Initialization Heuristics for Greedy Bayesian Network Structure Learning

Walter Perez Urcia Denis Deratani Mauá

Universidade de São Paulo
Instituto de Matemática e Estadística
Departamento de Ciências da Computação

KDMiLE 2015

Contents

1. Introduction
2. Bayesian Network
3. Learning BN
4. Initializing Heuristics
5. Experiments

Introduction

- Variables $X_1, \ldots, X_n$ takes values in $\Omega_1, \ldots, \Omega_n$
  - $X_1 : \mathit{Gender}, \Omega_1 = \{\mathit{Male}, \mathit{Female}\}$
  - $X_2 : \mathit{City\ size}, \Omega_2 = \{\mathit{Big}, \mathit{Small}\}$
- Factored possibility space $\Omega = \Omega_1 \times \ldots \times \Omega_n$
- Event is a subset of Ω
- Probability function maps events α and β into real values such that
  - $0 \leq \mathbb{P}(\alpha) \leq 1$
  - $\mathbb{P}(\Omega) = 1$
  - $\mathbb{P}(\alpha \cup \beta) = \mathbb{P}(\alpha) + \mathbb{P}(\beta) - \mathbb{P}(\alpha \cap \beta)$

- Every assignment of value to a variable correspond to an event:

$$Gender = M \leftrightarrow \alpha = \{(M, s), (M, b)\}$$

- The probability distribution of a variable X maps assignments of the variable to the respective probabilities:

$$\mathbb{P}(X = x) = \mathbb{P}(\{\omega : \omega \text{ consistent with } x\})$$

- We denote the probability distribution of X as $\mathbb{P}(X)$ and the probability of an arbitrary event $\{X = x\}$ as $\mathbb{P}(x)$
- It follows from the properties of probability function that

$$\sum_X \mathbb{P}(X) = \sum_{x \in \Omega_X} \mathbb{P}(X = x) = 1$$

- A joint assignment to a set of variables is an event

$$\textit{Gender} = M \text{ and } \textit{City size} = b \leftrightarrow \alpha = \{(M, b)\}$$

- The joint probability distribution of a set of variables is a function that maps joint assignments to their event probabilities:

$$\mathbb{P}(X = x, Y = y) = \mathbb{P}(\{\omega : \omega \text{ consistent with } x, y\})$$

- We denote the probability distribution of X and Y as $\mathbb{P}(X)$ and the probability of an arbitrary joint event as $\mathbb{P}(x, y)$
- It follows from the properties of probability function that

$$\sum_{X,Y} \mathbb{P}(X, Y) = \sum_{x \in \Omega_X} \sum_{x \in \Omega_Y} \mathbb{P}(X = x, Y = y) = 1$$

- Maps assignments of two variables to conditional probabilities:

$$\mathbb{P}(X = x \mid Y = y) = \frac{\mathbb{P}(X = x, Y = y)}{\mathbb{P}(Y = y)}$$

- Represented as $\mathbb{P}(X \mid Y)$
- Analogously, we can define join conditional probability distribution $\mathbb{P}(X, Y \mid Z, W)$

By definition of conditional probability:

$$\mathbb{P}(\alpha \mid \beta)\mathbb{P}(\beta) = \mathbb{P}(\alpha \cap \beta)$$

For events $\alpha_1, \ldots, \alpha_n$ it follows that

$$\mathbb{P}(\alpha_1 \cap \ldots \cap \alpha_n) = \mathbb{P}(\alpha_1) \prod_{i=2}^{n} \mathbb{P}(\alpha_i \mid \alpha_1 \cap \ldots \cap \alpha_{i-1})$$

In terms of variables:

$$\mathbb{P}(A, B, C) = \mathbb{P}(A)\mathbb{P}(B \mid A)\mathbb{P}(C \mid B, A)$$

$$\mathbb{P}(\beta \mid \alpha) = \frac{\mathbb{P}(\alpha \mid \beta)}{\mathbb{P}(\alpha)}\mathbb{P}(\beta)$$

- Prior probability: $\mathbb{P}(\beta)$
- Posterior probability: $\mathbb{P}(\beta \mid \alpha)$
- Data Likelihood: $\mathbb{P}(\alpha \mid \beta)$
- Evidence probability: $\mathbb{P}(\alpha)$

- Bayes' rule can be seen as a way of revising beliefs in light of new information/knowledge: start with $\mathbb{P}(\beta)$, observe α then set $\mathbb{P}(\beta)' = \mathbb{P}(\beta \mid \alpha)$
- This way of thinking is known as Bayesian Reasoning

Events α and β are independent if:

$$\mathbb{P}(\alpha \cap \beta) = \mathbb{P}(\alpha)\mathbb{P}(\beta)$$

- The following are equivalent definitions:
  - Either $\mathbb{P}(\alpha \mid \beta) = \mathbb{P}(\alpha)$ or $\mathbb{P}(\beta) = 0$
  - Either $\mathbb{P}(\beta \mid \alpha) = \mathbb{P}(\beta)$ or $\mathbb{P}(\alpha) = 0$
- Knowing β is irrelevant to determining the value of α
- Knowing α is irrelevant to determining the value of β

Variables A and B are independent if:

$$\mathbb{P}(A = a, B = b) = \mathbb{P}(A = a)\mathbb{P}(B = b)$$

for all values of a and b.
Another way to write this is:

$$\mathbb{P}(A, B) = \mathbb{P}(A)\mathbb{P}(B)$$

Events α and β are independent conditional on event γ if:

$$\mathbb{P}(\alpha \cap \beta \mid \gamma) = \mathbb{P}(\alpha \mid \gamma)\mathbb{P}(\beta \mid \gamma)$$

The following are equivalent definitions:

- Either $\mathbb{P}(\alpha \mid \beta, \gamma) = \mathbb{P}(\alpha \mid \gamma)$ or $\mathbb{P}(\beta \mid \gamma) = 0$
- Either $\mathbb{P}(\beta \mid \alpha, \gamma) = \mathbb{P}(\beta \mid \gamma)$ or $\mathbb{P}(\alpha \mid \gamma) = 0$

Analogously, variables A and B are conditionally independent given C if

$$\mathbb{P}(A, B \mid C) = \mathbb{P}(A \mid C)\mathbb{P}(B \mid C)$$

for every assignment to A, B and C

The presence of independences reduces the number of probability values to specify:

- No independences: $\mathbb{P}(A, B, C)$, k^3 values
- A, B and C are dependent, and A and B are conditionally independent given C: $\mathbb{P}(A, B \mid C) = \mathbb{P}(A \mid C)\mathbb{P}(B \mid C)$, $k + 2k^2$ values
- A, B and C are independent: $\mathbb{P}(A, B, C) = \mathbb{P}(A)\mathbb{P}(B)\mathbb{P}(C)$, $3k$ values

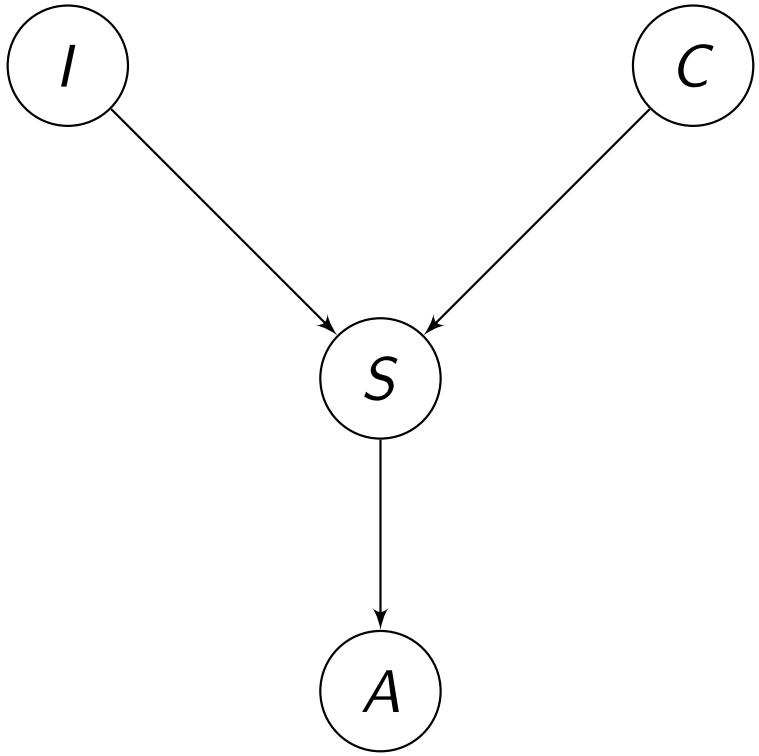
Bayesian Network

Markov property

Given its parents, every variable is conditionally independent from its non-descendants non-parents

Factorization property

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



The directed acyclic graph (DAG) above has joint probability distribution:

$$\mathbb{P}(I, C, S, A) = \mathbb{P}(I)\mathbb{P}(C \mid I)\mathbb{P}(S \mid C, I)\mathbb{P}(A \mid S, C, I)$$

$$= \mathbb{P}(I)\mathbb{P}(C)\mathbb{P}(S \mid C, I)\mathbb{P}(A \mid S)$$

A Bayesian Network consists of

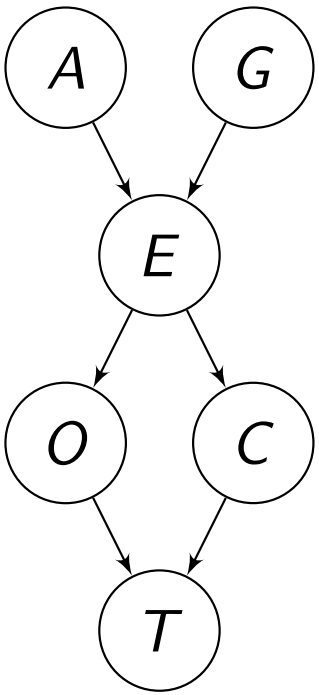
- A DAG G over a set of variables $X_1, \ldots, X_n$
- Probability constraints: $\mathbb{P}(X_i = k \mid Pa(X_i) = j) = \theta_{ijk}$

Joint Probability Distribution

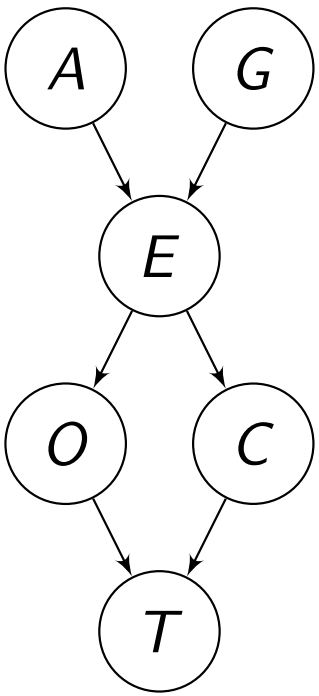
There is a unique probability function consistent with a BN:

$$\mathbb{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathbb{P}(X_i \mid Pa(X_i)) = \prod_{i=1}^{n} \theta_{ijk}$$

- Age (A): young, adult, old
- Gender (G): male, female
- Education (E): primary, high school, university
- Occupation (O): employee, self-employed
- City size (C): big, small
- Transport (T): private (car), public (bus, train, etc)

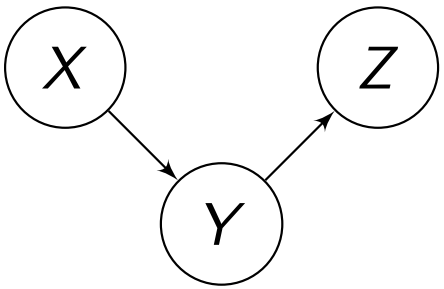


- Education rates have been increasing over years; young people are more likely to have university degrees than old people
- Women are more likely to invest in their education than men; women outnumber men in the vast majority of university-level courses
- High education levels is key to getting prestigious professions; jobs requiring university degrees are more easily available in big cities
- Preferred means of transport depends on occupation and city size

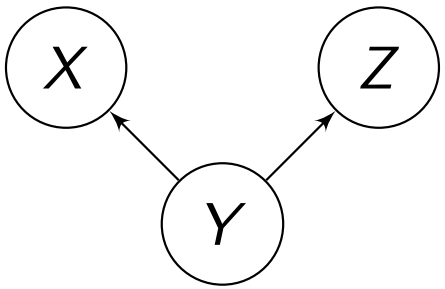


$$\mathbb{P}(A = young) = 0.3$$
$$\mathbb{P}(A = adult) = 0.5$$
$$\mathbb{P}(A = old) = 0.2$$
$$\mathbb{P}(E = high \mid A = young, G = F) = 0.7$$
$$\vdots$$
$$\mathbb{P}(C = small \mid E = high) = 0.25$$
$$\vdots$$

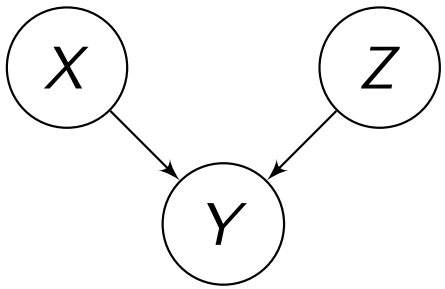
An arc $X \rightarrow Y$ can be interpreted as "X causes Y"



causal chain

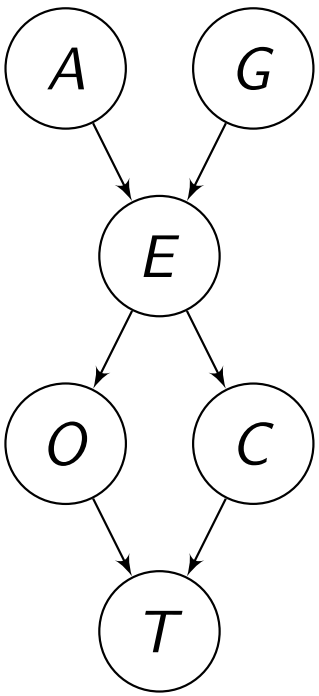


common cause



common effect

Defining and verifying causality is difficult and controversial: We can loosely define X causes Y if X temporarily precedes and direct influences Y



We can query a Bayesian Network about unspecified probabilities

- Are women more likely to prefer public transport over men: $\mathbb{P}(T = \textit{public} \mid G = F) > \mathbb{P}(T = \textit{public} \mid G)$?
- What is the distribution of ages for people who use private means of transport: $\mathbb{P}(A \mid T = \textit{private})$?

Learning BN

Constraint-based approaches

Perform multiple conditional independence hypothesis testing in order to build a DAG

Score-based approaches

Associate every DAG with a polynomial-time computable score value and search for structure with high score values

Learning as optimization

Given dataset D, select G that maximizes decomposable score function $sc(G, D)$

- Score $sc(G, D)$ is usually a mix of data fitness F and model complexity P:

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

with $\psi(N) \geq 0$ is a function of data size $|D| = N$

- We usually ommit dependence on D: $sc(G)$

Learning as optimization

Select G that maximizes decomposable score function

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

$$G^* = arg \max_G \sum_i sc(X_i, Pa(X_i))$$

Greedy Search is a popular approach to find an approximate solution. It relies on the definition of a neighborhood space among solutions and on local moves that search for improving solution in the neighborhood of an incumbent solution

```
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
```

Different neighborhoods and local moves rise to different methods such as:

- Structure-based
- Equivalence-based
- Order-based

Based on the observation that the problem of learning a Bayesian network can be written as

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

$$G^* = \arg\max_{<} \sum_{i=1}^{n} \max_{P \subseteq \{X_j < X_i\}} sc(X_i, P)$$

which means that if an optimal ordering over the variables is known, an optimal DAG can be found by maximizing the local scores independently

```
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)
```

where $swap(L, i, i+1)$ swaps the values $L[i]$ and $L[i+1]$

Initializing Heuristics

The selection of a good initial solution is crucial for avoiding convergence to poor local maxima in Order-Based Learning. Traditionally, this is attempted by randomly generating initial solutions (i.e., a node ordering) in order to cover as much as possible of the search space. But this also means that we have $n!$ possible orders and this is a huge disadvantage for problems with lots of variables.
We propose two different approaches to reduce the space of possible orders:

- DFS-based approach
- FAS-based approach

We can have an upper bound for $sc(G^*)$ by getting $sc(\overline{G})$ where

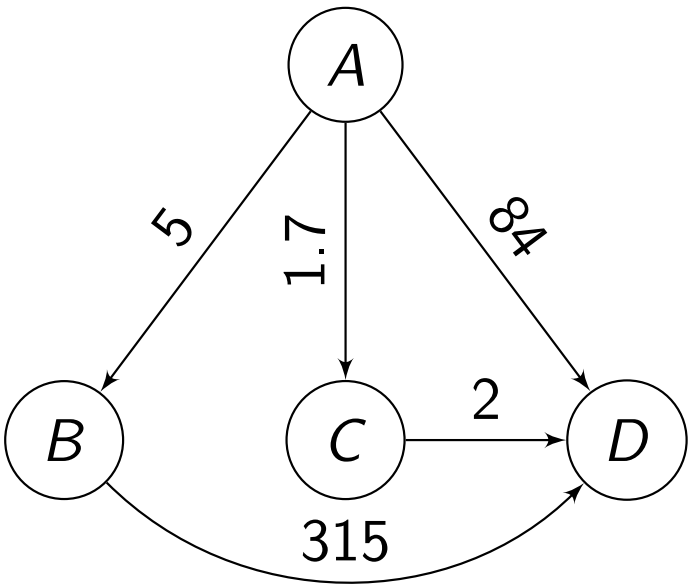
$$\overline{G} = \arg \sum_i \max_{Pa(X_i)} sc(X_i, Pa(X_i))$$

is the directed graph where the parents $Pa(X_i)$ of each node X_i are selected as to maximize the local score $sc(X_i, Pa(X_i))$. We call the parents of a variable in $\overline{G}$ the best parent set. Note that $\overline{G}$ usually contains cycles and it is thus not a solution to the main problem.

- We can exploit the information provided by $\overline{G}$ and avoid generating orders which are guaranteed sub-optimal
- Assume best parent set is unique
- Consider two variables X_i and X_j in $\overline{G}$, where X_j is parent of X_i, but there is no arc from X_i to X_j
- No optimal ordering can have X_i preceding X_j

The number of these orderings can be much smaller than the full space of orderings

Example



Graph $\overline{G}$

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

The algorithm

- Take as input $\overline{G}$ and mark all nodes unvisited
- Start with an empty list L
- While there is an unvisited node
  - Select an unvisited X_i uniformly random
  - Perform a depth-first search (DFS) rooted at X_i and add to L the visited nodes
- Return L

Disadvantage of DFS approach

This approach can be seen as removing edges from $\overline{G}$ such as to make it a DAG and then extract a topological order. But not all edges are equally relevant in terms of avoiding poor local maxima.

Estimating the relevance

We can estimate the relevance of an edge $X_j \rightarrow X_i$ by

$$W_{ji} = sc(X_i, Pa^*(X_i)) - sc(X_i, Pa^*(X_i) \setminus \{X_j\})$$

where $Pa^*(X_i)$ represents the best parent set for X_i.
The weight W_{ji} represents the cost of removing X_j from the set $Pa^*(X_i)$ and it is always a positive number.
Small values mean that parent X_j is not very relevant to X_i, while large values denotes the opposite.

We then wish to find a topological ordering of $\overline{G}$ that violates the least cost of edges.

Feedback Arc Set

- Given a directed graph $G = (V, E)$, a set $F \subseteq E$ is called a Feedback Arc Set (FAS) if every (directed) cycle of G contains at least one edge in F.
- In other words, F is an edge set that if removed makes the graph G acyclic.
- This could be reduced to the following equation:

$$F = \min_{G-F \text{ is a DAG}} \sum_{X_j \to X_j \in E} W_{ij}$$

Finding FAS F

```
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
```

The algorithm

- Take the weighted graph $\overline{G}$ with weights W_{ij} as input
- Find minimum-cost FAS F
- Remove the edges in F from $\overline{G}$
- Return a topological order from $\overline{G} - F$

Experiments

Configuration

For each dataset considered:

- Limit parent set size to 3
- Perform 1000 runs of Order-based Greedy Search
- At most 100 iterations ($K = 100$)
- Use BIC score
- Find best parent sets by exhaustive search

Datasets

Dataset	n (#attributes)	N (#instances)	Density of $\overline{G}$
Census	15	30168	2.85
Letter	17	20000	2.41
Image	20	2310	2.45
Mushroom	23	8124	2.91
Sensors	25	5456	3.00
SteelPlates	28	1941	2.18
Epigenetics	30	72228	1.87
Alarm	37	1000	1.98
Spectf	45	267	1.76
LungCancer	57	27	1.44

Table 1: Data sets characteristics

Dataset	Approach	Best Score	Avg. Initial Score	Avg. Best Score	Avg. It.
Census	Random	**-212186.79**	-213074.18 ± 558.43	-212342.26 ± 174.21	7.26 ± 2.90
	DFS-based	-212190.05	-212736.80 ± 379.96	-212339.83 ± 152.26	5.90 ± 2.61
	FAS-based	-212191.64	**-212287.99 ± 92.54**	**-212222.12 ± 70.99**	**3.28 ± 1.67**
Letter	Random	-138652.66	-139774.54 ± 413.74	-139107.13 ± 329.15	6.07 ± 2.50
	DFS-based	-138652.66	-139521.38 ± 396.61	**-138999.84 ± 310.06**	5.75 ± 2.35
	FAS-based	-138652.66	**-139050.43 ± 70.55**	-139039.26 ± 87.97	**2.24 ± 0.96**
Image	Random	**-12826.08**	-13017.13 ± 44.35	-12924.24 ± 41.39	7.59 ± 2.71
	DFS-based	-12829.10	-12999.09 ± 38.56	-12921.13 ± 37.88	7.10 ± 2.47
	FAS-based	-12829.10	**-12930.63 ± 20.83**	**-12882.30 ± 26.43**	**5.05 ± 1.72**
Mushroom	Random	**-55513.38**	-58450.72 ± 1016.54	-56563.84 ± 616.59	7.59 ± 2.76
	DFS-based	**-55513.38**	-58367.11 ± 871.25	-56472.72 ± 546.19	7.75 ± 2.58
	FAS-based	-55574.71	**-56450.49 ± 154.54**	**-56198.66 ± 174.64**	**4.65 ± 1.63**
Sensors	Random	**-62062.13**	-63476.33 ± 265.46	-62726.60 ± 251.26	9.22 ± 2.94
	DFS-based	-62083.21	-63392.60 ± 255.90	-62711.50 ± 257.79	9.65 ± 3.12
	FAS-based	-62074.88	**-62530.26 ± 133.44**	**-62330.94 ± 121.82**	**5.17 ± 2.24**
SteelPlates	Random	-13336.14	-13566.50 ± 65.80	-13429.13 ± 52.14	8.96 ± 3.43
	DFS-based	**-13332.91**	-13572.77 ± 81.12	-13432.30 ± 57.57	9.30 ± 3.38
	FAS-based	-13341.73	**-13485.26 ± 38.27**	**-13397.08 ± 29.53**	**7.77 ± 2.24**

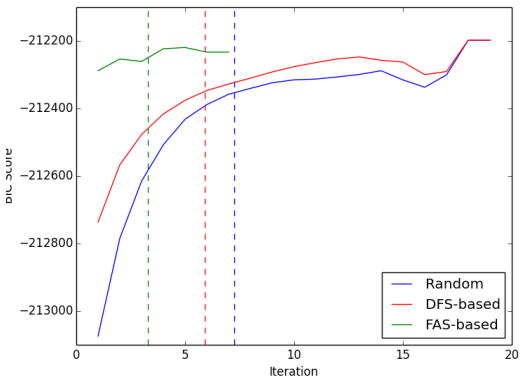


Figure 1: Census

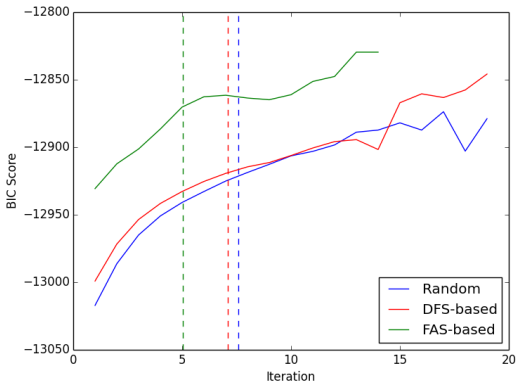


Figure 3: Image

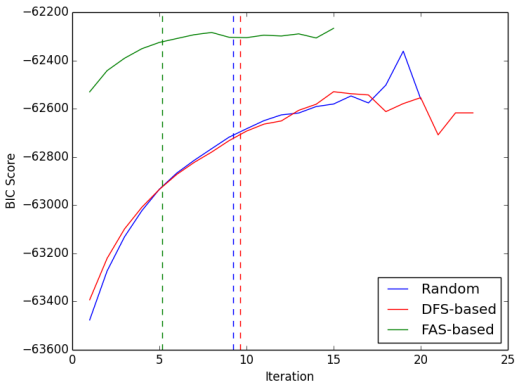


Figure 5: Sensors

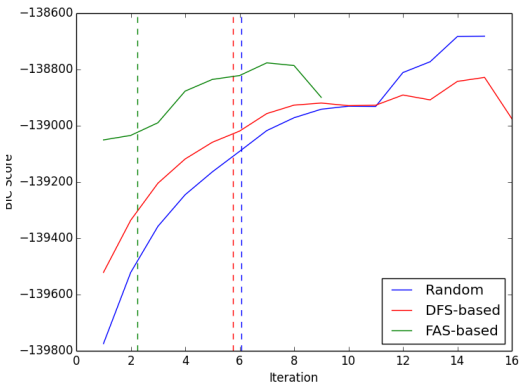


Figure 2: Letter

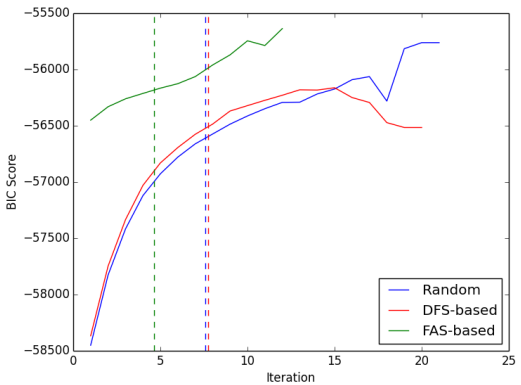


Figure 4: Mushroom

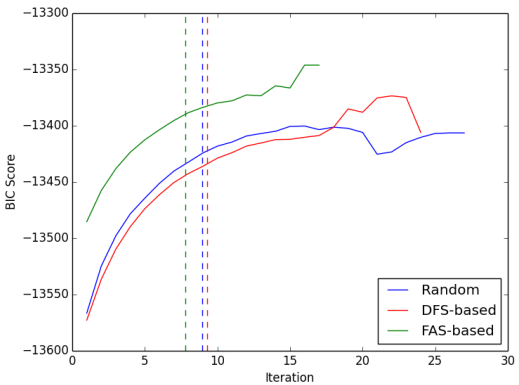


Figure 6: SteelPlates

Dataset	Approach	Best Score	Avg. Initial Score	Avg. Best Score	Avg. It.
Epigenetics	Random	-56873.76	-57722.30 ± 228.44	-57357.60 ± 222.12	5.89 ± 2.67
	DFS-based	**-56868.87**	**-57615.36 ± 189.17**	**-57308.93 ± 165.18**	6.42 ± 2.47
	FAS-based	**-56868.87**	-57660.09 ± 146.45	-57379.59 ± 148.42	**5.33 ± 2.28**
Alarm	Random	-13218.22	-13324.52 ± 30.49	-13245.43 ± 15.63	10.92 ± 3.24
	DFS-based	**-13217.97**	-13250.72 ± 17.70	-13236.71 ± 12.02	**4.32 ± 2.32**
	FAS-based	-13220.55	**-13249.77 ± 2.57**	**-13233.98 ± 6.19**	6.34 ± 1.74
Spectf	Random	-8176.81	-8202.03 ± 5.23	-8189.69 ± 4.65	7.20 ± 2.17
	DFS-based	**-8172.37**	-8200.04 ± 4.08	-8187.29 ± 4.91	7.86 ± 2.49
	FAS-based	-8172.51	**-8176.98 ± 2.01**	**-8176.07 ± 2.05**	**2.27 ± 1.11**
LungCancer	Random	**-711.23**	-723.79 ± 2.69	-718.03 ± 2.84	5.46 ± 1.78
	DFS-based	-711.36	-720.47 ± 2.51	**-715.29 ± 1.86**	5.02 ± 1.50
	FAS-based	-711.39	**-716.13 ± 0.89**	-715.67 ± 1.19	**2.73 ± 1.79**

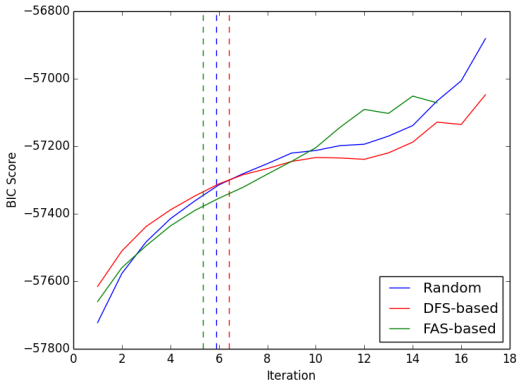


Figure 7: Epigenetics

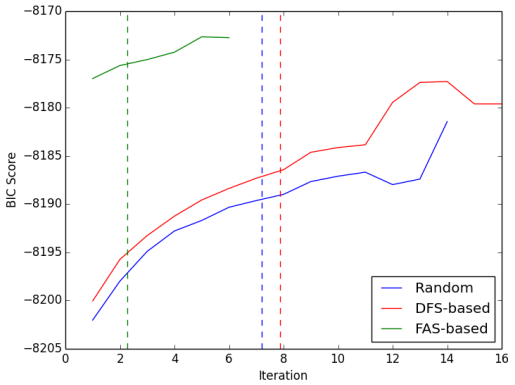


Figure 9: Spectf

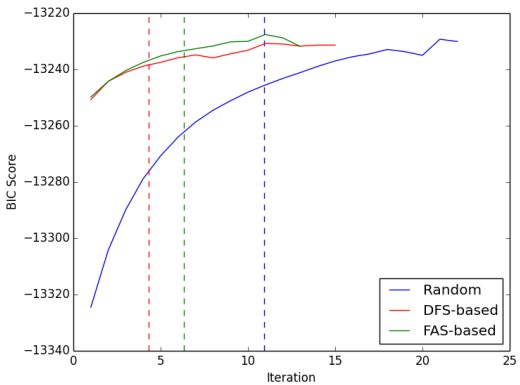


Figure 8: Alarm

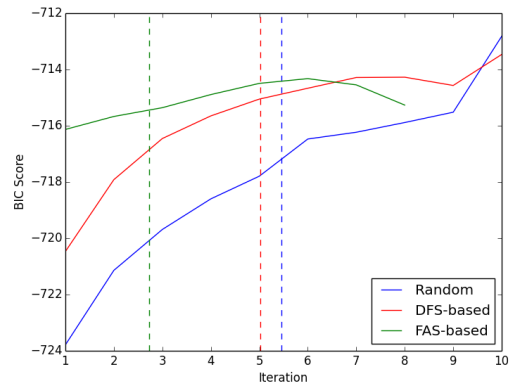


Figure 10: LungCancer

The results show that:

- In most of the datasets with less than 25 attributes, the Random strategy finds the highest-scoring networks over all runs, even though it finds worse networks on average
- The best initial solutions are found by the FAS-based strategy followed by the DFS-based strategy
- For datasets with more than 25 variables, Random is less effective in finding high-scoring networks, except for the LungCancer (which has very little data)
- These results suggest that more informed approaches to generating initial orderings might be more effective in high dimensionality domains, or when the number of restarts is limited

The results show that:

- The proposed strategies are also more robust, which can be seen by the smaller variance of the average initial and best scores
- The results also suggest that the proposed strategies are more effective than Random in datasets for which the graph G is sparser (smaller density), showing that pruning the space of orderings can be effective in those cases
- The initial orderings provided by the proposed strategies speed up convergence of the local search, as can be seen by the smaller number of average iterations for those strategies in the table
- It is expected better results with datasets with higher dimensionality

- Experiments with real-world datasets containing from 15 to 57 variables demonstrate that compared to the commonly used strategy of generating initial ordering uniformly at random the proposed heuristics lead to better solutions on average, and increase the convergence of the search with only a small overhead
- Although the gains observed in our experiments are small, we expect larger differences for datasets with more variables
- Our proposed techniques could be adapted to generate initial solutions also for Structure- and Equivalence-based local search methods by returning directed acyclic graphs instead of node orderings
- Another extension of this work is to employ the proposed heuristics in branch-and-bound solvers for finding optimal solutions

Thanks!