# Better Initialization Heuristics for Order-based 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.** An effective approach for learning Bayesian network structures is to perform a local search on the space of topological orderings, followed by a systematic search of compatible parent sets. Typically, the local search is initialized with an ordering generated uniformly at random. This can lead to poor local optima, slow down convergence and hurt the performance of the method. In this work we develop two informed heuristics for generating initial solutions to order-based structure learning. Both heuristics rely on the solution of a relaxed version of the problem in which cycles are permitted. The heuristics remove less relevant arcs of the relaxed solution in order to produce a directed acyclic graph, which is then used to produce topological orderings. Experiments with a large collection of real-world data sets demonstrate that our heuristics increase the quality of the solutions found with a negligible overhead.

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

Keywords: Bayesian Networks, Model Selection, Local Search, Parent Set Selection

## 1. INTRODUCTION

Bayesian Networks are space-efficient representations of multivariate probability distributions [Pearl 1988]. They are defined by two components: an acyclic directed graph (DAG) encoding the independences among the variables, and a collection of local conditional probability distributions for each variable given its parents.

Manually specifying a Bayesian network is a difficult task, and practitioners often resort to *learning* the model from data, that is, to inferring a DAG and the necessary conditional probabilities from a dataset of observations. Bayesian network learning is usually decomposed into two steps: first a DAG is obtained (structure learning), and then the numerical parameters (i.e., the conditional probabilities) are estimated for a fixed DAG (parameter learning). In this work, we focus on structure learning with complete data (i.e., the dataset does not contain missing or corrupted values).

A common approach to structure learning consists in associating every DAG with a polynomial-time computable score value, and optimizing over the space of DAGs [Cooper and Dietterich 1992; Lam and Bacchus 1994; Margaritis 2003; Tessyer and Koller 2005]. Typical score functions reward DAGs with high probability of generating the dataset (i.e., the data likelihood) while penalizing the complexity of the model (i.e., the number of parameters). Some examples are the Bayesian Information Criterion (BIC) [Schwarz 1978], the Akaike Information Criterion (AIC) [Akaike 1974], Minimum Description

We thank Mauro Scanagatta for providing us with the datasets. This work was developed with the help of computing resources available by the Superintendência de Informação da Universidade de São Paulo. The first author was partially supported by CAPES. The second author was partially supported by São Paulo Research Foundation (FAPESP) grant #2016/01055-1.

Length (MDL) [Suzuki 1996] and the Bayesian Dirichlet score (BD) [Heckerman et al. 1995]. An alternative approach is to learn the DAG by performing multiple conditional independence tests while enforcing acyclicity [Spirtes and Meek 1995; Cheng et al. 2002]. Although both approaches are consistent (for common score functions) in that they recover the "true" DAG (if one exists) given infinite data and computational resources, testing for independence can be computationally demanding, is highly sensitive to the significance level used, and often generates many false correlations; this last problem is particularly important in small-sample or high-dimensional datasets [Heckerman et al. 1999]. There are also hybrid approaches that employ statistical testing to generate good initial solutions for a score-based approach [Tsamardinos et al. 2006]. This hybrid approach alleviates the false positive problem at an increased computational cost.

Chickering [1996] showed that score-based structure learning is NP-hard for a wide class of score functions. This result was later strengthened to show NP-hardness for large-sample domains [Chickering et al. 2004]. Note that the latter includes the case of hybrid learning methods. Those results support the development of approximate methods that can scale for large domains.

There is vast literature on combinatorial optimization techniques to solve structure learning. The most successful approaches include integer- and constraint-programming [Bartlett and Cussens 2015], systematic search [de Campos and Ji 2011; Yuan and Malone 2013], and evolutionary algorithms [Larranaga et al. 1996; Larranaga et al. 2013]. While these approaches generate high quality solutions, they are computationally expensive in time and memory requirements.

An effective approach to structure learning in large domains is to perform a local search, for example, in the space of DAG structures or score-equivalent graphs [Friedman et al. 1999; Chickering 2002]. Based on a common observation that score-based structure learning decomposes into smaller independent problems when a topological ordering is imposed [Buntine 1991], Tessyer and Koller [2005] developed a local search method that continues to be the state-of-the-art in structure learning for large domains [Scanagatta et al. 2015]. The method performs a 1-neighborhood local search in the space of topological orderings, where two orderings are neighbors if they differ in at most one position.

As with most local search approaches, a good initialization of order-based structure learning is crucial to the quality of the solution found. Typically, the search is initialized with an ordering sampled uniformly at random. While this allows good coverage of the search space, it can lead to poor local optimal and slow down convergence. Although this issue can be alleviated by employing more sophisticate techniques for escaping local optima [Glover 1989; Granville et al. 1994; Elidan et al. 2002], this usually adds a significant computational overhead, and reduces scalability. An alternative, less demanding remedy is to initialize the search in high-scoring regions.

In this work we develop two new initialization heuristics for order-based Bayesian network structure learning with complete data. The heuristics rely on the (cyclic) graph obtained as the solution of the relaxed version of the problem where cycles are permitted. The first heuristic generates orderings by performing a depth-first traversal of that graph, adopting the in-degree of children as tie breaking criterion. Although this heuristic biases the search away from regions which may be sub-optimal, it ignores important information available at the score of the relaxed solution. Our second heuristic refines the first one by selecting high scoring orderings among the ones that are consistent with the relaxed version solution. We do this by reducing the problem to a minimum-cost feedback arc set problem, which is the problem of transforming a weighted cyclic directed graph into a DAG by removing the minimum weight of edges [Demetrescu and Finocchi 2003]. Experiments with real-world high-dimensional datasets show that our heuristics improve the quality of state-of-the-art order-based local search at an almost negligible additional computational cost, even when the relaxed problem is solved only approximately.

This work extends our previous work on initialization heuristics [Perez and Mauá 2015] with an

improved variant of the first heuristic, and a deeper empirical analysis that includes more and larger datasets and an investigation of the effects of approximately solving the relaxed problem.

The rest of this document is structured as follows: we begin in Section 2 explaining the basic concepts of score-based Bayesian network structure learning. Then in Section 3 we describe the methods for parent set selection, which is a necessary sub-step of local search approaches. Order-based structure learning is reviewed in Section 4. In Section 5 we describe our heuristics for generating initial solutions. Section 6 contains the experimental results. Finally, in Section 7, we state some conclusions of the main results and a discussion on future work.

## 2. LEARNING BAYESIAN NETWORKS

A Bayesian network specification contains a DAG $G = (V, E)$, where $V = \{X_1, X_2, \ldots, X_n\}$ is the set of (categorical) random variables, and a collection of conditional probability distributions $P(X_i \mid Pa_i^G)$, $i = 1, \ldots, n$, where $Pa_i^G$ are the parents of $X_i$ in $G$ and $P(X_i|\emptyset) = P(X_i)$. The Bayesian network is assumed to induce a joint probability distribution over all the variables through the equation

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid Pa_i^G)\,,$$

The number of parameters required to specify a Bayesian network with DAG $G$ is $size(G) = \sum_{i=1}^{n}(r_i - 1)\prod_{X_j \in Pa_i^G} r_j$, where $r_k$ denotes the number of states variable $X_k$ can assume.

A *score function* $sc(G)$ assigns a real-value to any DAG $G$ indicating its goodness in representing a given dataset.[1] For example, the BIC score function (which we use in our experiments), is given by

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

where $LL(G) = \sum_{i=1}^{n}\sum_k\sum_j N_{ijk}\log\frac{N_{ijk}}{N_{ij}}$ is the data loglikelihood, $N_{ijk}$ the number of instances where variable $X_i$ takes its $k$th value and its parents take the $j$th configuration (for some arbitrary fixed ordering of the configurations of the parents' values), and similarly for $N_{ij}$. We require that the score function be *decomposable*, meaning that it can be written as $sc(G) = \sum_{i=1}^{n} sc_i(Pa_i^G)$. Most score functions used, including BIC, are decomposable [Chickering and Meek 2002].

Given a score function, the score-based Bayesian network structure learning problem is to find an optimal DAG $G^*$ such that

$$sc(G^*) = \max_{<} \sum_{i=1}^{n} \max_{\mathbf{Y}\in\{X_j < X_i\}} sc_i(\mathbf{Y})\,, \tag{1}$$

where the first optimization is performed over complete orderings $<$ of the variables. The constraint that parents of a variable be strictly smaller than a variable in a given ordering ensures the graph is a DAG.

The structure learning problem is usually performed in two steps: first, an exact or approximate technique is used to obtain a list of candidate parent sets $\mathbf{C}_i$ for each variable $X_i$ (parent set selection); then, these candidate parent sets are used to search for a graph $\tilde{G}$ such that

$$sc(\tilde{G}) = \max_{<} \sum_{i=1}^{n} \max_{\mathbf{Y}\in\mathbf{C}_i^{<}} sc_i(\mathbf{Y})\,, \tag{2}$$

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

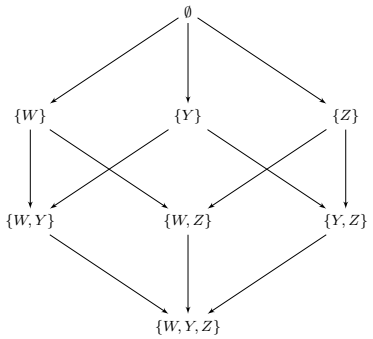



Fig. 1: Parental graph of a variable $X$ with respect to domain $X, Y, Z, W$.

where $\mathbf{C}_i^{<} = \{S \in \mathbf{C}_i \mid \forall X_j \in S, X_j < X_i\}$ are the parent sets in $\mathbf{C}_i$ consistent with the order $<$. The quality of the learned structure depends on the quality of both steps [Yuan and Malone 2013; Bartlett and Cussens 2015; Scanagatta et al. 2015].

Under the BIC and MDL score functions, the best parent set of any variable has at most $\log N$ variables (where $N$ is the size of the dataset) [de Campos and Ji 2011]. Thus, parent set selection can be solved in time $O(n^{1+\log N})$. This naive approach however is impractical even for moderately large domains and datasets [Scanagatta et al. 2015]. Koivisto [2006] showed that finding the best parent set of a variable is LOGSNP-hard even to approximate for the MDL, BIC and AIC score functions.[2]

## 3. PARENT SET SELECTION

The theoretical hardness of parent set selection justifies the use of approximate techniques. We now review the two most commonly used procedures for selecting parents sets (sequential selection and greedy selection), as well as a recently developed procedure by Scanagatta et al. [2015].

These techniques can all be seen as heuristic search in the parental graph (lattice diagram) of a variable $X_i$, which is the graph where each node is a subset of $V \setminus \{X_i\}$, and an arc connects a subset/node $\mathbf{Y}$ to a subset/node $\mathbf{Z}$ if $\mathbf{Z} = \mathbf{Y} \cup \{X_i\}$ for some variable $X_i$ [Yuan and Malone 2013]. Each node $\mathbf{X}$ in the parental graph of $X_i$ is associated with a score $sc_i(\mathbf{X})$. Figure 1 depicts the parental graph of variable $X$ in a small domain containing variables $X, Y, Z, W$. Note that the parental graph of an $n$-dimensional domain has $2^{n-1}$ nodes; thus, exhaustive search is impracticable for large domains.

Algorithm 1 shows a pseudo-code for a general search procedure on the parental graph. The functions $h_i$ and $g_i$ denote the heuristic and actual score of a parent set; *open* is a list of parent sets to be explored sorted by the values $h_i(\mathbf{Y})$, and *closed* is a list of parent sets already explored (those for which $g_i(\mathbf{Y})$ has been computed). The techniques differ in terms of the initialization of *open* and *closed* (line 1), and in the definitions of $h_i$ and $g_i$.

### 3.1 Sequential Selection

Sequential Selection is adopted by the state-of-the-art exact learning algorithms [Yuan and Malone 2013; Bartlett and Cussens 2015]. This technique consists in performing a breadth-first search in the parental graph up to a given depth $k$. The list *closed* is initialized empty, and *open* is initialized containing the empty set (hence not empty). The function $g_i$ is set as the local score function $sc_i$, and $h_i(\mathbf{Y}) = -|\mathbf{Y}|$. Pruning rules can be applied to detect suboptimal paths while conducting the search [de Campos and Ji 2011]. The worst-case running time of the procedure is $O(n^k)$. Sequential selection

[2]The class LOGSNP contains problems that are believed to exhibit runtime $O(s^{\log s})$, where $s$ is the size of the input, yet are thought not be NP-hard (note that $(n^{1+\log N})$ is sub-exponential).

**Algorithm 1:** Parent Set Selection

**Input** : Variable $X_i$, domain $V \setminus \{X_i\}$
**Output**: Score cache *closed*

```
Initialize closed and open
while open is not empty do
    Find Y ← arg max_{Y∈open} h_i(Y)
    for X_j ∈ V \ {X_i} do
        if Y ∪ {X_j} is not in open and closed then
            Calculate h_i(Y ∪ {X_j})
            Add Y ∪ {X_j} to open
        end
    end
    Calculate g_i(Y)
    Remove Y from open
    Add Y to closed
end
```

is optimal given a sufficiently large $k$; however, when working with large datasets (e.g., $n > 100$), it is necessary to set $k$ to low values (e.g. $k = 2$), which can severely hurt the quality of the produced parent sets [Bartlett and Cussens 2015].

### 3.2 Greedy Selection

A faster approach is to perform a greedy search that selects the best node to expand at each iteration [Cooper and Dietterich 1992]. This is accomplished by setting $h_i = g_i = sc_i$ as the local score function, and defining *open* and *closed* as before. The algorithm can in fact be made simpler as it is unnecessary to store the values of $g_i$ for explored nodes. The time efficiency of this method often comes at a decreased quality of the parent sets found.

### 3.3 Independence Selection

Independence selection attempts at improving the quality of the parent sets without compromising much the time efficiency [Scanagatta et al. 2015]. The technique can be seen as an $A^*$ search with an inadmissible heuristic. The heuristic function is the $\mathrm{BIC}_i^*$ defined as:

$$\mathrm{BIC}_i^*(\mathbf{Y}, \mathbf{Z}) = \mathrm{BIC}_i(\mathbf{Y}) + \mathrm{BIC}_i(\mathbf{Z}) + \mathrm{inter}_i(\mathbf{Y}, \mathbf{Z}), \tag{3}$$

where $\mathbf{Y}$ and $\mathbf{Z}$ are two non-empty disjoint sets of variables and $\mathrm{inter}_i(\mathbf{Y}, \mathbf{Z}) = \frac{\log N}{2}(r_i - 1)(\prod_{X_j \in \mathbf{Y}} r_j + \prod_{X_j \in \mathbf{Z}} r_j - \prod_{X_j \in \mathbf{Y} \cup \mathbf{Z}} r_j - 1) - \mathrm{BIC}_i(\emptyset)$. This approximate scoring function can be calculated in constant time (i.e., $O(1)$ complexity) if $\mathrm{BIC}_i(\mathbf{Y})$ and $\mathrm{BIC}_i(\mathbf{Z})$ are cached. Additionally, the local score $\mathrm{BIC}_i$ can also be efficiently computed from the $\mathrm{BIC}_i^*$ scores as

$$\mathrm{BIC}_i(\mathbf{Y} \cup \mathbf{Z}) = \mathrm{BIC}_i^*(\mathbf{Y}, \mathbf{Z}) + N \cdot I_i(\mathbf{Y}, \mathbf{Z}), \tag{4}$$

where $I_i$ is the *Interaction Information* estimated from data.

The approach uses $h_i(\mathbf{Y} \cup \mathbf{Z}) = \mathrm{BIC}_i^*(\mathbf{Y}, \mathbf{Z})$ and $g_i(\mathbf{Y}) = \mathrm{BIC}_i(\mathbf{Y})$. The list *closed* is initialized containing all the parent sets with cardinality at most one (including the empty set), and *open* is initialized with all the parent sets of cardinality two.

Since the scores are calculated for increasingly larger parent set sizes, pruning rules can be used to detect unnecessary computations [de Campos and Ji 2011]; also the $\mathrm{BIC}_i^*$ are available from cached scores at each iteration, and $\mathrm{BIC}_i$ score values can be obtained efficiently by computing $I_i(\mathbf{Y}, \mathbf{Z})$.

## 4. ORDER-BASED STRUCTURE LEARNING

As discussed in the introduction, local-search in the space of orderings continues to be one of the most competitive approximate methods for structure learning with high-dimensional datasets. The method described in Algorithm 2 receives a list of candidate sets with their pre-computed scores and performs a search over the space of orderings $<$ represented as a list of variables $L$. The search starts with an initial ordering (line 1), and, for a maximum number of iterations $K$, attempts to improve the incumbent solution by exchanging two adjacent variables in $L$ (this is the neighborhood of the search space). The score of an ordering is the maximum value of the score of a DAG consistent with the ordering and the candidate parent sets selected:

$$sc(L) = \sum_{i=1}^{n} \max_{\mathbf{Y} \in \mathbf{C}_i^L} sc_i(\mathbf{Y})\,. \tag{5}$$

The choice of neighborhood makes the relative difference of scores of the incumbent and proposed solutions efficiently computable. Additionally, an early stop condition verifies whether a local optimum has been reached. The optimal DAG consistent with ordering $L$ and the candidate parents sets $\mathbf{C}_i$ is computed and returned in line 12. Usually, several re-starts are performed in order to escape poor local optima.

**Algorithm 2:** Order-Based Greedy Search

```
   Input  : Candidate sets C_i with their scores pre-computed
   Output: A DAG G
 1 initialize L
 2 for j = 1 to K do
 3 |   L_j ← L
 4 |   for i = 1 to n − 1 do
 5 |   |   swap L_j[i] and L_j[i + 1]
 6 |   |   if sc(L_j) > sc(L) then
 7 |   |   |   L ← L_j
 8 |   |   end
 9 |   |   swap L_j[i] and L_j[i + 1]
10 |   end
11 end
12 obtain DAG G consistent with ordering L
```

## 5. GENERATING INFORMED INITIAL SOLUTIONS

As with most local search approaches, the selection of a good initial solution is crucial for avoiding convergence to poor local maxima in order-based structure learning. Typically, this is attempted by randomly generating initial orderings. While this guarantees a good coverage of the search space when sufficiently many restarts are performed, in large domains it can lead to poor solutions and require many iterations until a local optimum is reached. In this section, we devise heuristics that take advantage of the structure of the problem to produce better initial solutions. Our heuristics rely on a relaxed solution $H^*$ of the structure learning problem satisfying

$$sc(H^*) = \sum_{i} \max_{\mathbf{Y} \subseteq V \setminus \{X_i\}} sc_i(\mathbf{Y})\,. \tag{6}$$

The sets $Pa_i^{H^*}$ that maximize each local score are called the *best parent sets* (for $X_i$), and the corresponding graph $H^*$ is the *best parent set graph*. Note that $H^*$ usually contains cycles, and it is thus not a solution to Eq. 1. In practice, obtaining the graph $H^*$ can be difficult, and we often

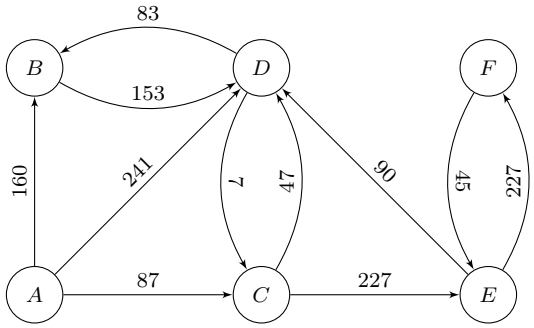



Fig. 2: An example of a parent set graph.

resort to an approximate solution $H$ where the parents $Pa_i^H$ are obtained using one of the parent set selection methods described in Section 3.

### 5.1 DFS-Based Approach

We can exploit the information provided by the best parent set graph (Eq. 6) or an approximation of it to bias the generation of topological orderings towards high-scoring regions. To see this, consider a(n approximation of the) best parent set graph with nodes $X_i$ and $X_j$ such that $X_i$ is the single parent of $X_j$ and has no parents. Then, there is an optimal ordering starting with $X_i$ (this can easily be shown by contradiction). We can delete $X_i$ from the graph and repeat the argument to conclude the existence of an optimal ordering starting with $X_i, X_j$. Now consider a case when there are two or more selectable nodes (by the previous explanation) in graph $H$. Instead of picking a random selectable node we can define the *goodness* of a node by:

$$goodness(X_i) = \prod_{X_j \in Ch_i^H \cap unvisited} |Pa_j^H \cap unvisited| \tag{7}$$

where $Ch_i^H$ is the set of $X_i$'s children and *unvisited* the set of unvisited nodes. Small values of *goodness* mean that removing $X_i$ from the graph will make more nodes to be selectable. Ties are resolved by picking one of the best selectable nodes uniformly at random.

For example, in the (best parent set) graph in Figure 2, we can safely constrain the orderings to start with $A$, since it has no parents, and remove it from the graph. At this time, we have three selectable nodes $B$, $C$ and $F$, each one with same in-degree, but with different goodness value. Since $F$ has the least goodness value, we select it. Performing previous steps repeatedly we get that the candidate optimal orderings are $A, F, C, E, B, D$ and $A, F, C, E, D, B$. Note that this is a significant decrease from the full space of $6! = 720$ possible orderings. This difference is likely to increase as the number of variables increases, and as the best parent set becomes sparser (the sparsity of the best parent set is related to the score function used, and the ratio between the domain dimension and the dataset size).

Motivated by the previous argument, we propose the *DFS-Based initialization heuristic* described in Algorithm 3 which takes a (best parent set) graph $H$ as input. The algorithm starts with all nodes labeled as unvisited and repeatedly selects an unvisited node using as criterias the in-degree and goodness of the node in increasing order. At each step, it marks the node as visited and consider it as removed from $H$. The algorithm returns the ordering when all nodes were visited.

### 5.2 FAS-Based Approach

The DFS-based approach can be seen as removing edges from a graph $H$ so as to make it a DAG (more specifically, a tree), and then extracting a consistent topological ordering. The selection of an edge to

**Algorithm 3:** DFS-Based ordering generation.

```
Function: DFS( Graph H )
unvisited ← all nodes
L ← ∅
while unvisited is not empty do
    O ← unvisited nodes ordered by unvisited in-degree and goodness
    B ← best nodes from O
    if B has more than one node then
        select a node X_r from B uniformly at random
    else
        select the unique node X_r from B
    end
    L ← L ∪ {X_r}
    unvisited ← unvisited \ {X_r}
end
return L
```

remove is often performed locally (among the children of a node), and considers only the qualitative information of the graph (i.e., the parent relationships). An arguably better approach is to use the score function to assess the relevance of each edge, and to consider the removal of edges globally (not only in a local neighborhood). We estimate the relevance of an edge $X_j \to X_i$ in a graph $H$ by

$$W_{ji} = sc_i(Pa_i^H) - sc_i(Pa_i^H \setminus \{X_j\}), \tag{8}$$

The weight $W_{ji}$ represents the cost of removing $X_j$ from the set $Pa_i^H$, and it is always a positive number if $H$ is the best parent set graph since $Pa_i^H$ maximizes the score for $X_i$. A small value of $W_{ji}$ suggests that the parent $X_j$ is not very relevant to $X_i$. For instance, in the weighted graph in Figure 2, the edge $C \to D$ is less relevant than the edge $B \to D$, which in turn is less relevant than the edge $A \to D$.

The main idea of our second heuristic is to penalize orderings which violate an edge $X_i \to X_j$ in $H$ by their associated cost $W_{ij}$. We then wish to find a topological ordering of $H$ that violates the least cost of edges. Given a directed graph $H = (V, E)$, a set $F \subseteq E$ is called a Feedback Arc Set (FAS) if every (directed) cycle of $H$ contains at least one edge in $F$. In other words, $F$ is an edge set that if removed makes the graph $H$ acyclic [Demetrescu and Finocchi 2003]. If we assume that the cost of an ordering of $H$ is the sum of the weights of the violated (or removed) edges, we can formulate the problem of finding a minimum cost ordering of $H$ as a Minimum Cost Feedback Arc Set Problem (min-cost FAS): given the weighted directed graph $H$ with weights $W_{ij}$, find a min-cost FAS $F$ such that

$$F = \arg \min_{H-F \text{ is a DAG}} \sum_{X_i \to X_j \in E} W_{ij}. \tag{9}$$

The min-cost FAS problem have been proved to be NP-complete for directed graphs [Gavril 1977], but there are efficient and effective approximation algorithms [Eades et al. 1993; Eades and Lin 1995; Demetrescu and Finocchi 2003] like the one shown in Algorithm 4 with complexity $O(nm)$, where $m$ is the number of edges on the graph.

We can now describe our second heuristic for generating initial solutions, based on the min-cost FAS problem: take the weighted graph $H$ with weights $W_{ij}$ as input, and find a min-cost FAS $F$; remove the edges in $F$ from $H$ and return a topological order of the DAG $H - F$ (this can be done by performing a depth-first search traversal starting at root nodes).

**Algorithm 4:** Minimum Cost FAS approximation

**Input** : Graph $H$
**Output**: Feedback Arc Set $F$

```
F ← ∅
while there is a cycle C on H do
    W_min ← arg min_{(u,v)∈C} W_uv
    for (u, v) ∈ C do
        W_uv = W_uv − W_min
        if W_uv = 0 then
            F = F + {(u, v)}
        end
    end
end
for (u, v) ∈ F do
    if (u, v) does not build a cycle then
        H = H + (u, v)
        F = F \ {(u, v)}
    end
end
```

## 6. EXPERIMENTS, RESULTS AND DISCUSSION

We compare the performance of order-based local search with different parent set selection procedures and different initialization heuristics on a selected set of real-world datasets listed in Table I.[3] One can see that these datasets range from small domains (with a few tens of variables) to large domains (with thousands of variables). Our experiments aim at evaluating the performance of structure learning for given initial orderings. The algorithms were implemented in C++, using a few utilities from the URLearning package for learning Bayesian networks.[4] All experiments were performed in a 20-node computer cluster; each computer has an Intel Xeon CPU 2.40GHz processor and 512 GB RAM.

For each dataset we ran sequential, greedy and independence selection with no limit on the maximum parent set size $k$, but using a time limit of two minutes per variable. After that, we performed 1000 re-starts of order-based local search, each taking at most 500 iterations ($K = 500$), and using the candidate parent sets obtained from the previous step. For the sake of readability, we report the quality of a DAG found by its *relative score* given by $RC(G) = \frac{sc(G)-sc(\emptyset)}{|sc(\emptyset)|}$, where $sc(\emptyset)$ is the score of an empty DAG.

We refer to the variant of the order-based structure learning algorithm with random initialization, depth-first based initialization and min-cost FAS-based initialization as RND, DFS and FAS, respectively.

### 6.1 Evaluation of Parent Set Selection Approaches

We evaluate the effect the different initialization heuristics have on the quality of the solutions generated by order-based local search for the three parent set selection techniques described. Tables I and II contain relevant statistics about the parent sets selected for each method. The columns $Nps_x$ and $M_x$ represent, respectively, the number of candidate parent sets and maximum in-degree in the best parent set graph using method $x$. We see from these results that the maximum degree using greedy or

[3]We used the same datasets used in [Scanagatta et al. 2015].
[4]Available at `http://urlearning.org/`.

Table I: Datasets characteristics: number of variables ($n$), number of instances ($N$), number of candidate parent sets ($Nps_x$) and maximum in-degree ($M_x$) using parent set selection method $x$

| Dataset | n | N | $Nps_{seq}$ | $M_{seq}$ | $Nps_{gre}$ | $M_{gre}$ | $Nps_{ind}$ | $M_{ind}$ |
|---|---|---|---|---|---|---|---|---|
| Nltcs | 16 | 21574 | 365.1K | 6 | 13.6K | 5 | 365.1K | 6 |
| Msnbc | 17 | 388434 | 712.0K | 9 | 16.7K | 6 | 243.2K | 11 |
| Kdd | 64 | 234954 | 4.8M | 4 | 1.4M | 5 | 593.0K | 6 |
| Plants | 69 | 23215 | 47.2M | 4 | 11.9M | 5 | 1.6M | 7 |
| Baudio | 100 | 20000 | 65.8M | 4 | 17.1M | 6 | 2.2M | 6 |
| Bnetflix | 100 | 20000 | 59.6M | 4 | 16.3M | 6 | 2.3M | 6 |
| Jester | 100 | 14116 | 74.3M | 4 | 21.4M | 5 | 3.8M | 5 |
| Accidents | 111 | 17009 | 47.0M | 4 | 20.9M | 7 | 3.4M | 7 |
| Tretail | 135 | 29387 | 1.9M | 4 | 4.8M | 7 | 1.2M | 4 |
| Pumsb_star | 163 | 16349 | 95.5M | 3 | 38.8M | 5 | 4.9M | 6 |
| Dna | 180 | 3186 | 9.0M | 3 | 62.5M | 4 | 7.1M | 4 |
| Kosarek | 190 | 44500 | 62.6M | 3 | 14.2M | 6 | 3.3M | 6 |
| Msweb | 294 | 37711 | 6.9M | 3 | 14.0M | 7 | 3.8M | 8 |
| Book | 500 | 11598 | 85.3M | 3 | 71.1M | 4 | 4.1M | 4 |
| Tmovie | 500 | 6117 | 96.2M | 3 | 64.8M | 5 | 4.8M | 5 |
| Cwebkb | 839 | 4199 | 43.6M | 2 | 81.2M | 5 | 5.9M | 5 |
| Cr52 | 889 | 9100 | 40.3M | 2 | 66.5M | 5 | 6.6M | 5 |
| C20ng | 910 | 18821 | 99.0M | 2 | 55.5M | 5 | 8.3M | 5 |
| Bbc | 1058 | 2225 | 11.8M | 2 | 50.4M | 4 | 4.7M | 4 |
| Ad | 1556 | 3279 | 4.0M | 2 | 78.6M | 4 | 3.2M | 4 |
| **Average** | | | **42.6M** | **3.5** | **30.6M** | **4.9** | **3.2M** | **5.3** |

independence selection is, in average, greater than sequential selection, while the amount of candidate parent sets is considerably smaller. Some exceptions are the datasets Nltcs and Msnbc where $M$ using greedy selection is the lowest from all procedures probably caused by their small number of variables.

The columns labeled $H^*_{seq}$, $H^*_{greedy}$ and $H^*_{ind}$ in Table II contain the scores of the best parent set graph obtained for the sequential, greedy and independence selection, respectively, while the columns $D(H^*)$ report the average number of parents in the graph $H^*$. It can be noticed that the difference of $D$ values is greater while the number of variables increases. Also, on average, graphs $H^*$ have not only higher score values when used greedy and independence selection, but also higher average in-degree.

### 6.2 Results Using Sequential Selection

We first analyze the performance of the different heuristics when sequential parent set selection is used. Notice that without the time limit, sequential selection generates the optimal best parent sets. Looking at the results in Table III we can notice that DFS and FAS obtain better results under any criteria in almost all datasets. An special case is Kdd dataset where the maximum number of iterations $K$ is almost reached using any of the initialization heuristics due to its huge number of instances.

To verify whether the performance differences are statistically significant, we performed *Friedman Test* [Demsar 2006]. In this and all following experiments, we adopt the statistical significance level $\alpha = 0.05$. The computed p-values are 0.0078, 0.0005, 0.0863 and 0.1423 for best score, average best score, average initial score and average iterations, respectively. Hence, there are statistically significant difference between the heuristics in all criteria except with respect to average number of iterations.

We also performed the *post-hoc Nemenyi Test* for the criteria with statistically significant differences in order to decide which pairwise comparisons were significant; this was carried out by calculating the average ranking of each heuristic considering all datasets. Then, we compute the critical distance $CD = q_\alpha \sqrt{\frac{k(k+1)}{6m}}$, where $k$ is the number of models to compare (i.e. heuristics), $m$ the number of datasets used and $q_\alpha$, the penalization factor for multiple comparison, was taken from [Demsar 2006]. The results are presented graphically in Figure 3. Each point represents the average ranking of the

Table II: Score $sc(H_x)$ and average in-degree $D(H_x)$ of the best parent set graph obtained by using parent set selection method $x$

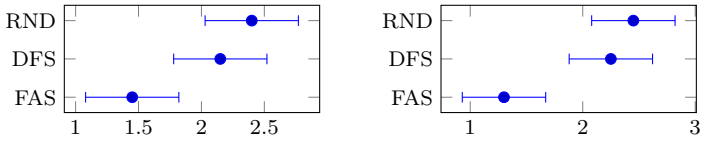
| Dataset | $sc(H^*_{seq})$ | $D(H^*_{seq})$ | $sc(H^*_{gre})$ | $D(H^*_{gre})$ | $sc(H^*_{ind})$ | $D(H^*_{ind})$ |
|---|---|---|---|---|---|---|
| Nltcs | 0.4515 | 4.750 | 0.4359 | 3.812 | 0.4515 | 4.750 |
| Msnbc | 0.1660 | 8.412 | 0.0867 | 4.941 | 0.1752 | 9.235 |
| Kdd | 0.1806 | 3.422 | 0.1759 | 3.672 | 0.1820 | 4.047 |
| Plants | 0.6828 | 3.884 | 0.6685 | 3.971 | 0.6871 | 4.928 |
| Baudio | 0.1877 | 3.590 | 0.1914 | 4.520 | 0.1978 | 4.940 |
| Bnetflix | 0.1336 | 3.550 | 0.1343 | 4.420 | 0.1433 | 4.970 |
| Jester | 0.1672 | 3.680 | 0.1690 | 4.490 | 0.1726 | 4.830 |
| Accidents | 0.5761 | 3.126 | 0.6314 | 3.847 | 0.5285 | 3.739 |
| Tretail | 0.0746 | 2.111 | 0.0793 | 2.081 | 0.0765 | 2.089 |
| Pumsb_star | 0.8194 | 2.325 | 0.7890 | 2.528 | 0.7684 | 2.595 |
| Dna | 0.4062 | 2.844 | 0.4004 | 2.806 | 0.4004 | 2.600 |
| Kosarek | 0.2303 | 2.668 | 0.2332 | 2.979 | 0.2374 | 3.300 |
| Msweb | 0.1852 | 2.031 | 0.1870 | 3.463 | 0.1888 | 3.367 |
| Book | 0.1435 | 2.118 | 0.1513 | 2.790 | 0.1523 | 2.866 |
| Tmovie | 0.3249 | 2.068 | 0.3610 | 2.954 | 0.3640 | 3.072 |
| Cwebkb | 0.1471 | 1.896 | 0.1692 | 2.777 | 0.1702 | 2.900 |
| Cr52 | 0.1775 | 1.900 | 0.2139 | 2.955 | 0.2143 | 3.100 |
| C20ng | 0.0843 | 1.902 | 0.1111 | 3.489 | 0.1118 | 3.680 |
| Bbc | 0.0865 | 1.863 | 0.0979 | 2.403 | 0.0984 | 2.483 |
| Ad | 0.8012 | 1.075 | 0.8251 | 1.223 | 0.8208 | 1.220 |
| **Average** | **0.3013** | **2.9607** | **0.3056** | **3.3060** | **0.3071** | **3.7355** |

corresponding approach, and the intervals indicate the critical distance. A method $A$ is considered statistically significant better than a method $B$ (w.r.t. to a specific criterion) if $A$ has a smaller average ranking and their intervals do not overlap.

We see from the figure that FAS outperforms RND under the two criteria and that sequential selection leads to significantly better results for FAS, but there is no statistically significant difference between RND and DFS in any criteria.

### 6.3 Results Using Greedy Selection

Table IV shows the results for similar experiments using greedy selection. Again, the search takes few iterations except for the Kdd dataset, where it often takes the maximum before it is aborted. We performed the same statistical analysis as before and obtained p-values of 0.1572, 0.5488, 0.7047 and 0.6376. In this case, there is no significant difference between the heuristics under any criteria; thus, they are not analyzed with the Nemenyi test. Although greedy selection obtains larger candidate parent sets, the results show that it considerably hurts the performance obtained by our heuristics compared to RND because of the poor quality of the parent sets processed. Finally, experiments show that the quality of candidate parent sets obtained are important for the performance of our heuristics used for the structure learning problem.




(a) Best Score (b) Average Best Score obtained

Fig. 3: Visualiation of the Nemenyi post-hoc analysis using sequential selection.

Table III: Best score obtained, Average best score obtained, Average initial score generated, Average number of iterations (Avg. It.) using Sequential selection (best values in bold)

| Dataset | Approach | Best Score | Avg. Best Score | Avg. Initial Score | Avg. It. |
|---|---|---|---|---|---|
| Nltcs | RND | 0.4079 | 0.3851 ± 0.009 | 0.3503 ± 0.014 | 12.6940 ± 24.412 |
| | FAS | **0.4125** | 0.3845 ± 0.008 | **0.3602 ± 0.000** | **10.6640 ± 10.642** |
| | DFS | 0.4120 | **0.3851 ± 0.009** | 0.3504 ± 0.014 | 11.4340 ± 10.061 |
| Msnbc | RND | 0.0945 | 0.0858 ± 0.005 | 0.0575 ± 0.010 | 20.5370 ± 17.282 |
| | FAS | 0.0953 | **0.0877 ± 0.002** | **0.0729 ± 0.000** | **9.5610 ± 5.515** |
| | DFS | **0.0958** | 0.0861 ± 0.005 | 0.0573 ± 0.010 | 21.6580 ± 27.461 |
| Kdd | RND | **0.1746** | 0.1720 ± 0.001 | 0.1431 ± 0.007 | 488.9210 ± 53.251 |
| | FAS | 0.1742 | **0.1722 ± 0.001** | **0.1613 ± 0.002** | **464.7260 ± 96.993** |
| | DFS | 0.1739 | 0.1720 ± 0.001 | 0.1427 ± 0.008 | 489.4970 ± 52.632 |
| Plants | RND | 0.6254 | 0.6059 ± 0.007 | 0.5792 ± 0.009 | **18.3200 ± 8.857** |
| | FAS | **0.6386** | **0.6063 ± 0.007** | 0.5632 ± 0.007 | 21.2740 ± 19.664 |
| | DFS | 0.6345 | 0.6061 ± 0.007 | **0.5798 ± 0.009** | 19.1220 ± 18.007 |
| Baudio | RND | 0.1660 | 0.1602 ± 0.002 | 0.1534 ± 0.003 | **18.6060 ± 6.625** |
| | FAS | **0.1671** | **0.1618 ± 0.002** | **0.1535 ± 0.003** | 18.8820 ± 6.152 |
| | DFS | 0.1659 | 0.1601 ± 0.002 | 0.1533 ± 0.003 | 18.6270 ± 6.910 |
| Bnetflix | RND | 0.1164 | 0.1077 ± 0.002 | **0.1016 ± 0.002** | 22.0230 ± 32.259 |
| | FAS | **0.1179** | **0.1080 ± 0.002** | 0.1010 ± 0.001 | 21.8210 ± 29.022 |
| | DFS | 0.1176 | 0.1076 ± 0.002 | 0.1014 ± 0.002 | **20.7890 ± 22.476** |
| Jester | RND | 0.1545 | 0.1484 ± 0.002 | 0.1422 ± 0.002 | **22.4180 ± 15.032** |
| | FAS | 0.1545 | **0.1504 ± 0.001** | 0.1418 ± 0.001 | 26.9350 ± 10.420 |
| | DFS | **0.1552** | 0.1483 ± 0.002 | **0.1422 ± 0.002** | 22.9240 ± 14.255 |
| Accidents | RND | 0.4233 | 0.3531 ± 0.019 | 0.3015 ± 0.023 | 18.0250 ± 7.290 |
| | FAS | **0.4294** | **0.3801 ± 0.017** | **0.3347 ± 0.019** | 18.4930 ± 5.913 |
| | DFS | 0.4090 | 0.3534 ± 0.019 | 0.3029 ± 0.023 | **17.2980 ± 6.901** |
| Tretail | RND | 0.0438 | 0.0232 ± 0.006 | 0.0194 ± 0.005 | 6.9490 ± 3.222 |
| | FAS | **0.0474** | **0.0235 ± 0.004** | **0.0202 ± 0.003** | **6.2610 ± 2.822** |
| | DFS | 0.0443 | 0.0232 ± 0.007 | 0.0194 ± 0.006 | 6.9440 ± 3.452 |
| Pumsb_star | RND | 0.7437 | 0.6942 ± 0.018 | 0.6434 ± 0.022 | **22.1570 ± 10.813** |
| | FAS | 0.7442 | 0.6921 ± 0.021 | 0.6047 ± 0.032 | 24.7550 ± 12.707 |
| | DFS | **0.7570** | **0.6945 ± 0.017** | **0.6446 ± 0.022** | 22.4480 ± 14.377 |
| Dna | RND | 0.2908 | 0.2218 ± 0.016 | 0.1998 ± 0.009 | 27.0660 ± 48.309 |
| | FAS | **0.2955** | **0.2352 ± 0.021** | **0.2027 ± 0.021** | 54.7050 ± 85.196 |
| | DFS | 0.2881 | 0.2212 ± 0.015 | 0.1999 ± 0.009 | **24.9160 ± 40.314** |
| Kosarek | RND | **0.2008** | 0.1484 ± 0.011 | 0.1307 ± 0.009 | 28.0490 ± 30.240 |
| | FAS | 0.1925 | **0.1595 ± 0.008** | **0.1474 ± 0.008** | **22.1030 ± 28.265** |
| | DFS | 0.1874 | 0.1484 ± 0.011 | 0.1306 ± 0.009 | 27.6220 ± 25.337 |
| Msweb | RND | 0.1067 | 0.0517 ± 0.012 | 0.0441 ± 0.011 | 8.9800 ± 4.263 |
| | FAS | **0.1204** | **0.0640 ± 0.016** | **0.0560 ± 0.017** | 9.4780 ± 4.940 |
| | DFS | 0.0894 | 0.0514 ± 0.012 | 0.0440 ± 0.010 | **8.7760 ± 3.911** |
| Book | RND | 0.1252 | 0.1181 ± 0.003 | 0.1150 ± 0.003 | 16.7710 ± 6.592 |
| | FAS | **0.1287** | **0.1195 ± 0.004** | 0.1142 ± 0.005 | 19.1240 ± 8.144 |
| | DFS | 0.1256 | 0.1181 ± 0.003 | **0.1151 ± 0.003** | **16.7240 ± 6.367** |
| Tmovie | RND | 0.3005 | 0.2848 ± 0.004 | 0.2804 ± 0.004 | 15.2110 ± 6.886 |
| | FAS | **0.3079** | **0.2987 ± 0.003** | **0.2953 ± 0.004** | 16.1440 ± 5.606 |
| | DFS | 0.2948 | 0.2849 ± 0.004 | 0.2806 ± 0.004 | **14.8790 ± 5.382** |
| Cwebkb | RND | 0.1174 | 0.1102 ± 0.002 | 0.1086 ± 0.002 | 11.1670 ± 4.633 |
| | FAS | **0.1276** | **0.1187 ± 0.003** | **0.1173 ± 0.003** | 11.6920 ± 7.123 |
| | DFS | 0.1197 | 0.1101 ± 0.002 | 0.1086 ± 0.002 | **11.0490 ± 6.347** |
| Cr52 | RND | 0.1509 | 0.1381 ± 0.005 | 0.1358 ± 0.005 | **13.3120 ± 5.186** |
| | FAS | **0.1584** | **0.1513 ± 0.002** | **0.1498 ± 0.002** | 13.8880 ± 6.398 |
| | DFS | 0.1545 | 0.1379 ± 0.005 | 0.1356 ± 0.005 | 13.4390 ± 5.424 |
| C20ng | RND | 0.0745 | 0.0691 ± 0.002 | 0.0681 ± 0.001 | 18.3333 ± 25.167 |
| | FAS | 0.0741 | **0.0708 ± 0.001** | **0.0697 ± 0.001** | **14.9890 ± 6.353** |
| | DFS | **0.0757** | 0.0692 ± 0.001 | 0.0682 ± 0.001 | 15.6810 ± 15.985 |
| Bbc | RND | 0.0764 | 0.0712 ± 0.004 | 0.0610 ± 0.002 | **364.9752 ± 190.215** |
| | FAS | **0.0777** | **0.0738 ± 0.003** | **0.0653 ± 0.002** | 389.4690 ± 182.781 |
| | DFS | 0.0766 | 0.0711 ± 0.004 | 0.0609 ± 0.002 | 366.7570 ± 190.450 |
| Ad | RND | **0.7142** | **0.6870 ± 0.010** | **0.6842 ± 0.009** | **7.4800 ± 11.620** |
| | FAS | 0.6562 | 0.6055 ± 0.018 | 0.5830 ± 0.017 | 12.1930 ± 6.465 |
| | DFS | 0.7107 | 0.6866 ± 0.010 | 0.6838 ± 0.009 | 7.6610 ± 11.443 |

## 6.4 Results Using Independence Selection

The experiments using independence selection are shown in Table V. Again, we see that FAS overall outperforms the other methods. The Friedman test obtained p-values of 0.0224, 0.0045, 0.0006 and 0.8607, which shows that average number of iterations is the only criteria without statistically significant difference.

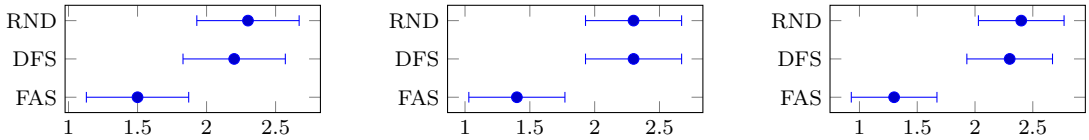
The post-hoc Nemenyi test graphics are shown in Figure 4. As in the results with sequential selection, we see that the FAS advantage is significant compared to other heuristics, and that RND is compares equally with DFS.

Table IV: Best score obtained, Average best score obtained, Average initial score generated, Average number of iterations (Avg. It.) using Greedy selection (best values in bold)

| Dataset | Approach | Best Score | Avg. Best Score | Avg. Initial Score | Avg. It. |
|---|---|---|---|---|---|
| Nltcs | RND | **0.4018** | **0.3724 ± 0.010** | **0.3396 ± 0.015** | 9.0260 ± 12.332 |
| | FAS | 0.4007 | 0.3710 ± 0.011 | 0.3161 ± 0.000 | 9.9080 ± 3.964 |
| | DFS | 0.4007 | 0.3721 ± 0.010 | 0.3385 ± 0.015 | **8.4960 ± 3.881** |
| Msnbc | RND | 0.0637 | 0.0557 ± 0.004 | 0.0410 ± 0.008 | **11.0410 ± 7.009** |
| | FAS | **0.0641** | **0.0565 ± 0.004** | **0.0417 ± 0.000** | 11.7240 ± 5.415 |
| | DFS | 0.0633 | 0.0559 ± 0.004 | 0.0406 ± 0.007 | 11.6530 ± 16.311 |
| Kdd | RND | **0.1685** | 0.1658 ± 0.001 | 0.1365 ± 0.007 | 496.7130 ± 37.134 |
| | FAS | 0.1685 | **0.1658 ± 0.001** | **0.1428 ± 0.003** | **494.9910 ± 45.889** |
| | DFS | 0.1684 | 0.1658 ± 0.001 | 0.1362 ± 0.007 | 497.5640 ± 31.910 |
| Plants | RND | 0.6107 | **0.5905 ± 0.007** | **0.5657 ± 0.009** | 16.1000 ± 6.093 |
| | FAS | **0.6135** | 0.5898 ± 0.008 | 0.5519 ± 0.005 | 17.9310 ± 5.770 |
| | DFS | 0.6096 | 0.5904 ± 0.007 | 0.5652 ± 0.010 | **15.9600 ± 5.687** |
| Baudio | RND | 0.1678 | **0.1609 ± 0.002** | **0.1539 ± 0.003** | 19.7680 ± 7.911 |
| | FAS | 0.1665 | 0.1608 ± 0.002 | 0.1507 ± 0.001 | 22.6070 ± 7.653 |
| | DFS | **0.1680** | 0.1608 ± 0.002 | 0.1538 ± 0.003 | **19.0880 ± 7.770** |
| Bnetflix | RND | **0.1123** | **0.1060 ± 0.002** | **0.1004 ± 0.002** | **17.2260 ± 6.110** |
| | FAS | 0.1116 | 0.1054 ± 0.002 | 0.0969 ± 0.001 | 19.1720 ± 6.481 |
| | DFS | 0.1121 | 0.1060 ± 0.002 | 0.1004 ± 0.002 | 17.2900 ± 6.231 |
| Jester | RND | **0.1545** | 0.1488 ± 0.002 | 0.1428 ± 0.002 | **22.6200 ± 11.114** |
| | FAS | 0.1533 | 0.1482 ± 0.002 | 0.1376 ± 0.000 | 24.0840 ± 11.852 |
| | DFS | 0.1544 | **0.1489 ± 0.002** | **0.1429 ± 0.002** | 23.6730 ± 14.374 |
| Accidents | RND | 0.4441 | 0.3616 ± 0.021 | 0.3083 ± 0.022 | 17.1620 ± 6.662 |
| | FAS | **0.4497** | **0.3938 ± 0.019** | **0.3564 ± 0.021** | **15.9890 ± 6.872** |
| | DFS | 0.4344 | 0.3610 ± 0.022 | 0.3070 ± 0.022 | 17.4430 ± 6.938 |
| Tretail | RND | 0.0509 | 0.0239 ± 0.007 | 0.0199 ± 0.006 | 6.9840 ± 3.372 |
| | FAS | **0.0514** | **0.0275 ± 0.007** | **0.0245 ± 0.005** | **5.7950 ± 2.982** |
| | DFS | 0.0496 | 0.0236 ± 0.007 | 0.0196 ± 0.006 | 7.0120 ± 3.228 |
| Pumsb_star | RND | 0.7103 | 0.6573 ± 0.020 | 0.6062 ± 0.025 | 17.3870 ± 7.296 |
| | FAS | 0.7074 | 0.6510 ± 0.023 | 0.5821 ± 0.027 | 17.5620 ± 6.883 |
| | DFS | **0.7115** | **0.6578 ± 0.019** | **0.6080 ± 0.024** | **17.0840 ± 7.125** |
| Dna | RND | **0.2917** | **0.2195 ± 0.016** | 0.1979 ± 0.009 | **26.5570 ± 48.460** |
| | FAS | 0.2909 | 0.2129 ± 0.024 | 0.1769 ± 0.024 | 39.6480 ± 70.295 |
| | DFS | 0.2898 | 0.2189 ± 0.016 | **0.1982 ± 0.009** | 27.0920 ± 57.135 |
| Kosarek | RND | 0.1977 | 0.1488 ± 0.012 | 0.1300 ± 0.009 | 32.7100 ± 33.459 |
| | FAS | 0.1933 | **0.1533 ± 0.010** | **0.1343 ± 0.008** | **31.4660 ± 30.109** |
| | DFS | **0.2030** | 0.1491 ± 0.012 | 0.1301 ± 0.009 | 34.4910 ± 41.046 |
| Msweb | RND | **0.0920** | **0.0504 ± 0.011** | 0.0434 ± 0.010 | 8.5560 ± 3.890 |
| | FAS | 0.0843 | 0.0445 ± 0.010 | 0.0356 ± 0.008 | 9.0730 ± 4.823 |
| | DFS | 0.0860 | 0.0503 ± 0.011 | **0.0436 ± 0.011** | **8.2830 ± 3.701** |
| Book | RND | 0.1299 | 0.1229 ± 0.003 | 0.1195 ± 0.003 | 17.6490 ± 6.657 |
| | FAS | **0.1387** | **0.1309 ± 0.003** | **0.1284 ± 0.004** | **16.2770 ± 7.752** |
| | DFS | 0.1312 | 0.1231 ± 0.003 | 0.1197 ± 0.003 | 17.4060 ± 6.128 |
| Tmovie | RND | 0.3308 | 0.3123 ± 0.004 | 0.3072 ± 0.005 | 15.8570 ± 9.715 |
| | FAS | **0.3343** | **0.3275 ± 0.002** | **0.3254 ± 0.002** | **13.7210 ± 8.022** |
| | DFS | 0.3285 | 0.3124 ± 0.004 | 0.3070 ± 0.005 | 16.7230 ± 11.119 |
| Cwebkb | RND | 0.1364 | 0.1248 ± 0.003 | 0.1229 ± 0.003 | **12.9630 ± 7.088** |
| | FAS | **0.1471** | **0.1379 ± 0.002** | **0.1365 ± 0.001** | 17.5410 ± 23.581 |
| | DFS | 0.1322 | 0.1246 ± 0.003 | 0.1226 ± 0.003 | 12.9970 ± 5.986 |
| Cr52 | RND | 0.1780 | 0.1632 ± 0.005 | 0.1600 ± 0.005 | 15.4350 ± 6.634 |
| | FAS | **0.1847** | **0.1783 ± 0.002** | **0.1765 ± 0.002** | **14.0780 ± 5.352** |
| | DFS | 0.1792 | 0.1632 ± 0.005 | 0.1601 ± 0.005 | 15.4390 ± 7.422 |
| C20ng | RND | 0.0957 | 0.0876 ± 0.002 | 0.0862 ± 0.002 | 17.5910 ± 18.741 |
| | FAS | **0.0973** | **0.0961 ± 0.000** | **0.0957 ± 0.000** | **13.0530 ± 3.817** |
| | DFS | 0.0957 | 0.0876 ± 0.002 | 0.0862 ± 0.002 | 17.5910 ± 18.741 |
| Bbc | RND | 0.0840 | 0.0787 ± 0.005 | 0.0681 ± 0.002 | 365.7339 ± 190.091 |
| | FAS | **0.0852** | **0.0812 ± 0.003** | **0.0747 ± 0.001** | 365.5450 ± 197.503 |
| | DFS | 0.0840 | 0.0781 ± 0.005 | 0.0680 ± 0.002 | **342.3483 ± 199.543** |
| Ad | RND | **0.7355** | 0.7102 ± 0.009 | 0.7078 ± 0.009 | **4.7060 ± 3.823** |
| | FAS | 0.6781 | 0.6221 ± 0.019 | 0.6005 ± 0.017 | 10.9940 ± 7.159 |
| | DFS | 0.7322 | **0.7106 ± 0.009** | **0.7083 ± 0.009** | 4.7480 ± 4.253 |

## 6.5 Evaluation of Initialization Heuristics

Finally, we look at the aggregated influence of the initialization heuristics (averaging over all parent set selection procedures). The Friedman test obtains p-values of 0.0007, 0.00004, 0.0007 and 0.44933



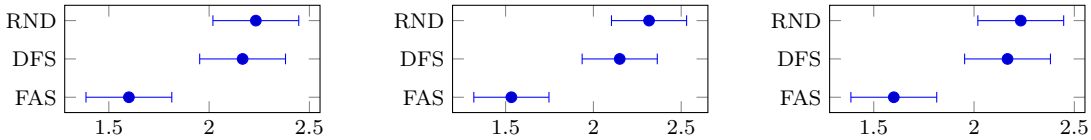

(a) Best Score (b) Average Best Score obtained (c) Average Initial Score obtained

Fig. 4: Visualization of the Nemenyi post-hoc analysis using independence selection.

Table V: Best score obtained, Average best score obtained, Average initial score generated, Average number of iterations (Avg. It.) using Independence selection (best values in bold)

| Dataset | Approach | Best Score | Avg. Best Score | Avg. Initial Score | Avg. It. |
|---|---|---|---|---|---|
| Nltcs | RND | 0.4073 | **0.3852 ± 0.009** | 0.3503 ± 0.014 | 12.0940 ± 18.022 |
| | FAS | **0.4125** | 0.3845 ± 0.008 | **0.3602 ± 0.000** | **10.6640 ± 10.642** |
| | DFS | 0.4120 | 0.3851 ± 0.009 | 0.3504 ± 0.014 | 11.4340 ± 10.061 |
| Msnbc | RND | 0.0950 | 0.0797 ± 0.007 | 0.0509 ± 0.010 | 16.8610 ± 22.530 |
| | FAS | **0.0956** | **0.0825 ± 0.005** | **0.0557 ± 0.001** | **15.1570 ± 22.275** |
| | DFS | 0.0934 | 0.0798 ± 0.007 | 0.0513 ± 0.010 | 16.6540 ± 21.971 |
| Kdd | RND | 0.1741 | **0.1656 ± 0.007** | 0.1158 ± 0.008 | 441.8780 ± 141.884 |
| | FAS | **0.1750** | 0.1653 ± 0.007 | **0.1336 ± 0.005** | **411.4250 ± 175.666** |
| | DFS | 0.1727 | 0.1654 ± 0.007 | 0.1161 ± 0.008 | 431.8180 ± 152.080 |
| Plants | RND | 0.5958 | **0.5634 ± 0.011** | **0.5349 ± 0.013** | 10.8540 ± 4.234 |
| | FAS | 0.5953 | 0.5608 ± 0.012 | 0.5090 ± 0.010 | 12.6340 ± 4.608 |
| | DFS | **0.5962** | 0.5632 ± 0.011 | 0.5349 ± 0.012 | **10.8190 ± 4.365** |
| Baudio | RND | 0.1447 | 0.1324 ± 0.004 | 0.1240 ± 0.004 | 10.2630 ± 4.756 |
| | FAS | **0.1528** | **0.1418 ± 0.002** | **0.1388 ± 0.002** | **7.1500 ± 7.124** |
| | DFS | 0.1470 | 0.1320 ± 0.005 | 0.1239 ± 0.005 | 10.0910 ± 4.546 |
| Bnetflix | RND | **0.1073** | 0.0918 ± 0.004 | 0.0854 ± 0.003 | 10.1970 ± 6.347 |
| | FAS | 0.1043 | **0.0928 ± 0.002** | **0.0881 ± 0.001** | **7.9620 ± 3.576** |
| | DFS | 0.1029 | 0.0917 ± 0.004 | 0.0853 ± 0.003 | 9.7740 ± 4.795 |
| Jester | RND | 0.1527 | 0.1282 ± 0.009 | 0.1148 ± 0.005 | **50.5090 ± 112.764** |
| | FAS | **0.1550** | **0.1398 ± 0.006** | **0.1301 ± 0.000** | 109.1880 ± 152.073 |
| | DFS | 0.1541 | 0.1288 ± 0.009 | 0.1145 ± 0.005 | 55.4230 ± 119.877 |
| Accidents | RND | 0.3565 | 0.2768 ± 0.030 | 0.2416 ± 0.023 | **10.9270 ± 9.189** |
| | FAS | **0.3656** | **0.3258 ± 0.010** | **0.2943 ± 0.005** | 16.0710 ± 9.204 |
| | DFS | 0.3471 | 0.2770 ± 0.029 | 0.2416 ± 0.023 | 11.0310 ± 9.871 |
| Tretail | RND | 0.0485 | 0.0232 ± 0.007 | 0.0195 ± 0.005 | 6.6140 ± 3.265 |
| | FAS | **0.0530** | **0.0329 ± 0.010** | **0.0289 ± 0.009** | 7.1600 ± 3.213 |
| | DFS | 0.0460 | 0.0230 ± 0.007 | 0.0192 ± 0.005 | **6.5240 ± 3.174** |
| Pumsb_star | RND | 0.6637 | 0.5888 ± 0.022 | **0.5491 ± 0.026** | 10.9380 ± 6.825 |
| | FAS | 0.6562 | **0.5945 ± 0.022** | 0.5469 ± 0.025 | 11.5510 ± 5.697 |
| | DFS | **0.6698** | 0.5880 ± 0.021 | 0.5489 ± 0.025 | **10.8220 ± 6.502** |
| Dna | RND | 0.2736 | 0.2046 ± 0.012 | 0.1902 ± 0.009 | **12.1380 ± 18.851** |
| | FAS | **0.3233** | **0.2879 ± 0.008** | **0.2596 ± 0.009** | 50.7770 ± 88.382 |
| | DFS | 0.2775 | 0.2044 ± 0.012 | 0.1902 ± 0.009 | 12.6830 ± 27.589 |
| Kosarek | RND | 0.2022 | 0.1457 ± 0.027 | 0.1166 ± 0.009 | 118.4560 ± 157.163 |
| | FAS | 0.2017 | **0.1610 ± 0.016** | **0.1385 ± 0.006** | **118.1580 ± 153.684** |
| | DFS | **0.2035** | 0.1468 ± 0.027 | 0.1166 ± 0.009 | 122.8750 ± 156.455 |
| Msweb | RND | 0.0851 | 0.0455 ± 0.010 | 0.0400 ± 0.010 | **6.3460 ± 3.115** |
| | FAS | **0.1166** | **0.0519 ± 0.016** | **0.0452 ± 0.016** | 7.6380 ± 3.646 |
| | DFS | 0.0869 | 0.0457 ± 0.010 | 0.0403 ± 0.009 | 6.4600 ± 3.162 |
| Book | RND | 0.1241 | 0.1129 ± 0.003 | 0.1099 ± 0.003 | 11.7360 ± 4.658 |
| | FAS | **0.1324** | **0.1256 ± 0.004** | **0.1241 ± 0.004** | **10.0690 ± 5.148** |
| | DFS | 0.1222 | 0.1130 ± 0.003 | 0.1100 ± 0.003 | 12.0530 ± 4.779 |
| Tmovie | RND | 0.3236 | 0.2789 ± 0.015 | 0.2619 ± 0.007 | 60.7770 ± 90.231 |
| | FAS | **0.3318** | **0.3102 ± 0.005** | **0.3071 ± 0.005** | **19.4600 ± 41.326** |
| | DFS | 0.3262 | 0.2798 ± 0.016 | 0.2617 ± 0.007 | 69.0670 ± 101.116 |
| Cwebkb | RND | 0.1322 | 0.1113 ± 0.003 | 0.1095 ± 0.003 | 9.1510 ± 8.210 |
| | FAS | **0.1354** | **0.1278 ± 0.002** | **0.1269 ± 0.002** | **8.2730 ± 3.202** |
| | DFS | 0.1201 | 0.1113 ± 0.003 | 0.1095 ± 0.003 | 9.0640 ± 3.367 |
| Cr52 | RND | 0.1705 | 0.1414 ± 0.007 | 0.1382 ± 0.006 | 14.0720 ± 22.640 |
| | FAS | 0.1760 | **0.1705 ± 0.001** | **0.1696 ± 0.001** | **7.6830 ± 3.006** |
| | DFS | **0.1798** | 0.1420 ± 0.007 | 0.1387 ± 0.007 | 14.3220 ± 23.750 |
| C20ng | RND | 0.0826 | 0.0695 ± 0.003 | 0.0682 ± 0.003 | 10.9640 ± 10.297 |
| | FAS | **0.0867** | **0.0840 ± 0.001** | **0.0836 ± 0.001** | **8.3390 ± 3.016** |
| | DFS | 0.0810 | 0.0694 ± 0.003 | 0.0682 ± 0.003 | 10.6720 ± 13.253 |
| Bbc | RND | 0.0820 | 0.0778 ± 0.003 | 0.0623 ± 0.002 | **476.1260 ± 95.441** |
| | FAS | **0.0857** | **0.0816 ± 0.001** | **0.0740 ± 0.001** | 493.0780 ± 52.860 |
| | DFS | 0.0821 | 0.0780 ± 0.003 | 0.0624 ± 0.002 | 482.3720 ± 83.473 |
| Ad | RND | **0.7220** | 0.6954 ± 0.010 | 0.6922 ± 0.010 | **7.7850 ± 10.265** |
| | FAS | 0.6659 | 0.6171 ± 0.018 | 0.5942 ± 0.017 | 13.2410 ± 6.021 |
| | DFS | 0.7189 | **0.6958 ± 0.010** | **0.6925 ± 0.010** | 8.3710 ± 10.238 |

for best score, average best score, average initial score and average number of iterations, respectively. Since the first three values are lower than $\alpha$, the differences are considered statistically significant, and we perform the Nemenyi post-hoc test for them. The results are displayed in Figure 5.




(a) Best Score obtained (b) Average Best Score obtained (c) Average Initial Score obtained

Fig. 5: Visualization of the Nemenyi post-hoc analysis considering all parent set selection approaches.

While in the previous analyses segmented by the parent set selection technique we observed that RND and DFS were almost the same under any criteria, this analysis shows that DFS tends to be a better alternative to RND, but without having a statistically significant difference. Also, results shows that FAS outperforms DFS and RND without doubt considering any parent set selection method. These results suggest that DFS biases the search towards high scoring regions, but has problems diversifying its initial solutions. FAS on the other hand achieves a good trade-off between generating high score initial orderings and diversifying between consecutive runs.

In summary, these results indicate the advantage of using informed approaches to generating initial orderings in high dimensionality domains.

## 7. CONCLUSIONS AND FUTURE WORK

Learning Bayesian networks from data is a notably difficult problem, and practitioners often resort to approximate solutions. A state-of-the-art approach for large domains is order-based structure learning, which performs a local search in the space of variable orderings. As with many local search approaches, the quality of the solutions produced by order-based learning strongly depends on the initialization strategy. In this work, we proposed two new informed heuristics for generating initial solutions for order-based structure learning. Both heuristics are based on the best parent set graph, which is the directed graph obtained as the solution of the relaxed problem when cycles are permitted. The first heuristic performs a depth-first search traversal of the parent set graph that orders nodes based on their in-degree. The second heuristic uses the score function to assess the relevance of edges in the parent graphs, and finds a minimum weight ordering by solving a minimum-cost feedback arc set problem. Experiments with 20 real-world datasets containing from 16 to 1556 variables demonstrate that our initialization heuristics improve the accuracy of order-based greedy search. To our knowledge, these are the largest domains considered in the literature.

The initialization heuristics can also be used in other methods that search the space of orderings such as tabu search [Glover 1989], simulated annealing [Granville et al. 1994] or data perturbation [Elidan et al. 2002]. We leave this as future work.

## REFERENCES


Akaike, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control* 19 (6): 716–723, 1974.

Bartlett, M. and Cussens, J. Integer linear programming for the Bayesian network structure learning problem. *Artificial Intelligence*, 2015.

Buntine, W. Theory refinement on Bayesian networks. In *Proc. of the Seventh Annual Conference on Uncertainty Artificial Intelligence*. San Francisco, CA, USA, pp. 52–60, 1991.

Cheng, J., Greiner, R., Kelly, J., Bell, D., and Liu, W. Learning bayesian networks from data: An information-theory based approach. *Artificial Intelligence* 137 (1–2): 43–90, 2002.

Chickering, D. Learning Bayesian networks is NP-complete. In *Learning from Data: Artificial Intelligence and Statistics V*. Springer New York, pp. 121–130, 1996.

Chickering, D., Heckerman, D., and Meek, C. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research* vol. 5, pp. 1287–1330, 2004.

Chickering, D. and Meek, C. Finding optimal Bayesian networks. In *Proc. of the Eighteenth Conference on Uncertainty in Artificial Intelligence*. pp. 94–102, 2002.

Chickering, D. M. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research* vol. 2, pp. 445–498, 2002.

Cooper, G. F. and Dietterich, T. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning* 9 (4): 309–347, 1992.

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. *Information Processing Letters* 86 (3): 129–136, 2003.

Demsar, J. Statistical comparison of classifiers over multiple data sets. *Journal of Machine Learning Research* vol. 7, pp. 1–30, 2006.

Eades, P. and Lin, X. A new heuristic for the feedback arc set problem. *Australian Journal of Combinatorics* vol. 12, pp. 15–26, 1995.

Eades, P., X.Lin, and Smyth, W. F. A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters* 47 (6): 319–323, 1993.

Elidan, G., Ninio, M., Friedman, N., and Schuurmans, D. Data perturbation for escaping local maxima in learning. In *Proc. of the Eighteenth National Conference on Artificial Intelligence*. pp. 132–139, 2002.

Friedman, H., Nachman, I., and Peér, D. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *Proc. of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. pp. 206–215, 1999.

Gavril, F. Some NP-complete problems on graphs. In *Proc. of the 11th Conference on Information Sciences and Systems*. pp. 91–95, 1977.

Glover, F. Tabu search - part I. *Operations Research Society of America Journal on Computing* 1 (3): 190–206, 1989.

Granville, V., Krivanek, M., and Rasson, J.-P. Simulated annealing: A proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16 (6): 652–656, 1994.

Heckerman, D., Geiger, D., and Chickering, D. Learning Bayesian networks: The combination of knowledge and statistical data. *Journal of Machine Learning Research* 20 (3): 197–243, 1995.

Heckerman, D., Meek, C., and Cooper, G. A Bayesian approach to causal discovery. *Computation, causation, and discovery*, 1999.

Koivisto, M. Parent assignment is hard for the MDL, AIC, and NML costs. In *Proc. of the 19st annual conference on Learning Theory*. pp. 289–303, 2006.

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

Larranaga, P., Karshenas, H., Bielza, C., and Santana, R. A review on evolutionary algorithms in Bayesian network learning and inference tasks. *Information Sciences* 233 (1): 109–125, 2013.

Larranaga, P., Poza, M., Yurramendi, Y., Murga, R., and Kuijpers, C. M. H. Structure learning of Bayesian networks by genetic algorithms: Performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18 (9): 912–926, 1996.

Margaritis, D. *Learning Bayesian Network Model Structure from Data*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh (PA), USA, 2003.

Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.

Perez, W. and Mauá, D. Initialization heuristics for greedy Bayesian network structure learning. In *Proc. of the 3rd Symposium on Knowledge Discovery, Mining and Learning*. pp. 58–65, 2015.

Scanagatta, M., de Campos, C. P., Corani, G., and Zaffalon, M. Learning Bayesian networks with thousands of variables. In *Proc. of the 29th Annual Conference on Neural Information Processing Systems*. pp. 1855–1863, 2015.

Schwarz, G. Estimating the dimension of a model. *The Annals of Statistics* 6 (2): 461–464, 1978.

Spirtes, P. and Meek, C. Learning Bayesian networks with discrete variables from data. In *Proc. of the 1st Int. Conference on Knowledge Discovery and Data Mining*. pp. 294–299, 1995.

Suzuki, J. Learning Bayesian belief networks based on the minimum description length principle. In *Proc. of the Thirteenth Int. Conference on Machine Learning*. pp. 462–470, 1996.

Tessyer, M. and Koller, D. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proc. of the 21st Conference on Uncertainty in Artificial Intelligence*. pp. 584–590, 2005.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning* 65 (1): 31–78, 2006.

Yuan, C. and Malone, B. Learning optimal Bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research* 48 (1): 23–65, 2013.