https://doi.org/10.31449/inf.v47i1.4497 Informatica **47** (2023) 83–96 **83**

Learning the Structure of Bayesian Networks from Incomplete Data Using a Mixture Model

Issam Salman[1], Jiří Vomlel[2]
[1]Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague, Trojanova 13, 120 01, Prague, CZ
[2]Institute of Information Theory and Automation of the CA, Pod Vodárenskou věží 4, 182 00, Prague, CZ
E-mail: Issam.Salman@fjfi.cvut.cz, vomlel@utia.cas.cz

Keywords: Bayesian network, belief-noisy-OR, structure learning, incomplete data, EM-mixture

Received: November 8, 2022

In this paper, we provide an approach to learning optimal Bayesian network (BN) structures from incomplete data based on the BIC score function using a mixture model to handle missing values. We have compared the proposed approach with other methods. Our experiments have been conducted on different models, some of them Belief Noisy-Or (BNO) ones. We have performed experiments using datasets with values missing completely at random having different missingness rates and data sizes. We have analyzed the significance of differences between the algorithm performance levels using the Wilcoxon test. The new approach typically learns additional edges in the case of Belief Noisy-or models. We have analyzed this issue using the Chi-square test of independence between the variables in the true models; this approach reveals that additional edges can be explained by strong dependence in generated data. An important property of our new method for learning BNs from incomplete data is that it can learn not only optimal general BNs but also specific Belief Noisy-Or models which is using in many applications such as medical application.

Povzetek: Razvita je metoda za določitev optimalne Bayesove mreže ob nepopolnih podatkih.

1 Introduction

Bayesian networks (BNs) have been used in a variety of applications. The challenge of learning a BN can be categorized into two parts: (1) structural learning, which involves identifying the topology of the BN; and (2) parametric learning, which involves estimating the conditional probabilities for a given network. The challenge of learning the structure of a BN is by far more difficult than the other one. Most methods, such as [1] and [2], require complete data, while in practical applications we are often confronted with values missing from the dataset; this problem regards both parts (1 and 2) mentioned above and affects the performance of the model learning. A record with a missing value should be omitted from the dataset.

An earlier work [3] studied the impact of learning the parameters and the structure of a BN using hard EM and soft EM with a comprehensive simulation study covering incomplete data.

In this paper, we study the problem of learning the optimal BN structure from incomplete data, adopting a new approach of using the product distribution mixture models to handle missing values; the latter will be used with [2] to estimate the missing values and learn the optimal structure. In addition, we show in this paper that our new approach is able to learn the structure of a Belief Noisy-OR (BNO) [4] model from incomplete data.

2 Bayesian network

A Bayesian network encodes a joint probability distribution over a set of random variables $U = \{X_1, X_2, \ldots, X_m\}$. We consider only discrete variables in this work, which is the most common current usage of BNs. A finite set of states of a variable X_i will be denoted by $\mathscr{X}_i$. Conditional probability distributions (CPDs) are attached to each variable in the network. Their purpose is to quantify the strength of the relationships depicted in the BN through its structure: these CPDs mathematically describe the behavior of that variable under every possible value assignment of its parents. Since to specify this behavior one needs a number of parameters exponential in the number of parents, and since this number is typically smaller than the number of variables in the domain, this approach results in exponential savings in space and time.

Formally, a Bayesian network for U is a pair $B = \langle G, \Theta \rangle$. Its first component, G, is a directed acyclic graph whose vertices correspond to the random variables U, and whose edges represent direct dependencies between these variables. The graph G encodes independence assumptions: each variable X_i is independent of its non-descendants given its parents in G. The second component of the pair, namely Θ, represents the set of parameters that quantify the network. It contains parameter $\theta_{x_i|\Pi_{x_i}} = f(x_i|\Pi_{x_i})$ for each possible value x_i of X_i and Π_{x_i} of Π_{X_i}, where Π_{X_i} denotes

the set of parents of X_i in G. Accordingly, a Bayesian network B defines a unique joint probability distribution over U given by:

$$\begin{aligned} F(X_1 = x_1, \ldots, X_m = x_m) &= \prod_{i=1}^{m} F(X_i = x_i | \Pi_{X_i} = \Pi_{x_i}) \\ &= \prod_{i=1}^{m} \theta_{x_i|\Pi_{x_i}} \end{aligned}$$

for each Π_{X_i} which is a parent of X_i.

2.1 Structure learning of BN

Note that a BN can be viewed from two perspectives: as an effective coding of an independence relationship on the one hand, and as an effective encoding of a high-dimensional distribution of probabilities on the other hand.

One option of learning the structure is to rely on the specialists in the field through a conscious and meticulous process of knowledge gathering. This involves training experts in probabilistic graphical modeling, validating expert opinions, and extracting and testing information. This process all too often leads to disagreements among experts and a lack of reliability pertaining to the model. Nonetheless, in many fields, where data is scarce, this is one of the key approaches to model building.

Another mechanism is the automatic derivation of the model based on a data set. It is this machine learning approach (ML) that we follow here (so that we avoid the very rich field of human knowledge acquisition). For a data set $D = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$, where $\mathbf{u}_i$ is an instantiation of all variables in U, the BN structure learning translates to the problem of learning a network structure from D. Suppose $\mathbf{u}$ is complete and discrete. Consequently, finding the optimal Bayesian network is reduced to finding the optimal structure. The optimal structure can be learned by three approaches coming from the area of ML.

The first is the constraint-based approach to structure learning, which takes advantage of the first perspective and attempts to reconstruct a Bayesian network by analyzing data independence. These algorithms require an infinite amount of data to learn independence with certainty; high-order independence tests can be unreliable unless the sample size is truly huge [5]. The second is the score-based approach, which invests in the second perspective and looks for Bayesian networks that adequately describe the available data with the best score. The core of this approach is to assign a score value $s(G)$ to each acyclic directed graph G. The score function defines an overall order (up to equivalences) over the structures in such a way that a structure with a better description of the data is assigned a higher value. The last approach is a hyper-approach, which mixes the two previous approaches together.

2.2 Score-based

Score-based learning is a technique frequently used for determining the optimal structure. In this process, each candidate is assigned a BN score to measure the goodness-of-fit of a structure to the data. The goal of the learning problem is then to find the optimal scoring structure. The score usually measures how well this BN describes the data set D.

Definition (1): Let $B = \langle G, \Theta \rangle$ be a Bayesian network, and let $D = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ be a training set, where each $\mathbf{u}_i$ assigns a value to all variables in $\mathbf{U}$. The *MDL* scoring function of a network B given a training data set D, written $MDL(B|D)$, is given by:

$$MDL(G|D) = LL(G|D) - \frac{\log n}{2} |G|$$

where $|G|$ is the number of parameters in the network. The first term represents the loglikelihood, i.e., it measures the model fit. The second term penalizes the model complexity. The penalty term for MDL is greater than that for most other evaluation functions, since optimal networks with the MDL are usually sparser than optimal networks with function scoring. As its name suggests, an optimal network with MDL minimizes the scoring function rather than maximizing it. The Bayesian information criterion (BIC) [6] is a scoring function whose calculation is equivalent to MDL for Bayesian networks, but it is derived on the basis of the models' asymptotic behavior. Where the score is decomposable, it can be written as a sum of the scores of each variable and its parent set:

$$\begin{aligned} BIC(G|D) &= \sum_{i=1}^{m} BIC(X_i|\Pi_{X_i}) \\ &= \sum_{i=1}^{m} \{LL(X_i|\Pi_{X_i}) - Penalty(X_i|\Pi_{X_i})\} \end{aligned}$$

The score-based algorithms' aim is to optimize this score and return the structure G that maximizes it. As the space of all possible structures is at least exponential in the number of variables m, this presents a number of problems. There are $m(m-1)/2$ possible undirected edges and $2^{m(m-1)/2}$ possible structures for every subset of these edges. Moreover, there may be more than one orientation of the edges for each such choice. One popular choice is hill-climbing [7].

2.3 Structural learning with pruning

Statistical testing is a method of reducing the set of potential DAGs. Another approach to reducing this set is to use constraints provided by experts. Besides that, we can use structural constraints similar to in [2]. The structural constraints can be applied locally as long as they include only one node and its parents.

Algorithm 1 represents an approach to learning the optimal structure of a BN using the constraint rules and a decomposable score [8]. The main function of the algorithm is to compute a collection of candidate parent sets for each variable. Then we optimize across this collection by selecting one parent set for each variable, without creating

directed cycles while maximizing the total score. The following theorem can be used to reduce the numbers of the collections for candidate parents.

Lemma 2.1. *Let X_i be a variable and $\Pi^{'}$ be a candidate parent set for X_i. Suppose that $BIC(X_i|\Pi^{'}) < BIC(X_i|\{\})$. Then $\Pi^{'}$ can be safely ignored from the candidate parent sets.*

Proof. The proof uses the decomposability of the BIC score. Let $G^{'}$ and G be DAGs that differ only on the parent set of X_i where $\Pi^{'}$ is the parent set of X_i in $G^{'}$ and Π is the parent set of X_i in G. Suppose $\Pi \subset \Pi^{'}$. Therefore, if $G^{'}$ does not contain directed cycles then G cannot contain them either. This fact, together with $\text{BIC}(X_i|\Pi) > \text{BIC}(X_i|\Pi^{'})$, implies thast $G^{'}$ is not BIC optimal. This statement also holds if the candidate subset is the empty set $\Pi = \{\}$. □

Algorithm 1 Parent sets evaluation for the BN structure learning algorithm

Input:
$\mathscr{D}$: a data set
m: an integer representing the number of variables in $\mathscr{D}$
Output: Accepted sets of parents for each node
Phase 1: initialize the parameters
$g_i = (V,E)$ // A DAG containing a node and its candidate parent set
S_i: BIC score of g_i
Q_i: priority queue of triples (X_i, Π_{X_i}, S_i) ordered by S_i
Phase 2: mscour($X_i, \mathscr{D}$) function to find the min(BIC) score
S_i = the BIC score of g_i where only X_i is included
return S_i
Phase 3: find the accepted Q_i for X_i
Q_i is empty
S^* = mscour($X_i, \mathscr{D}$)
Π_{X_i} is a parent set for X_i
add (X_i, Π_{X_i}, S^*) to Q_i
For each $X_k, k \in \{1,\ldots,m\}$ do:
  add X_k to Π_{X_i}
  S_{ki} = the BIC score of the updated Π_{X_i}
  if($S_{ki} > S^*$)
    add (X_i, Π_{X_i}, S_{ki}) to Q_k
    For each $X_j, j \in \{1,\ldots,m\}, i \neq j \neq k$, do:
      add X_j to Π_{X_i}
      S_{ki} = the BIC score of the updated Π_{X_i}
      if($S_{ki} > S^*$)
        add (X_i, Π_{X_i}, S_{ki}) to Q_k
      else delete X_j from Π_{X_i}
    end for
  else delete X_k from Π_{X_i}
end for

The $g_i = (V,E)$ in Phase 1 from Algorithm 1 is a DAG containing the set of nodes $V = \{X_i, \Pi_{X_i}\}$ and the set of arcs $E = \{(X_o, X_i), \forall X_o \in \Pi_{X_i}\}$. Algorithm 1 considers all possible parent sets that can lead to an optimal BN. Its implementation is based on [8]. After Phase 3, we find a DAG with the highest BIC from among the variables given the candidate parent sets of each variable. That is done using GOBNILP [9] tool[1] which is a smart algorithm using integer linear programming. We will refer to this algorithm as A1.

Let us also note that, if a dataset is generated from a BN having the empty graph as its structure and this dataset is large enough then, for any parent set $\{\Pi_{X_i} \neq \phi\}$, it holds that $BIC(X_i|\{\}) > BIC(X_i|\Pi_{X_i})$. This implies that the variables are independent and the penalty for larger parent sets makes the BIC value worse for all nonempty parent sets.

One of the axioms of the pruning rules stated in the literature states that if a candidate subset has a better score than another candidate set and the first candidate set is a subset of the second candidate set, it is safe to disregard that second candidate set due to the decomposability of score functions. We have applied the pruning rule as formalized in the theorem 2.1 in Algorithm 1. That algorithm will reduce the collection of accepted parent sets for each node by discarding all parent sets which do not meet the criteria.

3 Incomplete datasets

One of the widespread problems in data mining and machine learning is incomplete data. Values may be missing even from training instances. Nowadays more and more datasets are available, but most of them are incomplete. Therefore, machine learning must cope with this problem. Normally, to learn the BN structure using A1 algorithm [2], we need complete data, such that all instances $\mathbf{u}_i \in D,\ i \in \{1,\ldots,n\}$ are complete and don't have any missing values. In the case of incomplete data and an instance which has a missing value, A1 does not use this instance in the BN structure learning.

3.1 Product distribution mixtures to handle incomplete data

Because of incomplete data, most methods in machine learning cannot be applied. An easy way to deal with this problem is completing the data by simply omitting the incomplete vectors or removing the incomplete variables. But this approach has a weakness: we may lose a massive part of the available information. Another alternative is to use an estimation to replace the missing values [10] (i.e., put in estimates of the missing values). However, for certain reasons, the estimated values have to be typical, and the natural variability of the data will be partially restricted. For that, the product mixture model gives us a better way to directly apply the EM algorithm to complete the dataset [11]. We will refer to this approach as EM-Mixture.

Considering finite mixtures we assume that:

[1] https://www.cs.york.ac.uk/aig/sw/gobnilp/

$$P(X) = \sum_{j=1}^{r} w_j F(X|j), \quad (1)$$

$$F(X|j) = \prod_{i=1}^{m} F_i(X_i|j), \quad \sum_{j=1}^{r} w_j = 1 \quad (2)$$

where $w_j > 0$ is a probabilistic weight of the j-th mixture component, F_i is the conditional distribution of the variable $X_i, i \in 1,\ldots,m$, and r is the number of components. Note that the product components do not imply that the involved variables are independent. In this sense, the mixture model (1) is not restrictive [12]. It is easy to verify that, by increasing the number of components r, we can describe any discrete probability distribution in the form (1). In our experiments, it was selected based on the number of variables in a dataset.

To estimate the mixture parameters, we maximize the log-likelihood function:

$$LL = \sum_{k=1}^{n} \log P(\mathbf{u}^{(k)})$$

where n is the number of records in the dataset D and $\mathbf{u}^{(k)}$ is the k-th datavector from D. We will use the EM algorithm to maximize the log-likelihood function.

Next, we explain how the learned product mixture model will be used to fill in the missing values. Let $C = \{i_1, i_2, \ldots, i_k\}$ be a subset of $M = \{1,2,\ldots,m\}$ such that the corresponding sub-vector

$$\mathbf{u}_C = (x_{i_1}, x_{i_2}, \ldots, x_{i_k})$$

is complete. Then, under the product mixture model, we can compute the marginal probability of $\mathbf{u}_C$ as

$$P_C(\mathbf{u}_C) = \sum_{j=1}^{r} w_j F_C(\mathbf{u}_C|j) \quad (3)$$

$$F_C(\mathbf{u}_C|j) = \prod_{i \in C} F_i(x_i|j) \ . \quad (4)$$

Let z be an index of a variable unobserved in $\mathbf{u}$, i.e., $z \in M \setminus C$. Under the product mixture model, we can compute the conditional distribution of the missing value $\mathbf{u}_z$ given the complete part $\mathbf{u}_C$ with $P_C(\mathbf{u}_C) > 0$ as

$$P_{z|C}(\mathbf{u}_z|\mathbf{u}_C) = \frac{P_{z,C}(\mathbf{u}_z, \mathbf{u}_C)}{P_C(\mathbf{u}_C)} = \sum_{j=1}^{r} W_j(\mathbf{u}_C) F_z(\mathbf{u}_z|j)$$

where $W_j(\mathbf{u}_C)$ are the conditional component weights:

$$W_j(\mathbf{u}_C) = \frac{w_j F_C(\mathbf{u}_C|j)}{\sum_{j=1}^{r} w_j F_C(\mathbf{u}_C|j)} \ .$$

We thus compute the probability distribution $P_{z|C}(\mathbf{u}_z|\mathbf{u}_C)$ for each missing value of each data vector $\mathbf{u} \in D$ with a missing value. There are several ways of using this probability distribution to fill in the missing value of X_z in $\mathbf{u}$ – in this paper, we select value $\mathbf{u}_z$ maximizing $P_{z|C}(\mathbf{u}_z|\mathbf{u}_C)$ over all values of X_z.

The last step of our presentation is the description of adapting the EM algorithm for learning product mixture models such that it is applicable to incomplete data. Given a data vector $\mathbf{u} \in D$ and a variable X_i with index $i \in \{1,2,\ldots,n\}$, let $\mathscr{N}(\mathbf{u})$ be the subset of indices of the available variables (i.e., observed in that data) of $\mathbf{u}$, and $D(i) \subset D$ be the subset of vectors with observed values of variable X_i:

$$\mathscr{N}(\mathbf{u}) = \{v \in \{1,2,\ldots,n\} : \mathbf{u}_v \text{ observed in } \mathbf{u}\}$$
$$D(i) = \{\mathbf{u} \in D : i \in \mathscr{N}(\mathbf{u})\}$$

In Algorithm 2, we present the modification of the EM algorithm for the product mixture model for incomplete data. For $x_v \in \mathscr{X}_v$, $v \in \{1,2,\ldots,n\}$, and $j = 1,\ldots,r$, we use $F_v(x_v|j)$ to denote the conditional probability of observing value x_v of variable X_v given the component j. The initialization of the EM-Mixture algorithm (presented in Algorithm 2) is performed using the partitions obtained from agglomerative hierarchical clustering implemented in the function *hc* of the R package *mclust* [13]. In our algorithm, the symbol $\delta(x,y)$ denotes the standard delta function equal to one if $x = y$ and equal to zero otherwise.

Algorithm 2 EM-Mixture

Input: D is a data set
Output: a completed data set
Phase 1: initializing:
$w_j,\ j = 1,\ldots,r$
$F_v(x_v|j)$, for $x_v \in \mathscr{X}_v$, $v \in \{1,2,\ldots,n\}$, and $j = 1,\ldots,r$
$L = -\infty$
Phase 2:
repeat
E-Step:

$$q(j|\mathbf{u}) = \frac{w_j \prod_{v \in \mathscr{N}(\mathbf{u})} F_v(\mathbf{u}_v|j)}{\sum_{l=1}^{r} w_l \prod_{v \in \mathscr{N}(\mathbf{u})} F_v(\mathbf{u}_v|l)}, \text{ for } \mathbf{u} \in D,\ j = 1,\ldots,r$$

$$w_j = \frac{1}{|D|} \sum_{\mathbf{u} \in D} q(j|\mathbf{u}), \text{ for } j = 1,\ldots,r$$

M-Step: for $x_v \in \mathscr{X}_v$, $v \in \{1,2,\ldots,n\}$, and $j = 1,\ldots,r$

$$F_v(x_v|j) = \frac{\sum_{\mathbf{u} \in D(v)} \delta(x_v, \mathbf{u}_v) \cdot q(j|\mathbf{u})}{\sum_{\mathbf{u} \in D(v)} q(j|\mathbf{u})}$$

$$L' = \sum_{\mathbf{u} \in D} \log \left[\sum_{j=1}^{r} w_j \prod_{v \in \mathscr{N}(\mathbf{u})} F_v(\mathbf{u}_v|j) \right]$$

$$\mathscr{Q} = L' - L$$

$$L = L'$$

until $\mathscr{Q} \leq \varepsilon$

The EM algorithm converges monotonically to a local

or global maximum or a saddle point of the log-likelihood function L in the sense that the sequence of $\{L^t\}_{t=0}^{\infty}$ does not decrease. The presence of a local maximum makes the starting point of the procedure influential; hence it is selected at random. We use the value of $\varepsilon = 0.005$ to terminate the main loop of the algorithm. The sequence of log-likelihood values generated by E-Step and M-Step is non-decreasing [11] (i.e., $L^{'}L$).

We adapt the BN structure learning algorithm A1 so that it can learn from incomplete data. We use the EM-Mixture algorithm, i.e., Algorithm 2, to make the incomplete data complete in Phase 3. The whole algorithm will be referred to as A2.

3.2 Experiments

The experiments have been repeated ten times on ten different subsets in each MCAR rate on different models, using the generated datasets from the true models summarized in Table 8 in A. We have compared our approach denoted as A2 with three other methods. By A1 we denote the BIC optimal learning from complete data created by omitting all rows containing a missing value. In [3], the authors proposed the soft and hard EM algorithms to fill in the missing values and learn an optimal BN structure from the completed data by Tabu search [14], which we refer to as A3 and A4, respectively.

The test scenarios, which include more than 700 incomplete datasets, are summarized in Figure 1. The resulting BNs of the simulations within each scenario are shown in Tables 1, 2, and 3.

The decision tree shown in Figure 1 is intended to guide practitioners as to which imputation algorithm appears to perform the best, depending on the characteristics of their problem with incomplete data. Each leaf of the decision tree corresponds to a subset of the scenario that we studied, grouped according to the values of the experimental factors, to recommend which algorithm has the best average Structure Hamming Distance [15] (SHD) values between the essential graph of the learned model and the essential graph of the true model. The dominance of the algorithms has been tested using the Wilcoxon test [16]. We say that an algorithm is better than another if it has a lower average SHD and their confidence intervals do not overlap, i.e., the p-value of the Wilcoxon test is lower than 5%.

In the results based on the SHD, A2 has scored the best results. For the results based on the SHD and the Wilcoxon test, we have observed some important general trends:

- A2 appears to be a good algorithm in all scenarios.
- A2 is significantly better than other Algorithms for Model M2 in Leaves B and G.
- A2 is significantly better than other Algorithms for the model Child in Leaf C.
- A2 and A3 are significantly better than A1 and A4 for Models M1 in Leaves C, D, P, and K.

Table 1: Recommended algorithm by decision tree leaf where MCAR rate in [5 - 10] -Group 1.

Leaf	Size	Bayesian Network	Recommended Algorithm
A	Size >5000	Weather	A1, **A2**, A3, A4
		M1	**A2**, A3, A4
B	Size in [3000 - 5000]	Weather	**A2**, A3, A4
		M1	**A2**, A3, A4
		M2	**A2**
		Child	**A2**, A3, A4
C	Size in [1500 - 2500]	Weather	**A2**, A3, A4
		M1	**A2**, A3
		M2	**A2**, A3, A4
		Child	**A2**
D	Size <1000	Weather	**A2**, A3, A4
		M1	**A2**, A3
		M2	**A2**
		Child	**A2**, A3

- A1 is significantly worse than other Algorithms in all scenarios where the data size is smaller than 5,000.

Figure 2 represents the algorithm results of all models with all dataset sizes and all MCAR rates.

Table 2: Recommended algorithm by decision tree leaf where MCAR rate in [15 - 25] - Group 2.

Leaf	Size	Bayesian Network	Recommended Algorithm
E	Size >5000	Weather	A1, **A2**, A3, A4
		M1	**A2**, A3, A4
F	Size in [3000 - 5000]	Weather	**A2**, A3, A4
		M1	**A2**, A3, A4
		M2	**A2**, A3
		Child	**A2**, A3
G	Size in [1500 - 2500]	Weather	**A2**, A3, A4
		M1	**A2**, A3, A4
		M2	**A2**
		Child	**A2**, A3
P	Size <1000	Weather	**A2**, A3, A4
		M1	**A2**, A3
		M2	**A2**
		Child	**A2**, A3

Table 3: Recommended algorithm by decision tree leaf where MCAR rate in [35 - 50] - Group 3.

Leaf	Size	Bayesian Network	Recommended Algorithm
H	Size >5000	Weather	A1, **A2**, A3, A4
		M1	**A2**, A3, A4
G	Size in [3000 - 5000]	Weather	**A2**, A3, A4
		M1	**A2**, A3
K	Size in [1500 - 2500]	Weather	**A2**, A3, A4
		M1	**A2**, A3
M	Size <1000	Weather	**A2**, A3, A4
		M1	**A2**

4 Belief Noisy-Or model

The Belief Noisy-Or (BNO) model is suitable for describing a specific class of uncertain relationships in Bayesian networks [4] common in several practical applications of BNs. As an example, let us mention the QMR-DT network [17]. In Figure 3 we present the structure of a CPT

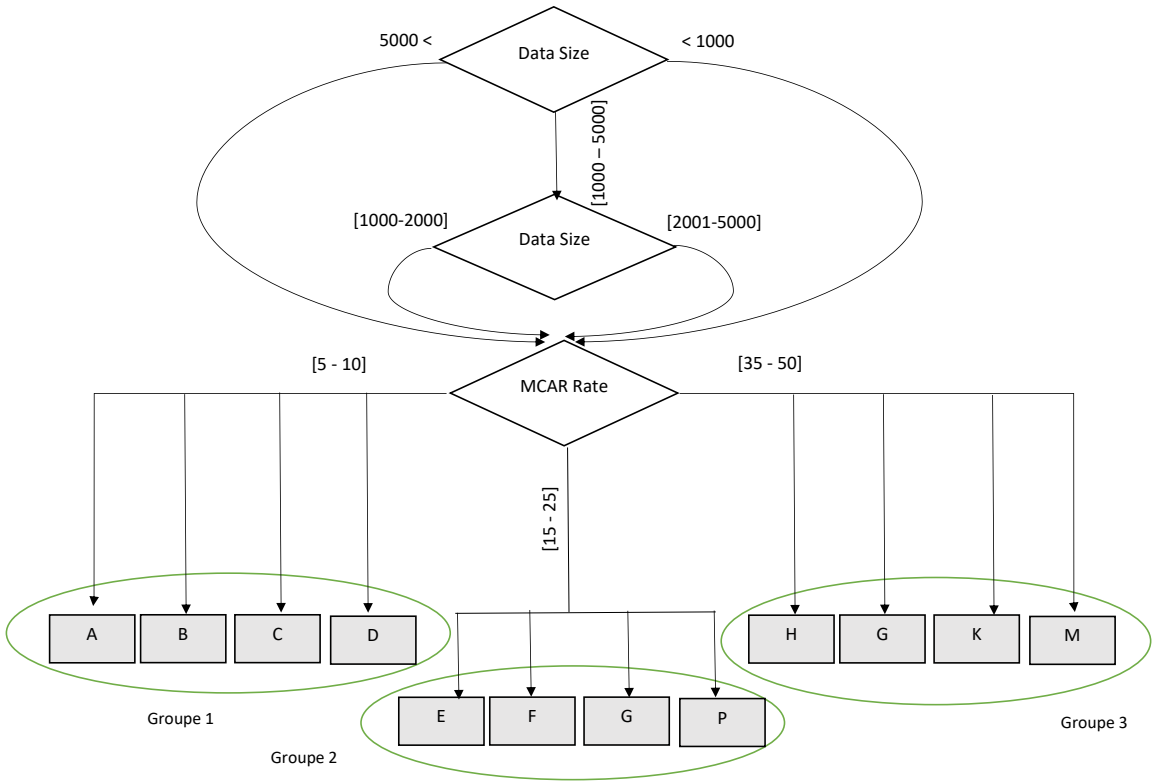


Figure 1: The decision tree for different test scenarios.

$F(Y|X_1,\ldots,X_n)$ where auxiliary nodes $X'_1,\ldots,X'_n$ are added to explicitly separate the noisy relations from the logical OR relation. For a CPT with multiple parent variables $X_1,\ldots,X_n$ the noisy-or is defined as follows[2]:

$$\begin{aligned} F(X_i^{'}=0|X_i=0) &= 1-\alpha, \quad F(X_i^{'}=1|X_i=0)=\alpha \\ F(X_i^{'}=0|X_i=1) &= p_i, \qquad F(X_i^{'}=1|X_i=1)=1-p_i \end{aligned}$$

where $i \in \{1,\ldots,n\}$ and $p_i \in [0,1]$ is the parameter which defines the probability that the positive value x_i of variable X_i is inhibited – it is referred to as the inhibition probability and the parameter α specifies the possibility of a positive value even if the value of the corresponding parent variable is negative. In most experiments, we will set $\alpha = 0$. The CPT of $F(Y|X'_1,\ldots,X'_n)$ represents the deterministic logical OR function, i.e.,

$$F(Y=0|X'_1=x'_1,\ldots,X'_n=x'_n) = \begin{cases} 1 & \text{if } x'_1=0,\ldots \\ & x'_n=0 \\ 0 & \text{otherwise.} \end{cases}$$

Consequently, the CPT of $F(Y|X_1,\ldots,X_n)$, which represents the noisy-or function, is computed as follows:

$$\begin{aligned} F(Y=0|X_1=x_1,\ldots,X_n=x_n) &= \prod_{i=1}^{n} F(X_i^{'}=0|X_i=x_i) \\ &= \prod_{1}^{n} (p_i)^{x_i}(1-\alpha)^{1-x_i} \\ F(Y=1|X_1=x_1,\ldots,X_n=x_n) &= 1-\prod_{1}^{n} (p_i)^{x_i}(1-\alpha)^{1-x_i} \end{aligned}$$

4.1 Analysis of BNO models

In this Section, we analyze the BNO models represented in Table 9 in A where $\alpha = 0$. Tables 5, 6, and 7 show the marginal probability distributions (MPD) of the variables in BNO models N1, N2, and BN2O, respectively; look at Figures 11 and 12. The Tables illustrate the decrease of the marginal probability values for F($C_i = 0$) in the case of a node having more than one parent. See Table 7. This decrease is due to the properties of the product of probabilities in (5). On the other hand, they also illustrate the increase of that marginal probability with a higher number of its predecessors in previous layers; that increase depends on the number of layers above and also on the numbers of the edges in those layers. See Table 5 and Table 6.

Using the conditional probability distributions of the variables given their parents, we can easily calculate joint

[2] In the case of one parent variable, we use probability values as specified in Table 4.

Table 4: $F(X_i^{'}|X_i)$ table

		X_i	
		0	1
$X_i^{'}$	0	$1-\alpha$	0.2
	1	α	0.8

Table 5: N1 (Figure 11): Marginal probability distributions

	C1	C2	C3	C4	C5	C6
$F(C_i=0)$	.5	.6	.68	.744	.795	.837
$F(C_i=1)$	.5	.4	.32	.256	.205	.163

Table 6: N2 (Figure 11): Marginal probability distributions

	C1	C2	C3	C4	C5	C6
$F(C_i=0)$	.5	.6	.536	.539	.716	.707
$F(C_i=1)$	.5	.4	.464	.461	.284	.293

Table 7: BN2O (Figure 12): Marginal probability distributions

	C1	C2	C3	C4	C5	C6	C7	C8
$F(C_i=0)$	.5	.5	.5	.5	.5	.129	.36	.36
$F(C_i=1)$	.5	.5	.5	.5	.5	.871	.64	.64

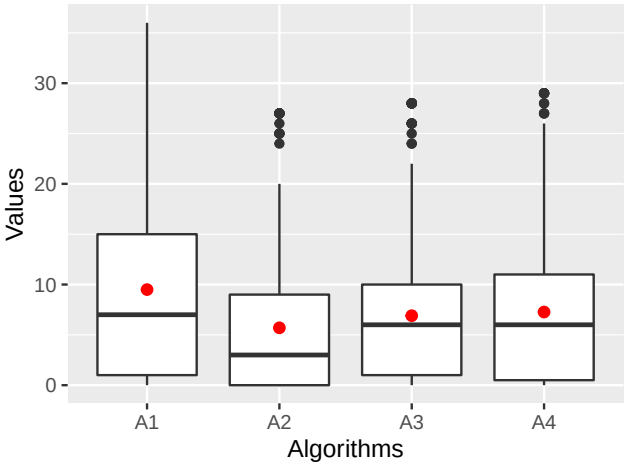


Figure 2: The Structural Hamming Distance to the true models from the resulting models of the structure learning algorithms using data generated from all models, summarized in Table 8 averaged over all data sizes and all MCAR rates.

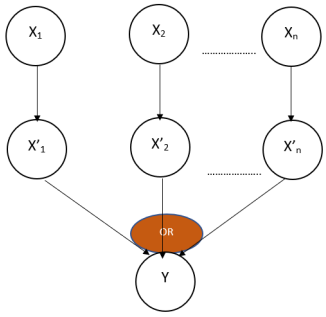


Figure 3: Noisy-or

probability distributions $F(\mathbf{U})$ using formula (1) and conditional probability distributions (CPD) $F(\mathbf{X}_A|\mathbf{X}_B)$, where $\mathbf{X}_A \subseteq \mathbf{U}$ and $\mathbf{X}_B \subseteq \mathbf{U} \setminus \mathbf{X}_A$. Recall that a CPD for a particular configuration $\mathbf{x}_B$ of parent nodes $\mathbf{X}_B$ can be computed as[3]:

$$F(\mathbf{X}_A|\mathbf{X}_B=\mathbf{x}_B) = \frac{F(\mathbf{X}_A,\mathbf{X}_B=\mathbf{x}_B)}{F(\mathbf{X}_B=\mathbf{x}_B)} \quad (5)$$

The Kullback-Leibler Distance (KLD) of two conditional probability distributions $F(\mathbf{X}_A|\mathbf{X}_B)$ and $G(\mathbf{X}_A|\mathbf{X}_B)$ defined on the same state space is computed as the weighted average KLD of $F(\mathbf{X}_A|\mathbf{X}_B=\mathbf{x}_B)$ and $G(\mathbf{X}_A|\mathbf{X}_B=\mathbf{x}_B)$ over all configurations $\mathbf{x}_B$:

$$\begin{aligned} D(F(\mathbf{X}_A|\mathbf{X}_B)||G(\mathbf{X}_A|\mathbf{X}_B)) &= \sum_{\mathbf{x}_B} F(\mathbf{X}_B=\mathbf{x}_B) \\ &* \sum_{\mathbf{x}_A} F(\mathbf{X}_A=\mathbf{x}_A|\mathbf{X}_B=\mathbf{x}_B) \\ &* \log \frac{F(\mathbf{X}_A=\mathbf{x}_A|\mathbf{X}_B=\mathbf{x}_B)}{G(\mathbf{X}_A=\mathbf{x}_A|\mathbf{X}_B=\mathbf{x}_B)} \\ &= \sum_{\mathbf{x}_A,\mathbf{x}_B} F(\mathbf{X}_A=\mathbf{x}_A,\mathbf{X}_B=\mathbf{x}_B) \\ &* \log \frac{F(\mathbf{X}_A=\mathbf{x}_A|\mathbf{X}_B=\mathbf{x}_B)}{G(\mathbf{X}_A=\mathbf{x}_A|\mathbf{X}_B=\mathbf{x}_B)} \end{aligned}$$

We will use KLD of conditional probability distributions estimated from the true data to support our arguments when we explain the results.

4.2 Experiments

We have performed experiments on different Belief noisy-or (BNO) models with their CPTs defined in Table 4 where $\alpha \in \{0, 0.2\}$ and the CPT of a node which has no parent is uniform, i.e., $F(X_i=1|\{\})=0.5, F(X_i=0|\{\})=0.5$. The experiments have been repeated ten times on ten different datasets generated from BNO models with different MCAR rates as specified in Table 9 in Appendix A. In all Figures, we will denote additional edges by blue dashed lines, missing edges by red lines, and edges with different arrows by orange lines.

[3] Please, note that all BNs considered in this paper satisfy the condition $F(\mathbf{X}_B=\mathbf{x}_B)>0$ for all $\mathbf{x}_B$.

4.2.1 Model N1

The true N1 model is shown in Figure 11 in Appendix A. We use this model as an example of a simple model with a chain structure. This model is motivated by some applications, e.g., from telecommunications. Let us summarize the results of this model:

- All algorithms learn the true structure when $\alpha \neq 0$ in all data sizes and all MCAR rates.
- The algorithms A2, A3, and A4 learn structures different from the true model in some cases with $\alpha = 0$, MCAR rate 15% and data size of 1,000. For example, A3 and A4 learn additional edge $C2 \rightarrow C4$, also, A2 learn $C4 \rightarrow C6$ instead of $C5 \rightarrow C6$.
- Using equation (5), we calculate $F(C6|C5)$ and $F(C6|C4)$ from the true model N1. We have found that their KLD value (computed using equation (4.1)) is very small, it is only 0.001. Also, the chi-square test of independence, whose p-value is smaller than 0.0001, reveals that there is a strong dependence between $C6$ and $C4$ in addition to the relationship between $C6$ and $C5$, already explicitly present in the true model. Also, the BIC of the learned structure[4] is -2,252.93 and the BIC of the true model from the same dataset is -2,255.64. This can be explained by the deterministic conditional distribution $F(C6|C5 = 0)$ for $C5 = 0$. For these reasons, we can conclude that we can accept that A2 has learned $C4 \rightarrow C6$ instead of $C5 \rightarrow C6$.

4.2.2 Model N2

The true N2 model is shown on the right hand side of Figure 11 in Appendix A. We use this model as an example of a model more complicated than the previous model N1. This model is motivated by some applications, e.g., by computer networks. We summarize the results of the experiments performed with this model:

- Figure 4 represents the Structure Hamming Distance (SHD) for all tested MCAR rates and models with $\alpha = 0$. We can observe that, as expected, the algorithm's performance is getting better with increasing the data size.
- We can see that A2 on average has a smaller SHD distance to the true model than other algorithms.
- In Figure 5, we compare the models learned from the datasets of size 5,000 with MCAR rate 10% using all four algorithms. We can see that A2 and A3 have the same SHD but they differ in that A3 has a missing edge $C4 \rightarrow C5$ while A2 has an additional edge $C3 \rightarrow C6$. This additional edge can be explained by observing that there is a chain of nodes $C3 \rightarrow C4 \rightarrow C6$ which the state 0 is propagated through because of $\alpha = 0$. In other words, we calculate $F(C3, C6|C1, C2, C4, C5)$ and the product $F(C3|C1, C2, C4, C5) \cdot F(C6|C4, C5)$ from the true model 11 using equation (5). The KLD value (computed as explained in (4.1)) of these two distributions is very small, it is only 0.02. Also, the chi-square test of independence of $C3$ and $C6$ reveals these variables are dependent (the test's p-value is smaller than 0.0001). The additional edge can be also supported by a comparison of BIC values of the learned structure with and without the additional edge $C3 \rightarrow C6$; they are -9,813.67 for the model with the additional edge and -9,880.5 for the true model.
- If $\alpha > 0$ then no additional edge is learned anymore, no matter what the MCAR rate is. Algorithms A2 and A4 we are always able to learn the true structure when the data size exceeds 1,000. Also, A1 and A3 learn the true structure when the data size is larger than 1,500.

[4]We report the BIC value for one of ten datasets since the results for the remaining nine are similar.

4.2.3 BN2O models

These models are motivated by health-care applications, for example by the QMR-DT network [17]. We created 60 different BN2O models consisting of two layers with $N = 20$ nodes in total. They differ in the numbers of nodes in the first layer, namely $L_1 \in \{5, 8, 12, 15\}$; the numbers of nodes in the second layer are $L_2 = 20 - L_1$. The numbers of edges between layers are generated randomly with three different options $\frac{N}{2}$, $\frac{2 \cdot N}{2}$, and $\frac{4 \cdot N}{2}$; each option repeated five times. Using these models, we have generated multiple incomplete datasets where the sizes of the datasets are 3,000 and 5,000 and the MCAR are 10% and 15%. Figure 8 shows the boxplot of additional and missing numbers of edges learned in all instances for each algorithm where the dataset size is 3,000, and for all MCAR rates. The results show that A2 has better results on average (i.e., the distance to the true model is smaller) than other algorithms.

Next we discuss in more detail one simpler example of a BN2O. The structure of this model is shown in Figure 12 in Appendix A.

- Figure 6 represents the SHD of all learned models grouped by MCAR rates with models where $\alpha = 0$.
- The learned models from the dataset of size 5,000, MCAR rate 10% and $\alpha = 0$ using all algorithms are shown in Figure 7. We can see that A2 performs better (i.e., the SHD distance to the true model is smaller) than other algorithms.
- Note the additional edge $C7 \rightarrow C8$ learned by A2 for most datasets. The argument supporting this additional edge is similar to that valid for the additional edge in the N2 model. Again, we can see that KLD of $F(C7, C8|C2, C5)$ and the product $F(C7|C2, C5).F(C8|C2, C5)$ is very small; it is only 0.002. Also, the chi-square test of independence of $C7$

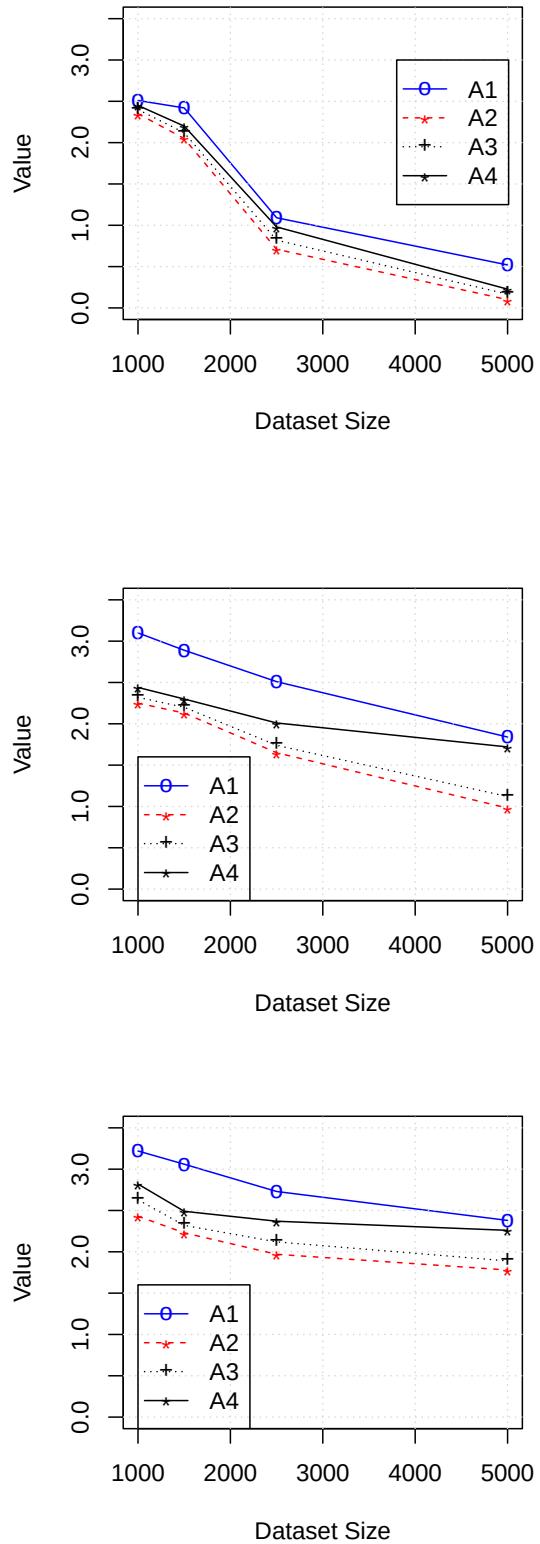


Figure 4: The Structural Hamming Distance of the resulting models of the structure learning algorithms to the true model (with $\alpha = 0$) using the data generated from the N2 model (the true model is presented in Figure 11) using the average over ten experiments for different data sizes and for the MCAR rates of 5%, 10%, and 15%, respectively.

and $C8$ has the p-value smaller than 0.0001 and there is always only a very small difference between BIC of the model with the extra edge and the true model; for example,, BIC of the model with the extra edge is -7,331.8 while the BIC of the true model is -7,338.5 for one of the en generated datasets.

– In the experiments with models having $\alpha > 0$ no additional edge has been learned and the true model is learned successfully when the data size is 2,500 or larger for all MCAR rates.

4.2.4 A large BN2O model

We have performed experiments with a model shown in Figure 13 in Appendix A. This model consists of 25 variables; 14 in the first layer and 11 in the second layer. All algorithms required a data size of more than 3,000 to give a good performance. With the data size of 3,000 (and the MCAR rate of 10%) the recorded SHD of algorithms A1, A2, A3, and A4 still have not been very good – namely, 14.6, 10.2, 10, and 9.8, respectively. With the data size of 5000 and 7500 (and the MCAR rate of 10%) the recorded average SHD of A1, A2, A3, and A4 are already much better – namely, 7.2, 4.2, 4.3, and 5.1, respectively. See Figure 9 for the learned models. With the data size of 10,000 we already get the true models except for the additional edges in the case of A2, as discussed in Section 4.2.3.

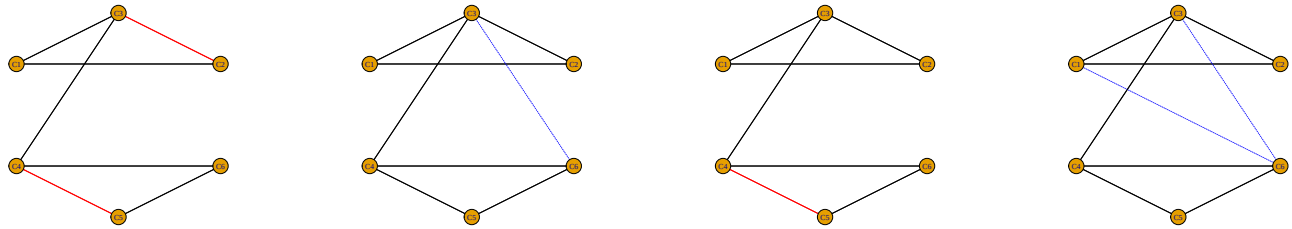


Figure 5: Models learned by A1, A2, A3, and A4, respectively, for most of ten datasets generated from the true N2 model (presented in Figure 11) (for $\alpha = 0$) with the MCAR rate 10 and the data size of 5,000.

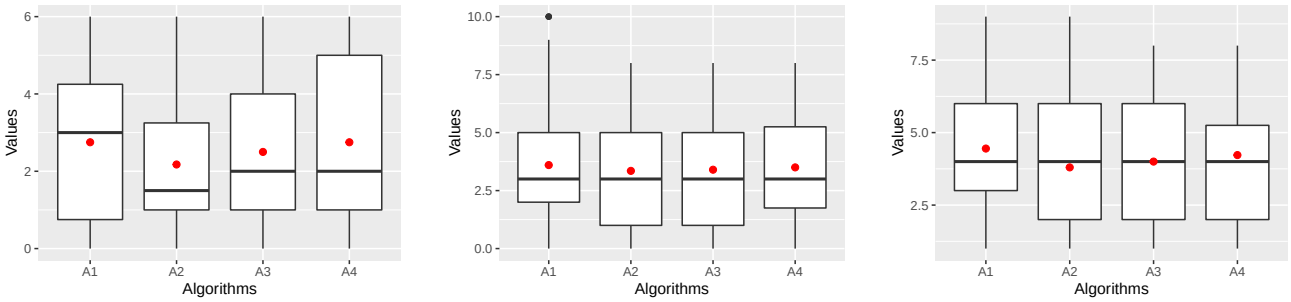


Figure 6: The Structural Hamming Distance to the true models of the resulting models of the structure learning algorithms using data generated from the BN2O model (the true model is presented in Figure 12) (with $\alpha = 0$) averaged over all data sizes for MCAR rates of 5%, 10%, and 15%, respectively.

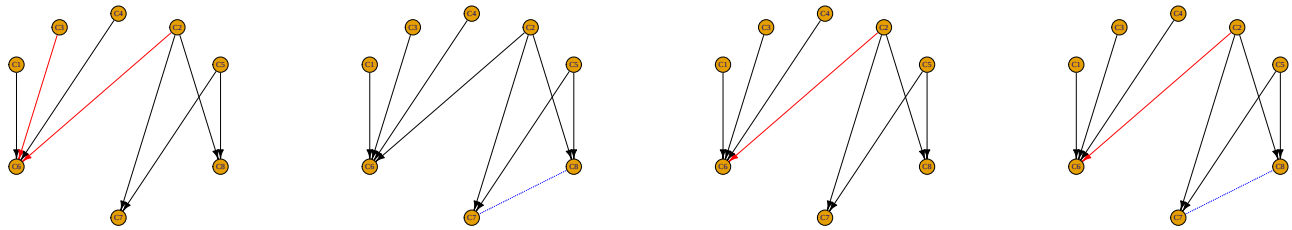


Figure 7: Models learned by A1, A2, A3, and A4, respectively, using data generated for most of ten datasets generated from the true BN2O model (presented in Figure 12) (for $\alpha = 0$) with the MCAR rate 10 and the data size of 5,000.

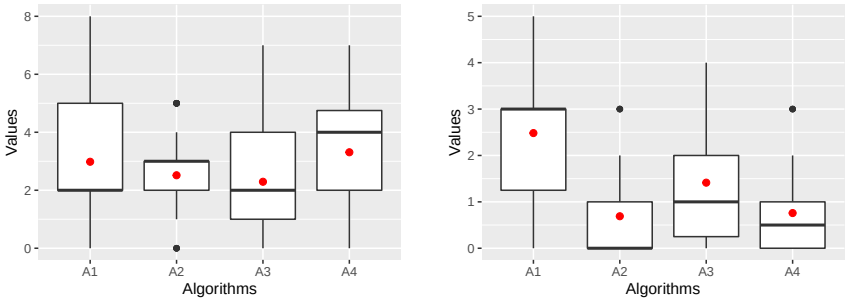


Figure 8: Results of the structure learning algorithms using data generated from the BN2O model (with $\alpha = 0$) with the data size of 3,000 and averaged over all tested MCAR rates. The plot on LHS displays the average number of additional edges and the plot on RHS displays the average number of missing edges.

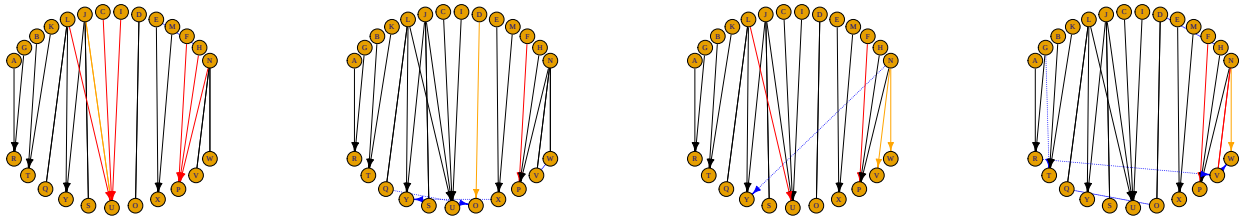


Figure 9: Models learned by A1, A2, A3, and A4, respectively, using the data generated from the large BN2O model consisting of 25 variables (for $\alpha = 0$) with the MCAR rate of 10% and the data size of 7,500 (true model is presented in Figure 13).

5 Conclusion

In this paper, we provide an approach to learning the optimal BN structure from incomplete data by adapting the considerations of [8]. This adaptation imputes missing values using product mixtures learned by the EM algorithm [11]. We have shown that the sequence of log-likelihood values generated by E-Step and M-Step of the EM algorithm is non-decreasing and the algorithm converges. Theorem 2.1 helps us reduce the collection of candidate parent sets for a variable, which can speed-up the learning algorithm.

We have performed experiments on incomplete data generated from different types of BN models to compare the proposed Algorithm A2 with other algorithms, namely with A1 [8], soft and hard EM [3], referred to as A3 and A4, respectively. In our comparisons, we use Structure Hamming Distance of CPDAGs of learned DAGs to CPDAGs of the original models.

Such comparisons have been undertaken on (a) general Bayesian networks and (b) Belief Noisy-or [4] (BNO) models with partially deterministic and nondeterministic conditional probability distributions. The experiments with models of type (b) are motivated by uncertain relationships in Bayesian networks, which are common in practical applications of BNs. We have obtained the following results in detailed simulation studies.

(a) General BN models:

- The A2 algorithm appears to be the best choice from among the tested algorithms for learning the structure of BNs from any incomplete data whatever the data size and the missing MCAR rate are.
- In most scenarios corresponding to different datasizes and MCAR rates, Algorithm A2 is significantly better than other algorithms and in no scenario is it significantly worse than any other algorithm according to the Wilcoxon test.

(b) BNO models:

- A2 is able to recover all true edges in the tested models except for the N1 model (shown in Figure 11) at size 1,000 and a missing rate of 15%. The different learned structure of the model N1 is acceptable because the Chi-square($\mathscr{X}^2$) test and the Kullback-Leibler distance (KLD) between the related conditional probabilities suggest there is a high degree of relationship between the connected variables.

- A2 has learned an additional edge in the case of Models N2 (shown in Figure 11) and BN2O (shown in Figure 12). The additional edge is acceptable since the $\mathscr{X}^2$ test and KLD suggest there is a high degree of relationship between these variables. We have seen that BIC of the learned structure is almost equal to BIC of the true model. For example, BIC of the model learned using A2 (shown in Figure 7) is -7,331.8 and BIC of the true model is -7,338.5. Similar behavior has been observed in other BNO models.

- A2 is always able to recover all edges while other algorithms are not.

- For large BN2O models, all algorithms require data sizes large than 3000 to have a good performance; e.g., for the BN2O with 25 variables A2 needs at least 10,000 data records to learn the correct model (with the exception of additional edges as discussed in Section 4.2.3).

We have empirically shown that our Algorithm A2 behaves better than other tested algorithms on several studied BNs and in different scenarios. Based on these experiments, we can recommend this algorithm for practitioners that use BNs or BNOs with incomplete data especially in the medical domain where BNO could be used to study the hidden relationship between symptoms and diseases. An interesting topic for future research might be learning the structure of large BN2O networks from incomplete data and optimize the number of components in the EM-Mixture .

Acknowledgement

This work was supported by Student Grant CTU SGS20/132/OHK4/2T/14

References

[1] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 20(2-3):131—163, 1997. URL: https://doi.org/10.1023/a:1007465528199.

[2] Cassio P de Campos, Mauro Scanagatta, Giorgio Corani, and Marco Zaffalon. Entropy-based pruning for learning Bayesian networks using BIC. *Artificial Intelligence*, 260:42—50, 2018. URL: https://doi.org/10.1016/j.artint.2018.04.002.

[3] Andrea Ruggieri, Francesco Stranieri, Fabio Stella, and Marco Scutari. Hard and soft EM in Bayesian network learning from incomplete data. *Algorithms*, 13(12):329, 2020. https://doi.org/10.3390/a13120329 doi:10.3390/a13120329.

[4] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

[5] Nir Friedman and Moises Goldszmidt. Learning Bayesian networks with local structure. In *Learning in graphical models*, page 421—459. Springer, 1998. URL: https://doi.org/10.1007/978-94-011-5014-9_15.

[6] Zhifa Liu, Brandon Malone, and Changhe Yuan. Empirical evaluation of scoring functions for Bayesian network model selection. In *Proceedings of the Ninth Annual MCBIOS Conference. Dealing with the Omics Data Deluge*, Oxford, MS, USA., 2012. BMC Bioinformatics. https://doi.org/10.1186/1471-2105-13-S15-S14 doi:10.1186/1471-2105-13-S15-S14.

[7] Poh Choo Song, Hui Yee Chong, Hong Choon Ong, and Sing Yan Looi. A model of Bayesian network analysis of the factors affecting student's higher level study decision: The private institution case. *journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 8(2):105—109, 2016.

[8] Cassio P de Campos, Zhi Zeng, and Qiang Ji. Structure learning of Bayesian networks using constraints. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 113—120, New York, NY, USA, 2009. Association for Computing Machinery. https://doi.org/10.1145/1553374.1553389 doi:10.1145/1553374.1553389.

[9] James Cussens. Bayesian network learning with cutting planes. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, page 153—160, Arlington, Virginia, USA, 2011. AUAI Press.

[10] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. B*, 39:1—38, 1977. URL: https://doi.org/10.1111/j.2517-6161.1977.tb01600.x.

[11] Jirí Grim, Jan Hora, Pavel Boček, Petr Somol, and Pavel Pudil. Statistical model of the 2001 Czech census for interactive presentation. *Journal of Official Statistics*, 26(4):673—694, 2010.

[12] J. Grim and P. Boček. Statistical model of prague households for interactive presentation of census data. In *SoftStat 95. Advances in Statistical Software 5. Conference on the Scientific Use of Statistical Software*, Heidelberg, DE, 1996.

[13] Luca Scrucca, Michael Fop, T. Brendan Murphy, and Adrian E. Raftery. mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1):289—317, 2016.

[14] Fred Glover. Tabu search-part I. *ORSA Journal on computing*, 1(3):190—206, 1989.

[15] Marco Scutari and Jean-Baptiste Denis. *Bayesian Networks: with Examples in R*. Chapman & Hall, Boca Raton, 2014. URL: https://doi.org/10.1111/biom.12856.

[16] M Neuhäuser and Mann-Whitney Test. *International Encyclopedia of Statistical Science*. Springer Berlin Heidelberg, 2011.

[17] Michael A Shwe, Blackford Middleton, David E Heckerman, Max Henrion, Eric J Horvitz, Harold P Lehmann, and Gregory F Cooper. Probabilistic diagnosis using a reformulation of the internist-1/qmr knowledge base. *Methods of information in Medicine*, 30(04):241—255, 1991. URL: https://doi.org/10.1055/s-0038-1634846.

[18] B Abramson, J Brown, Ward E, Allan Murphy, and Robert L Winkler. Hailfinder: A bayesian system for forecasting severe weather. *International Journal of Forecasting*, 12(1):57–71, 1996. Probability Judgmental Forecasting. URL: https://doi.org/10.1016/0169-2070(95)00664-8.

[19] A Philip Dawid. Prequential analysis, stochastic complexity and Bayesian inference. *Bayesian statistics*, 4:109—125, 1992.

A Appendix A. Simulation Scenarios

This Appendix provides an inclusive list of all experiments in the simulation study described in Sections 3.2 and 4.2, organized by their main characteristics in Tables 8 and 9, respectively. The number of components in each experiment selected based on the number of variables in the datasets. The true models mentioned in the Table 8 are shown in Figure 10. The true models mentioned in the Table 9 are shown in Figures 11 and 12.

Table 8: Description of the key factors of all BN experiments in the simulation study.

Network	Missing Rate (MCAR)	Replicates	Sample Size
Weather [18]	10	10	100, 500,1000,5000,10000
	25	10	100, 500,1000,5000,10000
	50	10	100, 500,1000,5000,10000,13000
Child [19]	10	10	1000, 2000,3000,5000
	15	10	1000, 2000,3000,5000
	50	10	1000, 2000,3000,5000
M2 (Figure 10)	5	10	500,1000,1500,2500,5000
	10	10	500,1000,1500,2500,5000
	15	10	500,1000,1500,2500,5000
	25	10	500,1000,1500,2500,5000
M1 (Figure 10)	10	10	500,1500,2500,5000,10000,13000
	20	10	500,1500,2500,5000,10000,13000
	35	10	500,1500,2500,5000,10000,13000
	50	10	500,1500,2500,5000,10000,13000

Table 9: Description of the key factors of all Belief Noisy-OR experiments in the simulation study (true models are presented in Figures 11 and 12).

Network	Missing Rate (MCAR)	Replicates	Sample Size
BN2O	5	10	1000,1500,2500,5000
	10	10	1000,1500,2500,5000
	15	10	1000,1500,2500,5000
N1	5	10	1000,1500,2500,5000
	10	10	1000,1500,2500,5000
	15	10	1000,1500,2500,5000
N2	5	10	1000,1500,2500,5000
	10	10	1000,1500,2500,5000
	15	10	1000,1500,2500,5000
large BN2O	10	10	5000, 7500

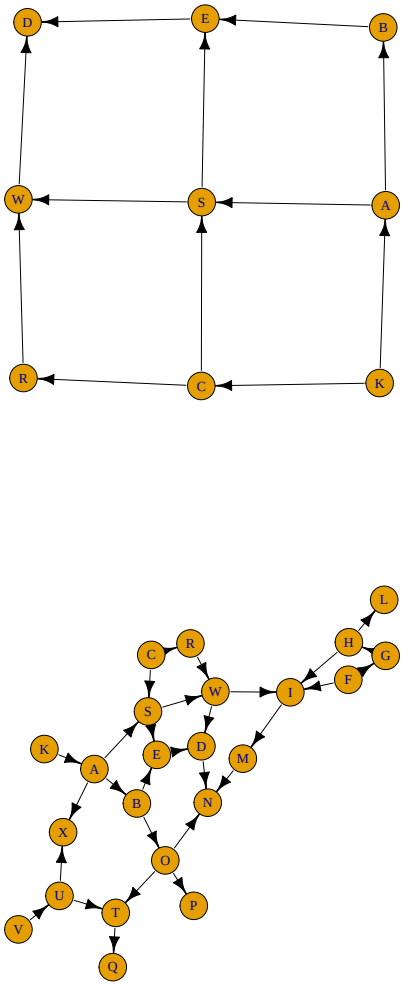
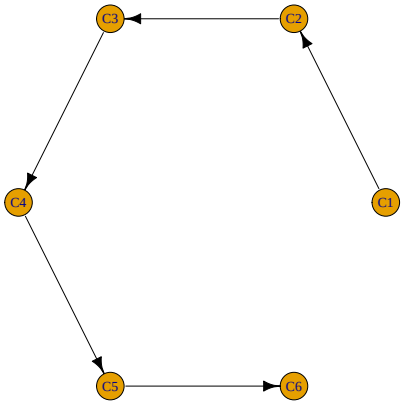


Figure 10: M1 and M2 true models, respectively



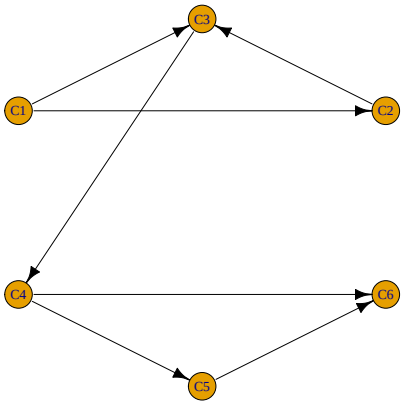


Figure 11: N1 and N2 true models, respectively. Their marginal probability distributions are summarized in Tables 5 and 6

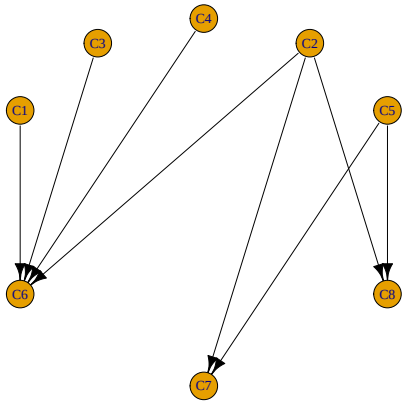


Figure 12: BN2O true model. Its marginal probability distributions are summarized in Table 7

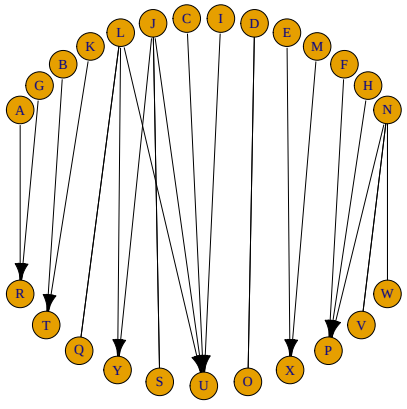


Figure 13: Example of a large BN2O model with 25 variables (whose learned models are presented in Figure 9).