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 method 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.

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 the atributtes of a data set. They are defined by two components: (i) a directed acyclic graph (DAG), where the nodes are the atributes of the data set and the edges encode the (in)dependence relationships among the atributes; and (ii) a collection of local conditional probability distributions of each atribute 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) attributes, 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 atributes 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 attributes through the equation

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

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

A common approach to learning Bayesian networks from a given data set is the search-and-score approach, which consists of associating every graph structure with a polynomial-time computable score value and searching for structures with high score values. 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 commonest approach to solve the problem is to use local search methods such as searching over the space of structures, searching over Markov equivalence classes, and searching over the space of topological orders. Local search methods are well-known to be vulnerable to poor local maxima unless a strategy for avoinding 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. Another stragegy 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 methods, but our technique can be exploited by any local search procedure for learning Bayesian network structures. 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 algorithm. 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 mathematically the search-and-score approach to the Bayesian network learning problem and we review some popular solutions.

2.1 Definition of the problem

The problem of learning the structure of a Bayesian network is stated as: Given a training data set D, select the network (DAG) G that maximizes the scoring function $sc(G, D)$. A general definition of a scoring function is as follows:

$$sc(G, D) = F(G, D) - \varphi(N) \times P(G), \tag{2}$$

where N is the size of the data set D, $F(G, D)$ 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 instance, the Bayesian information criterion (BIC) is defined as

$$BIC(G, D) = LL(G) - \frac{\log N}{2} size(G), \tag{3}$$

where

$$LL(G) = \sum_{i=1}^{n} \sum_{k} \sum_{j} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} \tag{4}$$

is the data loglikelihood,

$$size(G) = \sum_{i=1}^{n} (|\Omega_i| - 1) \prod_{X_j \in Pa(X_i)} |\Omega_j|, \tag{5}$$

represents the "size" of a model with structure G, N is the number of instances in the data set, n is the number of atributes, 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 scoring functions, BIC included, are *decomposable*, meaning that they can be written as a sum of local scoring functions $sc(X_i, Pa(X_i))$. Under the assumption of *score decomposability*, the Bayesian network structure learning problem is defined as finding

$$G = \max_{\{Pa(X_i)\}_i, G \text{ is acyclic}} \sum_{i=1}^{n} sc(X_i, Pa(X_i)). \tag{6}$$

In words, the structure learning problem is to select for each attribute the set parent attributes that maximize the local score while ensuring that the graph is acyclic. Exhaustively searching for the optimal parent set $Pa(X_i)$ for each attribute takes $O(2^n)$ time. To avoid the exponential blow-up we constrain the maximum number of parents to a value d, hence reducing the search space to 2^d; this means $|Pa(X_i)| \leq d$ for all attributes. Typical values for d are 3, 4 and 5, depending on the cardinality of variables. Additional speed-up can be achieved by using pruning schemes that remove suboptimal subsets of parent sets without resorting to inspection [de Campos and Ji 2011]. Notice that G is not necessarily equal to the graph G^* defined in equation 7 because G^* has for each attribute X_i its best parent set P (the one that maximizes $sc(X_i, Pa(X_i))$) and it can contain cycles (by definition, a network is a DAG).

$$G^* = \sum_{i=1}^{n} \max sc(X_i, Pa(X_i)), \tag{7}$$

2.2 Greedy Search Algorithm

The Greedy Search algorithm is a popular heuristic method used to find an approximation for a solution of learning Bayesian network problem using the concept of neighborhood between solutions. Depending on the focus of the algorithm this could be classified as Equivalence-based, Structure-based, Order-based, etc. Algorithm 1 shows a general pseudocode for this algorithm.

Algorithm 1: Greedy Search

```
1   G = Initial_Solution(X_1, ..., X_n, conf)
2   For a number of iterations K
3           best_neighbor = find_best_neighbor(G)
4           if score(best_neighbor) > score(G) then
5                   G = best_neighbor
6   Return G
```

The main idea of this algorithm is to generate an initial solution based on the $conf$ parameter (for instance, $conf$ can say that the solution has to be random generated). After that, for a number of iterations K explore the search space and selects the best neighbor of the best solution until that moment. Then calculate its score in order to know if it is a better solution. Additionally to the stop condition could be added that the solution converges (i.e. its score does not improve). Finally, return the best solution when the stop condition holds.
As mentioned before, there are lot of different approaches using this algorithm, but all of them only change lines 1, 3 and 4 depending on its own way to generate an initial solution, its search space and the scoring function used, respectively.

2.2.1 *Structure-based.*

In this version the initial solution G is a network, possibly random generated, and its possible neighbors are other networks that only differs by one arc. This means that an arc can be inserted, deleted or inverted its direction to obtain a new neighbor.

2.2.2 *Equivalence-based.* In the same way, the initial solution G is a network and uses the equivalence relation between networks to obtain new neighbors. Two networks L_1 and L_2 are equivalent if they have the same probability distribution [Chickering and Meek 2004].

2.2.3 *Order-based.* Finally, a Order-based Greedy Search is a popular and effective solution to the problem of learning the structure of a Bayesian network where the initial solution L is an order of the attributes and not a network like in previous approaches. Algorithm 2 shows its pseudocode, where the function *swap* (in line 5) swaps the values $L[i]$ and $L[i+1]$ in the order L.

Algorithm 2: Order-based Greedy Search

```
1   L = Get_Order(X_1, ..., X_n, conf)
2   For a number of iterations K
3           current_sol = L
4           For each i = 1 to n-1 do
5                   L_i = swap(L, i, i+1)
6                   if score(L_i) > score(current_sol)
7                           current_sol = L_i
8           if score(current_sol) > score(L) then
9                   L = current_sol
10  Return network(L)
```

So we need to calculate the score for an order of the attributes and we know from subsection 2.1 that, in general, to obtain the score for a network is an NP-hard problem because of the exponential number of possible sets of parents (even in the bounded case). But if we know an order of the attributes, the problem can be reduced as follows

$$G = \max_{G} \sum_{i=1}^{n} sc(X_i, Pa(X_i)) = \sum_{i=1}^{n} \max_{P \subseteq \{X_j < X_i\} : |P| \leq d} sc(X_i, P), \tag{8}$$

where $X_j < X_i$ means that X_j appears before X_i in the order of the attributes. This means that we can maximize the score for every attribute independently the other ones.
In section 4, the Order-based Greedy Search algorithm will be used for learning structure of Bayesian networks using multiple data sets.

3. GENERATING INFORMED INITIAL SOLUTIONS

From section 2.2, we already know the Greedy Search algorithm and the more popular approaches using it. In all of the approaches showed can be noticed that a very important step in the algorithm is in the first line, where an initial solution is generated because if we have a good one, the solution will converge faster. We focus in the Order-based one ((in subsection 2.2.3) . At the start of this section, the most common method will be explained. After that, we explain every new proposed method to generate an initial solution for Order-based Greedy Algorithm and the Feedback Arc Set (FAS) Problem.

3.1 Random solution

This is the most common and easy-to-implement approach used. An easy way to shuffle a list L with size n is the Fisher-Yates algorithm [Knuth 1998].

3.2 DFS-based solution

Generating random solutions means having a huge search space ($n!$, where n is the number of attributes in the dataset). In order to reduce this search space we can use a graph G^* (in Figure 1 that tell us

some information about the relationships between the attributes.

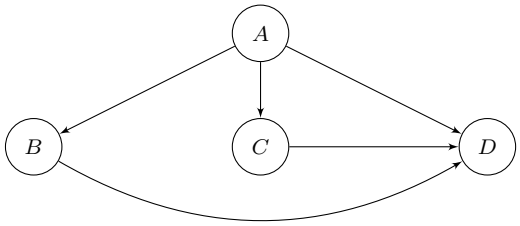


Fig. 1: Graph G^* built using equation 7

From G^*, we know that some orders does not generate some of the parent sets showed there. For instance, we could have attribute C before D, or A before B,C or D in the order generated. Having in consideration all these relations we have the following possible orders:

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

As can be noticed we have 14 possible orders given G^*, but $4! = 24$ possible orders (the number of permutations using 4 nodes) using a random approach. So we reduce the search space and also improve the possibilities that getting a good initial order because the relations are the best parent sets for each attribute. Also, the difference could be greater in problems with more attributes. Taking into consideration the previous analysis, we propose the following algorithm to generate initial solutions:

(1) Find the best parent set $Pa(X_i)$ for each attribute X_i
(2) Build a graph G^* using the relations between each attribute X_i and $Pa(X_i)$ as edges
(3) Start with an empty list L
(4) While there are nodes not visited do
    (a) Choose a random node X_k not visited
    (b) Perform a DFS on graph G^* using X_k as root and add each node X_v visited to the order L
(5) Return L

3.3 FAS-based solution

In this subsection, we first explained the generic problem and one of the variants of the Feedback Arc Set (FAS) Problem. After that, a reduction of the learning structure bayesian network problem will be explained in order to propose a new algorithm for generating initial solutions.

3.3.1 *Feedback Arc Set Problem.* The generic problem is stated as: Given a graph $G^* = (V, E)$, a set $F \subseteq E$ is called the Feedback Arc Set (FAS) if every cycle of G^* has at least one edge in F. In other words, F is the edge set that if you put if off the graph G^*, it becomes a directed acyclic graph (DAG).
There are some variants of this problem, but we only focus on finding the minimum cost FAS F from a weighted directed graph G^* defined in equation 9.

$$F_{G^*} = \min_{G^* becomes DAG} \sum_{(u,v)\in E} W_{uv}, \tag{9}$$

where W_{uv} is the weight of the edge from u to v on G^*. Although this variant becomes NP-hard, there are some approximation algorithms like the one that is below.

(1) Start with $F = \emptyset$
(2) While there is a cycle C in G do
   (a) Find the lowest weight W_{min} of all the edges $(u, v) \in C$
   (b) For each edge $(u, v) \in C$, do $W_{uv} = W_{uv} - W_{min}$
   (c) If some $W_{uv} = 0$ for some edge $(u, v) \in C$, then add (u, v) to F
(3) For each edge $(u, v) \in F$, add it to G only if it does not build a cycle on the graph
(4) Return F

3.3.2 *Algorithm for initial solutions.* First, we have the following relation for the problem:

$$G = \max_{\{Pa(X_i)\}_i, G \text{ is acyclic}} \sum_{i=1}^{n} sc(X_i, Pa(X_i)) \leq G^* = \sum_{i=0}^{n} \max sc(X_i, Pa(X_i)), \tag{10}$$

The left part is the original definition of the problem for learning structure of bayesian networks (given in equation 6) and the right part is the graph G^* showed in equation 7. In other words, the bayesian network G could be an acyclic subgraph of G^*. In order to prove that the relation is true, we need to define the weight edges as follows:

$$W_{ji} = sc(X_i, P) - sc(X_i, P \setminus \{X_j\}), \tag{11}$$

where W_{ji} is the weight of the edge that goes from X_j to X_i and P is the best parents 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. When the value is so small, it means the parent X_j could be more important. On the contrary if the weight is so big, this means that X_j is not so important in P.
Then we can prove that the problem can be reduce with the following equations:

$$\min \sum_{i=0}^{n} W_{uv} = \min \sum_{i=0}^{n} (sc(X_i, P) - sc(X_i, P \setminus X_j)) \tag{12}$$

$$\min \sum_{i=0}^{n} sc(X_i, P) - \min \sum_{i=0}^{n} sc(X_i, P \setminus X_j) \tag{13}$$

$$\max \sum_{i=0}^{n} sc(X_i, P) + \min \sum_{i=0}^{n} sc(X_i, P \setminus X_j) \tag{14}$$

$$sc(G) + \min \sum_{i=0}^{n} sc(X_i, P \setminus X_j) \approx sc(G^*) \tag{15}$$

This means that the score of DAG G plus the score of the deleted arcs of G^* give an approximation, and also that G could be a subgraph of G^*. Finally, as we already know how to become G^* to a DAG G using the FAS solution given in 3.3.1, we propose the new algorithm for getting an initial solution using it as follows:

(1) Find the best parent set $Pa(X_i)$ for each attribute X_i
(2) Build the graph G^* using all the relationships $X_j \implies X_i, X_j \in Pa(X_i)$ and put its score as W_{ji}
(3) Find the minimum cost FAS F
(4) Delete all edges $(u, v) \in F$ from G^* to obtain a network G
(5) Return a topological order of G

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

4. EXPERIMENTS AND RESULTS

After implementing all the approaches in section 3, some experiments were done in order to compare them. First, we show the setup for all experiments and then show the final results.

4.1 Experiments Setup

Table I shows the characteristics of each data set used in the experiments. Also, the values for each

Dataset	n (#attributes)	N (#instances)
Census	15	30168
Letter	17	20000
Image	20	2310
Mushroom	23	8124
Sensors	25	5456
SteelPlates	28	1941
Epigenetics	30	72228

Table I: Data sets characteristics

general parameter are given below.

—Maximum number of parents (d): 3

—Number of iterations in Greedy Search (K): 100

—Scoring function used: Bayesian Criterion Information (BIC)

—Number of different initial solutions generated (S): 100

The parameter S is used to have better statistics about using every approach compared with the others. It is possible to generate such a number of different solutions with every approach because there are a lot of possible orders given a graph and also $n!$ possible orders in the random approach.

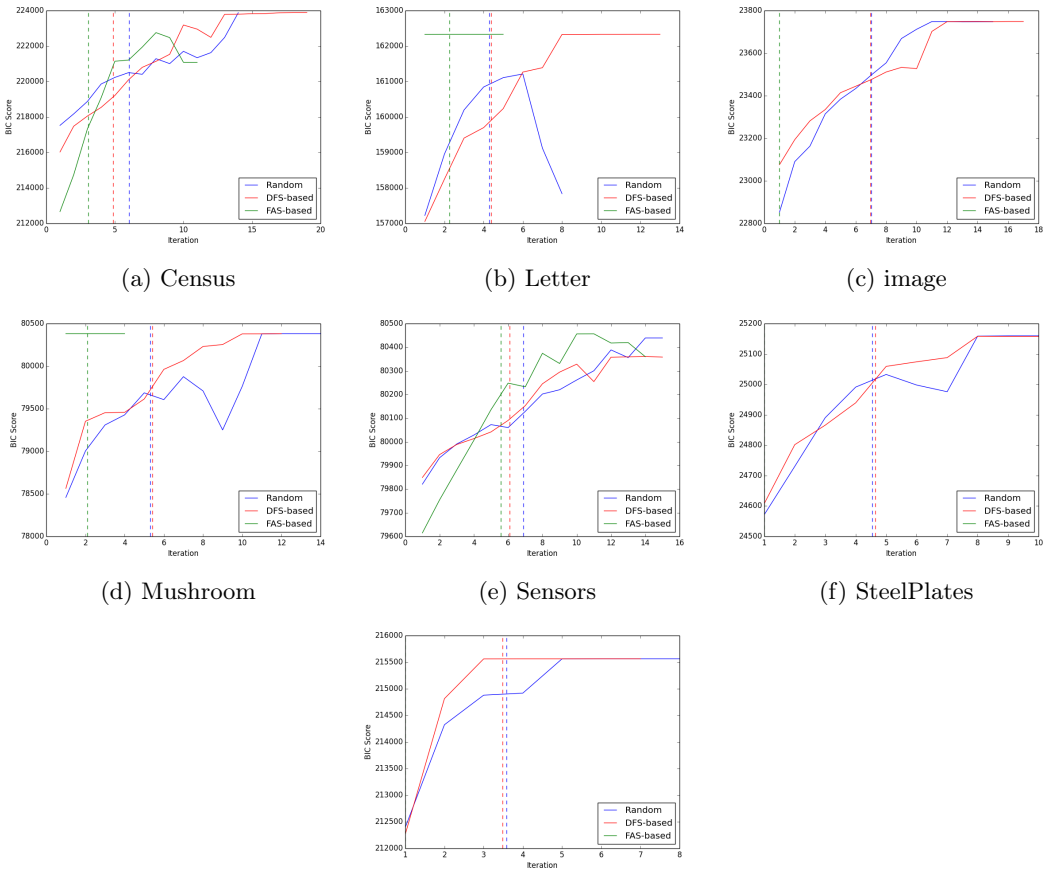
4.2 Results

After running the three versions explained in section 3 for each data set, we compare the score for the best network found, the percentage of solutions that converge to the best score and the average number of iterations. The results are showed in Table II.

Finally, Figure 2 shows the converge curve for each data set using each approach.

Dataset	Approach	Best Score	% Solutions	Avg. It.
Census	Random	-223893.767	**9.00**	6.03 ± 2.55
	DFS-based	-223893.767	2.00	4.87 ± 2.67
	FAS-based	-223893.767	3.00	**3.07 ± 2.09**
Letter	Random	-162332.183	5.00	4.30 ± 1.57
	DFS-based	-162332.183	11.00	4.38 ± 1.82
	FAS-based	-162332.183	**51.00**	**2.25 ± 0.83**
Image	Random	-23749.089	3.00	7.02 ± 2.74
	DFS-based	-23749.089	6.00	6.96 ± 3.16
	FAS-based	-23749.089	**100.00**	**1.00 ± 0.00**
Mushroom	Random	-80384.174	3.00	5.30 ± 2.56
	DFS-based	**-80384.107**	3.00	5.42 ± 2.31
	FAS-based	-80384.174	**74.00**	**2.11 ± 0.62**
Sensors	Random	-80545.820	1.00	6.89 ± 2.53
	DFS-based	**-80542.640**	1.00	6.10 ± 2.55
	FAS-based	-80545.820	1.00	**5.58 ± 2.26**
SteelPlates	Random	-25159.708	3.00	4.54 ± 1.68
	DFS-based	-25159.706	5.00	4.64 ± 1.64
	FAS-based	**-25157.926**	**100.00**	**1.00 ± 0.00**
Epigenetics	Random	-215563.134	94.00	3.58 ± 1.19
	DFS-based	-215563.134	90.00	3.48 ± 1.18
	FAS-based	-215563.134	**100.00**	**1.00 ± 0.00**

Table II: Best score obtained, Percentage of solutions that converge to best score (% Solutions), and Average number of iterations (Avg. It.) using each approach (best values in bold)



(a) Census (b) Letter (c) image

(d) Mushroom (e) Sensors (f) SteelPlates

(g) Epigenetics

Fig. 2: Converge curve for datasets, vertical lines are in avg. number of iterations

5. CONCLUSIONS AND FUTURE WORKS

We conclude that:

—Although all the approaches converge to the same score in most cases, the new methods could converge to a better solution

—The new methods converge in less or equal number of iterations, in average

—DFS-based approach improve, in some cases, the percentage number of solutions that converge to the maximum compared to the random approach, but the difference is not significantly

—DFS-based approach converges to a better solution as in mushroom and sensors dataset

—FAS-based approach take less iterations than the other ones and take significantly less iterations to converge (in average)

—FAS-based approach has a greater percentage of good initial solutions that has the best score, in lot of cases they do not need to perform a local search

—The new methods explained in this article could be used in other types of Greedy Search doing some modifications

—The scoring function could be changed to another one in order to compare robustness between them and BIC score

—Experiments could be done with bigger datasets (more than 60 attributes) in order to compare the efficiency of the methods

REFERENCES

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.

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.

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.

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.