Heuristics for Initializing Order-Based Bayesian Network Structure Learning

Walter Perez Urcia
and
Denis Deratani Mauá

Universidade de São Paulo, Brasil
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 models that represent efficiently probability distributions over the atributes of a data set. They are defined by two components:

—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.
—The conditional probability distributions of the atributes given their parents. The latter are defined by the network's structure.

Formally, a Bayesian network is defined by:

$$G = (V, E),$$

where $V = \{X_1, X_2, \dots, X_n\}$ is the set of attributes and

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

where $Pa_G(X_i)$ is the set of atributes that are parents of X_i in G. This definition shows that having less parents for an atribute decrease the number of parameters required in the specification, but learning Bayesian networks from data is a NP-hard problem [David M. Chickering 2004] even when the in-degree (maximum number of parents) is bounded, for this reason, the common approach to solve this problem is to use local search methods, commonly using a scoring function, like the bayesian information criterion (BIC) [Cover and Thomas 1991] or the minimum description length (MDL) [Wai Lam 1994] or Bayesian Dirichlet score (BD) [D. Heckerman and Chickering 1995].

In this article we propose new heuristics for generating good initial solutions to be fed into local search methods. We focus on order-based methods, but our technique can be exploited by any Bayesian net search procedure. We propose a new methods for generating informed initial solutions motivated by solutions of the Feedback Arc Set Problem (FASP). Our experiments show that using these new methods it is possible to learn better networks from data sets.

The article is structured as follows: We begin in Section 2 explaining the Greedy Search algorithm. In Section 3, we describe the new algorithm for gener3ating initial solutions. Section 4 shows the experiments using both approaches and comparing them (in time and scoring) 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 learning Bayesian network problem and then reduce it to one formula in order to be used as a heuristic score. Finally we explain some popular methods to solve them.

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:

$$sc(G, D) = F(G) + \varphi(N) \times P(G),$$

where N is the size of the data set D, $F(G)$ is a data fitness function, $\varphi(N)$ is a function of data size and $P(G)$ measures the model complexity of G.
For instance, using the Bayesian information criterion (BIC) as scoring function we have:

$$sc(G, D) = BIC(G, D) = LL(G) + \frac{\log N}{2} size(G),$$

where

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

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

N is the number of instances in the data set, n the number of atributes, N_{ijk} the number of instances where attributes has the values i, j and k (the same way for N_{ij}). Finally, Ω_i is the size of the set of possible values for the attribute X_i. Most scoring functions, BIC included, has the property to be decomposable, so we can reduce the problem to the following formula:

$$G = \max_{G} \sum_{i=1}^{n} sc(X_i, Pa(X_i)),$$

where $Pa(X_i)$ is the set of parents of attribute X_i in G. In other words, to get a better network, we need to pick a configuration that has the maximum sum of scores among their attributes. But the main problem with searching $Pa(X_i)$ is that we have 2^{n-1} possible set of parents for each attribute. It means, an exponential number of possible sets based on the number of attributes in the data set. In order to avoid this problem is added a new constraint d, where d is the maximum number of parents for each attribute, so we have only 2^d possible sets. This means $|Pa(X_i)| \leq d$ for all attributes.

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   $L = Initial\_Solution(X_1, \ldots, X_n, conf)$
2   $best\_score = score(L)$
3   For a number of iterations $K$ or until $L$ converges do
4           $best\_neighbor = find\_best\_neighbor(L)$
5           if $score(best\_neighbor) > score(L)$ then
6                   $L = best\_neighbor$
7   Return $L$
```

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 or until the solution converges (the best solution's score does not improve) 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. 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, 4 and 2 depending on its own way to generate an initial solution, its search space and the scoring function used, respectively.
For example, in the structure-based version the initial solution is a network, possibly random generated, and its 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.
In the same way, the equivalence-based version [?] uses the equivalence relation between networks to obtain new neighbors. This relation two networks L_1 and L_2 are equivalent if they have the same probability distribution.

Finally, a Order-based Greedy Search is a popular and effective solution to the problem of learning the structure of a Bayesian network. Algorithm 2 shows its pseudocode, where the function *swap* (in line 6) swaps the values $L[i]$ and $L[i+1]$ in the order L.

Algorithm 2: Order-based Greedy Search

```
1   $L = Get\_Order(X_1, \ldots, X_n, conf)$
2   $best\_score = score(L)$
3   For a number of iterations $K$ or until $L$ converges do
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 $L$
```

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:

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

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

As stated in Section 1, the solutions proposed in this article are motivated by the Feedback Arc Set Problem (FASP) that will be explained at the start of this section. After that, we explain every new method to generate an initial solution for Order-based Greedy Algorithm.

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 If we focus on unweighted undirected graphs and we want to find a FAS with minimum size, the problem becomes NP-hard, but there are some approximation algorithms like the one that is below.

(1) Select a random node $X \in V$
(2) Do a depth-first search (DFS) on G using X as root
(3) Put all the back edges encountered by the DFS in F. A back edge is the one that makes a cycle with size greater than two
(4) Return F

In the same way, another NP-hard problem is to find a minimum weight FAS from a weighted undirected graph G, but again we have some approximation algorithms in order to solve this problem.

(1) Negate all edge weights
(2) Find the minimum spanning tree (MST) M performing Kruskal algorithm
(3) Edges that are not in M are included in F
(4) Return F

The variants of the problem where the graph is directed (with weighted or unweighted edges) will not be explained in this article.

3.2 Informed initial solutions

From section 2.2, we already know the generic Greedy Search algorithm and the more popular approaches using it. It 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. So if we now focus only in the Order-based one showed in Algorithm 2, we first explain the common approach and then we will propose two new methods for generating initial solutions using information from data motivated by the solutions in 3.1.

3.2.1 *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, showed in Algorithm 3.

Algorithm 3: Fisher-Yates shuffle algorithm

```
For $i$ in $n-1$ to 1 do
    $j = random(0, i)$
    Swap $L[i]$ and $L[j]$
Return $L$
```

3.2.2 *Using FAS with unweighted edges.* Remembering the definition of a Bayesian network given in 2.1 we have:

$$G = \max_G \sum_{i=1}^{n} sc(X_i, Pa(X_i))$$

Although the Bayesian network G has the maximum sum of the scores of every field X_i given their parents $(Pa(X_i))$, this do not implies that every field have their best parents because the graph builded with them will probably not be a network as it can contain cycles. Besides that, calculating the best parents is computationally costly unless we bound the size of the parent set as mentioned in 2.1. Knowing those concepts and the approximation algorithm for unweighted FAS problem, we can propose the following algorithm:

(1) For each node X_i : Find the $Pa(X_i)$ set with maximum score
(2) Build the graph S using all the relationships $X_j \implies X_i, X_j \in Pa(X_i)$
(3) Choose a node X_k from S
(4) Start with an empty graph G
(5) Perform a DFS on using X_k as root with the following characteristics:
—Add the direct edge e that encounters to G, unless e builds a cycle
—If e builds a cycle, then add e to the feedback arc set F
(6) For each edge in F: Invert its direction
(7) Add edges in F to the DAG G
(8) Return the topological order of G

This solution is similar to the one for the unweighted FAS problem using the graph S, except that we use the tree builded by the DFS to get a network and then add the Feedback Arc Set with their directions inverted. Finally, as we want a field order, we return the topological order of the network G.

3.2.3 *Using FAS with weighted edges.* In this version, the edges have its weight defined as follows:

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

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
The algorithm for getting an initial solution is very similar to the previous one and it uses the main idea of the weighted FAS approximation algorithm.

(1) For each node X_i : Find the $Pa(X_i)$ set with maximum score
(2) Build the graph S using all the relationships $X_j \implies X_i, X_j \in P = Pa(X_i)$ and put its score as $-W_j i$

(3) Make the graph S undirected and perform Kruskal algorithm for getting the MST M
(4) Choose a node X_k from M
(5) Start with an empty graph G
(6) Perform a DFS on M and add the directed edges on G
(7) Return the topological order of G

In step 3, we have to make the graph S undirected because Kruskal algorithm can be used only in undirected weighted graphs. Finally, we perform a DFS to get a directed tree with root X_k and we only have to return its topological order.

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 3.2, 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)	Avg. #vals/var
Census	13	45000	7.5
Dataset 2	n_2	N_2	m_2
Dataset 3	n_3	N_3	m_3
Dataset 4	n_4	N_4	m_4

Table I: Data sets characteristics

general parameter are given below.

—Maximum number of parents (d): 4
—Number of iterations in Greedy Search (K): 100
—Scoring function used: Bayesian Criterion Information (BIC)

Finally, as in subsections 3.2.2 and 3.2.3 we have to choose a field as a root for performing the DFS, we have n possible different DAGs and then n possible different orders and use each one as an initial solution for Greedy Search. Also, for the random version, we generate n different random orders.

4.2 Results

After running the three versions explained in 3.2 for each data set, we compare the score for the best network found, the CPU Time to generate the initial solution and the CPU Time for getting the final network. Tables II and III show them respectively.

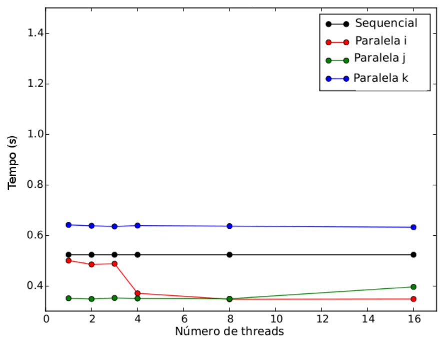
Dataset	Random Sol.	Unweighted Sol.	Weighted Sol.
Census	-165350	-14683	m_1
Dataset 2	n_2	N_2	m_2
Dataset 3	n_3	N_3	m_3
Dataset 4	n_4	N_4	m_4

Table II: Maximum cost using each approach

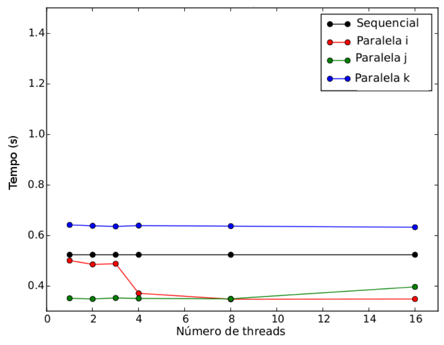
Dataset	Random Sol.	Unweighted Sol.	Weighted Sol.
Census	10.2 / 72.14	93.51 / 48.89	m_1
Dataset 2	n_2	N_2	m_2
Dataset 3	n_3	N_3	m_3
Dataset 4	n_4	N_4	m_4

Table III: CPU Time (in seconds) for initializing and total using each approach

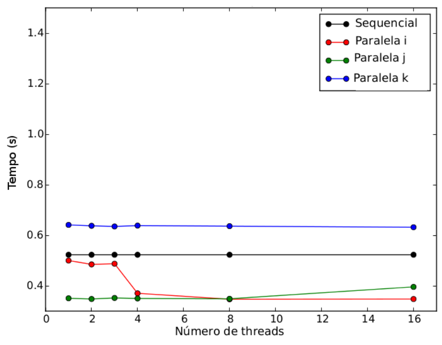
Finally, Figure 1 shows the converge curve for each data set using each approach.



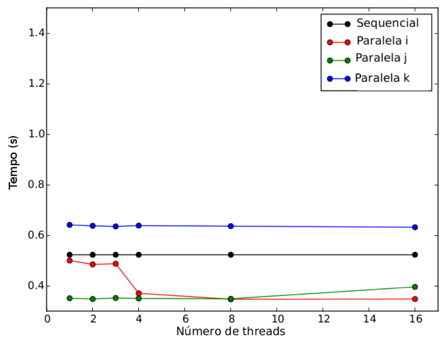
(a) Census



(b) Dataset #2



(c) Dataset #3



(d) Dataset #4

Fig. 1: Converge curve for datasets

5. CONCLUSIONS AND FUTURE WORK

We conclude that:

—Using the random initial solution take more iterations to find a converged solution than the new methods and also more CPU time

—The unweighted approach gives best networks than the other ones

—Initial solutions using the unweighted approach are the most time consuming to be generated and the random ones the least.

—A higher number of attributes implies more CPU time for the new methods with a significant difference compared to the random approach

—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 20 fields) in order to compare the efficiency of the methods

REFERENCES

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

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

David M. Chickering, David Heckerman, C. M. Large-Sample Learning of Bayesian Networks is NP-Hard. 5 (1): 1287–1330, 2004.

Wai Lam, F. B. Learning Bayesian Belief Networks. An approach based on the MDL principle. 10 (4): 31, 1994.