entropy

MDPI

Article

Polynomial-Time Algorithm for Learning Optimal BFS-Consistent Dynamic Bayesian Networks

Margarida Sousa and Alexandra M. Carvalho *

Instituto de Telecomunicações, Instituto Superior Técnico, Universidade de Lisboa, 1049-001 Lisboa, Portugal; margarida.sousa@tecnico.ulisboa.pt

* Correspondence: alexandra.carvalho@tecnico.ulisboa.pt; Tel.: +351-218-418-454

Received: 22 March 2018; Accepted: 10 April 2018; Published: 12 April 2018

Abstract: Dynamic Bayesian networks (DBN) are powerful probabilistic representations that model stochastic processes. They consist of a prior network, representing the distribution over the initial variables, and a set of transition networks, representing the transition distribution between variables over time. It was shown that learning complex transition networks, considering both intra- and inter-slice connections, is NP-hard. Therefore, the community has searched for the largest subclass of DBNs for which there is an efficient learning algorithm. We introduce a new polynomial-time algorithm for learning optimal DBNs consistent with a breadth-first search (BFS) order, named bcDBN. The proposed algorithm considers the set of networks such that each transition network has a bounded in-degree, allowing for p edges from past time slices (inter-slice connections) and k edges from the current time slice (intra-slice connections) consistent with the BFS order induced by the optimal tree-augmented network (tDBN). This approach increases exponentially, in the number of variables, the search space of the state-of-the-art tDBN algorithm. Concerning worst-case time complexity, given a Markov lag m, a set of n random variables ranging over r values, and a set of observations of N individuals over T time steps, the bcDBN algorithm is linear in N, T and m; polynomial in n and r; and exponential in p and k. We assess the bcDBN algorithm on simulated data against tDBN, revealing that it performs well throughout different experiments.

Keywords: dynamic Bayesian networks; optimum branching; score-based learning; theoretical-information scores

1. Introduction

Bayesian networks (BN) represent, in an efficient and accurate way, the joint probability of a set of random variables [1]. Dynamic Bayesian networks (DBN) are the dynamic counterpart of BNs and model stochastic processes [2]. DBNs consist of a prior network, representing the distribution over the initial attributes, and a set of transition networks, representing the transition distribution between attributes over time. They are used in a large variety of applications such as protein sequencing [3], speech recognition [4] and clinical forecasting [5].

The problem of learning a BN given data consists in finding the network that best fits the data. In a score-based approach, a scoring criterion is considered, which measured how well the network fits the data [6–10]. In this case, learning a BN reduces to the problem of finding the network that maximizes the score, given the data. Methods for learning DBNs are simple extensions of those considered for BNs [2]. Not taking into account the acyclicity constraints, it was proved that learning BNs does not have to be NP-hard [11]. This result can be applied to DBNs, not considering the intra-slice connections, as the resulting unrolled graph, which contains a copy of each attribute in each time slice, is acyclic. Profiting from this result, a polynomial-time algorithm for learning optimal DBN was proposed using the Mutual Information Tests (MIT) [12]. However, none of these algorithms

learns general mth-order Markov DBNs such that each transition network has inter- and intra-slice connections. More recently, a polynomial-time algorithm was proposed that learns both the inter- and intra-slice connections in a transition network [13]. The search space considered, however, is restricted to the tree-augmented network structures, resulting in the so-called tDBN.

By looking into lower-bound complexity results for learning BNs, it is known that learning tree-like structures is polynomial [14]. However, learning 2-polytrees is already NP-hard [15]. Learning efficiently structures richer than branchings (a.k.a. tree-like structures) has eluded the community, that resorted to use heuristic approaches. Carvalho et al. [16] suggested to search over graphs consistent with the topological order of an optimal branching. The advantage of this approach is that the search space increased exponentially with respect to branchings, while keeping the learning complexity in polynomial time. Later, the breadth-first search (BFS) order of an optimal branching was also considered [17], further improving the previous results in terms of search space.

In this paper, we propose a generalization of the tDBN algorithm, considering DBNs such that each transition network is consistent with the order induced by the BFS order of the optimal branching of the tDBN network, that we call bcDBN. Furthermore, we prove that the search space increases exponentially, in the number of attributes, comparing with the tDBN algorithm, while running in polynomial time.

We start by reviewing the basic concepts of Bayesian networks, dynamic Bayesian networks and their learning algorithms. Then, we present the proposed algorithm and the experimental results. The paper concludes with a brief discussion and directions for future work.

2. Bayesian Networks

Let X denote a discrete random variable that takes values over a finite set $\mathcal{X}$. Furthermore, let $\mathbf{X} = (X_1, \ldots, X_n)$ represent an n-dimensional random vector, where each X_i takes values in $\mathcal{X}_i = \{x_{i1}, \ldots, x_{ir_i}\}$, and $P(\mathbf{x})$ denotes the probability that $\mathbf{X}$ takes the value $\mathbf{x}$. A Bayesian network encodes the joint probability distribution of a set of n random variables $\{X_1, \ldots, X_n\}$ [1].

Definition 1 (Bayesian Network). *A n-dimensional Bayesian Network (BN) is a triple $B = (\mathbf{X}, G, \Theta)$, where:*

- $\mathbf{X} = (X_1, \ldots, X_n)$ *and each random variable X_i takes values in the set $\{x_{i1}, \ldots, x_{ir_i}\}$, where x_{ik} denotes the k-th value X_i takes.*
- $G = (\mathbf{X}, E)$ *is a directed acyclic graph (DAG) with nodes in $\mathbf{X}$ and edges E representing direct dependencies between the nodes.*
- $\Theta = \{\Theta_{ijk}\}_{i \in 1 \ldots n, j \in 1 \ldots q_i, k \in 1 \ldots .., r_i}$ *encodes the parameters of the network G, a.k.a. conditional probability tables (CPT):*

$$\Theta_{ijk} = P_B(X_i = x_{ik} | \Pi_{X_i} = w_{ij}), \tag{1}$$

where Π_{X_i} denotes the set of parents of X_i in the network G and w_{ij} is the j-th configuration of Π_{X_i}, among all possible configurations given by $\{w_{i1}, \ldots, w_{iq_i}\}$, with $q_i = \prod_{X_j \in \Pi_{X_i}} r_j$ denoting the total number of parent configurations.

A BN B induces a unique joint probability distribution over $\mathbf{X}$ given by:

$$P_B(X_1, \ldots, X_n) = \prod_{i=1}^{n} P_B(X_i | \Pi_{X_i}). \tag{2}$$

Let N_{ijk} be the number of instances in data set D of size N, where variable X_i takes the value x_{ik} and the set of parents Π_{X_i} takes the configuration w_{ij}. Denote the number of instances in D where the set of parents Π_{X_i} takes the configuration w_{ij} by

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk}.$$

Observe that,

$$X_i|\Pi_{X_i} \sim \text{Multinomial}(N_{ij}, \theta_{ij1}, \ldots, \theta_{ijr_i}),$$

for $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, q_i\}$.

Intuitively, the graph of a BN can be viewed as a network structure that provides the skeleton for representing the joint probability compactly in a factorized way, and making inferences in the probabilistic graphical model provides the mechanism for gluing all these components back together in a probabilistic coherent manner [18].

An example of a BN is depicted in Figure 1. It describes cash compensation and overnight accommodation to air passengers in the event of long flight delays. A flight may be delayed due to aircraft maintenance problems or severe weather (hurricane, blizzard, etc.). Whenever the delay is not caused by an external event to the airline company, a passenger may be entitled to a monetary compensation. Regardless of the cause, if the delay is long enough, the passenger might be offered an overnight accommodation. As a result of the dependences encoded by the graph, the joint probability distribution of the network can be factored as

$$P(M, S, F, O, C) = P(M)P(S)P(F|M, S)P(O|F)P(C|F, S),$$

where only the first letter of a variable name is used: M—Maintenance problems; S—Severe weather; F—Flight delay; O—Overnight accommodation; and C—Cash compensation. In this simple example, all variables are Bernoulli (ranging over `T` and `F`). Inside the callouts only the CPTs for variables taking the value `T` are given.

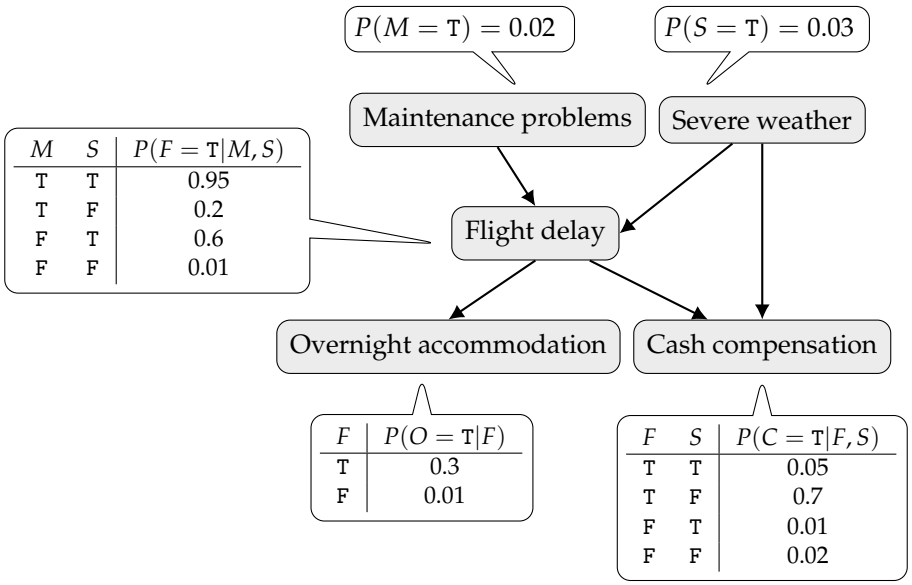


Figure 1. A BN example regarding airline regulations with conditional probability tables.

3. Learning Bayesian Networks

Learning a Bayesian network is two-fold: parameter learning and structure learning. When learning the parameters, we assume the underlying graph G is given, and our goal is to estimate the set of parameters of the network Θ. When learning the structure, the goal is to find a structure G, given only the training data. We assume data is complete, i.e., each instance is fully observed, there are no missing values nor hidden variables, and the training set $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is given by a set of N i.i.d. instances. Using general results of the maximum likelihood estimate in a multinomial distribution we get the following estimate for the parameters of a BN B:

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}, \tag{3}$$

that is denoted by observed frequency estimate (OFE).

In score-based learning, a scoring function $\phi : \mathcal{S} \times \mathcal{X} \to \mathbb{R}$ is required to measure how well a BN B fits the data D (where $\mathcal{S}$ denotes the search space). In this case, the learning procedure can be extremely efficient if the employed score is decomposable. A scoring function ϕ is said to be decomposable if the score can be expressed as a sum of local scores that depends only on each node and its parents, that is, in the form:

$$\phi(B, D) = \sum_{i=1}^{n} \phi_i(\Pi_{X_i}, D).$$

Well-known decomposable scores are divided in two classes: Bayesian and information-theoretical. Herein, we focus only on two information-theoretical criteria, namely Log-Likelihood (LL) and Minimum Description Length (MDL) [19]. Information-theoretical scores are based on the compression achieved to describe the data, given an optimal code induced by a probability distribution encoded by a BN.

The LL is given by:

$$LL(B|D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log(\theta_{ijk}). \tag{4}$$

This criterion favours complete network structures, and does not generalize well, leading to the overfitting of the model to the training data. The MDL criterion, proposed by Rissanen [19], imposes that the parameters of the model, ignored in the LL score, must also be accounted. The MDL score for learning BNs is defined by:

$$MDL(B|D) = LL(B|D) - \frac{1}{2} \ln(N)|B|, \tag{5}$$

where $|B|$ corresponds to the number of parameters Θ of the network, given by:

$$|B| = \sum_{i=1}^{n} (r_i - 1) q_i. \tag{6}$$

The penalty introduced by MDL creates a trade off between fitness and model complexity, providing a model selection criterion robust to overfitting.

The structure learning reduces to an optimization problem: given a scoring function, a data set, a search space and a search procedure, find the network that maximizes this score. Denote the set of BNs with n random variables by $\mathcal{B}_n$.

Definition 2 (Learning a Bayesian Network)**.** *Given a data $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and a scoring function ϕ, the problem of learning a Bayesian network is to find a Bayesian network $B \in \mathcal{B}_n$ that maximizes the value $\phi(B, D)$.*

The space of all Bayesian networks with n nodes has a superexponential number of structures, $2^{\mathcal{O}(n^2)}$. Learning general Bayesian networks is a NP-hard problem [20–22]. However, if we restrict the search space $\mathcal{S}$ to tree-like structures [14,23] or to networks with bounded in-degree and a known ordering over the variables [24], it is possible to obtain a global optimal solution for this problem. Polynomial-time algorithms to learn BNs with underlying consistent k-graphs (CkG) [16] and breadth-first search consistent k-graphs (BCkG) [17] network structures were proposed. The sets of CkG and BCkG graphs are exponentially larger, in the number of variables, when compared with branchings [16,17].

Definition 3 (k-graph)**.** *A k-graph is a graph where each node has in-degree at most k.*

Definition 4 (Consistent k-graph)**.** *Given a branching R over a set of nodes V, a graph $G = (V, E)$ is said to be a consistent k-graph (CkG) w.r.t. R if it is a k-graph and for any edge in E from X_i to X_j the node X_i is in the path from the root of R to X_j.*

Definition 5 (BFS-consistent k-graph)**.** *Given a branching R over a set of nodes V, a graph $G = (V, E)$ is said to be a BFS-consistent k-graph (BCkG) w.r.t. R if it is a k-graph and for any edge in E from X_i to X_j the node X_i is visited in breadth-first search (BFS) of R before X_j.*

Observe that the order induced by the optimal branching might be partial, while its BFS order is always total (and refines it). Given a BFS-consistent k-graph, there can only exist an edge from X_i to X_j if X_i is less than or as deep as X_j in R. We assume that if $i < j$ and X_i and X_j are at the same level, then the BFS over R reaches X_i before X_j. An example is given in Figure 2.

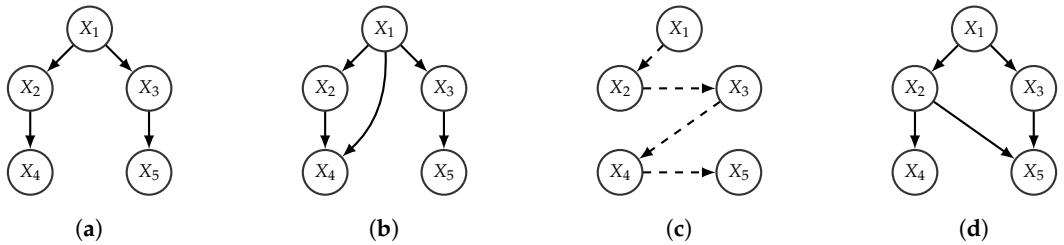


Figure 2. Given the branching R represented in (**a**); (**b**) represents a consistent 2-graph with respect to R; (**c**) represents the BFS of R and (**d**) represents a BFS-consistent 2-graph of R (not consistent with R).

4. Dynamic Bayesian Networks

Dynamic Bayesian networks (DBN) model the stochastic evolution of a set of random variables over time [2]. Consider the discretization of time in time slices given by the set $\mathcal{T} = \{0, \ldots, T\}$. Let $\mathbf{X}[t] = (X_1[t], \ldots, X_n[t])$ be a random vector that denotes the value of the set of attributes at time t. Furthermore, let $\mathbf{X}[t_1 : t_2]$ denote the set of random variables $\mathbf{X}$ for the interval $t_1 \leq t \leq t_2$. Consider a set of individuals $\mathcal{H}$ measured over T sequential instants of time. The set of observations is represented as $\{\mathbf{x}^h[t]\}_{h \in \mathcal{H}, t \in \mathcal{T}}$, where $\mathbf{x}^h[t] = (x_1^h, \ldots, x_n^h)$ is a single observation of n attributes, measured at time t and referring to individual h.

In the setting of DBNs the goal is to define a probability joint distribution over all possible trajectories, i.e., possible values for each attribute X_i and instant t, $X_i[t]$. Let $P(\mathbf{X}[t_1 : t_2])$ denote the joint probability distribution over the trajectory of the process from $\mathbf{X}[t_1]$ to $\mathbf{X}[t_2]$. The space of possible trajectories is very large, therefore in order to define a tractable problem it is necessary to make assumptions and simplifications.

Observations are viewed as i.i.d. samples of a sequence of probability distributions $\{P_{\theta[t]}\}_{t \in \mathcal{T}}$. For all individuals $h \in \mathcal{H}$, and a fixed time t, the probability distribution is considered constant, i.e., $\mathbf{x}^h[t] \sim P_{\theta[t]}, h \in \mathcal{H}$. Using the chain rule the joint probability over $\mathbf{X}$ is given by:

$$P\big(\mathbf{X}[0:T]\big) = P\big(\mathbf{X}[0]\big) \prod_{t=0}^{T-1} P\big(\mathbf{X}[t+1] | \mathbf{X}[0:t]\big).$$

Definition 6 (mth-Order Markov assumption)**.** *A stochastic process over $\mathbf{X}$ satisfies the mth-order Markov assumption if, for all $t \geq 0$*

$$P\big(\mathbf{X}[t+1] | \mathbf{X}[0:t]\big) = P\big(\mathbf{X}[t+1] | \mathbf{X}[t-m+1:t]\big). \tag{7}$$

In this case m is called the Markov lag of the process.

If all conditional probabilities in Equation (7) are invariant to shifts in time, that is, are the same for all $t \in \mathcal{T}$, then the stochastic process is called a stationary mth-order Markov process.

Definition 7 (First-order Markov DBN)**.** *A non-stationary first-order Markov DBN consists of:*

- *A prior network B^0, which specifies a distribution over the initial states $\mathbf{X}[0]$.*
- *A set of transition networks B_t^{t+1} over the variables $\mathbf{X}[t : t+1]$, representing the state transition probabilities, for $0 \leq t \leq T-1$.*

We denote by G_{t+1} the subgraph of B_t^{t+1} with nodes $\mathbf{X}[t+1]$, that contains only the intra-slice dependencies. The transition network B_t^{t+1} has the additional constraint that edges between slices (inter-slice connections) must flow forward in time. Observe that in the case of a first-order DBN a transition network encodes the inter-slice dependencies (from time transitions $t \rightarrow t+1$) and intra-slice dependencies (in the time slice $t+1$).

Figure 3 shows an example of a DBN, aiming to infer a driver behaviour. The model describes the state of a car, including its velocity and distance to the following vehicle, as well as, the weather and the type of road (highway, arterial, local road, etc.). In the beginning, the speed depends only if there is a car nearby. After that, the velocity depends on: (i) the previous weather (the road might be icy because it snowed last night); (ii) the current weather (it might be raining now); (iii) how close the car was from another (if it gets too close the driver might need to break); and (iv) the current type of road (with different velocity limits). The current distance to the following car depends on the previous car velocity and on the previous distance to the next vehicle. Figure 4 joins the prior and transition networks and extends the unrolled DBN to a third time slice.

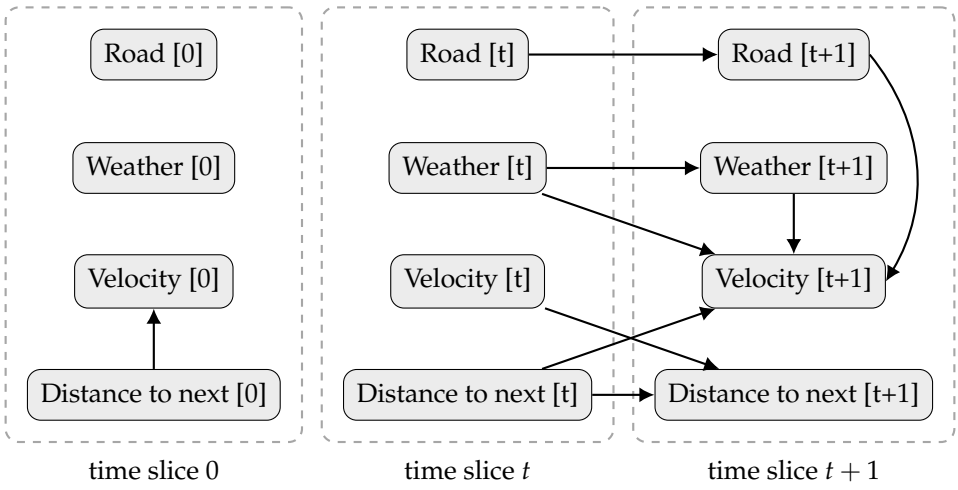


Figure 3. A simple example of a first-order Markov stationary DBN. On the left, the prior network B_0, for $t = 0$. On the right, a two-slice transition network B_t^{t+1}.

Learning DBNs, considering no hidden variables or missing values, i.e., considering a fully observable process, reduces simply to applying the methods described for BNs for each transition of time [25]. Several algorithms for learning DBNs are concerned with identifying inter-slice connections only, disregarding intra-slice dependencies or assuming they are given by some prior network and kept fixed over time [11,12,26]. Recently, a polynomial-time algorithm was proposed that learns both the inter and intra-slice connections in a transition network [13]. However, the search space is restricted to tree-augmented network structures (tDBN), i.e., acyclic networks such that each attribute has one parent from the same time slice, but can have at most p parents from the previous time slices.

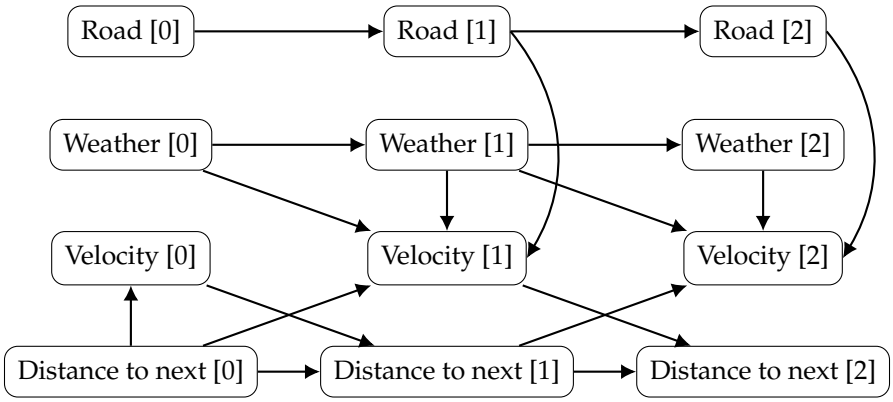


Figure 4. The DBN example from Figure 3 is unrolled for the first three time slices.

Definition 8 (Tree-augmented DBN)**.** *A dynamic Bayesian network is called tree-augmented (tDBN) if for each transition network* $\{t-m+1,\ldots,t\} \rightarrow t+1$ *each attribute* $X_i[t+1]$ *has exactly one parent in the time slice* $t+1$*, except the root, and at most* p *parents from the preceding time slices* $\{t-m+1,\ldots,t\}$*.*

5. Learning Consistent Dynamic Bayesian Networks

We introduce a polynomial-time algorithm for learning DBNs such that: the intra-slice network has in-degree at most k and is consistent with the BFS order of the tDBN; the inter-slice network has in-degree of at most p. The main idea of this approach is to add dependencies that were lost due to the tree-augmented restriction of the tDBN and, furthermore, remove irrelevant ones that might be present because a connected graph was imposed. Moreover, we also consider the BFS order of the intra-slice network as an heuristic for a causality order between variables. We make this concept rigorous with the following definition.

Definition 9 (BFS-consistent k-graph DBN)**.** *A dynamic Bayesian network is called BFS-consistent k-graph (bcDBN) if for each intra-slice network* G_{t+1}*, with* $t \in \{0,\ldots,T-1\}$*, the following holds:*

- G_{t+1} *is a k-graph, i.e., each node has in-degree at most k;*
- *Given an optimal branching* R_{t+1} *over the set of nodes* $\mathbf{X}[t+1]$*, for every edge in* G_{t+1} *from* $X_i[t+1]$ *to* $X_j[t+1]$*, the node* $X_i[t+1]$ *is visited in the BFS of* R_{t+1} *before* $X_j[t+1]$*.*

Moreover, each node $X_i[t+1]$ *has at most* p *parents from previous time slices.*

Before we present the learning algorithm, we need to introduce some notation, namely, the concept of ancestors of a node.

Definition 10 (Ancestors of a node)**.** *The ancestors of a node* X_i *in time slice* $t+1$*, denoted by* $\alpha_{i,t+1}^{BFS}$ *, are the set of nodes in slice* $t+1$ *connecting the root of the BFS of an optimal branching* R_{t+1} *and* $X_i[t+1]$*.*

We will now describe briefly the proposed algorithm for learning a transition network of a mth-order bcDBN. Let $\mathcal{P}_{\leq p}(\mathbf{X}[t-m+1:t])$ be the set of subsets of $\mathbf{X}[t-m+1:t]$ of cardinality less than or equal to p. For each node $X_i[t+1] \in \mathbf{X}[t+1]$, the optimal set of past parents ($\mathbf{X}_{ps} \in \mathcal{P}_{\leq p}(\mathbf{X}[t-m+1:t])$) and maximum score ($s_i$) is found,

$$s_i = \max_{\mathbf{X}_{ps}[t-m+1:t] \in \mathcal{P}_{\leq p}(\mathbf{X}[t-m+1:t])} \phi_i(\mathbf{X}_{ps}[t-m+1:t], D_{t-m+1}^{t+1}), \tag{8}$$

where ϕ_i is the local contribution of $X_i[t+1]$ for the overall score ϕ and D_{t-m+1}^{t+1} is the subset of observations concerning the time transition $t-m+1 \rightarrow t+1$. For each possible edge in $t+1$, $X_j[t+1] \rightarrow X_i[t+1]$, the optimal set of past parents and maximum score (s_{ij}) is determined,

$$s_{ij} = \max_{\mathbf{X}_{ps}[t-m+1:t] \in \mathcal{P}_{\leq p}(\mathbf{X}[t-m+1:t])} \phi_i(\mathbf{X}_{ps}[t-m+1:t] \cup \{X_j[t+1]\}, D^{t+1}_{t-m+1}). \quad (9)$$

We note that the set $\mathbf{X}_{ps}[t-m+1:t]$ that maximizes Equations (8) and (9) needs not to be the same. The one in Equation (8) refers to the best set of p parents from past time slices, and the one in Equation (9) concerns the best set of p parents from the past time slices when $X_j[t+1]$ is also a parent of $X_i[t+1]$.

A complete directed graph is built such that each edge $X_j[t+1] \to X_i[t+1]$ has the following weight,

$$e_{ij} = s_{ij} - s_i, \quad (10)$$

that is, the gain in the network score of adding $X_j[t+1]$ as a parent of $X_i[t+1]$. Generally $e_{ij} \neq e_{ji}$, as the edge $X_i[t+1] \to X_j[t+1]$ may account for the contribution from the inter-slice parents and, in general, inter-slice parents of $X_i[t+1]$ and $X_j[t+1]$ are not the same. Therefore, Edmond's algorithm is applied to obtain a maximum branching for the intra-slice network [27]. In order to obtain a total order, the BFS order of the output maximum branching is determined and the set of candidate ancestors $\alpha^{BFS}_{i,t+1}$ is computed. For node $X_i[t+1]$, the optimal set of past parents $\mathbf{X}_{ps}[t-m+1:t]$ and intra-slice parents, denoted by $\mathbf{X}_{ps}[t+1]$, are obtained in a one-step procedure by finding

$$\max_{\mathbf{X}_{ps}[t-m+1:t] \in \mathcal{P}_{\leq p}(\mathbf{X}[t-m+1:t])} \max_{\mathbf{X}_{ps}[t+1] \in \mathcal{P}_{\leq k}(\alpha^{BFS}_{i,t+1})} \phi_i(\mathbf{X}_{ps}[t-m+1:t] \cup \mathbf{X}_{ps}[t+1], D^{t+1}_{t-m+1}), \quad (11)$$

where $\mathcal{P}_{\leq k}(\alpha^{BFS}_{i,t+1})$ is the set of all subsets of $\alpha^{BFS}_{i,t+1}$ of cardinality less than or equal to k. Note that, if $X_i[t+1]$ is the root, $\mathcal{P}_{\leq k}(\alpha^{BFS}_{i,t+1}) = \{\emptyset\}$, so the set of intra-slice parents $\mathbf{X}_{ps}[t+1]$ of $X_i[t+1]$ is always empty.

The pseudo-code of the proposed algorithm is given in Algorithm 1. As parameters, the algorithm needs: a dataset D, a Markov lag m, a decomposable scoring function ϕ, a maximum number of inter-slice parents p and a maximum number of intra-slice parents k.

Algorithm 1 Learning optimal mth-order Markov bcDBN

1: **for** each transition $\{t-m+1, \ldots, t\} \to t+1$ **do**
2: Build a complete directed graph in $\mathbf{X}[t+1]$.
3: Weight all edges $X_j[t+1] \to X_i[t+1]$ of the graph with e_{ij} as in Equation (10) (Algorithm 2).
4: Apply Edmond's algorithm to the intra-slice network, to obtain an optimal branching.
5: Build the BFS order of the output optimal branching.
6: **for** all nodes $X_i[t+1]$ **do**
7: Compute the set of parents of $X_i[t+1]$ as in Equation (11) (Algorithm 3).
8: **end for**
9: **end for**
10: Collect the transition networks to obtain the optimal bcDBN structure.

The algorithm starts by building the complete directed graph in Step 2, after which the graph is weighted according to Equation (10); this procedure is described in detail in Algorithm 2. The Edmonds' algorithm is then applied to the intra-slice network, resulting from that an optimal branching (Step 4). The BFS order of this branching is computed (Step 5) and the final transition network is redefined to be consistent with it. This is done by computing the parents of $X_i[t+1]$ given by Equation (11) (Steps 6–7), further detailed in Algorithm 3.

Theorem 1. *Algorithm 1 finds an optimal mth-order Markov bcDBN, given a decomposable scoring function ϕ, a set of n random variables, a maximum intra-slice network in-degree of k and a maximum inter-slice network in-degree of p.*

Proof. Let B be the optimal bcDBN and B' be the DBN output of Algorithm 1. Consider without loss of generality the time transition $\{t-m+1,\ldots,t\} \rightarrow t+1$. The proof follows by contradiction, assuming that the score of B' is lower than B. The contradiction found is the following: the optimal branching algorithm applied to the intra-slice graph, Step 4 of Algorithm 1, outputs an optimal branching; moreover, all sets of parents with cardinality of at most k consistent with the BFS order of the optimal branching and all sets of parents from the previous time slices with cardinality of at most p are checked in the for-loop at Step 6. Therefore, the optimal set of parents is found for each node. Finally, note that the selected graph is acyclic since: (i) in the intra-slice network the graph is consistent with a total order (so no cycle can occur); and (ii) in the inter-slice network there are only dependencies from previous time slices to the present one (and not on the other way). □

Algorithm 2 Compute all the weights e_{ij}

for all nodes $X_i[t+1]$ **do**
  Let $s_i = -\infty$.
  for $\mathbf{X}_{ps}[t-m+1:t] \in \mathcal{P}_{\leq p}(\mathbf{X}[t-m+1:t])$ **do**
    if $\phi_i(\mathbf{X}_{ps}[t-m+1:t], D^{t+1}_{t-m+1}) > s_i$ **then**
      Let $s_i = \phi_i(\mathbf{X}_{ps}[t-m+1:t], D^{t+1}_{t-m+1})$.
    end if
  end for
  for all nodes $X_j[t+1] \neq X_i[t+1]$ **do**
    Let $s_{ij} = -\infty$.
    for $\mathbf{X}_{ps}[t-m+1:t] \in \mathcal{P}_{\leq p}(\mathbf{X}[t-m+1:t])$ **do**
      if $\phi_i(\mathbf{X}_{ps}[t-m+1:t] \cup \{X_j[t+1]\}, D^{t+1}_{t-m+1}) > s_{ij}$ **then**
        Let $s_{ij} = \phi_i(\mathbf{X}_{ps}[t-m+1:t] \cup \{X_j[t+1]\}, D^{t+1}_{t-m+1})$.
      end if
    end for
  end for
  Let $e_{ij} = s_{ij} - s_i$.
end for

Algorithm 3 Compute the set of parents of $X_i[t+1]$

Let max $= -\infty$.
for $\mathbf{X}_{ps}[t-m+1:t] \in \mathcal{P}_{\leq p}(\mathbf{X}[t-m+1:t])$ **do**
  for $\mathbf{X}_{ps}[t+1] \in \mathcal{P}_{\leq k}(\alpha^{BFS}_{i,t+1})$ **do**
    if $\phi_i(\mathbf{X}_{ps}[t-m+1:t] \cup \mathbf{X}_{ps}[t+1], D^{t+1}_{t-m+1}) >$ max **then**
      Let max $= \phi_i(\mathbf{X}_{ps}[t-m+1:t] \cup \mathbf{X}_{ps}[t+1], D^{t+1}_{t-m+1})$.
      Let the parents of $X_i[t+1]$ be $\mathbf{X}_{ps}[t-m+1:t] \cup \mathbf{X}_{ps}[t+1]$.
    end if
  end for
end for

Theorem 2. *Algorithm 1 takes time*

$$\max\{\mathcal{O}(n^{p+3}(m+1)^3 m^p r^{p+2} N(T-m+1)), \mathcal{O}(n^{p+k+2} m^p (m+1) r^{p+k+1} N(T-m+1))\},$$

given a decomposable scoring function ϕ, a Markov lag m, a set of n random variables, a bounded in-degree of each intra-slice transition network of k, a bounded in-degree of each inter-slice transition network of p and a set of observations of N individuals over T time steps.

Proof. For each time transition $\{t-m+1,\ldots,t\} \rightarrow t+1$, in order to compute all weights e_{ij} (Algorithm 2), it is necessary to iterate over all the edges, that takes time $\mathcal{O}((n(m+1))^2)$. The number of subsets of parents from the preceding time slices with at most p elements is given by:

$$|\mathcal{P}_{\leq p}(\mathbf{X}[t])| = \sum_{i=0}^{p} \binom{nm}{i} < \sum_{i=0}^{p} (nm)^i \in \mathcal{O}((nm)^p). \tag{12}$$

Calculating the score of each parent set (Step 11 of Algorithm 2), considering that the maximum number of states a variable may take is r, and that each variable has at most $p+1$ parents (p from the past and 1 in $t+1$), the number of possible configurations is given by r^{p+2}. The score of each configuration is computed over the set of observations D_{t-m+1}^{t+1}, therefore taking $\mathcal{O}((m+1)r^{p+2}nN)$. Applying Edmond's optimal branching algorithm to the intra-slice network and computing its BFS order, in Steps 4 and 5, takes $\mathcal{O}(n^2)$ time. Hence, Steps 1–5 take time $\mathcal{O}(n^{p+3}(m+1)^3m^pr^{p+2}N)$. Step 6 iterates over all nodes in time slice $t+1$, therefore iterates $\mathcal{O}(n)$ times. In Algorithm 3, Step 7, the number of subsets with at most p elements from the past and k elements from the present is upper bounded by $\mathcal{O}((nm)^pn^k)$. Computing the score of each configuration takes time complexity of $\mathcal{O}((m+1)nr^{p+k+1}N)$. Therefore Steps 6–9 take time complexity of $\mathcal{O}(n^{p+k+2}m^p(m+1)r^{p+k+1}N)$. Algorithm 1 ranges over all $T-m+1$ time transitions, hence, takes time $\max\{\mathcal{O}(n^{p+3}(m+1)^3m^pr^{p+2}N(T-m+1)), \mathcal{O}((n^{p+k+2}m^p(m+1)r^{p+k+1}N(T-m+1))\}$. □

Theorem 3. *There are at least $2^{(nk-\frac{k^2}{2}-\frac{k}{2}-1)(T-m+1)}$ non-tDBN transition networks in the set of bcDBN structures, where n is the number of variables, T is the number of time steps considered, m is the Markov lag and k is the maximum intra-slice in-degree considered.*

Proof. Consider without loss of generality the time transition $\{t-m+1,\ldots,t\} \rightarrow t+1$ and the optimal branching in $t+1$, R_{t+1}. Let $(V, \subseteq_{BFS})$ be the total order induced by the BFS over R_{t+1}. For any two nodes $X_i[t+1]$ and $X_j[t+1]$, with $i \neq j$, we say that node $X_i[t+1]$ is lower than $X_j[t+1]$ if $X_i[t+1] \subseteq_{BFS} X_j[t+1]$. The i-th node of R_{t+1} has precisely $i-1$ lower nodes. When $i > k$, there are at least 2^k subsets of V with at most k lower nodes. When $i \leq k$, only 2^{i-1} subsets of V with at most k lower nodes exist. Therefore, there are at least

$$\left(\prod_{i=k+1}^{n} 2^k\right) \times \left(\prod_{i=1}^{k} 2^{i-1}\right) = 2^{nk-\frac{k^2}{2}-\frac{k}{2}}$$

BFS-consistent k-graphs.

Let X_R be the root of R_{t+1} and X_j its child node. Let $\emptyset$ denote the empty set. X_R and $\emptyset$ are the only possible ancestors of X_j. If $\emptyset$ is the optimal one, then the resultant graph will not be a tree-augmented network. Therefore there are at least

$$2^{nk-\frac{k^2}{2}-\frac{k}{2}-1}$$

non-tree-augmented graphs in the set of BFS-consistent k-graphs.

There are $T-m+1$ transition networks, hence, there are at least $2^{(nk-\frac{k^2}{2}-\frac{k}{2}-1)(T-m+1)}$ non-tDBN network structures in the set of bcDBN network structures. □

6. Experimental Results

We assess the merits of the proposed algorithm comparing it with one state-of-the-art DBN learning algorithm, tDBN [13]. Our algorithm was implemented in Java using an object-oriented paradigm

and was released under a free software license (https://margaridanarsousa.github.io/learn_cDBN/). The experiments were run on an Intel® Core™ i5-3320M CPU @ 2.60GHz×4 machine.

We analyze the performance of the proposed algorithm for synthetic data generated from stationary first-order Markov bcDBNs. Five bcDBN structures were determined, parameters were generated arbitrarily, and observations were sampled from the networks, for a given number of observations N. The parameters p and k were taken to be the maximum in-degree of the inter and intra-slice network, respectively, of the transition network considered.

In detail, the five first-order Markov stationary transition networks considered were:

- one intra-slice complete bcDBN network with $k = 2$ and at most $p = 2$ parents from the previous time slice (Figure 5a);
- one incomplete bcDBN network, such that each node in $t + 1$ has a random number of inter-slice ($p = 2$) and intra-slice ($k = 2$) parents between 0 and $p + k \leq 4$ (Figure 5b);
- two incomplete intra-slice bcDBN network ($k = 3$) such that each node has at most $p = 2$ parents from the previous time slice (Figure 5c,e);
- one tDBN ($k = 1$), such that each node has at most $p = 2$ parents from the previous time slice (Figure 5d).

The tDBN and bcDBN algorithms were applied to the resultant data sets, and the ability to learn and recover the original network structure was measured. We compared the original and recovered networks using the precision, recall and F_1 metrics:

$$\text{precision} = \frac{TP}{TP + FP}, \text{recall} = \frac{TP}{TP + FN} \text{ and } F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}},$$

where TP are the true positive edges, FP are the false positive edges and FN are the false negative edges.

The results are depicted in Table 1 and the presented values are annotated with a 95% confidence interval, over five trials. The tDBN+LL and tDBN+MDL denote, respectively, the tDBN learning algorithm with LL and MDL criteria. Similarly, the bcDBN+LL and bcDBN+MDL denote, respectively, the bcDBN learning algorithm with LL and MDL scoring functions.

Considering Network 1, the tDBN recovers a significantly lower number of edges, giving raise to lower recalls and similar precisions, when comparing with bcDBN for LL and MDL. The bcDBN+LL and bcDBN+MDL have similar performances. For $N = 2000$, bcDBN+LL and bcDBN+MDL are able to recover in average 99% of the total number of edges.

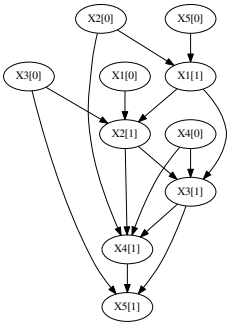
For Networks 2 and 5, considering incomplete networks, the tDBN has again lower recalls and similar precisions than bcDBN. However, in this case, the bcDBN+MDL clearly outperforms bcDBN+LL for all number of instances N considered.

Moreover, in Network 5, taking a maximum intra-slice in-degree $k = 3$, bcDBN only recovers 84% of the total number of edges, for $N = 2000$. These results suggest that a considerable number of observations are necessary to fully reconstruct the complex BFS-consistent k-structures.

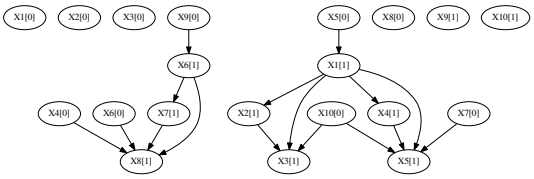
Curiously, the bcDBN+MDL algorithm has better results considering a complete tree-augmented initial structure (Network 4), with higher precision scores and similar recall, comparing with tDBN+MDL.

For both algorithms, in general, the LL gives raise to better results, when considering a complete network structure and a lower number of instances, whereas taking an incomplete network structure and a higher number of instances, the MDL outperforms LL. The complexity penalization term of MDL prevents the algorithms of choosing false positive edges and gives raise to higher precision scores. The LL selects more complex structures, such that each node has exactly $p + k$ parents.

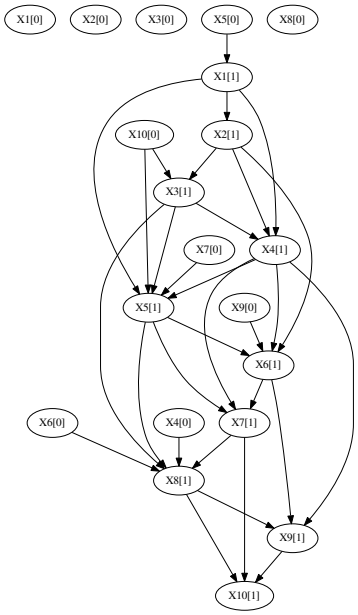
We stress that in all settings considered both algorithms improve their performance when increasing the number of observations N. In order to understand the number of instances N needed to fully recover the initial transition network, we designed two new experiments where five samples where generated from the first-order Markov transition networks depicted in Figure 6.



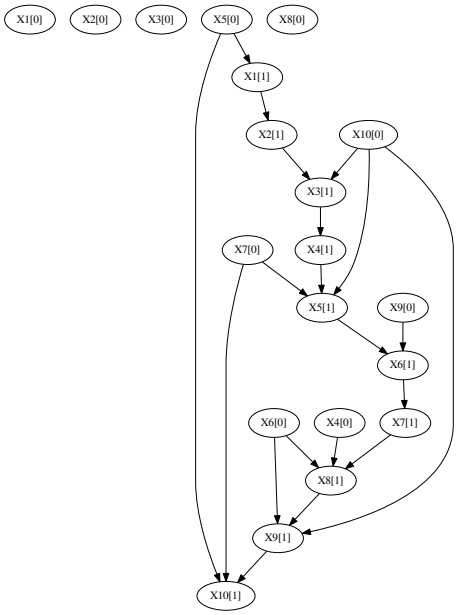
(**a**) Network 1 (bcDBN with $p = k = 2$).



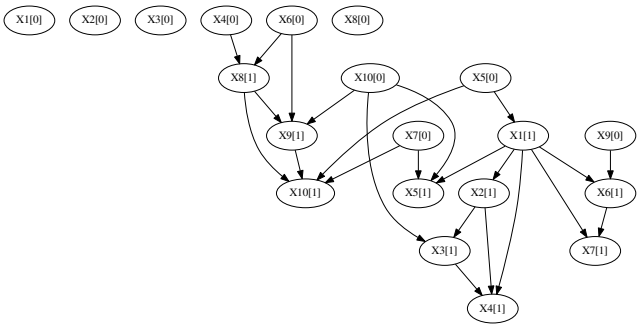
(**b**) Network 2 (bcDBN with $p = k = 2$).



(**c**) Network 3 (bcDBN with $p = 2$ and $k = 3$).



(**d**) Network 4 (tDBN with $p = 2$).

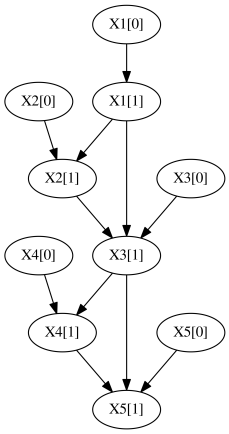


(**e**) Network 5 (bcDBN with $p = 2$ and $k = 3$).

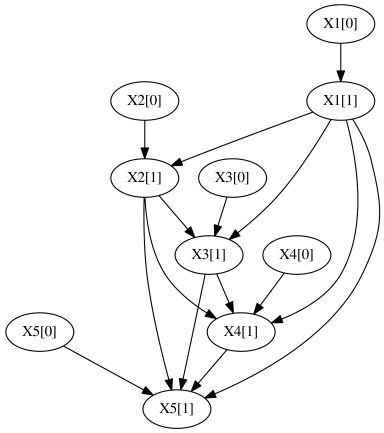
Figure 5. First-order Markov stationary transition networks considered in the experiments.

Table 1. Comparative structure recovery results for tDBN and bcDBN on simulated data. For each network, n is the number of network attributes, p is the maximum inter-slice in-degree, k is the maximum intra-slice in-degree, and r is the number of states of all attributes. On the left, N is the number of observations. Precision (Pre.), recall (Rec.) and F_1-measure (F_1) values are presented as percentages, running time is in seconds.

N	tDBN+LL				tDBN+MDL				bcDBN+LL				bcDBN+MDL			
	Pre.	**Rec.**	F_1	**Time**	**Pre.**	**Rec.**	F_1	**Time**	**Pre.**	**Rec.**	F_1	**Time**	**Pre.**	**Rec.**	F_1	**Time**
Network 1 ($n = 5, p = 2, k = 2, r = 2$)																
100	74 ± 12	69 ± 11	72 ± 12	0	91 ± 7	56 ± 8	69 ± 7	0	74 ± 16	84 ± 18	79 ± 16	0	97 ± 5	43 ± 9	58 ± 9	0
500	83 ± 3	77 ± 3	80 ± 3	0	98 ± 3	73 ± 5	84 ± 4	0	84 ± 5	95 ± 6	89 ± 5	0	98 ± 3	91 ± 8	94 ± 6	0
1000	81 ± 5	76 ± 5	79 ± 5	0	98 ± 3	79 ± 2	87 ± 3	0	85 ± 4	96 ± 5	90 ± 4	0	97 ± 5	96 ± 7	97 ± 6	0
2000	83 ± 5	77 ± 5	80 ± 5	0	95 ± 8	77 ± 5	85 ± 6	0	87 ± 2	99 ± 2	93 ± 2	0	98 ± 4	99 ± 2	98 ± 3	0
Network 2 ($n = 10, p = 2, k = 2, r = 2$)																
100	16 ± 5	29 ± 8	20 ± 6	0	31 ± 9	24 ± 5	27 ± 7	0	18 ± 4	41 ± 9	25 ± 5	0	36 ± 12	13 ± 5	18 ± 7	0
500	28 ± 4	51 ± 7	36 ± 5	0	58 ± 5	45 ± 4	51 ± 4	0	28 ± 6	65 ± 13	39 ± 8	2	81 ± 9	49 ± 8	61 ± 9	2
1000	33 ± 3	60 ± 6	43 ± 4	0	61 ± 5	46 ± 4	52 ± 5	0	32 ± 4	75 ± 8	45 ± 5	4	66 ± 7	55 ± 9	60 ± 8	4
2000	38 ± 2	69 ± 3	49 ± 2	0	72 ± 4	60 ± 3	65 ± 3	0	32 ± 3	75 ± 6	45 ± 4	9	77 ± 12	73 ± 7	74 ± 9	9
Network 3 ($n = 10, p = 2, k = 3, r = 2$)																
100	26 ± 1	26 ± 2	26 ± 2	0	54 ± 13	24 ± 5	33 ± 7	0	25 ± 7	40 ± 11	31 ± 9	1	59 ± 16	24 ± 7	34 ± 10	1
500	43 ± 6	45 ± 6	44 ± 6	0	71 ± 11	39 ± 7	50 ± 8	0	48 ± 6	75 ± 9	58 ± 7	8	75 ± 15	55 ± 14	63 ± 14	8
1000	43 ± 6	44 ± 6	44 ± 6	0	68 ± 7	41 ± 6	51 ± 6	0	47 ± 6	74 ± 9	58 ± 7	18	75 ± 10	61 ± 7	67 ± 8	18
2000	44 ± 5	46 ± 5	45 ± 5	0	77 ± 3	49 ± 1	60 ± 1	0	46 ± 8	72 ± 13	56 ± 10	37	82 ± 3	77 ± 3	80 ± 3	35
Network 4 ($n = 10, p = 2, k = 1, r = 2$)																
100	45 ± 10	65 ± 14	53 ± 11	0	57 ± 7	49 ± 8	52 ± 8	0	45 ± 10	65 ± 14	53 ± 11	0	85 ± 9	47 ± 10	60 ± 9	0
500	58 ± 4	84 ± 5	69 ± 4	0	85 ± 3	86 ± 4	86 ± 3	0	58 ± 4	84 ± 5	69 ± 4	0	100 ± 0	84 ± 3	91 ± 2	0
1000	63 ± 1	92 ± 2	75 ± 2	0	88 ± 2	91 ± 3	90 ± 3	0	63 ± 1	92 ± 2	75 ± 2	0	100 ± 0	88 ± 2	94 ± 1	0
2000	61 ± 4	88 ± 6	72 ± 5	0	87 ± 3	92 ± 2	89 ± 2	0	61 ± 4	88 ± 6	72 ± 5	1	100 ± 0	90 ± 0	95 ± 0	1
Network 5 ($n = 10, p = 2, k = 3, r = 2$)																
100	32 ± 9	43 ± 12	37 ± 10	0	55 ± 14	31 ± 7	39 ± 9	0	22 ± 6	45 ± 12	30 ± 8	1	69 ± 13	19 ± 8	29 ± 11	1
500	49 ± 4	65 ± 6	56 ± 5	0	80 ± 4	57 ± 4	67 ± 4	0	36 ± 4	72 ± 8	48 ± 5	8	92 ± 5	56 ± 2	70 ± 2	9
1000	50 ± 6	66 ± 7	57 ± 6	0	83 ± 6	64 ± 5	72 ± 5	0	40 ± 4	79 ± 9	53 ± 6	16	90 ± 8	71 ± 9	79 ± 8	17
2000	54 ± 6	71 ± 7	61 ± 6	0	86 ± 5	70 ± 5	77 ± 5	0	40 ± 3	81 ± 7	54 ± 5	33	91 ± 6	84 ± 5	87 ± 5	34



(**a**) Network 6 (bcDBN with $p = 1$ and $k = 2$).



(**b**) Network 7 (bcDBN with $p = 1$ and $k = 4$).

Figure 6. Two additional transition networks to test structure recovery in terms of number of observations N.

The number of observations needed for the bcDBN+MDL to recover the aforementioned networks are 1120.0 ± 478.18 (Figure 6a) and 2900.0 ± 1134.77 (Figure 6b), with a 95% confidence interval, where the five trials were done for each network. When increasing k, the number of necessary observations to totally recover the initial structure increases significantly.

When considering more complex BFS-consistent k-structures, the bcDBN algorithm achieved consistently significantly higher F_1 measures than tDBN. As expected, bcDBN+LL obtained better results for complete structures, whereas bcDBN+MDL achieved better results for incomplete structures.

7. Conclusions

The bcDBN learning algorithm has polynomial-time complexity with respect to the number of attributes and can be applied to stationary and non-stationary Markov processes. The proposed algorithm increases the search space exponentially, in the number of attributes, comparing with the state-of-the-art tDBN algorithm. When considering more complex structures, the bcDBN is a good alternative to the tDBN. Although a higher number of observations are necessary to fully recover the transition network structure, bcDBN is able to recover a significantly larger number of dependencies and surpasses, in all experiments, the tDBN algorithm in terms of F_1-measure.

A possible line of future research is to consider hidden variables and incorporate a structural Expectation-Maximization procedure in order to generalize hidden Markov models. Another possible path to follow is to consider mixtures of bcDBNs, both for classification and clustering.

Acknowledgments: This work was supported by national funds through FCT, Fundação para a Ciência e a Tecnologia, under contract IT (UID/EEA/50008/2013), and by projects PERSEIDS (PTDC/EMS-SIS/0642/2014), NEUROCLINOMICS2 (PTDC/EEI-SII/1937/2014), and internal IT projects QBigData and RAPID.

Author Contributions: Alexandra M. Carvalho and Margarida Sousa conceived the proposed algorithm and wrote the paper; Margarida Sousa performed the experimental results.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*; Morgan Kaufmann: San Francisco, CA, USA, 2014.
2. Murphy, K.P.; Russell, S. *Dynamic Bayesian Networks: Representation, Inference and Learning*; University of California: Berkeley, CA, USA, 2002.
3. Yao, X.Q.; Zhu, H.; She, Z.S. A dynamic Bayesian network approach to protein secondary structure prediction. *BMC Bioinform.* **2008**, *9*, 49.
4. Zweig, G.; Russell, S. Speech Recognition with Dynamic Bayesian Networks. Ph.D. Thesis, University of California, Berkeley, CA, USA, 1998.
5. Van Gerven, M.A.; Taal, B.G.; Lucas, P.J. Dynamic Bayesian networks as prognostic models for clinical patient management. *J. Biomed. Inform.* **2008**, *41*, 515–529.
6. Friedman, N.; Geiger, D.; Goldszmidt, M. Bayesian Network Classifiers. *Mach. Learn.* **1997**, *29*, 131–163.
7. Grossman, D.; Domingos, P. Learning Bayesian network classifiers by maximizing conditional likelihood. In Proceedings of the Twenty-First International Conference on Machine Learning, Banff, AB, Canada, 4–8 July 2004; pp. 46–53.
8. Carvalho, A.M.; Roos, T.; Oliveira, A.L.; Myllymäki, P. Discriminative Learning of Bayesian Networks via Factorized Conditional Log-Likelihood. *J. Mach. Learn. Res.* **2011**, *12*, 2181–2210.
9. Carvalho, A.M.; Adão, P.; Mateus, P. Efficient Approximation of the Conditional Relative Entropy with Applications to Discriminative Learning of Bayesian Network Classifiers. *Entropy* **2013**, *15*, 2716–2735.
10. Carvalho, A.M.; Adão, P.; Mateus, P. Hybrid learning of Bayesian multinets for binary classification. *Pattern Recognit.* **2014**, *47*, 3438–3450.
11. Dojer, N. Learning Bayesian networks does not have to be NP-hard. In *International Symposium on Mathematical Foundations of Computer Science*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 305–314.

12. Vinh, N.X.; Chetty, M.; Coppel, R.; Wangikar, P.P. Polynomial time algorithm for learning globally optimal dynamic Bayesian network. In Proceedings of the International Conference on Neural Information Processing, Shanghai, China, 14–17 November 2011.
13. Monteiro, J.L.; Vinga, S.; Carvalho, A.M. Polynomial-time algorithm for learning optimal tree-augmented dynamic Bayesian networks. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), Amsterdam, The Netherlands, 12–16 July 2015; pp. 622–631.
14. Chow, C.; Liu, C. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Inf. Theory* **1968**, *14*, 462–467.
15. Dasgupta, S. Learning Polytrees. In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, 30 July–1 August 1999; pp. 134–141.
16. Carvalho, A.M.; Oliveira, A.L. Learning Bayesian networks consistent with the optimal branching. In Proceedings of the Sixth International Conference on Machine Learning and Applications, Cincinnati, OH, USA, 13–15 December 2007; pp. 369–374.
17. Carvalho, A.M.; Oliveira, A.L.; Sagot, M.F. Efficient learning of Bayesian network classifiers. In Proceedings of the Australasian Joint Conference on Artificial Intelligence, Gold Coast, Australia, 2–6 December 2007.
18. Koller, D.; Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*; MIT Press: Cambridge, MA, USA, 2009.
19. Rissanen, J. *Minimum Description Length Principle*; Wiley Online Library: Hoboken, NJ, USA, 1985.
20. Cooper, G.F. The computational complexity of probabilistic inference using Bayesian belief networks. *Artif. Intell.* **1990**, *42*, 393–405.
21. Chickering, D.M. Learning Bayesian networks is NP-complete. In *Learning from Data*; Springer Science & Business Media: New York, NY, USA, 1996; Volume 112, pp. 121–130.
22. Dagum, P.; Luby, M. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artif. Intell.* **1993**, *60*, 141–153.
23. Heckerman, D.; Geiger, D.; Chickering, D.M. Learning Bayesian networks: The combination of knowledge and statistical data. *Mach. Learn.* **1995**, *20*, 197–243.
24. Cooper, G.F.; Herskovits, E. A Bayesian method for the induction of probabilistic networks from data. *Mach. Learn.* **1992**, *9*, 309–347.
25. Friedman, N.; Murphy, K.; Russell, S. Learning the structure of dynamic probabilistic networks. In Proceedings of the Fourteenth Conference on UAI, Madison, WI, USA, 24–26 July 1998; pp. 139–147.
26. Murphy, K.P. The Bayes Net Toolbox for MATLAB. *Comput. Sci. Stat.* **2001**, *33*, 2001.
27. Edmonds, J. Optimum branchings. *Math. Decis. Sci.* **1968**, *71B*, 233–240.