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 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. 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 [Jensen 2001]. 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.

There are two common approaches to learning Bayesian networks from a given data set: constraint-based and score-based. Constraint-based methods learn conditional independence statements among the variables using statistical tests on the data set and are more useful in data sets with missing values. 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]. However, since we are only

concerned on data sets without missing values, we will not refer to these any further. On the other hand, score-based 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].

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 methods on data sets without missing values, but our technique can be exploited by any local search procedure for learning Bayesian network structures that will be explained later. 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 the search-and-score approach learning 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

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

that it is the graph with edges of the form $X_j \to X_i$, where $X_j \in Pa(X_i)$, and $Pa(X_i)$ is the parent set that maximizes the score $sc(X_i, Pa(X_i))$, also called *best parent set*. 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 Algorithm

The Greedy Search algorithm is a popular heuristic method used to find an approximate solution for the Bayesian network learning. 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 algorithm.

Algorithm 1: Greedy Search

```
1  GreedySearch( Dataset D ) : return a BN G
2      G = Initial_Solution(X_1, ..., 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 algorithm is to generate an initial solution (e.g. a random generated one). After that, for a number of iterations K, the algorithm explores the search space and selects the best neighbor of the best solution until that moment. It then updates the lower bound on the score in case an improving solution has been found. 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). The algorithm terminates by returning the best solution found. 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. The initial solution is usually obtained by randomly generating a DAG, using one of the many methods available in the literature [Grzegorczyk and Husmeier 2008].

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). 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.* Given a topological ordering $<$ of the attributes, the problem of learning a Bayesian network simplifies to

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

This means that if an optimal ordering over the attributes is known, an optimal DAG can be found by maximizing the local scores independently. Since an exhaustive approach would take $O(2^n)$ time, the local optimization is usually constrained to the space of parents of cardinality at most a certain parameter d. For the BIC scoring function (as for others), limiting the cardinality of parent sets does not affect the optimality, as the number of maximal parents in the optimal BIC-maximizing DAG can be shown to be at most $\log N$ (so it suffices to select $d \geq \log N$).

Order-based Greedy Search is a popular and effective solution to the problem of learning the structure of a Bayesian network, which consists of searching the spaces of topological orderings of variables. 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 [Tessyer and Koller 2005]. 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 current solution.

Algorithm 2: Order-based Greedy Search

```
1   OrderBasedGreedySearch( Dataset D ) : return a BN
2       L = Get_Order(X_1, ..., 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)
```

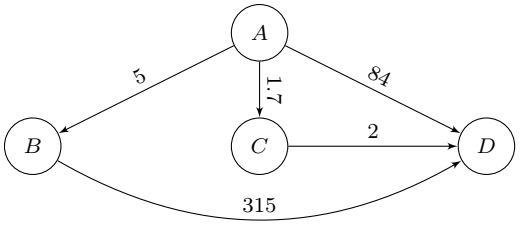
The standard approach to generate initial solutions is to sample a permutation of the attributes uniformly at random by some efficient procedure such as the Fisher-Yates algorithm [Knuth 1998; Tessyer and Koller 2005]. In the next section, we propose new strategies to informed generation of topological 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

It can be noticed that generating random solutions means having a huge search space ($n!$) for datasets with $n > 10$, but this search space can be reduced using the graph $\overline{G}$ (defined in equation 4). 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.
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	CDBA
ACDB	DABC
ADBC	DACB
ADCB	DBAC
BDAC	DBCA
BDCA	DCAB
CDAB	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:

Algorithm 3: DFS-based method

```
1   DFS_Based( Graph Ḡ ) : return Order L consistent with Ḡ
2       L = empty list
3       While there are nodes not visited on Ḡ do
4           X_k = random not visited node
5           DFS(Ḡ, X_k, L)
6       Return L
```

Where DFS performs a depth-first search on $\overline{G}$ using X_k as root and add each node X_v visited to the order L.

3.2 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.

3.2.1 *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 DAG [Demetrescu and Finocchi 2001].
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 6.

$$F_G = \min_{G-F \text{ is a DAG}} \sum_{(u,v)\in E} W_{uv}, \tag{6}$$

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.

Algorithm 4: FAS approximation

```
1  MinimumCostFAS( Graph G ) : Return FAS F
2      F = empty set
3      While there is a cycle C on G do
4          W_min = lowest weight of all edges in C
5          For each edge (u,v) ∈ C do
6              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
```

3.2.2 *Algorithm for initial solutions.* A possible problem with the previous method (DFS-based) is that uses $\overline{G}$ and generate orders consistent with it using a random node as root, but does not consider that some edges are more important 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 important than others in the best parent set of a variable. This importance of an edge could be represented as follows:

$$W_{ji} = sc(X_i, P) - sc(X_i, P \setminus \{X_j\}), \tag{7}$$

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 because P maximizes the score for X_i. 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. For instance, graph $\overline{G}$ in Figure 1a shows that C is more important than B and A as a parent of D because that edge has a lower value.
We can now describe our second heuristic for generating initial solutions, based on the minimum cost FAS problem:

Algorithm 5: FAS-based method

```
1  FAS_Based( Graph Ḡ ) : return Order consistent with Ḡ
2      Add weights W_ji to edges in Ḡ
3      F = MinimumCostFAS( Ḡ )
4      G = Ḡ − F
5      Return topological( 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. The greedy search was ran with a maximum number of parents ($d = 3$), a limit of 100 iterations ($K = 100$), and 1000 restarts ($S = 100$). We used the BIC scoring in all experiments and the parent sets were calculated by exhaustive search in all cases. The choice of the parameter S was adopted so as to have enough data to produce statistically significant results while still being able to ran all the experiments in a restricted amount of time.

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

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
Alarm	37	1000
Spectf	45	267
LungCancer	57	27

Table I: Data sets characteristics

We evaluated three different initialization strategies: random, which randomly generates a topological ordering, DFS-based, which uses the DFS heuristic, and FAS-based, which uses the heuristic based on the minimum cost FAS. For each approach, we compared the score of the best network found, the percentage of solutions that converged to the best score and the average number of iterations that local search took to converge. The results are shown in Table II. The experiment shows that in most

Dataset	Approach	Best Score	% Solutions	Avg. It.
Census	Random	**-212186.787**	0.30	7.26 ± 2.90
	DFS-based	-212190.051	0.20	5.90 ± 2.61
	FAS-based	-212191.642	**51.85**	**3.28 ± 1.67**
Letter	Random	-138652.663	4.30	6.07 ± 2.50
	DFS-based	-138652.663	**6.70**	5.75 ± 2.35
	FAS-based	-138652.663	0.34	**2.24 ± 0.96**
Image	Random	**-12826.077**	0.10	7.59 ± 2.71
	DFS-based	-12829.104	**0.40**	7.10 ± 2.47
	FAS-based	-12829.104	0.14	**5.05 ± 1.72**
Mushroom	Random	**-55513.382**	0.10	7.59 ± 2.76
	DFS-based	**-55513.382**	0.10	7.75 ± 2.58
	FAS-based	-55574.713	**0.20**	**4.65 ± 1.63**
Sensors	Random	**-62062.128**	0.10	9.22 ± 2.94
	DFS-based	-62083.211	0.10	9.65 ± 3.12
	FAS-based	-62074.876	**0.11**	**5.17 ± 2.24**
SteelPlates	Random	-13336.138	0.10	8.96 ± 3.43
	DFS-based	**-13332.908**	0.10	9.30 ± 3.38
	FAS-based	-13341.731	**0.30**	**7.77 ± 2.24**
Epigenetics	Random	-56873.763	0.10	5.89 ± 2.67
	DFS-based	**-56868.869**	**0.20**	6.42 ± 2.47
	FAS-based	**-56868.869**	0.10	**5.33 ± 2.28**
Alarm	Random	-13218.222	0.10	10.92 ± 3.24
	DFS-based	**-13217.970**	0.10	**4.32 ± 2.32**
	FAS-based	-13220.548	**0.80**	6.34 ± 1.74
Spectf	Random	-8176.805	1.00	7.20 ± 2.17
	DFS-based	**-8172.366**	1.00	7.86 ± 2.49
	FAS-based	-8172.507	1.00	**2.27 ± 1.11**
LungCancer	Random	**-711.228**	1.00	5.46 ± 1.78
	DFS-based	-711.358	1.00	5.02 ± 1.50
	FAS-based	-711.392	1.00	**2.73 ± 1.79**

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)

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 obtain better scores than FAS-based one. Moreover, the percentage of solutions obtained with best score is always greater using FAS-based method independently of the number of attributes of

the dataset (n), but as stated before, these best scores are not always the best possible ones. 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, it means that these initial solutions are close to a local maxima, even for the biggest datasets. So in general, the new methods work better with bigger datasets than smaller ones.

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 greedy Bayesian network structure search methods.
Experiments with benchmark data sets showed 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.
Also these methods could be adapted to be used for Structure-based and Equivalence-based methods because they does not affect the search method, but only the initial solution generated. The DFS heuristic change from saving a list to saving the DFS directed tree built and the FAS heuristic return the DAG obtained from removing the FAS instead of returning its topological order. Our heuristics can also be exploited by branch-and-bound solvers that find optimal solutions. These 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.

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.

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.

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.