consider that some arcs are more relevant than others and could be better to keep them on the graph because they maximize the score. It also means that some nodes could be more relevant than others in the best parent set of a variable. The weakness is that it considers all arcs equally (ir)relevant when obtaining a DAG out of $\overline{G}$, but we can estimate the relevance of an arc using Equation 6.

$$W_{ji} = sc(X_i, P) - sc(X_i, P \setminus \{X_j\}), \tag{6}$$

where W_{ji} is the weight of the edge that goes from X_j to X_i and P is the best parent set for X_i. This formula represents the cost of getting X_j out of the set P and it is always a positive number because P maximizes the score for X_i. A small value means the parent X_j is not very relevant to X_i, whereas a big value means that X_j is relevant. For instance, graph $\overline{G}$ in Figure 1a shows that C is less relevant than B and A as a parent of D because that edge has a lower value.
Another weakness is that the DFS-based method is that it is safe in the sense that it only prune non-optimal orderings, but it could be of little use if $\overline{G}$ is not sparse. When this is the case, the problem could relaxed by obtaining a sparser version of $\overline{G}$ removing the less relevant arcs. This problem is called Minimum Cost Feedback Arc Set Problem (min-cost FAS) that is stated as: given a weighted directed graph $G = (V, E)$, a set $F \subseteq E$ is called a minimum cost Feedback Arc Set if every cycle of

G has 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]. So this problem is to find a min-cost FAS F_G such that:

$$F_G = \min_{G-F \text{ is a DAG}} \sum_{(u,v)\in E} W_{uv}, \tag{7}$$

where W_{uv} is the weight of the edge from u to v on G. Even though the problem is NP-hard, there are some efficient approximation algorithms like the one described in Algorithm 3.

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

We can now describe our second heuristic for generating initial solutions, based on the minimum cost FAS problem: having the graph $\overline{G}$ as input, add weights W_{ji} to the arcs on the graph using Equation 6. Then, find the min-cost FAS F_G and remove its arcs from $\overline{G}$ to make it a DAG. Finally, return a topological order of the obtained graph $\overline{G} - F_G$.

These new methods for generating initial solutions will be used in next section to learn Bayesian networks with multiple data sets.

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 division of its number of edges and its number of nodes. The greedy search was ran with a maximum number of parents ($d = 3$), a limit of 100 iterations ($K = 100$), and 1000 restarts ($S = 1000$), except for LungCancer dataset where the running total time is big to perform such as number of restarts. We used the BIC score in all experiments and the parent sets were calculated by exhaustive search in all cases.

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

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

We compared our proposed initialization strategies, DFS- and FAS-based, against the standard approach of randomly generation a order. For each approach, we compared the best score obtained, the average initial score, the average best score 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 approach finds Bayesian networks with greater or equal score than the methods proposed, but it does not hold for bigger datasets where DFS-based method obtained better scores than FAS-based because the reduction in the search space done by these methods improves the possibilities to obtain better scores as shown in average best score column, also with a considerable lower standard deviation compared to the random approach. Moreover, results show that new methods work better in datasets where graph $\overline{G}$ is sparse (small density), possible because it has to do less choices for removing arcs in $\overline{G}$. Finally, the relation between the average number of iterations needed to converge from DFS-based method and random approach is increasing while n increases. However, FAS-method initial solutions converge faster than in any of the methods with a considerable difference which means that these initial solutions are close to good local maxima, even for the biggest datasets. Last statement can be proven to be true by looking at the average initial score column that shows that FAS-based method always generate solutions with the highest scores having the lowest standard deviation. So in general, the new methods work better with bigger datasets than smaller ones and this improves while the density of $\overline{G}$ decreases.

Dataset	Approach	Best Score	Avg. Initial Score	Avg. Best Score	Avg. It.
Census	Random	**-212186.79**	-213074.18 ± 558.43	-212342.26 ± 174.21	7.26 ± 2.90
	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**
Letter	Random	-138652.66	-139774.54 ± 413.74	-139107.13 ± 329.15	6.07 ± 2.50
	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**
Image	Random	**-12826.08**	-13017.13 ± 44.35	-12924.24 ± 41.39	7.59 ± 2.71
	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**
Mushroom	Random	**-55513.38**	-58450.72 ± 1016.54	-56563.84 ± 616.59	7.59 ± 2.76
	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**
Sensors	Random	**-62062.13**	-63476.33 ± 265.46	-62726.60 ± 251.26	9.22 ± 2.94
	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**
SteelPlates	Random	-13336.14	-13566.50 ± 65.80	-13429.13 ± 52.14	8.96 ± 3.43
	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**
Epigenetics	Random	-56873.76	-57722.30 ± 228.44	-57357.60 ± 222.12	5.89 ± 2.67
	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**
Alarm	Random	-13218.22	-13324.52 ± 30.49	-13245.43 ± 15.63	10.92 ± 3.24
	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
Spectf	Random	-8176.81	-8202.03 ± 5.23	-8189.69 ± 4.65	7.20 ± 2.17
	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**
LungCancer	Random	**-711.23**	-723.79 ± 2.69	-718.03 ± 2.84	5.46 ± 1.78
	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)

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 new heuristics for producing good initial solutions to be fed into order-based greedy Bayesian network structure search methods.

Experiments with benchmark data sets suggested that our heuristics lead to better solutions while the size of the datasets increases, and that in most cases it leads to faster convergence with a small overhead (compared to the commonly used methods of generating initial solutions). The advantage of our heuristics grows with the size of the data set.

Our proposed techniques could be adapted to generate initial solutions for Structure- and Equivalence-based methods by generating DAGs as follows: the DFS approach could be adapted to saving the DFS directed tree and the FAS approach to returning the DAG obtained from removing the FAS instead of returning its topological order. Our heuristics can also be exploited by branch-and-bound solvers such as [de Campos and Ji 2011] finding optimal solutions. These ideas are left for 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.