# Polynomial-time algorithm for learning optimal tree-augmented dynamic Bayesian networks

**José L. Monteiro**
IDMEC
Instituto Superior Técnico
Universidade de Lisboa
jose.libano.monteiro@tecnico.ulisboa.pt

**Susana Vinga**
IDMEC
Instituto Superior Técnico
Universidade de Lisboa
susanavinga@tecnico.ulisboa.pt

**Alexandra M. Carvalho**
Instituto de Telecomunicações
Instituto Superior Técnico
Universidade de Lisboa
alexandra.carvalho@tecnico.ulisboa.pt

## Abstract

The identification of conditional dependences in longitudinal data is provided through structure learning of dynamic Bayesian networks (DBN). Several methods for DBN learning are concerned with identifying inter-slice dependences, but often disregard the intra-slice connectivity. We propose an algorithm that jointly finds the optimal inter and intra time-slice connectivity in a transition network. The search space is constrained to a class of networks designated by tree–augmented DBN, leading to polynomial time complexity. We assess the effectiveness of the algorithm on simulated data and compare the results to those obtained by a state of the art DBN learning implementation, showing that the proposed algorithm performs very well throughout the different experiments. Further experimental validation is made on real data, by identifying non-stationary gene regulatory networks of *Drosophila melanogaster*.

## 1 INTRODUCTION

Longitudinal data, also known as multivariate time series in the machine learning community, are obtained by conducting repeated measurements on a set of individuals. They arise in several contexts, such as biomedical and clinical studies, socio-economics and meteorology, and provide an opportunity for studying changes over time. In multivariate longitudinal data, each measurement or observation is a vector of variables, whose joint evolution is subject of analysis.

Multivariate longitudinal data can be modelled as a set of $n$-dimensional observations of a stochastic process over $T$ sequential instants of time. The set of observations is expressed as $\{\boldsymbol{x}^i[t]\}_{i\in\mathcal{I},t\in\mathcal{T}}$, where $\mathcal{I}$ is the set of individuals being measured and $\mathcal{T}$ is the set of time indices. Thus, $\boldsymbol{x}^i[t] = (x^i_1[t], \ldots, x^i_n[t]) \in \mathbb{R}^n$ is a single observation of $n$ features, made at time $t$ and referring to the individual $i$.

Observations are assumed to result from independent samples of a sequence of probability distributions $\{\mathbb{P}_{\boldsymbol{\theta}[t]}\}_{t\in\mathcal{T}}$. While these distributions may be time-variant, they are considered constant across different individuals observed at the same time, such that $\boldsymbol{x}^i[t] \sim \mathbb{P}_{\boldsymbol{\theta}[t]}$ for all $i \in \mathcal{I}$. If the observations are also identically distributed over time, that is, $\boldsymbol{\theta}[t] = \boldsymbol{\theta}$ for all $t \in \mathcal{T}$, the process is said to have a stationary or time-invariant distribution (henceforth simply referred to as a stationary process).

The identification of conditional independences in data provides an approximation to estimating the underlying probability distribution (Chow and Liu, 1968). Bayesian networks (BN) are a popular machine learning tool for this purpose, being able to represent complex processes that involve uncertainty. Dynamic Bayesian networks (DBN) extend BN in order to model temporal processes and are usually defined according to strong simplifying assumptions (Friedman et al., 1998; Murphy, 2002). A common premise is to consider the first-order Markov property, which states that attributes in time-slice $t + 1$ only depend on those in time-slice $t$, but not on the past trajectory. Another frequent assumption is to consider a stationary process, which may be adequate in some cases, but does not hold in many interesting scenarios.

### 1.1 RELATED WORK

Methods for learning stationary DBN are essentially simple extensions of those employed to learn BN (Murphy, 2002). A common approach consists in defining a scoring function, which measures a network's goodness of fit to training data, and a search procedure to generate networks (Heckerman et al., 1995). In general, obtaining an optimal network is an NP-hard problem, because the search space is super-exponential in the number of attributes (Chickering et al., 1995).

Unlike the case of BN, however, it was recently shown that learning the inter-slice structure of DBN does not have to be NP-hard (Dojer, 2006). This new hardness result is due to the relaxation of the acyclicity constraint on the transition network, since the unrolled network is always acyclic. In the same article, the author derived a polynomial complexity bound in the number of variables when using the minimum description length (MDL) or the Bayesian Dirichlet equivalence (BDe) scores. Relying on this result, Vinh et al. (2011b) further proposed a polynomial-time algorithm for learning optimal DBN using the mutual information tests (MIT) score.

While there is plenty of literature regarding the process of learning stationary first-order Markov networks, there are only a few references to learning more general classes of DBN. In fact, it was not until recently that some authors started to relax the standard stationarity assumption underlying graphical models. The following paragraphs present a brief review of such realizations.

The problem of model selection, that is, identifying a system for probabilistic inference that is efficient, accurate and informative is discussed in Bilmes (2000). With the purpose of performing classification, the author proposes a class of models where the conditional independences are determined by the values of certain nodes in the graph, instead of being fixed. This is accomplished by extending hidden Markov models (HMM) to include dependences among observations of different time-slices. The idea of a network whose edges can appear and disappear is further explored by other authors in a temporal context to model non-stationary processes.

An extension of the traditional discrete DBN model is defined in Robinson and Hartemink (2010), where an initial network of dependences and a set of incremental changes on its structure are learnt. The authors assume that the process is piecewise-stationary, having the number and times of the transitions (change points) to be estimated a posteriori. Prior knowledge regarding both the initial structure and the evolutionary behaviour of the network can be incorporated. By considering conjugate Dirichlet priors on the parameters, which are assumed to have a multinomial distribution, the marginal likelihood is computed exactly, resulting in the BDe metric. The authors extend this metric to incorporate the changes introduced by their new model.

The same approach is considered in Dondelinger et al. (2010), but concerning continuous data. The authors also differentiate the penalties for adding and removing edges in a network and allow different nodes to have distinct penalty terms, instead of a single hyperparameter for penalizing disparities between structures.

In more recent work, Grzegorczyk and Husmeier (2012) argue that there should be a trade-off between the often unrealistic stationarity assumption, modelled with constant parameters, and the opposite case of complete parameter independence over time, ignoring the evolutionary aspect of the process. They acknowledge, however, that the latter case, which is considered in Dondelinger et al. (2010) and Robinson and Hartemink (2010), has the advantage of allowing the computation of the marginal likelihood in closed form. The authors introduce a scheme for coupling the parameters along time-slices, although keeping the network structure fixed.

Regarding undirected graphical models, Kolar et al. (2010) propose two methods for estimating the underlying time-varying networks of a stochastic process. They model each network as a Markov random field (MRF) with binary nodes. The first method assumes that the parameters change smoothly over time whereas the second considers piecewise constant parameters with abrupt changes. In both approaches, the estimator for the parameters is the result of a $l_1$-regularized convex optimization problem. These methods, however, only capture pairwise undirected relations between binary variables, resulting in a model which is far from being generally applicable.

## 1.2 OUR APPROACH

The algorithm we propose falls under the search and score paradigm. Many software implementations for learning DBN are concerned with identifying inter-slice dependences, but disregard the intra-slice connectivity or assume it is given by some prior network and kept fixed over time (Dojer et al., 2013; Murphy, 2001; Vinh et al., 2011a). We instead suggest an algorithm that simultaneously learns all these dependences.

As a consequence of considering intra-slice edges in the proposed algorithm, the relaxation of the acyclicity constraint proposed in Dojer (2006) no longer applies, and obtaining an optimal network becomes NP-hard. We approach this problem by limiting the search space to tree-augmented networks, that is, networks whose attributes have at most one parent in the same time-slice. This restriction does not prevent an attribute to have several parents from preceding slices, and also accounts for the algorithm's polynomial time complexity in the number of attributes. Moreover, even though tree structures appear to be a strong constraint, they have been shown to produce very good results in classification tasks, namely within the tree augmented naive Bayes (TAN) method (Friedman et al., 1997).

The remaining of the paper is organized as follows. Section 2 formally defines BN, provides a theoretical overview on learning this class of networks and introduces DBN by extension. Section 3 describes the proposed DBN structure learning algorithm and analyses its time complexity. Section 4 assesses its performance on simulated data and real data. Section 5 presents the conclusions of this work.

## 2 THEORETICAL BACKGROUND

A BN is a graphical representation of a joint probability distribution over a set of random variables (Pearl, 1988). It is defined as a triple $B = (\boldsymbol{X}, G, \boldsymbol{\theta})$, where:

- $\boldsymbol{X} = (X_1, \ldots, X_n)$ is a random vector. Discrete random variables with a finite domain are considered;
- $G = (\boldsymbol{X}, E)$ is a directed acyclic graph (DAG) whose nodes are the elements of $\boldsymbol{X}$ and edges $E$ specify conditional dependences between the variables: each $X_i$ is independent of its non-descendants given its parents $\boldsymbol{pa}(X_i)$ in $G$;
- $\boldsymbol{\theta} = \{\theta_{ijk}\}$ is a set of parameters, specifying the local probability distributions of the network via

  $$\theta_{ijk} = P_B(X_i = x_{ik} \mid \boldsymbol{pa}(X_i) = w_{ij}), \qquad (1)$$

  where $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, q_i\}$ and $k \in \{1, \ldots, r_i\}$. $r_i$ is the number of discrete states of $X_i$. The set of possible configurations of $\boldsymbol{pa}(X_i)$, i.e., the set of different combinations of values that the parents of $X_i$ can take, is denoted by $\{w_{i1}, \ldots, w_{iq_i}\}$, where $q_i = \prod_{X_j \in \boldsymbol{pa}(X_i)} r_j$ is the number of all possible configurations.

A BN $B$ defines a joint probability distribution over $\boldsymbol{X}$:

$$P_B(X_1, \ldots, X_n) = \prod_{i=1}^{n} P_B(X_i \mid \boldsymbol{pa}(X_i)). \qquad (2)$$

The problem of learning a BN, given a dataset comprising instances of $\boldsymbol{X}$, can be stated as finding the structure (DAG) and parameters that best match the training data. When measuring the goodness of fit of a network $B$ to data $D$ by means of a scoring function $\phi$, learning a BN consists of maximizing $\phi(B, D)$ over the space of all networks with $n$ attributes.

A decomposable scoring function can be expressed as a sum of local terms, each depending only on a node and its parents:

$$\phi(B, D) = \sum_{i=1}^{n} \phi_i(\boldsymbol{pa}(X_i), D). \qquad (3)$$

Decomposability simplifies the process of calculating scores and provides an efficient way of evaluating incremental changes on a network.

The log-likelihood (LL) is a decomposable score which favours networks that are more likely to have generated the data. For a fixed structure $G$, assuming an underlying multinomial distribution, the network parameters are determined by maximum-likelihood estimation (MLE):

$$\{\hat{\theta}_{ijk} = \hat{P}_D(X_i = x_{ik} \mid \boldsymbol{pa}(X_i) = w_{ij}) = \frac{N_{ijk}}{N_{ij}}\} \qquad (4)$$

where $\hat{P}_D$ is the distribution induced by the observed frequency estimates, $N_{ijk}$ is the number of instances where $X_i$ takes its $k$-th value $x_{ik}$ and the variables in $\boldsymbol{pa}(X_i)$ take their $j$-th configuration $w_{ij}$, and $N_{ij}$ is the number of instances where the variables in $\boldsymbol{pa}(X_i)$ take their $j$-th configuration $w_{ij}$ notwithstanding the value of $X_i$. Since the parameters are unambiguously found for a fixed network structure, the LL criterion depends only on the network $G$. Taking into account Eq. (2), and assuming that instances of $D$ are independent and identically distributed (i.i.d.), the LL scoring function is expressed (Heckerman et al., 1995) as

$$\begin{aligned}\phi_{LL}(B, D) &= \log P(D \mid B) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}}.\end{aligned} \qquad (5)$$

The minimum description length (MDL) score is an extension of the LL criterion, including a term for penalizing complex structures:

$$\phi_{MDL}(B, D) = \phi_{LL}(B, D) - \frac{1}{2} \log(N)|B|, \qquad (6)$$

where $N$ is the number of samples in $D$ and $|B|$ denotes the number of parameters of the network, given by:

$$|B| = \sum_{i=1}^{n} (r_i - 1) q_i. \qquad (7)$$

While a BN defines a joint probability distribution over a fixed set of variables, a DBN extends this representation to model temporal processes (Friedman et al., 1998). Let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be a random vector, composed by the attributes that the are changed by some process. Furthermore, let $\boldsymbol{X}[t] = (X_1[t], \ldots, X_n[t])$ denote the instantiation of the attributes at discrete time $t \in \mathbb{N}$. A DBN encodes the joint probability distributions over all possible trajectories of a process.

The first-order Markov property states that future values only depend on present ones, not on the past trajectory, s.t. $P(\boldsymbol{X}[t+1] \mid \boldsymbol{X}[0] \cup \cdots \cup \boldsymbol{X}[t]) = P(\boldsymbol{X}[t+1] \mid \boldsymbol{X}[t])$. A relaxation of this assumption is the higher-order Markov property, where nodes can have dependences on an arbitrary (but fixed) number of previous time-slices.

A non-stationary first-order Markov DBN describing a temporal process over $T$ time-slices consists of:

- a prior network $B^0$, which specifies a distribution over the initial states $\boldsymbol{X}[0]$;
- a set of transition networks $B_t^{t+1}$ over the variables $\boldsymbol{X}[t] \cup \boldsymbol{X}[t+1]$, specifying the state transition probabilities, for $0 \leq t < T$.

A stationary network contains only one prior network and one transition network, being the latter unrolled over time.

Learning DBN typically refers to the transition network(s), as learning the prior network can be done directly using BN methods. Learning a transition network has the additional requirement that edges between slices must flow forward in time.

## 3 PROPOSED METHOD

The proposed algorithm is based on learning tree-like Bayesian networks. The Chow-Liu algorithm finds a tree with maximum mutual information (Chow and Liu, 1968), or, equivalently, a tree with maximum LL score. The algorithm works as follows: (i) a complete undirected graph weighted with the mutual information between each pair of nodes is built; (ii) an undirected spanning tree is extracted; and (iii) the optimal branching is retrieved by choosing an arbitrary node as the tree root and then setting the direction of all edges to be outward from it.

It was shown that the Chow-Liu algorithm can be adapted to use any decomposable scoring criterion $\phi$ (Heckerman et al., 1995). In this case, the weight of an edge $X_j \to X_i$ is assigned as $\phi_i(\{X_j\}, D) - \phi_i(\{\}, D)$, expressing the contribution of the edge, as measured by $\phi$, to the total network score. If $\phi$ is score-equivalent, the weights of the edges $X_j \to X_i$ and $X_i \to X_j$ are the same, and so an undirected spanning tree is enough to retrieve the optimal branching. However, if the score is not score-equivalent, Edmond's algorithm (Edmonds, 1967) needs to be used to find the maximum branching from a complete directed weighted graph (Heckerman et al., 1995).

In the temporal domain, nodes in $\mathbf{X}[t+1]$ can also have parents from previous time-slices. Our approach for learning DBN jointly learns inter and intra time-slice dependences. We propose to learn a tree network structure for the intra-slice dependences while limiting the number of parents from the preceding time-slices. That is, for each node in the current time-slice $t+1$ we allow one parent from the same time-slice (with the exception of the root node) and at most $p$ parents from the preceding time-slices. We call the resulting transition network structure a tree-augmented DBN (tDBN). As we shall see next, the weight of an edge $X_j[t+1] \to X_i[t+1]$ will account for the contribution of inter and intra-slice parents simultaneously. Therefore, due to inter-slice parents, the weights $X_j[t+1] \to X_i[t+1]$ and $X_i[t+1] \to X_j[t+1]$ are, in general, not the same. This forces us to resort to Edmond's algorithm.

### 3.1 OPTIMAL TREE-AUGMENTED DBN STRUCTURE LEARNING

Considering, for the sake of simplicity, the first-order Markov DBN paradigm, parents from the past can only belong to the preceding slice. Let $\mathcal{P}_{\leq p}(\boldsymbol{X}[t])$ be the set of subsets of $\boldsymbol{X}[t]$ of cardinality less than or equal to $p$. If a node in $\boldsymbol{X}[t+1]$ is limited to having at most $p$ parents from the past, its set of parents must belong to $\mathcal{P}_{\leq p}(\boldsymbol{X}[t])$. The optimal tDBN structure learning algorithm proceeds as follows.

First, for each node $X_i[t+1] \in \boldsymbol{X}[t+1]$, the best score and the set of parents $\boldsymbol{X}_{ps}[t]$ in $\mathcal{P}_{\leq p}(\boldsymbol{X}[t])$ that maximizes it are found. This optimization is formally expressed as

$$s_i = \max_{\boldsymbol{X}_{ps}[t] \in \mathcal{P}_{\leq p}(\boldsymbol{X}[t])} \phi_i(\boldsymbol{X}_{ps}[t], D_t^{t+1}), \tag{8}$$

where $\phi_i$ denotes a local term of a decomposable scoring function $\phi$ and $D_t^{t+1}$ is the subset of observations of $D$ concerning the time transition $t \to t+1$. Then, also allowing one parent from the current time-slice, for each edge $X_j[t+1] \to X_i[t+1]$, the best score and the set of parents from the past that maximizes it are also found:

$$s_{ij} = \max_{\boldsymbol{X}_{ps}[t] \in \mathcal{P}_{\leq p}(\boldsymbol{X}[t])} \phi_i(\boldsymbol{X}_{ps}[t] \cup \{X_j[t+1]\}, D_t^{t+1}). \tag{9}$$

A complete directed graph with nodes in $\boldsymbol{X}[t+1]$ is built, being the weight of each edge $X_j[t+1] \to X_i[t+1]$ assigned as

$$e_{ij} = s_{ij} - s_i, \tag{10}$$

which expresses the gain in the total network score by including $X_j[t+1]$ as a parent of $X_i[t+1]$, as opposed to leaving $X_i[t+1]$ only with parents in $\boldsymbol{X}[t]$. In general, $e_{ij} \neq e_{ji}$, and therefore a directed spanning tree must be found. Thus, to obtain the $t \to t+1$ transition network structure, the Edmonds' algorithm for finding a maximum branching (Edmonds, 1967) is applied. The resulting directed tree immediately provides the network intra-slice connectivity (in $t+1$). In addition, for all the nodes except the root, their set of parents from the preceding time-slice is the solution for the optimization problem in Eq. (9). Similarly, the root node's parents are given by the solution for the problem in Eq. (8).

The described procedure can jointly obtain the intra and inter-slice connectivity in a transition network. By repeatedly applying it to all the available time transitions, it is possible to retrieve the structure of a tree-augmented non-stationary first-order Markov DBN. A global view of this method is presented in Algorithm 1.

**Theorem 1.** *Algorithm 1 finds an optimal tDBN under a given decomposable scoring function.*

*Proof.* Let $B$ be an optimal tDBN and $T$ be the tree structure of $B$ accounting only for the intra-slice dependences. Due to the optimality of $B$, its overall score is equal to

$$s_R + \sum_{X_j[t+1] \to X_i[t+1] \in T} s_{ij},$$

**Algorithm 1:** Optimal non-stationary first-order Markov tDBN structure learning

```
Input: $\boldsymbol{X}$: the set of network attributes;
        $D$: dataset of longitudinal observations over $T$ time-slices;
        $\phi$: a decomposable scoring function
Output: A tree-augmented DBN structure
1 For each transition $t \to t+1$ :
2     Build a complete directed graph in $\boldsymbol{X}[t+1]$
3     Calculate the weight of all edges and the optimal set of parents of all nodes (Algorithm 2)
4     Apply a maximum branching algorithm
5     Extract transition $t \to t+1$ network using the maximum branching and the optimal set of parents calculated in Algorithm 2
6 Collect transition networks to obtain DBN structure
```

according to Eq. (8) and Eq. (9), where $X_R$ is the root node of $T$.

Consider the constant $K = \sum_i s_i$. Finding an optimal tDBN for a given score $\phi$ is the same as finding the optimal tDBN up to the constant $K$; note that $s_i$ does not depend on the structure $T$, nor $B$. By subtracting $K$ to the score of $B$ we obtain

$$\sum_{X_j[t+1] \to X_i[t+1] \in T} e_{ij},$$

according to Eq. (10). Observe that an optimal branching of the complete directed graph, where each edge $X_j[t+1] \to X_i[t+1]$ is weighted with $e_{ij}$, is precisely $T$. Therefore, due to the soundness of Edmonds' algorithm, the output of Algorithm 1 is $T$, from which $B$ can be recovered. □

## 3.2 COMPLEXITY ANALYSIS

The derivation of a complexity bound on the running time of Algorithm 1 is presented in the following.

**Theorem 2.** *The time complexity of Algorithm 1 is polynomial in the number of attributes $n$, linear in the the number of observations $N$, and exponential in the number of parents $p$.*

*Proof.* For each transition, the step of determining the edge weights and optimal sets of parents takes the most number of operations and determines the algorithm's growth rate. The iterative process starting at line 11 in Algorithm 2 is the most expensive overall. It calculates the weights for all edges in a complete graph with $n$ nodes, which requires $\mathcal{O}(n^2)$ iterations. For any edge, a score is evaluated for each possible set of parents in the preceding time-slice. The total number of parent sets is given by:

$$|\mathcal{P}_{\leq p}(\boldsymbol{X}[t])| = \sum_{i=1}^{p} \binom{n}{i} < \sum_{i=1}^{p} n^i \in \mathcal{O}(n^p). \qquad (11)$$

**Algorithm 2:** Determining edge weights and optimal sets of parents (first-order Markov)

```
Input: $\boldsymbol{X}[t], \boldsymbol{X}[t+1]$: sets of $n$ nodes from two adjacent time-slices;
        $p$: upper bound on the number of parents from time-slice $t$;
        $D_t^{t+1}$: dataset of observations concerning the time transition $t \to t+1$;
        $\phi_{\text{child}[t+1]}(\text{parents}, \text{dataset})$: a local term of $\phi$
Output: $E_{[n \times n]}$: edge weights matrix;
         parentsPastSlice$_{[n]}$: optimal set of parents from time-slice $t$;
         parentsAllSlices$_{[n \times n]}$ : optimal set of parents from time-slices $t$ and $t+1$
1  allParentSets $\leftarrow \mathcal{P}_{\leq p}(\boldsymbol{X}[t])$
2  For each $X_i[t+1]$ :
3      bestScore $\leftarrow -\infty$
4      For each $\boldsymbol{X}_{ps}[t] \in$ allParentSets :
5          currentScore $\leftarrow \phi_i(\boldsymbol{X}_{ps}[t], D_t^{t+1})$
6          If bestScore < currentScore :
7              bestScore $\leftarrow$ currentScore
8              parentsPastSlice$_i \leftarrow \boldsymbol{X}_{ps}[t]$
9      For each $X_j[t+1]$ :
10         $E_{ij} \leftarrow -$bestScore
11 For each $X_i[t+1]$ :
12     For each $X_j[t+1]$ :
13         bestScore $\leftarrow -\infty$
14         For each $\boldsymbol{X}_{ps}[t] \in$ allParentSets :
15             currentScore
               $\leftarrow \phi_i(\boldsymbol{X}_{ps}[t] \cup \{X_j[t+1]\}, D_t^{t+1})$
16             If bestScore < currentScore :
17                 bestScore $\leftarrow$ currentScore
18                 parentsAllSlices$_{ij} \leftarrow \boldsymbol{X}_{ps}[t]$
19         $E_{ij} \leftarrow E_{ij}+$ bestScore
```

For calculating each score, all different network configurations must be considered. Assuming that $r$ is the maximum number of discrete states a variable can take, and that a variable $X_i[t+1]$ has $p+1$ parents (one in $\boldsymbol{X}[t+1]$ and $p$ in $\boldsymbol{X}[t]$), there are $\mathcal{O}(r^{p+2})$ different configurations. Each configuration needs to be counted over a dataset containing $|D_t^{t+1}|$ observations, which can be stored in a $|D_t^{t+1}| \times 2n$ sized matrix, thus requiring $\mathcal{O}(|D_t^{t+1}|n)$ comparisons. Taking into account all the internal loops, the complexity of the outer cycle is $\mathcal{O}(n^{p+3}\, r^{p+2}\, |D_t^{t+1}|)$.

The efficient implementation of Edmonds' algorithm described in (Tarjan, 1977) has quadratic complexity in the number of nodes, hence being irrelevant to the overall bound. Algorithm 1, which learns a network structure for each of $T$ time transitions, admits a worst-case complexity of $\mathcal{O}(n^{p+3} r^{p+2} N)$, where $N = |D| = \sum_{t=0}^{T-1} |D_t^{t+1}|$. □

### 3.3 EXTENSIONS TO STATIONARITY AND HIGHER-ORDER MARKOV

Algorithm 1 was presented in its non-stationary first-order Markov form. Nonetheless, adaptations for stationary or higher-order Markov processes are trivial and can be concisely described.

A stationary version of Algorithm 1 does not contain the for loop starting at line 1, because only one iteration is needed to find one transition network. In Algorithm 2, the entire dataset $D$ is used in each score evaluation. Overall, since the number of examinations of each observation for learning the DBN structure is the same as in the non-stationary version, the time complexity remains the same, that is, $\mathcal{O}(n^{p+3}\, r^{p+2}\, N)$.

In a $m$-th-order Markov version of tDBN, regardless of process stationarity, nodes are allowed to have parents from $m$ previous time-slices. Therefore, in Algorithm 2, allParentSets $\leftarrow \mathcal{P}_{\leq p}(\boldsymbol{X}[t-m+1] \cup \cdots \cup \boldsymbol{X}[t])$ and $D^{t+1}_{t-m+1}$ is used instead of $D^{t+1}_{t}$. These changes worsen the algorithm's time complexity, which can be roughly approximated by $\mathcal{O}((nm)^{p+3}\, r^{p+2}\, N)$.

## 4 EXPERIMENTAL RESULTS

In this section we describe the methodology used for evaluating the optimal tDBN learning algorithm and present the obtained results, in terms of speed and accuracy. Our algorithm was implemented in Java using an object-oriented approach and released under a free-software license[1]. Simulated data was first considered to evaluate the simpler stationary first-order Markov tDBN. As very good results were achieved, further assessment on real data with a non-stationary first-order Markov tDBN was used to learn time-varying gene regulatory networks from gene expression data of *Drosophila melanogaster*.

### 4.1 SIMULATED DATA

In the first set of experiments comprising simulated data, Banjo (et al., 2005), a state of the art DBN learning tool, was employed besides the tDBN learning algorithm for comparative purposes. Banjo was chosen for being able to also learn the intra-slice connectivity, as opposed to most DBN learning implementations. Throughout the experiments, an implementation's ability to recover a known network structure was measured. This was accomplished by specifying a DBN (both its structure and parameters), sampling the network to generate observations and inputting the produced datasets to each implementation, in order to learn the underlying structure. The original and recovered networks were then compared by evaluating the precision and recall metrics. In addition, to provide a unified score combining the two previous metrics, the $F$-measure was also calculated.

[1]Available at `http://josemonteiro.github.io/tDBN/` where additional experimental results are also provided.

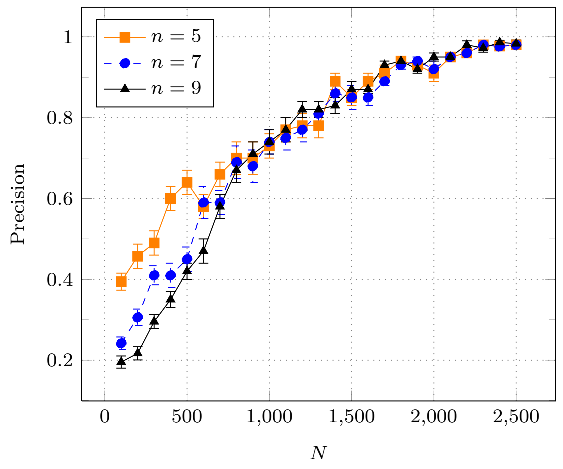


Figure 1: Plot of the precision values achieved by the tDBN+LL learning algorithm for different values of input observations $N$. Three lines are shown, corresponding to complete tree-augmented networks with different number of attributes $n$, each attribute taking $r = 8$ different states and having $p = 2$ parents from the previous time-slice. Each point results from averaging the precision over 25 sampled datasets, with error bars denoting standard errors. Precision generally increases with $N$ for every choice of $n$, attaining a plateau with $N > 2000$.

Because many methods only learn the inter-slice connectivity of DBN, Table 1 present the metrics taking into account (i) only the inter-slice edges, and (ii) all edges. The results in Table 1 are displayed as average statistics over 5 runs. Precision (Pre), recall (Rec) and $F$-measure ($F_1$) values are presented as percentages, running time is in seconds; $n$ is the number of network attributes, $p$ is the number of parents from the preceding time-slice, $r$ is the number of states of all attributes, and $N$ is the number of observations. Values in bold correspond to the highest $F$-measure score in groups (i) or (ii); that is, one concerning the recovery of the inter-slice edges only and another concerning the recovery of all edges. Additional results and details concerning other experiments employing the tDBN learning algorithm are shown in Figures 1 and 2.

In the comparative tests, the stationary tDBN learning algorithm was employed using the LL and MDL scores. Banjo's Markov lag interval was set between 0 and 1 to allow intra-slice edges. Its running time was set to 10 minutes, which was always longer than learning tDBN's, and simulated annealing was used as search procedure. In all cases, the maximum number of parents was set according to the original network. The experiments were run on an Intel i5-3570 @ 3.40 GHz machine.

Table 1: Comparative structure recovery results on simulated data. Banjo's running time is not shown, as it was always 600 seconds.

| $N$ | tDBN+LL | | | | | | | tDBN+MDL | | | | | | | Banjo | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Inter-slice | | | Global | | | | Inter-slice | | | Global | | | | Inter-slice | | | Global | | |
| | Pre | Rec | $F_1$ | Pre | Rec | $F_1$ | Time | Pre | Rec | $F_1$ | Pre | Rec | $F_1$ | Time | Pre | Rec | $F_1$ | Pre | Rec | $F_1$ |
| Complete tree-augmented network ($n = 20, p = 2, r = 2$) | | | | | | | | | | | | | | | | | | | | |
| 100 | $\mathbf{64 \pm 3}$ | $\mathbf{64 \pm 3}$ | $\mathbf{64}$ | $\mathbf{61 \pm 4}$ | $\mathbf{61 \pm 4}$ | $\mathbf{61}$ | 4 | $75 \pm 3$ | $54 \pm 3$ | 63 | $67 \pm 3$ | $54 \pm 3$ | 60 | 4 | $98 \pm 3$ | $17 \pm 1$ | 29 | $66 \pm 9$ | $15 \pm 1$ | 24 |
| 300 | $\mathbf{88 \pm 2}$ | $\mathbf{88 \pm 2}$ | $\mathbf{88}$ | $\mathbf{86 \pm 3}$ | $\mathbf{86 \pm 3}$ | $\mathbf{86}$ | 12 | $\mathbf{98 \pm 2}$ | $\mathbf{80 \pm 1}$ | $\mathbf{88}$ | $90 \pm 2$ | $79 \pm 1$ | 84 | 13 | $98 \pm 3$ | $18 \pm 1$ | 30 | $54 \pm 3$ | $20 \pm 1$ | 29 |
| 700 | $\mathbf{97 \pm 1}$ | $\mathbf{97 \pm 1}$ | $\mathbf{97}$ | $\mathbf{98 \pm 1}$ | $\mathbf{98 \pm 1}$ | $\mathbf{98}$ | 28 | $100 \pm 0$ | $93 \pm 0$ | 96 | $100 \pm 0$ | $95 \pm 0$ | 97 | 29 | $97 \pm 3$ | $19 \pm 1$ | 32 | $46 \pm 3$ | $19 \pm 1$ | 27 |
| Complete tree-augmented network ($n = 20, p = 2, r = 5$) | | | | | | | | | | | | | | | | | | | | |
| 100 | $\mathbf{15 \pm 2}$ | $\mathbf{15 \pm 2}$ | $\mathbf{15}$ | $\mathbf{13 \pm 2}$ | $\mathbf{13 \pm 2}$ | $\mathbf{13}$ | 68 | $21 \pm 2$ | $11 \pm 1$ | 14 | $\mathbf{16 \pm 1}$ | $\mathbf{11 \pm 1}$ | $\mathbf{13}$ | 68 | – | – | – | – | – | – |
| 300 | $\mathbf{84 \pm 3}$ | $\mathbf{84 \pm 3}$ | $\mathbf{84}$ | $\mathbf{83 \pm 3}$ | $\mathbf{83 \pm 3}$ | $\mathbf{83}$ | 209 | $51 \pm 5$ | $26 \pm 2$ | 34 | $45 \pm 5$ | $29 \pm 3$ | 35 | 213 | – | – | – | – | – | – |
| 700 | $\mathbf{100 \pm 0}$ | $\mathbf{100 \pm 0}$ | $\mathbf{100}$ | $\mathbf{100 \pm 0}$ | $\mathbf{100 \pm 0}$ | $\mathbf{100}$ | 491 | $97 \pm 2$ | $49 \pm 1$ | 65 | $96 \pm 2$ | $64 \pm 1$ | 77 | 489 | $100 \pm 0$ | $3 \pm 0$ | 6 | $100 \pm 0$ | $2 \pm 0$ | 4 |
| Incomplete tree-augmented network ($n = 20, \max p = 3, r = 2$) | | | | | | | | | | | | | | | | | | | | |
| 100 | $39 \pm 1$ | $81 \pm 3$ | 53 | $43 \pm 0$ | $70 \pm 1$ | 53 | 44 | $\mathbf{73 \pm 2}$ | $\mathbf{67 \pm 2}$ | $\mathbf{70}$ | $\mathbf{70 \pm 2}$ | $\mathbf{66 \pm 3}$ | $\mathbf{68}$ | 46 | $96 \pm 4$ | $17 \pm 1$ | 29 | $60 \pm 14$ | $18 \pm 3$ | 28 |
| 300 | $43 \pm 1$ | $90 \pm 1$ | 58 | $53 \pm 1$ | $87 \pm 2$ | 66 | 136 | $\mathbf{85 \pm 1}$ | $\mathbf{84 \pm 2}$ | $\mathbf{84}$ | $\mathbf{86 \pm 3}$ | $\mathbf{85 \pm 3}$ | $\mathbf{85}$ | 140 | $100 \pm 0$ | $19 \pm 1$ | 32 | $46 \pm 3$ | $23 \pm 1$ | 31 |
| 700 | $48 \pm 0$ | $99 \pm 1$ | 65 | $58 \pm 1$ | $96 \pm 2$ | 72 | 330 | $\mathbf{90 \pm 0}$ | $\mathbf{94 \pm 1}$ | $\mathbf{92}$ | $\mathbf{94 \pm 0}$ | $\mathbf{96 \pm 0}$ | $\mathbf{95}$ | 326 | $93 \pm 7$ | $20 \pm 2$ | 33 | $37 \pm 5$ | $21 \pm 3$ | 27 |
| Inter-sliced only network ($n = 20, p = 2, r = 2$) | | | | | | | | | | | | | | | | | | | | |
| 100 | $\mathbf{63 \pm 3}$ | $\mathbf{63 \pm 3}$ | $\mathbf{63}$ | $\mathbf{42 \pm 2}$ | $\mathbf{63 \pm 3}$ | $\mathbf{50}$ | 4 | $73 \pm 2$ | $51 \pm 3$ | 60 | $43 \pm 2$ | $51 \pm 3$ | 47 | 4 | $100 \pm 0$ | $18 \pm 1$ | 31 | $100 \pm 0$ | $18 \pm 1$ | 31 |
| 300 | $\mathbf{87 \pm 1}$ | $\mathbf{87 \pm 1}$ | $\mathbf{87}$ | $\mathbf{59 \pm 1}$ | $\mathbf{87 \pm 1}$ | $\mathbf{70}$ | 12 | $91 \pm 1$ | $79 \pm 1$ | 85 | $59 \pm 1$ | $79 \pm 1$ | 68 | 12 | $100 \pm 0$ | $18 \pm 0$ | 31 | $89 \pm 11$ | $18 \pm 0$ | 30 |
| 700 | $91 \pm 1$ | $91 \pm 1$ | 91 | $\mathbf{61 \pm 1}$ | $\mathbf{91 \pm 1}$ | $\mathbf{73}$ | 28 | $\mathbf{96 \pm 1}$ | $\mathbf{88 \pm 1}$ | $\mathbf{92}$ | $\mathbf{63 \pm 0}$ | $\mathbf{88 \pm 1}$ | $\mathbf{73}$ | 28 | $100 \pm 0$ | $18 \pm 0$ | 31 | $100 \pm 0$ | $18 \pm 0$ | 31 |

Four different network settings are considered in Table 1. The first two networks are complete tree-augmented DBN, in the sense that each attribute in $\boldsymbol{X}[t+1]$ has exactly $p$ parents in $\boldsymbol{X}[t]$ and at most one parent in $\boldsymbol{X}[t+1]$. In these settings, the number of edges is always $n(p+1)-1$. On the other hand, the third network is an incomplete tree-augmented DBN, because the number of parents from the preceding slice is chosen at random between 1 and $p$. In the fourth network, there are no edges inside $\boldsymbol{X}[t+1]$, corresponding to the traditional inter-sliced DBN concept.

According to each network setting, a network was created by randomly generating its structure and parameters using uniform distributions. An experimental group, corresponding to a line in Table 1, consisted of 5 independent datasets with the same number of observations, sampled from one of the generated networks. The presented values result from averaging the performance metrics over the datasets of a group.

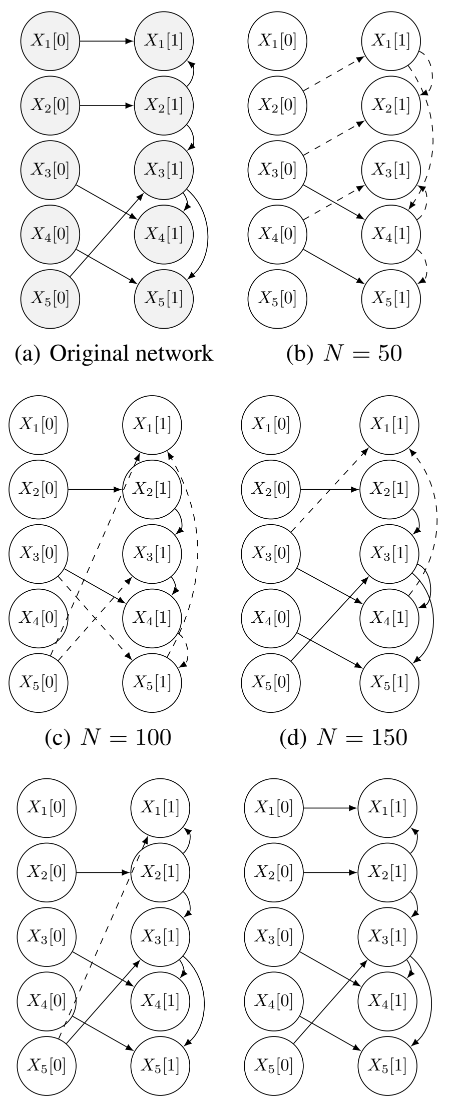
From the experimental results in Table 1, it can be seen that the performance of the proposed algorithm consistently increases with $N$. In the first setting, the tDBN learning algorithm performs very well with either score, with LL having a slight advantage. This result was expected, since a complete tree-augmented DBN is biased towards tDBN, and LL assures that a necessary and sufficient number of edges is recovered. Banjo obtains a very good inter-slice precision, but only recovers one fifth of the original edges. Notice that Banjo's global results deteriorate when $N$ increases, which can be explained by decreasing performance while identifying the intra-slice connectivity.

In the second setting, the tDBN+LL learning algorithm globally outperforms the other implementations. The regularization effect of MDL is observed through lower recall levels, since the number of network parameters increases with $r$. Nevertheless, tDBN+MDL greatly improves with $N$ and already achieves very high precision for $N = 700$. Comparing to the first setting, Banjo is more conservative adding edges, and chooses none for $N \leq 300$.

In the third setting, the inter-slice precision of tDBN+LL does not exceed $50\%$, due to the recovery of exactly $p$ parents, when the real number can be smaller than $p$. In this setting, tDBN+MDL clearly achieves the best performance. The penalizing term in MDL prevents false positive edges from being chosen, resulting in significantly higher precision values compared to LL. Banjo performs like in the first setting, being able to determine the correct parents and thus reaching high precision values with respect to the inter-slice connectivity.

Even though the last setting comprises a network without intra-slice edges, the tDBN learning algorithm performs well with either scoring function. In fact, the inter-slice metrics are comparable to the ones in the first setting. The $F$-measure values of LL and MDL are very similar, causing neither score to stand out. The recovery of an intra-slice tree, which is an inherent aspect of tDBN learning procedure, worsens the global performance of the algorithm. In this setting, Banjo obtains an excellent inter-slice result, correctly identifying all the dependences and reporting no false positives. On the other hand, the percentage of retrieved edges is quite low, resulting in unimpressive $F$-measure scores.

Overall, the tDBN learning algorithm obtained very good results. While tDBN+LL outperformed in the more complex second setting network, tDBN+MDL showed to be more robust, achieving at least one highest $F$-measure score in each setting. Banjo consistently identified high-precision sparse inter-sliced networks, but generally could not recover more than $20\%$ of the existing data depen-


(a) Original network (b) $N = 50$

(c) $N = 100$ (d) $N = 150$

(e) $N = 200$ (f) $N = 250$

Figure 2: Example of the tDBN+LL learning algorithm's ability to recover a known network. The original tree-augmented network has $n = 5$ attributes, each taking $r = 8$ different states and having one parent from the previous time-slice. Differing in the number of input observations $N$, five recovered networks are shown. Dashed edges are not present in the original network but are nevertheless recovered. As $N$ increases, the recovered network structures become more similar to the original, being identical for $N = 250$.

dences.

## 4.2 DROSOPHILA MELANOGASTER DATA

The following experiments consisted in applying the tDBN learning algorithm to identify non-stationary gene regulatory networks of *Drosophila melanogaster*. Arbeitman et al. (2002) published a dataset containing gene expression measurements of 4028 *Drosophila* genes over 67 time steps, covering the four major stages of morphogenesis: embryo, larva, pupa and adult. Some authors have focused on a small subset of this time series, consisting of eleven genes involved in wing muscle development (Guo et al., 2007; Robinson and Hartemink, 2010; Dondelinger et al., 2013). Consequently, to allow comparison to other methods, the same subset is considered herein. The *Drosophila* gene expression dataset was preprocessed in the same way as in the aforementioned references.

For learning the gene regulatory networks, the first-order Markov tDBN learning algorithm was employed with the MDL score, allowing at most two parents from the past. However, as the number of observations was small, there was not enough evidence for MDL to include more than one parent. Figure 3 presents the resulting networks in compact form, to facilitate comparison to networks inferred by other authors. Table 2 examines the identified gene interactions against the ones reported in other publications, in the form of matching percentages. Guo et al. (2007) predicted non-stationary undirected networks, while Dondelinger et al. (2013) inferred non-stationary DBN.

The results in Table 2 suggest that tDBN learning algorithm performed reasonably well. The embryonic and larval networks contain a significant number of known interactions that are present in at least one of the corresponding reported networks. On the other hand, the pupal and adult networks did not achieve this result. Some of the pupal interactions could be disjointly found on the networks of both authors, resulting in a higher combined matching rate. The adult network, however, retrieved few known interactions, even when comparing to both sources combined.

Table 2: Comparative structure learning results on *Drosophila melanogaster* data.

| Morphogenic stage | Embryonic | Larval | Pupal | Adult |
|---|---|---|---|---|
| Observed time-slices | 31 | 10 | 18 | 8 |
| Matches (Guo et al., 2007) | 25% | 50% | 40% | 30% |
| Matches (Dondelinger et al., 2013) | 75% | 60% | 30% | 20% |
| Matches (both sources) | 75% | 70% | 60% | 40% |

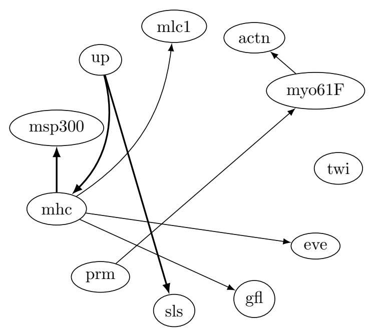
As acknowledged by Dondelinger et al. (2013), an objective assessment regarding the accuracy of the learnt networks is not possible due to limited biological knowledge available, which leads to the absence of a gold standard. Furthermore, there are three reasons for which the obtained results should be interpreted carefully. First, despite the best efforts to follow the procedure in Zhao et al. (2006), the resulting preprocessed dataset was possibly not the same. Second, learnt interactions are suggestions of causal regulatory effects. Additional biological experiments are necessary for validating the inferred networks, as noted in Guo et al. (2007). Third, the small number of observations leads to the existence of many equivalent networks with maximum score, but only one is reported by the tDBN learning algorithm.

## 5 CONCLUSION

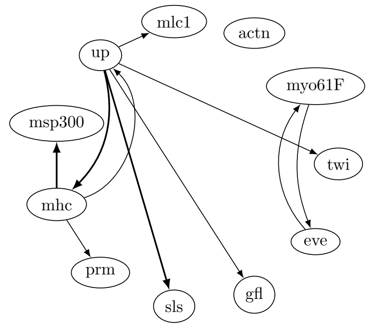
We have presented a simple yet effective algorithm for learning the structure of DBN, jointly recovering the inter and intra time-slice connectivity. The tDBN learning algorithm has polynomial time complexity with respect to the number of attributes and can be applied to non-stationary Markov processes.

The stationary version of the algorithm achieved very good results on simulated datasets, showing to be competitive with state of the art algorithms in recovering underlying structures. Furthermore, encouraging results were obtained on real data with the non-stationary and higher-order Markov versions of tDBN, indicating a broad scope of applications for the proposed algorithm.

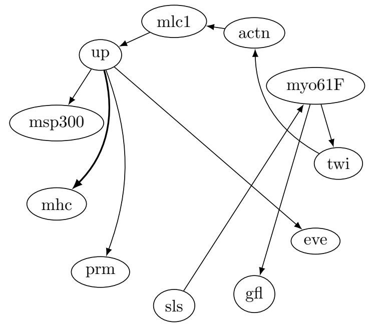
Finally, the application of our method to learn non-stationary processes could be improved in conjunction with change-point techniques, as investigated in Robinson and Hartemink (2010) and Dondelinger et al. (2010). In its current form, the tDBN learning algorithm cannot identify changes in the underlying distribution of data and thus the number of transition networks to learn, as well as adequate training data for each network, need to be previously specified.



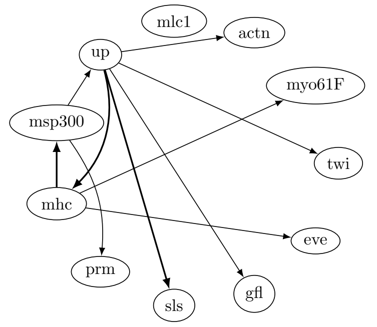
(a) Embryonic



(b) Larval



(c) Pupal



(d) Adult

Figure 3: *Drosophila* gene regulatory networks identified by the tDBN learning algorithm. Networks are shown in compact form, where each edge represents a dependence between a node at time-slice $t + 1$ and its parent at the previous time-slice $t$. Highlighted edges indicate a relation found in the majority of the morphogenic stages. Intra-slice edges are omitted.

## Acknowledgments

This work was supported by national funds through FCT, Fundação para a Ciência e a Tecnologia, under contracts LAETA (UID/EMS/50022/2013) and IT (UID/EEA/50008/2013), and by projects CancerSys (EXPL/EMS-SIS/1954/2013) and InteleGen (PTDC/DTP-FTO/1747/2012). SV acknowledges support by Program Investigador FCT (IF/00653/2012) from FCT, co-funded by the European Social Fund (ESF) through the Operational Program Human Potential (POPH).

## References

M. N. Arbeitman, E. E. M. Furlong, F. Imam, E. Johnson, B. H. Null, B. S. Baker, M. A. Krasnow, M. P. Scott, R. W. Davis, and K. P. White. Gene expression during the life cycle of Drosophila melanogaster. *Science*, 297 (5590):2270–2275, 2002.

J. A. Bilmes. Dynamic Bayesian multinets. In *Proc. of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 38–45, 2000.

D. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks: Search methods and experimental results. In *Proc. of the 5th International Workshop on Artificial Intelligence and Statistics*, pages 112–128, 1995.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

N. Dojer. Learning Bayesian networks does not have to be NP-hard. In *Mathematical Foundations of Computer Science 2006*, pages 305–314. 2006.

N. Dojer, P. Bednarz, A. Podsiadło, and B. Wilczyński. BNFinder2: Faster Bayesian network learning and Bayesian classification. *Bioinformatics*, page btt323, 2013.

F. Dondelinger, S. Lèbre, and D. Husmeier. Heterogeneous continuous dynamic Bayesian networks with flexible structure and inter-time segment information sharing. In *Proc. of the 27th International Conference on Machine Learning*, pages 303–310, 2010.

F. Dondelinger, S. Lèbre, and D. Husmeier. Non-homogeneous dynamic Bayesian networks with Bayesian regularization for inferring gene regulatory networks with gradually time-varying structure. *Machine learning*, 90(2):191–230, 2013.

J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240, 1967.

A. Hartemink et al. Banjo: Bayesian network inference with Java objects. http://www.cs.duke.edu/amink/software/banjo/, 2005.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3):131–163, 1997.

N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. In *Proc. of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 139–147, 1998.

M. Grzegorczyk and D. Husmeier. Bayesian regularization of non-homogeneous dynamic Bayesian networks by globally coupling interaction parameters. In *Journal of Machine Learning Research Workshop and Conference Proceedings*, volume 22, pages 467–476, 2012.

F. Guo, S. Hanneke, W. Fu, and E. P. Xing. Recovering temporally rewiring networks: A model-based approach. In *Proc. of the 24th international conference on Machine learning*, pages 321–328, 2007.

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

M. Kolar, L. Song, A. Ahmed, and E. P. Xing. Estimating time-varying networks. *The Annals of Applied Statistics*, 4(1):94–123, 2010.

K. Murphy. The Bayes net toolbox for Matlab. *Computing Science and Statistics*, 33:2001, 2001.

K. Murphy. *Dynamic Bayesian networks: representation, inference and learning*. PhD thesis, University of California, 2002.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Representation and Reasoning Series. Morgan Kaufmann, 1988. ISBN 9781558604797.

J. W. Robinson and A. J. Hartemink. Learning non-stationary dynamic Bayesian networks. *Journal of Machine Learning Research*, 11:3647–3680, 2010.

R. Tarjan. Finding optimum branchings. *Networks*, 7(1): 25–35, 1977.

N. Vinh, M. Chetty, R. Coppel, and P. Wangikar. GlobalMIT: Learning globally optimal dynamic Bayesian network with the mutual information test criterion. *Bioinformatics*, 27(19):2765–2766, 2011a.

N. Vinh, M. Chetty, R. Coppel, and P. Wangikar. Polynomial time algorithm for learning globally optimal dynamic Bayesian network. In *Neural Information Processing*, pages 719–729, 2011b.

W. Zhao, E. Serpedin, and E. R. Dougherty. Inferring gene regulatory networks from time series data using the minimum description length principle. *Bioinformatics*, 22 (17):2129–2135, 2006.