Journal of Artificial Intelligence Research 57 (2016) 1-37 Submitted 03/16; published 09/16

Learning Continuous Time Bayesian Networks in Non-stationary Domains

Simone Villa VILLA@DISCO.UNIMIB.IT
Fabio Stella STELLA@DISCO.UNIMIB.IT
Department of Informatics, Systems and Communication
University of Milano-Bicocca
Viale Sarca 336, 20126 Milan, Italy

Abstract

Non-stationary continuous time Bayesian networks are introduced. They allow the parents set of each node to change over continuous time. Three settings are developed for learning non-stationary continuous time Bayesian networks from data: known transition times, known number of epochs and unknown number of epochs. A score function for each setting is derived and the corresponding learning algorithm is developed. A set of numerical experiments on synthetic data is used to compare the effectiveness of non-stationary continuous time Bayesian networks to that of non-stationary dynamic Bayesian networks. Furthermore, the performance achieved by non-stationary continuous time Bayesian networks is compared to that achieved by state-of-the-art algorithms on four real-world datasets, namely drosophila, saccharomyces cerevisiae, songbird and macroeconomics.

1. Introduction

The identification of relationships and statistical dependencies between components in multivariate time-series, and the ability of reasoning about whether and how these dependencies change over time is crucial in many research domains such as biology, economics, finance, traffic engineering and neurology, to mention just a few. In biology, for example, knowing the gene regulatory network allows to understand complex biological mechanisms ruling the cell. In such a context, Bayesian networks (BNs) (Pearl, 1989; Segal, Pe'er, Regev, Koller, & Friedman, 2005; Scutari & Denis, 2014), dynamic Bayesian networks (DBNs) (Dean & Kanazawa, 1989; Zou & Conzen, 2005; Vinh, Chetty, Coppel, & Wangikar, 2012) and continuous time Bayesian networks (CTBNs) (Nodelman, Shelton, & Koller, 2002; Acerbi, Zelante, Narang, & Stella, 2014) have been used to reconstruct transcriptional regulatory networks from gene expression data. The effectiveness of discrete DBNs has been investigated to identify functional correlations among neuroanatomical regions of interest (Burge, Lane, Link, Qiu, & Clark, 2009), while a useful primer on BNs for functional magnetic resonance imaging data analysis has been made available (Mumford & Ramsey, 2014). However, the mentioned applications require the time-series to be generated from a stationary distribution, i.e. one which does not change over time. While stationarity is a reasonable assumption in many situations, there are cases where the data generating process is clearly non-stationary. Indeed, in the last years, researchers from different disciplines, ranging from economics to computational biology, to sociology and to medicine have become interested in representing relationships and dependencies which change over time.

Specifically, researchers have been interested in analyzing the temporal evolution of genetic networks (Lebre, Becq, Devaux, Stumpf, & Lelandais, 2010), the flow over neural information networks (Smith, Yu, Smulders, Hartemink, & Jarvis, 2006), heart failure (Liu, Hommersom, van der Heijden, & Lucas, 2016), complications in type 1 diabetes (Marini, Trifoglio, Barbarini, Sambo, Camillo, Malovini, Manfrini, Cobelli, & Bellazzi, 2015) and the dependence structure among financial markets during crisis (Durante & Dunson, 2014). According to the specialized literature on evolution models (Robinson & Hartemink, 2010), they can be divided into two main categories: *structurally non-stationary*, i.e. those models which are allowed to change their structure over time, and *parametrically non-stationary*, i.e. those models which only allow the parameters' values to change over time.

In this paper, the *structurally non-stationary continuous time Bayesian network* model (nsCTBN) is introduced. A nsCTBN consists of a sequence of CTBNs which improves expressiveness over a single CTBN. Indeed, a nsCTBN allows the parents set of each node to change over time at specific *transition times* and thus it allows to model non-stationary systems. To learn a nsCTBN, the Bayesian score for learning CTBNs is extended (Nodelman, Shelton, & Koller, 2003). The nsCTBN version of the Bayesian score is still decomposable by variable and it depends on the *knowledge setting* which can be: *known transition times*, where transition times are known, *known number of epochs*, where only the number of transition times is known, and *unknown number of epochs*, where the number of transition times is unknown. A learning algorithm for each knowledge setting is designed and developed. Experiments against non-stationary dynamic Bayesian networks (nsDBNs) (Robinson & Hartemink, 2010), i.e. the discrete time counterparts of nsCTBNs, have been performed. The main contributions of this paper are the following:

- definition of the structurally non-stationary continuous time Bayesian network model;
- derivation of the Bayesian score decomposition under each knowledge setting;
- the design of algorithms for learning nsCTBNs under different knowledge settings. A novel dynamic programming algorithm for learning nsCTBNs under the known transition times setting is described, while learning nsCTBNs under the others settings is performed by simulated annealing, exploiting the dynamic programming algorithm;
- performance comparison between nsCTBNs and nsDBNs under all knowledge settings for a rich set of synthetic data generated by nsCTBNs and nsDBNs;
- performance comparison between nsCTBNs and state-of-the-art algorithms on real-world datasets, namely *drosophila*, *saccharomyces cerevisiae* and *songbird*;
- a nsCTBN learned on a macroeconomics dataset consisting of variables evolving at different time granularities spanning from 1st January 1986 to 31st March 2015.

The rest of the paper is organized as follows. In Section 2 continuous time Bayesian networks are introduced together with their learning problem from complete data. Section 3 introduces non-stationary continuous time Bayesian networks, presents three learning settings and derives their corresponding Bayesian score functions. Algorithms for learning nsCTBNs under different learning settings are described in Section 4. Numerical experiments on synthetic and real-world datasets are presented in Section 5. Section 6 closes the paper by making conclusions and indicating directions for further research activities.

2. Continuous Time Bayesian Networks

Continuous time Bayesian networks combine Bayesian networks and homogeneous Markov processes together to efficiently model discrete state continuous time dynamical systems (Nodelman et al., 2002). They are particularly useful for modeling domains in which variables evolve at different time granularities, such as to model the presence of people at their computers (Nodelman & Horvitz, 2003), to study reliability of dynamical systems (Boudali & Dugan, 2006), to model failures in server farms (Herbrich, Graepel, & Murphy, 2007), to detect network intrusion (Xu & Shelton, 2008), to analyze social networks (Fan & Shelton, 2009), to model cardiogenic heart failure (Gatti, Luciani, & Stella, 2011) and to reconstruct gene regulatory networks (Acerbi & Stella, 2014; Acerbi, Viganò, Poidinger, Mortellaro, Zelante, & Stella, 2016). Recently, the complexity of inference in continuous time Bayesian networks has been studied (Sturlaugson & Sheppard, 2014).

2.1 Basics

The representation ability of continuous time Bayesian networks is inherent to the factorization of the system dynamics into local continuous time Markov processes that depend on a limited set of states. The continuous time Bayesian network model is defined as follows:

Definition 1. *Continuous time Bayesian network (Nodelman et al., 2002). Let $\boldsymbol{X}$ be a set of random variables $\boldsymbol{X} = \{X_1, X_2, \ldots, X_N\}$. Each X has a finite domain of values $Val(X) = \{x_1, x_2, \ldots, x_I\}$. A continuous time Bayesian network over $\boldsymbol{X}$ consists of two components: the first is an initial distribution $P^0_{\boldsymbol{X}}$, specified as a Bayesian network over $\boldsymbol{X}$, the second is a continuous time transition model specified as: a directed (possibly cyclic) graph $\mathcal{G}$ whose nodes are $X_1, X_2, \ldots, X_N$; a conditional intensity matrix (CIM), $\boldsymbol{Q}_X^{Pa(X)}$, for each variable $X \in \boldsymbol{X}$, where $Pa(X)$ denotes the set of parents of X in the graph $\mathcal{G}$.*

The conditional intensity matrix $\boldsymbol{Q}_X^{Pa(X)}$ consists of the set of intensity matrices

$$\boldsymbol{Q}_X^{pa_u} = \begin{bmatrix} -q_{x_1}^{pa_u} & \cdot & q_{x_1 x_I}^{pa_u} \\ q_{x_2 x_1}^{pa_u} & \cdot & q_{x_2 x_I}^{pa_u} \\ \cdot & \cdot & \cdot \\ q_{x_I x_1}^{pa_u} & \cdot & -q_{x_I}^{pa_u} \end{bmatrix},$$

where pa_u ranges over all possible configurations of the parents set $Pa(X)$, while $q_{x_i}^{pa_u} = \sum_{x_j \neq x_i} q_{x_i x_j}^{pa_u}$. Off-diagonal elements of $\boldsymbol{Q}_X^{pa_u}$, i.e. $q_{x_i x_j}^{pa_u}$, are proportional to the probability that the variable X transitions from state x_i to state x_j given the parents' state pa_u. The intensity matrix $\boldsymbol{Q}_X^{pa_u}$ can be equivalently summarized with two independent sets: $\boldsymbol{q}_X^{pa_u} = \{q_{x_i}^{pa_u} : 1 \leq i \leq I\}$, i.e. the set of intensities parameterizing the exponential distributions over *when* the next transition occurs, and $\boldsymbol{\theta}_X^{pa_u} = \{\theta_{x_i x_j}^{pa_u} = q_{x_i x_j}^{pa_u} / q_{x_i}^{pa_u} : 1 \leq i, j \leq I, j \neq i\}$, i.e. the set of probabilities parameterizing the multinomial distributions over *where* the state transitions. Note that the CTBN model assumes that only one single variable can change state at any specific instant, while its transition dynamics are specified by its parents via the CIM, and they are independent of all other variables given its *Markov Blanket*.

2.2 Structural Learning

Given a *fully observed dataset* $\mathcal{D}$, i.e. a dataset consisting of multiple *trajectories*[1] whose states and transition times are fully known, the problem of learning the structure of a CTBN has been addressed as the problem of selecting the graph $\mathcal{G}^*$ which maximizes the Bayesian score computed on the dataset $\mathcal{D}$ (Nodelman et al., 2003):

$$BS\left(\mathcal{G} : \mathcal{D}\right) = \ln P(\mathcal{G}) + \ln P(\mathcal{D}|\mathcal{G}). \tag{1}$$

where $P(\mathcal{G})$ is the *prior over the graph* $\mathcal{G}$ and $P(\mathcal{D}|\mathcal{G})$ is the *marginal likelihood.*

The prior $P(\mathcal{G})$ over the graph $\mathcal{G}$, which allows us to prefer some CTBN's structures over others, is usually assumed to satisfy the *structure modularity property* (Friedman & Koller, 2000), i.e. to decompose into the following product of terms:

$$P(\mathcal{G}) = \prod_{X \in \boldsymbol{X}} P(Pa(X) = Pa_{\mathcal{G}}(X)), \tag{2}$$

a term for each parents set $Pa_{\mathcal{G}}(X)$ in the graph $\mathcal{G}$. A uniform prior over $\mathcal{G}$ is often used.

The marginal likelihood $P(\mathcal{D}|\mathcal{G})$ depends on the prior over parameters $P(\boldsymbol{q}_{\mathcal{G}}, \boldsymbol{\theta}_{\mathcal{G}}|\mathcal{G})$ which is usually assumed to satisfy the *global parameter independence*, the *local parameter independence* and the *parameter modularity* properties, which are outlined below.

Global parameter independence (Spiegelhalter & Lauritzen, 1990) states that the parameters $\boldsymbol{q}_X^{Pa_{\mathcal{G}}(X)}$ and $\boldsymbol{\theta}_X^{Pa_{\mathcal{G}}(X)}$ associated with each variable X in a graph $\mathcal{G}$ are independent, thus the prior over parameters decomposes by variable as follows:

$$P(\boldsymbol{q}_{\mathcal{G}}, \boldsymbol{\theta}_{\mathcal{G}}|\mathcal{G}) = \prod_{X \in \boldsymbol{X}} P(\boldsymbol{q}_X^{Pa_{\mathcal{G}}(X)}, \boldsymbol{\theta}_X^{Pa_{\mathcal{G}}(X)}|\mathcal{G}). \tag{3}$$

Local parameter independence (Spiegelhalter & Lauritzen, 1990) asserts that the parameters associated with each configuration pa_u of the parents $Pa_{\mathcal{G}}(X)$ of a variable X are independent. Therefore, the parameters associated with each variable X are decomposable by parent configuration pa_u as follows:

$$P(\boldsymbol{q}_X^{Pa_{\mathcal{G}}(X)}, \boldsymbol{\theta}_X^{Pa_{\mathcal{G}}(X)}|\mathcal{G}) = \prod_{pa_u} \prod_{x_i} P(q_{x_i}^{pa_u}, \boldsymbol{\theta}_{x_i}^{pa_u}|\mathcal{G}). \tag{4}$$

Parameter modularity (Geiger & Heckerman, 1997) asserts that if a variable X has the same parents $Pa_{\mathcal{G}}(X) = Pa_{\mathcal{G}'}(X)$ in two distinct graphs $\mathcal{G}$ and $\mathcal{G}'$, then the probability density functions of the parameters associated with X must be identical:

$$P(\boldsymbol{q}_X^{Pa_{\mathcal{G}}(X)}, \boldsymbol{\theta}_X^{Pa_{\mathcal{G}}(X)}|\mathcal{G}) = P(\boldsymbol{q}_X^{Pa_{\mathcal{G}'}(X)}, \boldsymbol{\theta}_X^{Pa_{\mathcal{G}'}(X)}|\mathcal{G}'). \tag{5}$$

Furthermore, we also assume that the sets of parameters characterizing the exponential distributions are independent of the sets of parameters characterizing the multinomial distributions:

$$P(\boldsymbol{q}_{\mathcal{G}}, \boldsymbol{\theta}_{\mathcal{G}}|\mathcal{G}) = P(\boldsymbol{q}_{\mathcal{G}}|\mathcal{G})P(\boldsymbol{\theta}_{\mathcal{G}}|\mathcal{G}). \tag{6}$$

1. A trajectory is defined to be a sequence of pairs $(t,\ \boldsymbol{X}(t))$, where each transition time $t \in [0, T]$ is associated with the state $\boldsymbol{X}(t)$ of all the random variables corresponding to the nodes of the CTBN.

A Dirichlet distribution is selected as the prior for the parameters associated with the multinomial distribution, while a gamma distribution is selected as the prior for the parameters associated with the exponential distribution, i.e.

$$P(q_{x_i}^{pa_u}) \sim Gamma\left(\alpha_{x_i}^{pa_u}, \tau_{x_i}^{pa_u}\right), \tag{7}$$

$$P(\boldsymbol{\theta}_{x_i}^{pa_u}) \sim Dir\left(\alpha_{x_i x_1}^{pa_u}, \dots, \alpha_{x_i x_I}^{pa_u}\right), \tag{8}$$

where $\alpha_{x_i}^{pa_u}, \tau_{x_i}^{pa_u}, \alpha_{x_i x_1}^{pa_u}, \dots, \alpha_{x_i x_I}^{pa_u}$ are the priors' hyperparameters. In particular, the α hyperparameters represent the pseudocounts for the number of transitions from state to state, while the τ parameter represents the imaginary amount of time spent in each state before any data is observed. Note that the hyperparameter $\alpha_{x_i}^{pa_u}$ is inversely proportional to the number of joint states of the parents of X. Conditioning on the dataset $\mathcal{D}$, we obtain the following posteriors over parameters:

$$P(q_{x_i}^{pa_u}|\mathcal{D}) \sim Gamma\left(\alpha_{x_i}^{pa_u} + M_{x_i}^{pa_u}, \tau_{x_i}^{pa_u} + T_{x_i}^{pa_u}\right), \tag{9}$$

$$P(\boldsymbol{\theta}_{x_i}^{pa_u}|\mathcal{D}) \sim Dir\left(\alpha_{x_i x_1}^{pa_u} + M_{x_i x_1}^{pa_u}, \dots, \alpha_{x_i x_I}^{pa_u} + M_{x_i x_I}^{pa_u}\right), \tag{10}$$

where $T_{x_i}^{pa_u}$ and $M_{x_i x_j}^{pa_u}$ are the sufficient statistics of the CTBN (Nodelman et al., 2003). In particular, $T_{x_i}^{pa_u}$ is the amount of time spent by the variable X in the state x_i while its parents $Pa(X)$ are in state pa_u, while $M_{x_i x_j}^{pa_u}$ is the number of times that the variable X transitions from the state x_i to the state x_j while its parents $Pa(X)$ are in state pa_u[2].

In the Bayesian score (1) the term $P(\mathcal{G})$ does not grow with the size of dataset $\mathcal{D}$. Thus, the significant term is the marginal likelihood $P(\mathcal{D}|\mathcal{G})$. In the case of complete data, while exploiting the parameters' independence (6) and the global parameter independence property (3), the marginal likelihood can be written as follows:

$$P(\mathcal{D}|\mathcal{G}) = \prod_{X \in \boldsymbol{X}} ML(\boldsymbol{q}_X^{Pa_{\mathcal{G}}(X)}|\mathcal{D})\; ML(\boldsymbol{\theta}_X^{Pa_{\mathcal{G}}(X)}|\mathcal{D}), \tag{11}$$

where $ML(\boldsymbol{q}_X^{Pa_{\mathcal{G}}(X)}|\mathcal{D})$ is the marginal likelihood of $\boldsymbol{q}$ derived as follows:

$$\prod_{pa_u} \prod_{x_i} \frac{\Gamma\left(\alpha_{x_i}^{pa_u} + M_{x_i}^{pa_u} + 1\right) \left(\tau_{x_i}^{pa_u}\right)^{\left(\alpha_{x_i}^{pa_u}+1\right)}}{\Gamma\left(\alpha_{x_i}^{pa_u} + 1\right) \left(\tau_{x_i}^{pa_u} + T_{x_i}^{pa_u}\right)^{\left(\alpha_{x_i}^{pa_u} + M_{x_i}^{pa_u} + 1\right)}}, \tag{12}$$

and $ML(\boldsymbol{\theta}_X^{Pa_{\mathcal{G}}(X)}|\mathcal{D})$ is the marginal likelihood of $\boldsymbol{\theta}$ derived as follows:

$$\prod_{pa_u} \prod_{x_i = x_j} \frac{\Gamma\left(\alpha_{x_i}^{pa_u}\right)}{\Gamma\left(\alpha_{x_i}^{pa_u} + M_{x_i}^{pa_u}\right)} \times \prod_{x_i \neq x_j} \frac{\Gamma\left(\alpha_{x_i x_j}^{pa_u} + M_{x_i x_j}^{pa_u}\right)}{\Gamma\left(\alpha_{x_i x_j}^{pa_u}\right)}, \tag{13}$$

under the Bayesian-Dirichlet equivalent (BDe) metric version for CTBNs (Nodelman, 2007). In this case, the BDe metric uses the priors (7) and (8), while the parameter modularity (5), as well as the global (3) and the local (4) parameter independence properties are assumed to be satisfied.

2. Please note that the number of times the variable X leaves the state x_i while its parents $Pa(X)$ are in state pa_u is computed as follows $M_{x_i}^{pa_u} = \sum_{x_j \neq x_i} M_{x_i x_j}^{pa_u}$.

In conclusion, the Bayesian score (1) can be computed in closed form by assuming the structure modularity property (2) is satisfied, and using the BDe metric as follows:

$$BS(\mathcal{G} : \mathcal{D}) = \sum_{X \in \boldsymbol{X}} \ln P(Pa(X) = Pa_{\mathcal{G}}(X)) + \ln ML(\boldsymbol{q}_X^{Pa_{\mathcal{G}}(X)}|\mathcal{D}) + \ln ML(\boldsymbol{\theta}_X^{Pa_{\mathcal{G}}(X)}|\mathcal{D}). \quad (14)$$

Since the graph $\mathcal{G}$ of a CTBN does not have acyclicity constraints, it is possible to maximize the Bayesian score (14) by separately optimizing the parents set $Pa(X)$ for each variable X. It is worthwhile to mention that if the maximum number of parents is set, then the search of the optimal value of the Bayesian score (14) can be performed in polynomial time. The search can be performed by enumerating each possible parents set or by using a greedy hill-climbing procedure with operators to add, delete or reverse edges of the graph $\mathcal{G}$.

3. Non-stationary Continuous Time Bayesian Networks

Continuous time Bayesian networks are both *structurally stationary*, as the graph does not change over time, and *parametrically stationary*, as the conditional intensity matrices do not change over time. These stationarity assumptions are reasonable in many situations, but there are cases where the data generating process is intrinsically non-stationary and thus CTBNs can no longer be used. Therefore, in this section, we extend CTBNs to become structurally non-stationary. i.e. we allow the CTBN's structure to change over continuous time.

3.1 Definition

In the non-stationary continuous time Bayesian network model, the graph of the CTBN is replaced by a *graphs sequence* $\boldsymbol{\mathcal{G}} = (\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_E)$, where a graph $\mathcal{G}_e$ represents the causal dependency structure of the model for the epoch $e \in \{1, 2, \ldots, E\}$[3]. This model is structurally non-stationary because of the introduction of the graphs sequence and it can handle transition times that are common to the whole network and/or node-specific.

Following the notations and definitions used for non-stationary dynamic Bayesian networks, we let $\mathcal{T} = (t_1, \ldots, t_{E-1})$ be the *transition times* sequence, i.e. the times at which the causal dependency structure $\mathcal{G}_e$, active at epoch e, is replaced by the causal dependency structure $\mathcal{G}_{e+1}$, which becomes active at epoch $e+1$. An *epoch* is defined to be the period of time between two consecutive transitions, i.e. the epoch e is active during the period of time starting at t_{e-1} and ending at t_e. The graph $\mathcal{G}_{e+1}$, which is active during the epoch $e+1$, differs from the graph $\mathcal{G}_e$, which is active during the epoch e, in a set of edges that we call the *set of edge changes* $\Delta\mathcal{G}_e$.

Figure 1 shows a graphs sequence $\boldsymbol{\mathcal{G}} = (\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4)$ consisting of four epochs ($E = 4$) with transition times $\mathcal{T} = (t_1, t_2, t_3)$. Each epoch is associated with a set of edge changes. Specifically, the graph $\mathcal{G}_2$ differs from the graph $\mathcal{G}_1$ by the following set of edge changes $\Delta\mathcal{G}_1 = \{X_3 \to X_2, X_2 \not\to X_3, X_1 \not\to X_2\}$, the graph $\mathcal{G}_3$ differs from the graph $\mathcal{G}_2$ by the following set of edge changes $\Delta\mathcal{G}_2 = \{X_2 \to X_1\}$ and the graph $\mathcal{G}_4$ differs from the graph $\mathcal{G}_3$ by the following set of edge changes $\Delta\mathcal{G}_3 = \{X_3 \to X_4, X_4 \to X_1, X_1 \not\to X_4, X_4 \not\to X_3\}$.

3. It is worthwhile to mention that the first epoch, i.e. the epoch starting at time 0 and ending at time t_1 is associated with the graph $\mathcal{G}_1$, while the last epoch, i.e. the epoch starting at time t_{E-1} and ending at time T (the supremum of the considered time interval, i.e. [0,T]) is associated with the graph $\mathcal{G}_E$.

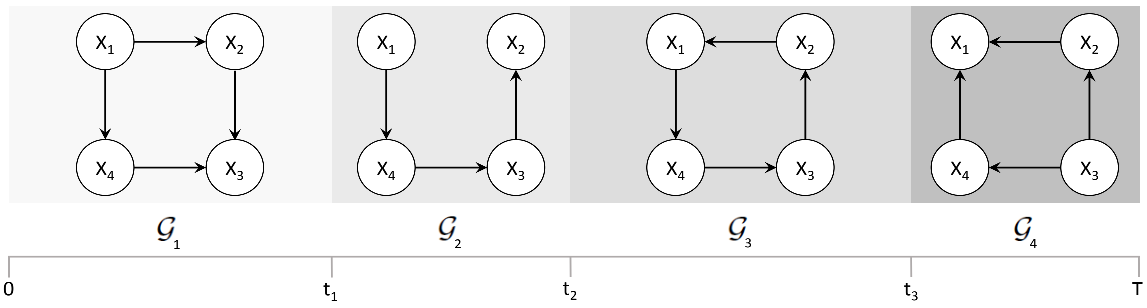


Figure 1: Graphs sequence $\boldsymbol{\mathcal{G}} = (\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4)$ of a nsCTBN with four epochs, $E = 4$, and three transition times, $\mathcal{T} = (t_1, t_2, t_3)$, where the edges are gained and lost over time.

Non-stationary continuous time Bayesian networks allow each node to have its own sequence of parents sets, each parents set being active at a given epoch. Therefore, we introduce the concept of *homogeneous interval* $H(X) = (h_1, \ldots, h_M)$ associated with node X, which is defined as the union of consecutive epochs during which the same parents set $Pa(X)$ is active for the node X. Note that if each epoch is associated with a different parents set, then M is equal to E.

A non-stationary continuous time Bayesian network is defined as follows.

Definition 2. *(Structurally) non-stationary continuous time Bayesian network. Let $\boldsymbol{X}$ be a set of random variables $X_1, \ldots, X_N$. Each X has a finite domain of values $Val(X) = \{x_1, \ldots, x_I\}$. A (structurally) non-stationary continuous time Bayesian network $\mathcal{N}_{ns} = (\mathcal{B}, \mathcal{M}_{ns})$ over $\boldsymbol{X}$ consists of two components:*

- *an initial distribution $P^0_{\boldsymbol{X}}$, specified as a Bayesian network $\mathcal{B}$ over $\boldsymbol{X}$,*
- *a non-stationary continuous time transition model $\mathcal{M}_{ns}$ specified as:*
  - *a sequence of directed (possibly cyclic) graphs $\boldsymbol{\mathcal{G}} = (\mathcal{G}_e)_{e=1}^{E}$ whose nodes are $X_1, \ldots, X_N$, where E represents the number of epochs;*
  - *a conditional intensity matrix, $\boldsymbol{Q}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}_{X,H(X)}$, $\forall X \in \boldsymbol{X}$, where $Pa_{\boldsymbol{\mathcal{G}}}(X)$ denotes the parents sets of X in $\boldsymbol{\mathcal{G}}$, and $H(X)$ denotes the intervals associated with X.*

The conditional intensity matrix $\boldsymbol{Q}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}_{X,H(X)}$ consists of a set of intensity matrices

$$\boldsymbol{Q}^{pa_u}_{X,h_m} = \begin{bmatrix} -q^{pa_u}_{x_1,h_m} & \cdot & q^{pa_u}_{x_1 x_I,h_m} \\ q^{pa_u}_{x_2 x_1,h_m} & \cdot & q^{pa_u}_{x_2 x_I,h_m} \\ \cdot & \cdot & \cdot \\ q^{pa_u}_{x_I x_1,h_m} & \cdot & -q^{pa_u}_{x_I,h_m} \end{bmatrix},$$

one for each configuration pa_u of each parents set $Pa(X) \in Pa_{\boldsymbol{\mathcal{G}}}(X)$ which is active during the interval $h_m \in H(X)$.[4]

4. Note that the following equation $q^{pa_u}_{x_i,h_m} = \sum_{x_j \neq x_i} q^{pa_u}_{x_i x_j,h_m}$ still holds.

3.2 Learning Framework

Learning a nsCTBN from a fully observed dataset $\mathcal{D}$ can be done using the Bayesian learning framework taking into account the entire graphs sequence $\boldsymbol{\mathcal{G}}$. In the nsCTBNs case, we must specify the prior probability over the graphs sequence $\boldsymbol{\mathcal{G}}$ and, for each possible sequence, the density measure over possible values of the parameters $\boldsymbol{q}_{\boldsymbol{\mathcal{G}}}$ and $\boldsymbol{\theta}_{\boldsymbol{\mathcal{G}}}$. Once they prior $P(\boldsymbol{\mathcal{G}})$ and the likelihood $P(\boldsymbol{q}_{\boldsymbol{\mathcal{G}}}, \boldsymbol{\theta}_{\boldsymbol{\mathcal{G}}}|\boldsymbol{\mathcal{G}})$ are given, the marginal likelihood $P(\mathcal{D}|\boldsymbol{\mathcal{G}})$ can be computed and the Bayesian score can be evaluated. It is important to note that we are focused on recovering the graphs sequence $\boldsymbol{\mathcal{G}}$ and not on detecting possible changes of the parameters. In fact, we identify non-stationarity in the parameters of the model, i.e. the entries of the conditional intensity matrices, that are significant enough to result in structural changes of the graph. Others changes are assumed to be small enough not to alter the graph structure.

3.2.1 Prior Probability over Graphs

Given the transition times $\mathcal{T}$, and thus the number of epochs E, we assume that the prior over the nsCTBN's structure $\boldsymbol{\mathcal{G}}$ can be written as follows:

$$P(\boldsymbol{\mathcal{G}}|\mathcal{T}) = P(\mathcal{G}_1, ..., \mathcal{G}_E|\mathcal{T}) = P(\mathcal{G}_1, \Delta\mathcal{G}_1, ..., \Delta\mathcal{G}_{E-1}|\mathcal{T}) = P(\mathcal{G}_1)P(\Delta\mathcal{G}_1, ..., \Delta\mathcal{G}_{E-1}|\mathcal{T}). \tag{15}$$

Equation (15) is justified because we assume that the probability distribution over edge changes only is a function of the number of changes performed, which can also be defined independently of the initial graph $\mathcal{G}_1$. If some knowledge about particular edges or the overall topology is available for the initial network, then we can use an informative prior $P(\mathcal{G}_1)$ otherwise we can resort to a uniform distribution. As in CTBNs, $P(\mathcal{G}_1)$ must satisfy the structure modularity assumption (2), while the prior over the set of edge changes $P(\Delta\mathcal{G}_1, \dots, \Delta\mathcal{G}_{E-1}|\mathcal{T})$ defines the way in which edges change through adjacent epochs.

3.2.2 Prior Probability over Parameters

The prior over parameters $P(\boldsymbol{q}_{\boldsymbol{\mathcal{G}}}, \boldsymbol{\theta}_{\boldsymbol{\mathcal{G}}}|\boldsymbol{\mathcal{G}}, \mathcal{T})$ is selected to satisfy the following assumptions: independence between the sets of parameters characterizing the exponential and the multinomial distributions (6), parameter modularity (5) and parameter independence. The latter assumption is divided into three components for nsCTBNs: *global parameter independence*, *interval parameter independence* and *local parameter independence*.

Global parameter independence asserts that the parameters associated with each node in a nsCTBN's graphs sequence are independent, so the prior over parameters decomposes by variable X as follows:

$$P(\boldsymbol{q}_{\boldsymbol{\mathcal{G}}}, \boldsymbol{\theta}_{\boldsymbol{\mathcal{G}}}|\boldsymbol{\mathcal{G}}, \mathcal{T}) = \prod_{X \in \boldsymbol{X}} P(\boldsymbol{q}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}, \boldsymbol{\theta}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}|\boldsymbol{\mathcal{G}}, \mathcal{T}). \tag{16}$$

Interval parameter independence states that the parameters associated with each interval of the active parents for each node are independent, so the parameters associated with each X and its parents sets $Pa_{\boldsymbol{\mathcal{G}}}(X)$ are decomposable by interval $h_m \in H(X)$ as follows:

$$P(\boldsymbol{q}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}, \boldsymbol{\theta}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}|\boldsymbol{\mathcal{G}}, \mathcal{T}) = \prod_{h_m} P(\boldsymbol{q}_{X,h_m}^{Pa_{\mathcal{G}}(X)}, \boldsymbol{\theta}_{X,h_m}^{Pa_{\mathcal{G}}(X)}|\boldsymbol{\mathcal{G}}, \mathcal{T}). \tag{17}$$

Local parameter independence states that the parameters associated with each state of a variable in a given interval are independent, thus the parameters associated with each X in the interval $h_m \in H(X)$ are decomposable by parent configuration pa_u as follows:

$$P(\boldsymbol{q}_{X,h_m}^{Pa_{\mathcal{G}}(X)}, \boldsymbol{\theta}_{X,h_m}^{Pa_{\mathcal{G}}(X)} | \boldsymbol{\mathcal{G}}, \mathcal{T}) = \prod_{pa_u} \prod_{x_i} P(q_{x_i,h_m}^{pa_u}, \boldsymbol{\theta}_{x_i,h_m}^{pa_u} | \boldsymbol{\mathcal{G}}, \mathcal{T}). \tag{18}$$

As in the CTBNs case, a Dirichlet distribution is used as prior for the parameters of the multinomial distribution and a gamma distribution is used as prior for the parameters of the exponential distribution. The sufficient statistics are modified as follows: $T_{x_i,h_m}^{pa_u}$ is the amount of time spent in state $X = x_i$ while $Pa(X) = pa_u$ in the interval $H(X) = h_m$, while $M_{x_i x_j,h_m}^{pa_u}$ is the number of transitions from state $X = x_i$ to state $X = x_j$ while $Pa(X) = pa_u$ in the interval $H(X) = h_m$. We let $M_{x_i,h_m}^{pa_u} = \sum_{x_j \neq x_i} M_{x_i x_j,h_m}^{pa_u}$ to be the number of times X leaves state x_i while its parents $Pa(X)$ are in state pa_u during the interval $H(X) = h_m$.

3.2.3 Marginal Likelihood

Given the graphs sequence $\boldsymbol{\mathcal{G}}$, and the transition times $\mathcal{T}$, the marginal likelihood $P(\mathcal{D}|\boldsymbol{\mathcal{G}}, \mathcal{T})$ of the dataset $\mathcal{D}$ can be computed in closed form using the priors and the sufficient statistics previously defined. To derive the Bayesian-Dirichlet equivalent metric for nsCTBNs, we make the same assumptions as those for CTBNs. In this case, the parameter independence assumption is divided into global (16), interval (17) and local (18) parameter independence. Therefore, the marginal likelihood becomes:

$$P(\mathcal{D}|\boldsymbol{\mathcal{G}}, \mathcal{T}) = \prod_{X \in \boldsymbol{X}} ML(\boldsymbol{q}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}|\mathcal{D}) \; ML(\boldsymbol{\theta}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}|\mathcal{D}). \tag{19}$$

The marginal likelihood of $\boldsymbol{q}$ in equation (19) can be calculated as follows:

$$ML(\boldsymbol{q}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}|\mathcal{D}) = \prod_{h_m} \prod_{pa_u} \prod_{x_i} \frac{\Gamma\left(\alpha_{x_i,h_m}^{pa_u} + M_{x_i,h_m}^{pa_u} + 1\right)\left(\tau_{x_i,h_m}^{pa_u}\right)^{(\alpha_{x_i,h_m}^{pa_u}+1)}}{\Gamma\left(\alpha_{x_i,h_m}^{pa_u} + 1\right)\left(\tau_{x_i,h_m}^{pa_u} + T_{x_i,h_m}^{pa_u}\right)^{(\alpha_{x_i,h_m}^{pa_u}+M_{x_i,h_m}^{pa_u}+1)}}, \tag{20}$$

while the marginal likelihood of $\boldsymbol{\theta}$ in equation (19) can be calculated as follows:

$$ML(\boldsymbol{\theta}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}|\mathcal{D}) = \prod_{h_m} \prod_{pa_u} \prod_{x_i = x_j} \frac{\Gamma\left(\alpha_{x_i,h_m}^{pa_u}\right)}{\Gamma\left(\alpha_{x_i,h_m}^{pa_u} + M_{x_i,h_m}^{pa_u}\right)} \times \prod_{x_i \neq x_j} \frac{\Gamma\left(\alpha_{x_i x_j,h_m}^{pa_u} + M_{x_i x_j,h_m}^{pa_u}\right)}{\Gamma\left(\alpha_{x_i x_j,h_m}^{pa_u}\right)}. \tag{21}$$

It is important to note that for nsCTBNs, the pseudocounts α as well as the imaginary amount of time τ are associated with each interval. This aspect requires a careful choice in order not to be too biased towards these values when small intervals are analyzed.

A possible correction is to weight the CTBN's hyperparameters by a quantity proportional to the time interval width $(h_m - h_{m-1})$, where h_M denotes the total time. Thus, the nsCTBN's hyperparameters could be defined as follows:

$$\alpha_{x_i x_j,h_m}^{pa_u} = \alpha_{x_i x_j}^{pa_u} \frac{(h_m - h_{m-1})}{h_M}, \tag{22}$$

$$\tau_{x_i,h_m}^{pa_u} = \tau_{x_i}^{pa_u} \frac{(h_m - h_{m-1})}{h_M}. \tag{23}$$

If you want to control the parameter priors using only two hyperparameters α and τ, then you can use the uniform BDe for nsCTBNs (BDeu). In this case, the hyperparameters defined in (22) and (23) are divided by the number U of possible configurations of the parents $Pa(X)$ of node X times the cardinality I of the domain of X, as follows:

$$\begin{aligned} \alpha_{x_i x_j, h_m}^{pa_u} &= \frac{\alpha}{U\,I} \frac{(h_m - h_{m-1})}{h_M}, && (24) \\ \tau_{x_i, h_m}^{pa_u} &= \frac{\tau}{U\,I} \frac{(h_m - h_{m-1})}{h_M}. && (25) \end{aligned}$$

Equations (22) and (23) rescale the hyperparameters in such a way not to be biased with respect to the epochs' length, while equations (24) and (25) are based on the uniform distribution and they have been used for performing all numerical experiments.

3.3 Bayesian Score Decomposition

The Bayesian score can be decomposed by variable based on the information available about the transition times. In this regard, three knowledge settings are used to derive the Bayesian score, namely: known transition times (KTT), known number of epochs (KNE) and unknown number of epochs (UNE).

3.3.1 Known Transition Times

In this setting, the transition times $\mathcal{T}$ are known. Thus, the prior probability over the graphs sequence $P(\boldsymbol{\mathcal{G}}|\boldsymbol{\mathcal{T}})$ decomposes as in equation (15), while the marginal likelihood decomposes by variable X according to equation (19).

Therefore, the Bayesian score $BS(\boldsymbol{\mathcal{G}} : \mathcal{D}, \mathcal{T})$ can be written as follows:

$$\begin{aligned} BS(\boldsymbol{\mathcal{G}} : \mathcal{D}, \mathcal{T}) &= \ln P(\mathcal{G}_1) + \ln P(\Delta\mathcal{G}_1, \ldots, \Delta\mathcal{G}_{E-1}|\mathcal{T}) \\ &+ \ln ML(\boldsymbol{q}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}|\mathcal{D}) + \ln ML(\boldsymbol{\theta}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}|\mathcal{D}). \qquad (26) \end{aligned}$$

In such a setting the structural learning problem of a non-stationary continuous time Bayesian network consists of finding the graph $\mathcal{G}_1$ active during the first epoch ($e = 1$) and the $E-1$ sets of edge changes $\Delta\mathcal{G}_1, \ldots, \Delta\mathcal{G}_{E-1}$ together with the corresponding parameters values, which maximize the Bayesian score defined in equation (26).

The graphs $\mathcal{G}_2, \ldots, \mathcal{G}_E$ are selected by making assumptions on the ways by which the edges change over continuous time. A common approach (Robinson & Hartemink, 2010) consists of assuming that the graphs sequence $\boldsymbol{\mathcal{G}} = (\mathcal{G}_1, \ldots, \mathcal{G}_E)$ depends on a parameter which controls the number of edge changes over continuous time. This approach uses a truncated geometric distribution, with parameter $p = 1 - \exp(-\lambda_c)$, to model the *number of parents' changes* occurring at transition time t_{e+1}:

$$c_e = \sum_{X \in \boldsymbol{X}} |\Delta\mathcal{G}_e(X)|. \qquad (27)$$

The variable c_e counts the number of edge changes between two consecutive graphs $\mathcal{G}_e$ and $\mathcal{G}_{e+1}$, while the parameter λ_c controls the impact of the number of edge changes c_e on the score function (26).

If the edge changes $\Delta\mathcal{G}_e$ are assumed to be mutually independent, then the probability for the edge changes through subsequent epochs can be written as follows:

$$P(\Delta\mathcal{G}_1, \ldots, \Delta\mathcal{G}_{E-1}|\mathcal{T}) = \prod_{e=1}^{E-1} \frac{(1-\exp(-\lambda_c))(\exp(-\lambda_c))^{c_e}}{1-(\exp(-\lambda_c))^{c_{max}+1}} \propto \prod_{e=1}^{E-1} (\exp(-\lambda_c))^{c_e}, \quad (28)$$

where c_{max} is the *truncation term*. Therefore, if we assume a truncated geometric distribution on the number of parents' changes occurring at each transition times and equation (28) holds, then the Bayesian score (26) decomposes by variable X as follows:

$$\begin{aligned} BS(\boldsymbol{\mathcal{G}} : \mathcal{D}, \mathcal{T}) &= \sum_{X \in \boldsymbol{X}} \ln P(Pa(X) = Pa_{\mathcal{G}_1}(X)) - \lambda_c \sum_{e=1}^{E-1} c_e \\ &+ \ln ML(\boldsymbol{q}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}|\mathcal{D}) + \ln ML(\boldsymbol{\theta}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}|\mathcal{D}). \end{aligned} \quad (29)$$

It is worthwhile to notice that the number of parents' changes c_e for each epoch e penalizes the Bayesian score, and thus it discourages sudden variations in the parents set between consecutive epochs, while the parameter λ_c controls the impact of such changes on the score function (26).

3.3.2 Known Number of Epochs

If the transition times $\mathcal{T}$ are unknown, then the Bayesian score can be written as follows:

$$BS(\boldsymbol{\mathcal{G}}, \mathcal{T} : \mathcal{D}) = \ln P(\boldsymbol{\mathcal{G}}, \mathcal{T}) + \ln P(\mathcal{D}|\boldsymbol{\mathcal{G}}, \mathcal{T}). \quad (30)$$

Assuming that $P(\boldsymbol{\mathcal{G}}, \mathcal{T}) = P(\boldsymbol{\mathcal{G}})P(\mathcal{T})$ the Bayesian score (30) becomes:

$$BS(\boldsymbol{\mathcal{G}}, \mathcal{T} : \mathcal{D}) = \ln P(\boldsymbol{\mathcal{G}}) + \ln P(\mathcal{T}) + \ln P(\mathcal{D}|\boldsymbol{\mathcal{G}}, \mathcal{T}). \quad (31)$$

If the number of epochs E is known, then the prior probability $P(\boldsymbol{\mathcal{G}})$ over the graphs sequence $\boldsymbol{\mathcal{G}}$ decomposes as in equation (15), while a truncated geometric distribution can be used on the number of parents' changes occurring at each transition time, as in the known transition times setting.

Any choice for $P(\mathcal{T})$ can be made to include prior knowledge about the set of transition times. However, if no information is available, a uniform prior on $P(\mathcal{T})$ is used, implying that all possible values of transition times are equally likely for a given number of epochs E. Thus, the Bayesian score (31) can be decomposed by variable X as follows:

$$\begin{aligned} BS(\boldsymbol{\mathcal{G}}, \mathcal{T} : \mathcal{D}) &= \ln P(\mathcal{T}) + \sum_{X \in \boldsymbol{X}} \ln P(Pa(X) = Pa_{\mathcal{G}_1}(X)) - \lambda_c \sum_{e=1}^{E-1} c_e \\ &+ \ln ML(\boldsymbol{q}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}|\mathcal{D}) + \ln ML(\boldsymbol{\theta}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)}|\mathcal{D}), \end{aligned} \quad (32)$$

where c_e counts the number of edge changes between two consecutive parents sets, while λ_c controls the impacts on $BS(\boldsymbol{\mathcal{G}}, \mathcal{T} : \mathcal{D})$ of such edge changes, as it happens under the KTT setting.

3.3.3 Unknown Number of Epochs

If the number of epochs E is unknown, then transition times $\mathcal{T}$ are unknown as well. Under this setting, we learn a nsCTBN by exploiting what introduced under the KTT and KNE settings. We assume that the structure of the non-stationary continuous time Bayesian network can evolve at different speeds over continuous time. Such an assumption is incorporated by using a truncated geometric distribution with parameter $p = 1 - \exp(-\lambda_e)$ on the number of epochs. In general, large values of λ_e encode the strong prior belief that the structure of the nsCTBN changes slowly (i.e. few epochs exist).

Following what we presented under the KTT setting, the Bayesian score can be obtained by subtracting the parameter λ_e times the number of epochs E. Therefore, the Bayesian score $BS(\boldsymbol{\mathcal{G}}, \mathcal{T} : \mathcal{D})$ decomposes by variable X as follows:

$$\begin{aligned} BS(\boldsymbol{\mathcal{G}}, \mathcal{T} : \mathcal{D}) &= \ln P(\mathcal{T}) - \lambda_e E + \sum_{X \in \boldsymbol{X}} \ln P(Pa(X) = Pa_{\mathcal{G}_1}(X)) - \lambda_c \sum_{e=1}^{E-1} c_e \\ &+ \ln ML(\boldsymbol{q}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)} | \mathcal{D}) + \ln ML(\boldsymbol{\theta}_{X,H(X)}^{Pa_{\boldsymbol{\mathcal{G}}}(X)} | \mathcal{D}). \end{aligned} \qquad (33)$$

Note that the Bayesian score (33) contains two parameters, namely λ_c and λ_e, which encode our prior belief about the structure of the nsCTBN. Specifically, the parameter λ_c regulates our prior belief about the smoothness of the edge changes (e.g. encouraging or discouraging the edge changes per epoch), while the parameter λ_e regulates our prior belief about the number of epochs (e.g. encouraging or discouraging the creation of epochs).

4. Structural Learning

The optimal structure of nsCTBNs can be found by separately maximizing the components of the Bayesian score associated with each node. This can be achieved by using an exact optimization algorithm based on dynamic programming when the transition times are given. By contrast, when only the number of epochs is known or no information about the transition times is available, we have to resort to approximate techniques based on Monte Carlo or on simulated annealing. We present the exact algorithm for solving the structural learning problem under the KTT setting. Then, we briefly outline the stochastic algorithms to solve the structural learning problem under the KNE setting and under the UNE setting.

4.1 Known Transition Times

Under this setting the Bayesian score decomposes according to equation (29). Thus, the *optimal graphs sequence* $\boldsymbol{\mathcal{G}}^*$ can be found by separately searching the *optimal parents sequence* $\boldsymbol{\mathcal{G}}_X^*$ for each node X. To solve the problem of finding the optimal parents sequence $\boldsymbol{\mathcal{G}}_X^*$ for node X we consider a sequence consisting of M intervals $H(X) = (h_1, \ldots, h_M)$ and S possible parents, so to have $Z = 2^S$ possible parents sets. To find the optimal parents sequence $\boldsymbol{\mathcal{G}}_X^*$ we must compute $M \times Z$ marginal likelihood terms associated with $\boldsymbol{q}$ and $\boldsymbol{\theta}$, one marginal likelihood term for each possible parents set $Pa_z(X)$ and each interval h_m. Then, an optimization algorithm can be used to find the maximum of the component of the Bayesian score associated with the node X.

An exhaustive search would be prohibitive, as it would require evaluating Z^M scores, one for each possible parents sequence $\boldsymbol{\mathcal{G}}_X$. Unfortunately, also a greedy search strategy that computes the parents set which maximizes the Bayesian score for each interval is not viable. In fact, the function that counts the parents' changes c_e in (29) binds the choice of the subsequent parents set, i.e. it binds $\mathcal{G}_e$ to $\mathcal{G}_{e+1}$.

However, the relation between the score of the variable X associated with the parents set $Pa(X)$ in the interval h_m, denoted as $BS^{Pa(X)}_{X,h_m}$, and the score associated with the parents set $Pa(X)$ in the interval h_{m-1}, denoted as $BS^{Pa(X)}_{X,h_{m-1}}$, can be defined by recursion as follows:

$$BS^{Pa(X)}_{X,h_m} = \max_{Pa_z} \left\{ BS^{Pa_z(X)}_{X,h_{m-1}} - \lambda_c c_{X,e} + \ln ML(\boldsymbol{q}^{Pa(X)}_{X,h_m}, \boldsymbol{\theta}^{Pa(X)}_{X,h_m} | \mathcal{D}) \right\}, \tag{34}$$

where $c_{X,e} = |\Delta\mathcal{G}_e(X)|$, while the marginal likelihoods of $\boldsymbol{q}$ and $\boldsymbol{\theta}$ are grouped together. The score $BS^{Pa(X)}_{X,h_m}$, associated with the parents set $Pa(X)$ for node X in the interval h_m, is introduced to clarify the recursion used in the Algorithm 1. Note that this score depends on all the components of the score up to h_m. In particular, not only the marginal likelihoods component is involved, but also the term $c_{X,e}$, which counts the parents' changes, is included as it binds the choice of subsequent parents sets. Equation (34) is exploited by dynamic programming to select the optimal parents sequence $\boldsymbol{\mathcal{G}}^*_X$ for each node X.

Algorithm 1 takes as input the marginal likelihoods of $\boldsymbol{q}$ and $\boldsymbol{\theta}$ for each interval and parents set, the prior probability about the initial parents set, the number of parents' changes, and the parameter λ_c. Algorithm 1 ensures the optimal parents sequence $\boldsymbol{\mathcal{G}}^*_X$ for the node X and its corresponding optimal Bayesian score. Its core is the computation of the $M \times Z$ score matrix, denoted by SC, through the dynamic programming recursion. The dynamic programming recursion for the interval h_1 ($m = 1$) is defined as follows:

$$SC^z_1 = \ln ML(\boldsymbol{q}^{Pa_z(X)}_{X,h_1}, \boldsymbol{\theta}^{Pa_z(X)}_{X,h_1} | \mathcal{D}) + \ln P(Pa_z(X) = Pa_{\mathcal{G}_{h_1}}(X)), \tag{35}$$

for $1 \le z \le Z$, while, for the intervals h_m ($m = 2, \ldots, M$), the recursion is:

$$SC^z_m = \max_{1 \le u \le Z} \left\{ SC^u_{m-1} + \ln ML(\boldsymbol{q}^{Pa_z(X)}_{X,h_m}, \boldsymbol{\theta}^{Pa_z(X)}_{X,h_m} | \mathcal{D}) - \lambda_c c_{X,e} \right\}.$$

After filling the score matrix SC, the value $\max_z\{SC[M, z]\}$ is the optimal Bayesian score, while the optimal parents sequence is reconstructed backwards from M to 1 by using the index matrix IN. The cost of computing the dynamic programming recursion is $O(M \times Z^2)$, which is polynomial for a fixed maximum number of parents S.

The problem of selecting the optimal parents sequence has an interesting graph representation. Indeed, it is possible to create a graph whose nodes are associated with marginal likelihoods of $\boldsymbol{q}$ and $\boldsymbol{\theta}$ for the interval h_m and for the parents set $Pa_z(X)$, while each node associated with the interval h_m is linked with all the nodes associated with the interval h_{m+1}. Each arc is associated with a weight computed as the difference between the marginal likelihoods in the interval h_m for the parents set $Pa_z(X)$ and the cost of switching from the parents set of the interval h_{m-1} to the parents set of the interval h_m. Two special nodes are added to represent the start and the end of the optimal parents sequence. Such a graph does not have cycles, thus the selection of the optimal parents sequence for each node can be reduced to the longest path problem from the start node to the end node of a directed acyclic graph, and thus it can be solved using either dynamic or linear programming.

Algorithm 1 LearnKTTX

Require: matrix containing the marginal likelihoods of $\boldsymbol{q}$ and $\boldsymbol{\theta}$ $MLX[M, Z]$, vector containing the prior probability about the initial parents set $PR[Z]$, matrix containing the number of parents' changes $C[Z, Z]$ and the parameter for the parents' changes λ_c.

Ensure: score matrix $SC[M, Z]$ and index matrix $IN[M, Z]$.

```
1:  Initialize $SC[m, z] \leftarrow -\infty, IN[m, z] \leftarrow 0$.
2:  for $m \leftarrow 1, \dots, M$ do
3:    for $z \leftarrow 1, \dots, Z$ do
4:      if $(m = 1)$ then
5:        $SC[m, z] \leftarrow \ln MLX[m, z] + \ln PR[z]$
6:      else
7:        for $w \leftarrow 1, \dots, Z$ do
8:          $score \leftarrow SC[m-1, w] + \ln MLX[m, z] - \lambda_c C[w, z]$
9:          if $(score > SC[m, z])$ then
10:           $SC[m, z] \leftarrow score$
11:           $IN[m, z] \leftarrow w$
12:         end if
13:       end for
14:     end if
15:   end for
16: end for
```

Learning a nsCTBN can be done following the following four steps procedure: *i)* use the dataset $\mathcal{D}$ to compute for each variable X the sufficient statistics $T^{pa_u}_{x_i,h_m}$ and $M^{pa_u}_{x_i x_j,h_m}$ according to the given transition times $\mathcal{T}$; *ii)* compute the marginal likelihoods (20) and (21), and then fill the MLX matrix; *iii)* run Algorithm 1 for each node X to get the corresponding optimal parents sequence; *iv)* collect the optimal parents sequence for each node X and compute the corresponding CIMs using the sufficient statistics already computed in step *i)*.

If we allow the intervals to differ from the transition times, i.e. they can be obtained as one of all the possible unions of transition times; then we have to repeat the learning procedure for all the $E \times (E-1)/2$ cases. It is possible to speed up the computation because the sufficient statistics can be aggregated through intervals. In such a way, we read the dataset once, while the precomputed marginal likelihoods can be stored and reused for the same intervals. Moreover, the computations can be performed in parallel for each node.

4.2 Known Number of Epochs

In this setting, we know the number of epochs, but the transition times are not given, so we cannot directly apply Algorithm 1. However, once a *tentative allocation* $\hat{\mathcal{T}}$ of the transition times is given, we can apply Algorithm 1 to obtain the optimal nsCTBN's structure, under the assumption that $\hat{\mathcal{T}}$ is not too different from the true transition times $\mathcal{T}$. To find an *optimal tentative allocation* $\hat{\mathcal{T}}^*$, i.e. an allocation that is as close as possible to $\mathcal{T}$, we apply the *simulated annealing* (SA) algorithm (Kirkpatrick, Gelatt, & Vecchi, 1983).

Simulated annealing is an iterative algorithm that attempts to find the global optimum $\boldsymbol{x^*}$ of a given function $f(\boldsymbol{x})$ through a stochastic search over the feasible region. At iteration k, when the SA algorithm is assumed to be in state $\boldsymbol{x_k}$, it samples a *proposal state* $\boldsymbol{x'}$ according to some *proposal distribution* $\boldsymbol{x'} \sim \mathcal{P}'(\cdot|\boldsymbol{x_k})$. Then, the SA algorithm computes the quantity $\alpha = \exp\left(-(f(\boldsymbol{x}) - f(\boldsymbol{x'}))/CT\right)$, where CT is the *computational temperature*. The SA algorithm accepts the proposal state $\boldsymbol{x'}$ with probability equal to $\min\{1, \alpha\}$. Concisely, SA always accepts any proposal state $\boldsymbol{x'}$ where $f(\boldsymbol{x'}) > f(\boldsymbol{x})$ by setting $\boldsymbol{x_{k+1}} = \boldsymbol{x'}$, while it accepts the proposal state $\boldsymbol{x'}$ when $f(\boldsymbol{x'}) < f(\boldsymbol{x})$ with probability α by setting $\boldsymbol{x_{k+1}} = \boldsymbol{x'}$ with probability α and $\boldsymbol{x_{k+1}} = \boldsymbol{x_k}$ with probability $(1 - \alpha)$, i.e. in this case the state of the SA algorithm does not change. The computational temperature reduces over iterations according to a *cooling schedule*. It has been shown that if one cools *sufficiently slowly*, then the algorithm will probably find the global optimum (Kirkpatrick et al., 1983). The design of the cooling schedule is an important part of the SA algorithm (Bertsimas & Tsitsiklis, 1993). A possible approach is to use an *exponential cooling schedule* defined as follows: $CT_k = CT_0 \times \zeta^k$, where CT_0 represents the *initial temperature*, typically set to 1.0, ζ is the *cooling rate*, usually set to be close to 0.8, while k is the current iteration (Murphy, 2012).

In the nsCTBNs case, the state of the SA algorithm $\boldsymbol{x}$ is associated with the tentative allocation $\hat{\mathcal{T}}$, while the function $f(\boldsymbol{x})$ is the Bayesian score (32). Algorithm 2 takes as input the sufficient statistics, the parameters used to run Algorithm 1 and the parameters of the SA algorithm. It solves the structural learning problem under the KNE setting for a given variable X by ensuring the optimal tentative allocation $\hat{\mathcal{T}}^*$ and its corresponding score.

Algorithm 2 LearnKNEX

```
Require: sufficient statistics $SuffStatsX$, prior probability $PR[]$, number of parents' changes $C[,]$, parameter $\lambda_c$, tentative allocation $\hat{\mathcal{T}}$, initial temperature $CT_0$, cooling rate $\zeta$, number of iterations $Iters$, truncation parameter $z$ and standard deviation $\sigma$.
Ensure: optimal tentative allocation $\hat{\mathcal{T}}^*$ and best Bayesian score $bestSC$.
1: Initialize $k \leftarrow 0$, $\hat{\mathcal{T}}^* \leftarrow \hat{\mathcal{T}}$.
2: $MLX \leftarrow$ GetMLX($SuffStatsX, \hat{\mathcal{T}}$)
3: $bestSC \leftarrow$ LearnKTTX($MLX$, $PR[], C[,], \lambda_c$)
4: while ($k < Iters$) do
5:     $\hat{\mathcal{T}} \leftarrow$ TentativeAllocation($\hat{\mathcal{T}}^*$, $z$, $\sigma$)
6:     $MLX \leftarrow$ GetMLX($SuffStatsX, \hat{\mathcal{T}}$)
7:     $tentSC \leftarrow$ LearnKTTX($MLX$, $PR[], C[,], \lambda_c$)
8:     $CT \leftarrow CT_0 \times \zeta^k$
9:     $accProb \leftarrow \min\left\{1, \exp\left(-\frac{(bestSC - tentSC)}{CT}\right)\right\}$
10:    $ur \leftarrow$ UniRand()
11:    if ($ur \leq accProb$) then
12:        $\hat{\mathcal{T}}^* \leftarrow \hat{\mathcal{T}}$
13:        $currSC \leftarrow tentSC$
14:    end if
15:    $k \leftarrow k + 1$
16: end while
17: $bestSC \leftarrow currSC$
```

The simulated annealing parameters we used include the tentative allocation $\hat{\mathcal{T}}$, the initial temperature CT_0, the cooling rate ζ and the number of iterations $Iters$ for the exponential cooling schedule. Moreover, the truncation parameter z and standard deviation σ are used for the selection of the new tentative allocation $\hat{\mathcal{T}}'$ according to the random procedure shown in Algorithm 3. This procedure selects a transition time through a discrete uniform distribution, UniRandDiscr($\hat{\mathcal{T}}$), and perturbs it according to a truncated normal distribution, StdNormRand(), having a standard deviation equal to σ, with the addition of point masses at z and $-z$, where z represents the truncation parameter.

Algorithm 3 TentativeAllocation

```
Require: tentative allocation $\hat{\mathcal{T}}$, truncation parameter $z$ and standard deviation $\sigma$.
Ensure: new tentative allocation $\hat{\mathcal{T}}'$.
 1: $t \leftarrow$ UniRandDiscr($\hat{\mathcal{T}}$)
 2: $\hat{\mathcal{T}}' \leftarrow \hat{\mathcal{T}} \setminus t$
 3: $nr \leftarrow$ StdNormRand()
 4: if $(nr < -z)$ then
 5:     $nr \leftarrow -z$
 6: end if
 7: if $(nr > z)$ then
 8:     $nr \leftarrow z$
 9: end if
10: $t \leftarrow t + nr \times \sigma$
11: $\hat{\mathcal{T}}' \leftarrow \hat{\mathcal{T}} \cup t$
```

4.3 Unknown Number of Epochs

In this setting the number of epochs is unknown; thus the structural learning algorithm must be able to move across a different number of epochs, as well as the corresponding transition times. Also in this case, we used a simulated annealing algorithm where the state $\boldsymbol{x}$ is the tentative allocation $\hat{\mathcal{T}}$ and the function to be optimized $f(\boldsymbol{x})$ is the Bayesian score shown in equation (33). The cooling schedule has been set the same as the one used under the KNE setting. The proposal distribution differs from the one used under the KNE setting as it uses two additional operators, namely the *split* and the *merge* operators. The split operator allows to split a given interval $[\hat{t}_m; \hat{t}_{m+1})$ into two subintervals $[\hat{t}_m; t)$ and $[t; \hat{t}_{m+1})$ where $\hat{t}_m, \hat{t}_{m+1} \in \hat{\mathcal{T}}$. The merge operator allows to merge contiguous intervals $[\hat{t}_{m-1}; \hat{t}_m)$ and $[\hat{t}_m; \hat{t}_{m+1})$ to form the wider interval $[\hat{t}_{m-1}; \hat{t}_{m+1})$ where $\hat{t}_{m-1}, \hat{t}_m, \hat{t}_{m+1} \in \hat{\mathcal{T}}$.

The new state is obtained by sampling the number of epochs changes ec from a multinoulli distribution with parameters (p_1, p_2, p_3), where p_1 represents the probability that the number of epochs of the next iteration $|\hat{\mathcal{T}}|$ is decreased by one; p_3 represents the probability that the number of epochs of the next iteration $|\hat{\mathcal{T}}|$ is increased by one, and p_2 represents the probability that number of epochs of the next iteration $|\hat{\mathcal{T}}|$ does not change with respect to the current one. If ec is equal to 2, then Algorithm 2 is invoked, if ec is equal to 1, then the merge operator is applied before invoking Algorithm 2, while if ec is equal to 3, then the split operator is applied before invoking Algorithm 2.

Algorithm 4 solves the structural learning problem of nsCTBN under the UNE setting for a given node X by ensuring the optimal tentative allocation $\hat{\mathcal{T}}^*$ and its corresponding Bayesian score. This algorithm is similar to the one used under the KNE settings, but it uses Algorithm 5 to apply the split and merge operators. The *left(t)* function in Algorithm 5 returns the transition time in $\hat{\mathcal{T}}$ which comes immediately before transition time t.

Algorithm 4 LearnUNEX

```
Require: sufficient statistics $SuffStatsX$, prior probability $PR[]$, number of parents' changes $C[,]$, parameter $\lambda_c$, parameter $\lambda_e$, tentative allocation $\hat{\mathcal{T}}$, initial temperature $CT_0$, cooling rate $\zeta$, number of iterations $Iters$, truncation parameter $z$, standard deviation $\sigma$, split probability $sp$ and merge probability $mp$.
Ensure: optimal tentative allocation $\hat{\mathcal{T}}^*$ and best Bayesian score $bestSC$.
 1: Initialize $k \leftarrow 0$, $\hat{\mathcal{T}}^* \leftarrow \hat{\mathcal{T}}$.
 2: $bestSC \leftarrow$LearnKTTX(GetMLX($SuffStatsX$, $\hat{\mathcal{T}}$), $PR[], C[,], \lambda_c$) $-\lambda_e\ |\hat{\mathcal{T}}|$
 3: while ($k < Iters$) do
 4:   $\hat{\mathcal{T}} \leftarrow$ SplitMerge($\hat{\mathcal{T}}^*$, $sp$, $mp$)
 5:   $\hat{\mathcal{T}} \leftarrow$ TentativeAllocation($\hat{\mathcal{T}}$, $z$, $\sigma$)
 6:   $tentSC \leftarrow$ LearnKTTX(GetMLX($SuffStatsX$, $\hat{\mathcal{T}}$), $PR[], C[,], \lambda_c$) $-\lambda_e\ |\hat{\mathcal{T}}|$
 7:   $CT \leftarrow CT_0 \times \zeta^k$
 8:   $accProb \leftarrow \min\left\{1, \exp\left(-\frac{(bestSC - tentSC)}{CT}\right)\right\}$
 9:   $ur \leftarrow$ UniRand()
10:   if ($ur \leq accProb$) then
11:     $\hat{\mathcal{T}}^* \leftarrow \hat{\mathcal{T}}$
12:     $currSC \leftarrow tentSC$
13:   end if
14:   $k \leftarrow k + 1$
15: end while
16: $bestSC \leftarrow currSC$
```

Algorithm 5 SplitMerge

```
Require: tentative allocation $\hat{\mathcal{T}}$, split probability $sp$ and merge probability $mp$.
Ensure: new tentative allocation $\hat{\mathcal{T}}'$.
 1: $\hat{\mathcal{T}}' \leftarrow \hat{\mathcal{T}}$
 2: $p \leftarrow$ UniRand()
 3: if ($p < mp$) then
 4:   $t \leftarrow$ UniRandDiscr($\hat{\mathcal{T}}$)
 5:   $\hat{\mathcal{T}}' \leftarrow \hat{\mathcal{T}} \setminus t$
 6: else
 7:   if ($p < (mp + sp)$) then
 8:     $t \leftarrow$ UniRandDiscr($\hat{\mathcal{T}} \cup T$)
 9:     $nt \leftarrow$ left($t$) $+ \frac{t - \text{left}(t)}{2}$
10:     $\hat{\mathcal{T}}' \leftarrow \hat{\mathcal{T}} \cup nt$
11:   end if
12: end if
```

5. Numerical Experiments

Numerical experiments are performed on both synthetic and real-world datasets. Synthetic datasets are used to compare nsCTBNs to nsDBNs under the KTT, KNE and UNE knowledge settings in terms of accuracy, precision, recall and F_1 measure. The following real-world datasets: *drosophila*, *saccharomyces cerevisiae* and *songbird*, are used to compare nsCTBNs to state-of-the-art algorithms, i.e. TSNI (a method based on ordinary differential equations), nsDBN (Robinson & Hartemink, 2010) and non-homogeneous dynamic Bayesian networks with Bayesian regularization (TVDBN) (Dondelinger, Lebre, & Husmeier, 2013), under the UNE knowledge setting. Drosophila, saccharomyces cerevisiae and songbird datasets are collected at fixed time intervals, thus we analyzed an additional real-world dataset, consisting of financial/economic variables evolving at different time granularities, to exploit the expressiveness of nsCTBNs when events occur asynchronously. Note that while the performance comparison using synthetic datasets benefits from the knowledge of the *ground truth*, the same does not apply to the performance comparison using real-world datasets because the ground truth is not available. In such cases, the comparison exploits partial and meta-knowledge available in the specialized literature.

5.1 Synthetic Datasets

Artificially generated datasets include data sampled from a rich set of nsDBN models, i.e. *nsDBN generated datasets*, and a rich set of nsCTBN models, i.e. *nsCTBN generated datasets*. Such nsDBN and nsCTBN models consist of five nodes associated with binary and ternary variables. Numerical experiments concern learning the parents sets, transition times and the number of epochs for a single node. This choice is motivated by the fact that structural learning for nsCTBN can be performed for each single node independently from the remaining ones. However, when transition times are unknown, having multiple parents sets changes could make it easier to correctly identify the times of change.

5.1.1 nsDBN Generated Datasets

nsDBN generated datasets were sampled from nsDBN models[5] associated with the following *number of epochs* $E \in \{2, 3, 4, 5\}$. In particular, for each number of epochs E, 10 different nsDBN instances were sampled to obtain a *number of datasets* equal to 10, each one consisting of a single trajectory. Thus, 40 synthetic datasets were used to learn the structure of nsDBN and nsCTBN (*number of models* =2) under the KTT, KNE and UNE settings.

Structural learning experiments were performed with $\lambda_c = \{1, 2, 4\}$ and $\lambda_e = \{5, 10, 15\}$ for nsCTBN and $\lambda_s = \{1, 2, 4\}$ and with $\lambda_m = \{10, 50, 100\}$ for nsDBN[6]. An overall number of 1,200 experiments have been performed. In particular, we performed *number of epochs* $\cdot$ *number of datasets* $\cdot$ *number of* λ_c *or* λ_s $\cdot$ *number of models* $= 4{\cdot}10{\cdot}3{\cdot}2 = 240$ experiments under the KTT setting, 240 under the KNE setting, while *number of epochs* $\cdot$ *number of datasets* $\cdot$ *number of* λ_c *or* λ_s $\cdot$ *number of* λ_e *or* λ_m $\cdot$ *number of models* $= 4{\cdot}10{\cdot}3{\cdot}3{\cdot}2 = 720$ experiments have been performed under the UNE setting.

5. Inter-slice arcs are allowed, while intra-slice arcs are not allowed. This holds true for all nsDBN models sampled to obtain the nsDBN generated datasets.
6. It is worthwhile to mention that the λ_s and λ_m parameters are the nsDBN counterparts of the λ_c and λ_e parameters for the nsCTBN.

The *nsdbn jar executable*[7] (Robinson & Hartemink, 2010) was used for structural learning of nsDBN, where we set the maximum number of proposed networks to 500,000 and the burn-in period to 50,000 for nsDBN. nsCTBN were learned by using the following parameters setting: $Iters$ = 1,000, CT_0 = 1,000, $\zeta = 0.8$, $z = 3$, $\sigma = 1$, $sp = 0.3$, $mp = 0.3$, $\alpha = 1$ and $\tau = 0.1$ using the BDeu metric. Furthermore, for nsDBN and nsCTBN we set the maximum number of parents to 4. Only arcs that occurred in more than 90 percent of the samples[8] belong to the inferred nsDBN and nsCTBN models. Accuracy (Acc), precision (Prc), recall (Rec) and F_1 measure (F_1) achieved by nsDBN and nsCTBN learned under the KTT, KNE and UNE settings are reported in Table 1, 2 and 3 respectively. It is worthwhile to mention that under the KNE and UNE settings, nsDBNs and nsCTBNs almost always identified the correct number of epochs and the location of their associated transition times. Accuracy, precision, recall and F_1 measure have been computed in two different ways. Firstly, we included all arcs of the true network for each epoch. Secondly, we excluded the *self-reference arcs*, i.e. those arcs connecting the same node in two consecutive time-slices of the true network for each epoch. In fact, while each node of a nsCTBN has the self-reference arc by default, the same does not happen for nsDBNs. This means that in the first case a nsDBN is required to learn arcs that a nsCTBN is not required to do. Therefore, to ensure a fair comparison of nsCTBN to nsDBN we adopted the second case. Tables 1, 2 and 3 report the performance measure values computed by excluding self-reference arcs from the set of arcs of the true networks for each epoch.

Table 1: nsCTBN compared to nsDBN under the KTT setting for nsDBN generated data. Average, min (subscript) and max (superscript) performance values over 10 networks and λ_c for nsCTBN and λ_s for nsDBN.

	Number of epochs E							
	2		3		4		5	
	nsDBN	nsCTBN	nsDBN	nsCTBN	nsDBN	nsCTBN	nsDBN	nsCTBN
Acc	$0.96^{1.00}_{0.93}$	$0.92^{1.00}_{0.75}$	$0.95^{0.98}_{0.93}$	$0.92^{1.00}_{0.75}$	$0.95^{0.99}_{0.89}$	$0.82^{1.00}_{0.63}$	$0.94^{0.97}_{0.89}$	$0.82^{0.95}_{0.55}$
$Prec$	$0.90^{1.00}_{0.67}$	$1.00^{1.00}_{1.00}$	$0.79^{1.00}_{0.50}$	$1.00^{1.00}_{1.00}$	$0.87^{1.00}_{0.40}$	$0.96^{1.00}_{0.75}$	$0.85^{1.00}_{0.50}$	$0.99^{1.00}_{0.80}$
Rec	$0.77^{1.00}_{0.40}$	$0.85^{1.00}_{0.50}$	$0.67^{0.83}_{0.43}$	$0.86^{1.00}_{0.63}$	$0.65^{0.90}_{0.25}$	$0.70^{1.00}_{0.38}$	$0.58^{0.80}_{0.25}$	$0.71^{0.91}_{0.33}$
F_1	$0.80^{1.00}_{0.57}$	$0.91^{1.00}_{0.67}$	$0.71^{0.91}_{0.47}$	$0.92^{1.00}_{0.77}$	$0.73^{0.95}_{0.31}$	$0.80^{1.00}_{0.50}$	$0.69^{0.86}_{0.35}$	$0.82^{0.95}_{0.47}$

According to Tables 1, 2 and 3, nsDBNs consistently achieve greater accuracy values than those achieved by nsCTBNs under the three settings. Furthermore, for nsDBNs the accuracy is stable with respect to the number of epochs E while the same does not happen for nsCTBNs. Indeed, when the number of epochs E is greater than 3, nsCTBNs achieve accuracy values which are significantly smaller than those achieved when the number of epochs E is equal to 2 or 3. The same does not happen to nsDBNs where the accuracy is robust with respect to the number of epochs E.

7. We acknowledge the precious help of Alex Hartemink who let us use the nsdbn jar executable program for learning nsDBN models. Furthermore, he also provided the drosophila and songbird datasets.
8. Samples are obtained under the same parameters values.

Table 2: nsCTBN compared to nsDBN under the KNE setting for nsDBN generated data. Average, min (subscript) and max (superscript) performance values over 10 networks and λ_c for nsCTBN and λ_s for nsDBN.

	Number of epochs E							
	2		3		4		5	
	nsDBN	nsCTBN	nsDBN	nsCTBN	nsDBN	nsCTBN	nsDBN	nsCTBN
Acc	$0.94_{0.88}^{1.00}$	$0.92_{0.75}^{1.00}$	$0.95_{0.92}^{0.98}$	$0.92_{0.75}^{1.00}$	$0.94_{0.89}^{0.99}$	$0.81_{0.63}^{1.00}$	$0.93_{0.87}^{0.96}$	$0.82_{0.55}^{0.95}$
Prec	$0.91_{0.69}^{1.00}$	$1.00_{1.00}^{1.00}$	$0.79_{0.50}^{1.00}$	$1.00_{0.95}^{1.00}$	$0.86_{0.55}^{0.97}$	$0.95_{0.75}^{1.00}$	$0.85_{0.55}^{0.96}$	$0.98_{0.80}^{1.00}$
Rec	$0.76_{0.39}^{1.00}$	$0.85_{0.50}^{1.00}$	$0.68_{0.35}^{0.84}$	$0.86_{0.63}^{1.00}$	$0.65_{0.25}^{0.91}$	$0.71_{0.38}^{1.00}$	$0.59_{0.24}^{0.78}$	$0.70_{0.33}^{0.91}$
F_1	$0.80_{0.58}^{1.00}$	$0.91_{0.67}^{1.00}$	$0.72_{0.51}^{0.91}$	$0.92_{0.77}^{1.00}$	$0.74_{0.38}^{0.95}$	$0.81_{0.50}^{1.00}$	$0.70_{0.35}^{0.74}$	$0.81_{0.47}^{0.95}$

Table 3: nsCTBN compared to nsDBN under the UNE setting for nsDBN generated data. Average, min (subscript) and max (superscript) performance values over 10 networks and λ_c, λ_e for nsCTBN and λ_s, λ_m for nsDBN.

	Number of epochs E							
	2		3		4		5	
	nsDBN	nsCTBN	nsDBN	nsCTBN	nsDBN	nsCTBN	nsDBN	nsCTBN
Acc	$0.95_{0.88}^{1.00}$	$0.92_{0.75}^{1.00}$	$0.95_{0.93}^{0.98}$	$0.92_{0.75}^{1.00}$	$0.93_{0.89}^{0.98}$	$0.81_{0.61}^{0.99}$	$0.92_{0.89}^{0.96}$	$0.81_{0.55}^{0.93}$
Prec	$0.91_{0.70}^{0.98}$	$1.00_{1.00}^{1.00}$	$0.80_{0.52}^{1.00}$	$1.00_{1.00}^{1.00}$	$0.87_{0.67}^{0.95}$	$0.95_{0.73}^{1.00}$	$0.84_{0.71}^{0.90}$	$0.98_{0.80}^{1.00}$
Rec	$0.75_{0.38}^{0.98}$	$0.85_{0.50}^{1.00}$	$0.66_{0.41}^{0.81}$	$0.86_{0.63}^{1.00}$	$0.65_{0.26}^{0.87}$	$0.70_{0.36}^{0.99}$	$0.57_{0.23}^{0.78}$	$0.69_{0.33}^{0.87}$
F_1	$0.79_{0.55}^{0.96}$	$0.91_{0.67}^{1.00}$	$0.70_{0.48}^{0.89}$	$0.92_{0.77}^{1.00}$	$0.74_{0.41}^{0.87}$	$0.80_{0.48}^{0.99}$	$0.68_{0.34}^{0.85}$	$0.81_{0.47}^{0.93}$

A different picture emerges when focusing on the task to discover *positive arcs*. Indeed, in such a case nsCTBNs achieve values of precision, recall and F_1 measure, which are always greater than those achieved by nsDBNs. nsCTBNs achieve precision values which are robust with respect to the knowledge settings and the number of epochs E. The same does not hold true for the recall performance measure. Indeed, nsCTBNs achieve a robust recall with respect to the knowledge settings (KTT, KNE and UNE), while the recall achieved by nsCTBNs significantly degrades when moving from 2 to 3 epochs under all knowledge settings. The same happens to the F_1 measure achieved by nsCTBNs. The results of numerical experiments suggest that nsCTBNs are more effective than nsDBNs to discover *positive arcs*, even if the datasets have been generated using nsDBNs. A possible explanation for this behavior is that learning nsDBNs is more difficult than learning nsCTBNs. In particular, nsDBNs must learn self-reference arcs while nsCTBNs do not. Furthermore, for each node, nsCTBNs *learn locally* the sequence of parents sets while the same does not happen for nsDBNs. In fact, nsDBNs *learn globally* the sequence of parents sets for all nodes, i.e. they globally learn the sequence of networks, and thus they solve a learning problem which is more difficult than the one solved by nsCTBNs.

5.1.2 nsCTBN Generated Datasets

We generated 40 synthetic datasets with $E \in \{2, 3, 4, 5\}$, these datasets are then used to learn the structure of nsCTBN under the three knowledge settings. The same parameters setting is used as the one used for nsCTBN learning from nsDBN generated datasets (for nsCTBN, we used $\alpha = 1$, $\tau = 0.1$ and the BDeu metric), while, in this case, we did not perform structural learning experiments for nsDBN models[9]. The graphical structures of the nsCTBN models sampled to obtain the datasets are the same as those sampled to obtain the nsDBN datasets. The goal of these experiments is to analyze the performance of nsCTBN structural learning algorithms under the three knowledge settings.

The analysis of data reported in Tables 4, 5 and 6 brings us to conclude that the nsCTBN structural learning algorithms work very well under the three settings according to the considered performance measures. Accuracy, recall and F_1 measure decrease slightly when the number of epochs increases from 2 to 5. In particular, the recall measure suffers the greatest decrease from 1 to 0.95 when the number of epochs increases from 2 to 5. Accuracy and F_1 measure are very robust with respect to the number of epochs, while precision is the most robust performance measure with respect to different datasets and different values of the number of epochs under all knowledge settings.

Table 4: nsCTBN under the KTT setting for nsCTBN generated data. Average, min (subscript) and max (superscript) performance values over 10 networks and λ_c.

	Number of epochs E			
	2	3	4	5
Acc	$1.00^{1.00}_{1.00}$	$0.99^{1.00}_{0.92}$	$0.99^{1.00}_{0.94}$	$0.98^{1.00}_{0.90}$
$Prec$	$1.00^{1.00}_{1.00}$	$1.00^{1.00}_{1.00}$	$1.00^{1.00}_{1.00}$	$1.00^{1.00}_{1.00}$
Rec	$1.00^{1.00}_{1.00}$	$0.99^{1.00}_{0.86}$	$0.99^{1.00}_{0.89}$	$0.96^{1.00}_{0.86}$
F_1	$1.00^{1.00}_{1.00}$	$0.99^{1.00}_{0.92}$	$0.99^{1.00}_{0.94}$	$0.98^{1.00}_{0.92}$

Table 5: nsCTBN under the KNE setting for nsCTBN generated data. Average, min (subscript) and max (superscript) performance values over 10 networks and λ_c.

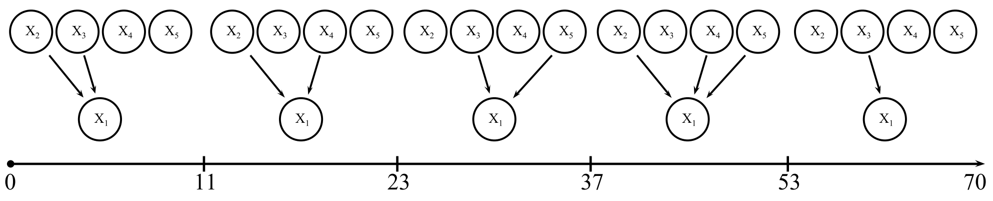
	Number of epochs E			
	2	3	4	5
Acc	$1.00^{1.00}_{1.00}$	$0.98^{1.00}_{0.92}$	$0.99^{1.00}_{0.94}$	$0.97^{1.00}_{0.90}$
$Prec$	$1.00^{1.00}_{1.00}$	$0.99^{1.00}_{0.90}$	$1.00^{1.00}_{0.99}$	$1.00^{1.00}_{0.97}$
Rec	$1.00^{1.00}_{1.00}$	$0.97^{1.00}_{0.86}$	$0.98^{1.00}_{0.89}$	$0.96^{1.00}_{0.85}$
F_1	$1.00^{1.00}_{1.00}$	$0.98^{1.00}_{0.88}$	$0.99^{1.00}_{0.94}$	$0.98^{1.00}_{0.90}$

9. nsCTBN generated data are asynchronous involving different time granularities, thus nsDBN cannot be directly applied. An option is to preprocess these datasets to adapt them to nsDBNs. Given that this would be strongly arbitrary and be penalizing for nsDBNs, we decided to learn only the nsCTBN models.

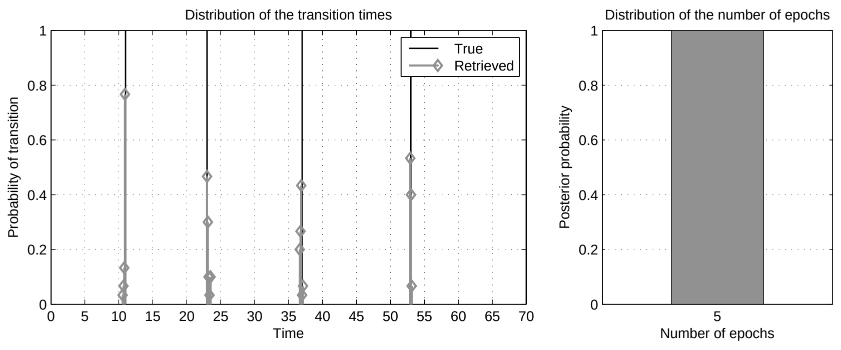
Table 6: nsCTBN under the UNE setting for nsCTBN generated data. Average, min (subscript) and max (superscript) performance values over 10 networks and λ_c and λ_e.

	Number of epochs E			
	2	3	4	5
Acc	$1.00^{1.00}_{1.00}$	$0.99^{1.00}_{0.92}$	$0.99^{1.00}_{0.94}$	$0.97^{1.00}_{0.90}$
$Prec$	$1.00^{1.00}_{1.00}$	$1.00^{1.00}_{1.00}$	$1.00^{1.00}_{1.00}$	$1.00^{1.00}_{0.98}$
Rec	$1.00^{1.00}_{1.00}$	$0.99^{1.00}_{0.86}$	$0.98^{1.00}_{0.89}$	$0.95^{1.00}_{0.81}$
F_1	$1.00^{1.00}_{1.00}$	$0.99^{1.00}_{0.92}$	$0.99^{1.00}_{0.94}$	$0.97^{1.00}_{0.89}$

The best and worst values of accuracy for $E = 5$ reported in Table 6 belong to the experiments performed on synthetic dataset number 3 and number 9 respectively. Their results are illustrated hereafter. Figure 2(a) shows the graphs sequence of the true nsCTBN for the synthetic datasets number 3, while Figure 2(b) displays the posterior distribution over epochs (right), together with the distribution of the corresponding transition times (left)[10] of the learned nsCTBN in the UNE case. Figure 3 shows the same information as those depicted in Figure 2, but for the synthetic dataset number 9. In the latter case, the distribution over epochs is slightly in favor of the correct number of epochs.



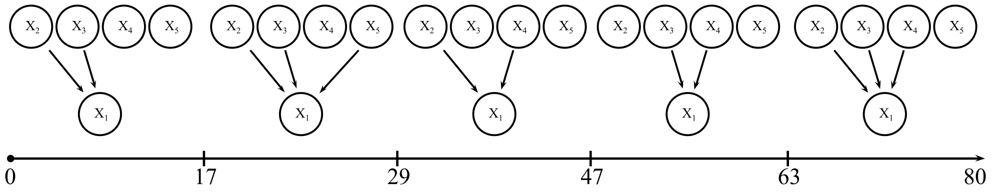
(a) True nsCTBN model.



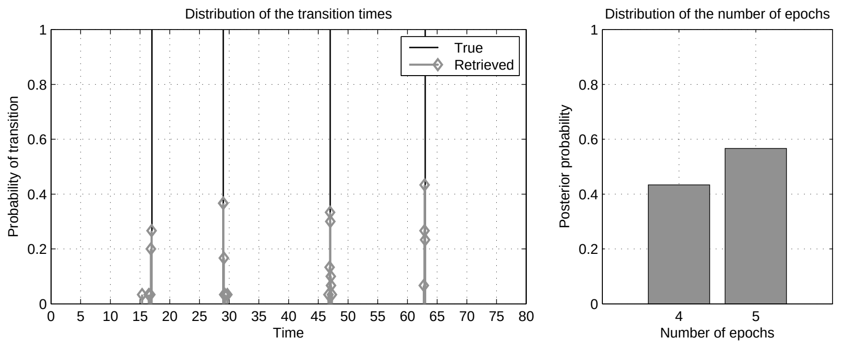
(b) Learned nsCTBN model results.

Figure 2: nsCTBN generated dataset number 3: (a) true graphs sequence over E=5 epochs and (b) distribution of the transition times (left) and posterior over epochs (right) associated with the nsCTBN inferred under the UNE setting.

10. Transition times whose distance is less than 0.1 have been aggregated.



(a) True nsCTBN model.



(b) Learned nsCTBN model results.

Figure 3: nsCTBN generated dataset number 9: (a) true graphs sequence over E=5 epochs and (b) distribution of the transition times (left) and posterior over epochs (right) associated with the nsCTBN inferred under the UNE setting.

5.2 Real-world Datasets

It is very difficult to find real-world datasets where the corresponding ground truth model is completely known and/or a uniform consensus from domain experts has been reached. Therefore, we decided to use the following three well-known datasets: drosophila, saccharomyces cerevisiae and songbird to compare the performance of nsCTBNs to that of nsDBNs and other state-of-the-art algorithms, i.e. TSNI and TVDBN. Such datasets are publicly available, clearly described and a rich and detailed discussion about their likely ground truth models is given in the specialized literature. Furthermore, a macroeconomics dataset is introduced and analyzed. This dataset consists of 17 financial/economic variables collected at different time granularity spanning from 1st January 1986 to 31st March 2015.

5.2.1 Drosophila

The drosophila dataset includes the mRNA expression levels of 4,028 genes at 67 successive time-points spanning the four stages of the *Drosophila melanogaster* life cycle (Lebre et al., 2010): the *embryonic* (31 time-points), *larval* (10 time-points) and *pupal* stage (18 time-points) and the first 30 days of *adulthood* (8 time-points). For comparative purposes (Dondelinger et al., 2013), we have analyzed the *reduced drosophila* dataset consisting of gene expression time-series of 11 genes involved in wing muscle development. Given that nsCTBNs are based on discrete variables, we binarized the expression level of the 11 genes for the reduced drosophila dataset as done in the literature (Zhao, Serpedin, & Dougherty, 2006; Guo, Hanneke, Fu, & Xing, 2007; Robinson & Hartemink, 2010).

Firstly, the network inference task of the embryonic, larval, pupal and adulthood morphogenic stages was performed under the KTT setting (Robinson & Hartemink, 2010; Dondelinger et al., 2013). The nsCTBN structural learning was performed using the following parameter values $\lambda_c = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ and by setting the maximum number of parents to 4. The nsCTBN learned with different λ_c values are combined, and only arcs that occurred in more than 20 percent of samples are included into the inferred non-stationary continuous time Bayesian network. No other techniques predict non-stationary directed networks (Robinson & Hartemink, 2010), so precision, recall and F_1 measure, computed with respect to networks inferred by Zhao et al. (2006) and Guo et al. (2007), are reported in Table 7 for nsDBN, nsCTBN and TVDBN (Dondelinger et al., 2013). The networks associated with the four epochs, as inferred with the nsCTBN on the reduced drosophila dataset under the KTT setting, are depicted in Figure 4.

Table 7: Precision (*Prec*), recall (*Rec*) and F_1 measure (F_1) achieved by nsCTBN, nsDBN, and TVDBN on the drosophila dataset are computed with respect to networks inferred by Zhao et al. (2006) and Guo et al. (2007). Average values (Average) of precision, recall and F_1 measure achieved by Zhao et al. (2006) and Guo et al. (2007) are also reported.

	Zhao et al. (2006)			Guo et al. (2007)			Average		
	Prec	*Rec*	F_1	*Prec*	*Rec*	F_1	*Prec*	*Rec*	F_1
nsDBN	0.58	0.38	0.46	0.47	0.34	0.39	0.52	0.36	0.42
nsCTBN	0.33	0.37	0.35	0.41	0.43	0.42	0.37	0.40	0.39
TVDBN	0.17	0.27	0.21	0.36	0.61	0.45	0.27	0.44	0.33

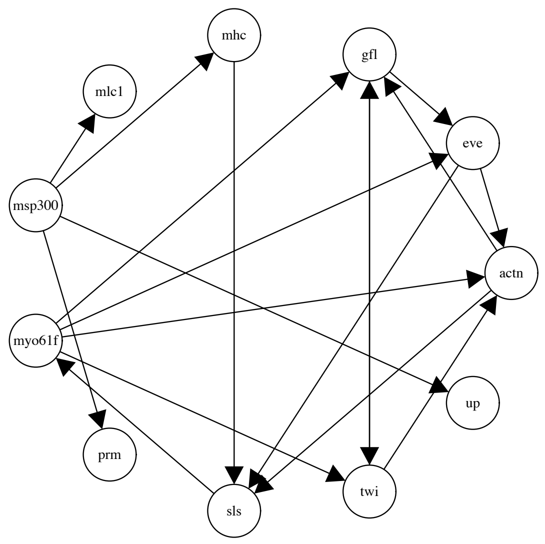
According to Table 7, no optimal algorithm exists for the reduced drosophila dataset. If the network retrieved by Zhao et al. (2006) is used as ground truth, then nsDBN is the best model, while if the network retrieved by Guo et al. (2007) network is used as ground truth, then TVDBN is the optimal one as far as the F_1 measure is concerned. If the average performance is computed, then nsDBN is the best model and TVDBN is the worst; while nsCTBN achieves an F_1 value that is close to the one achieved by nsDBN.

Secondly, we investigated whether the transition times inferred by structural learning of nsCTBN under the UNE setting correspond to the known transitions between stages (Lebre et al., 2010; Dondelinger et al., 2013). The network inference task was performed by learning nsCTBN under the UNE setting with the following parameter values $\lambda_c = \{0.2, 0.4, 1, 2\}$ and $\lambda_e = \{0.5, 1, 2, 5\}$. Furthermore, we set the maximum number of parents to 2, the number of iterations to 1,000 and the number of runs to 100.

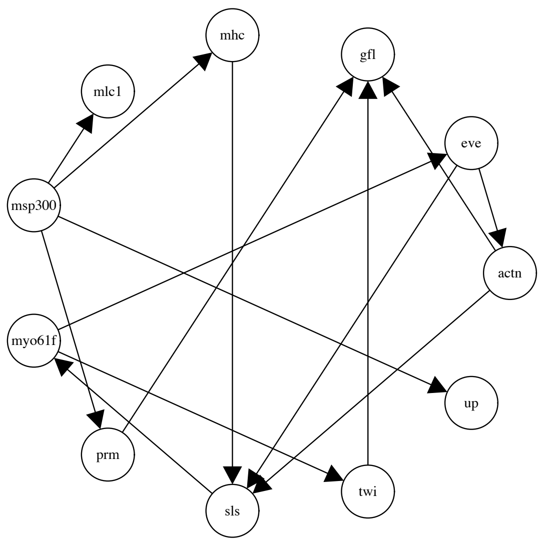
Figure 5 shows the distribution of the transition times[11] (left) and the posterior over the number of epochs (right). The number of epochs is correctly detected to be 4 even if a probability close to 0.1 is associated with 5 epochs. However, the transition times are not all correctly identified. The embryonic stage is not correctly identified, the larval stage is correctly discovered to start at time-point 31, while it is inferred to end at time-point 38

11. Each stem represents the posterior probability that the corresponding time-point starts a new epoch. Therefore, a stem at time-point t means that an epoch ends at time-point $t-1$, while the next epoch starts at time-point t.

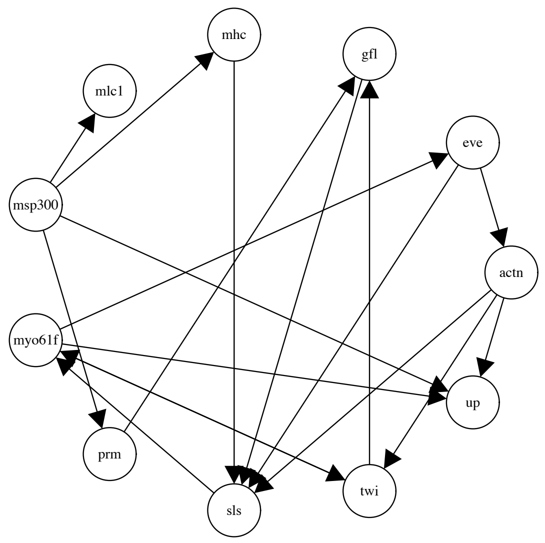
instead of 40. nsCTBN did not identify the pupal and the adulthood stages, but it identified two additional transition times (17 and 51). The same behavior is observed for nsDBNs, while TVDBNs are capable to correctly identify the pupal and the adulthood stages. However, the TVDBN-0, TVDBN-Exp and TVDBN-Bino inferred networks (Dondelinger et al., 2013) consist of a number of epochs ranging from 6 to 7.



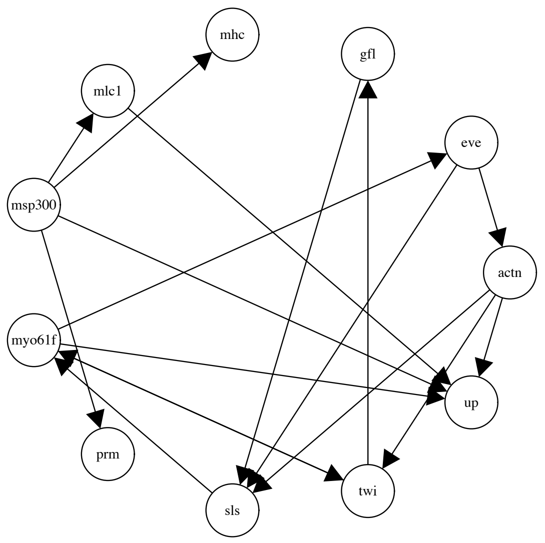
(a) Embryonic (epoch from 0 to 30).



(b) Larval (epoch from 31 to 41).



(c) Pupal (epoch from 42 to 59).



(d) Adulthood (epoch from 60 to 66).

Figure 4: Networks inferred with nsCTBN under the KKT setting on the reduced drosophila dataset. Only arcs that occurred in more than 20 percent of the networks associated with different λ_c values are included in the inferred nsCTBN model.

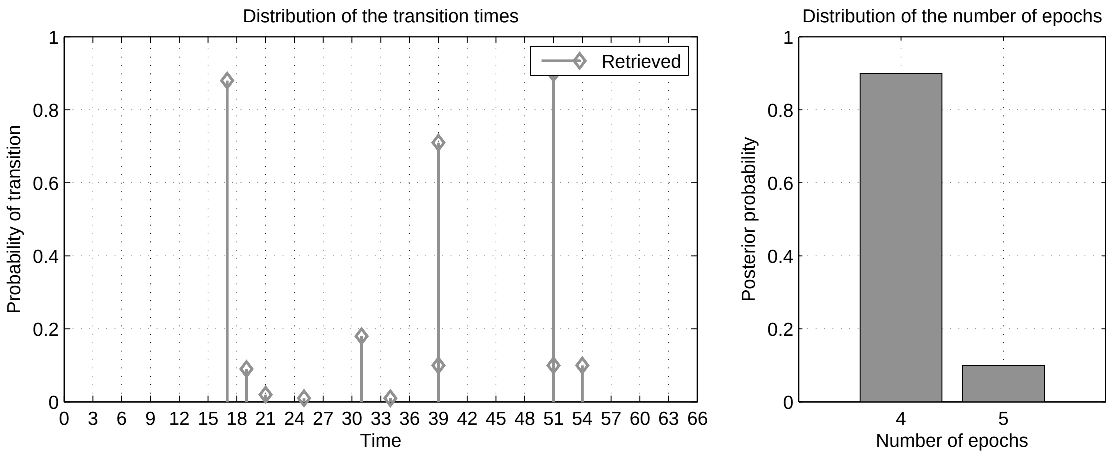


Figure 5: Transition time graph (left) and posterior probability histogram over the number of epochs E (right) associated with the nsCTBN model learned from the drosophila reduced dataset under the UNE setting when $\lambda_c = 0.2$ and $\lambda_e = 2$.

5.2.2 Saccharomyces Cerevisiae

The saccharomyces cerevisiae dataset is obtained from a synthetic regulatory network with 5 genes in saccharomyces cerevisiae (Cantone, Marucci, Iorio, Ricci, Belcastro, Bansal, Santini, di Bernardo, di Bernardo, & Cosma, 2009). It is obtained by measuring gene expression time-series with RT-PCR (reverse transcription polymerase chain reaction) for 16 and 21 time-points under two conditions related to the carbon source: galactose (*switch on* experimental condition) and glucose (*switch off* experimental condition). We merged the time-series from the two experimental conditions under exclusion of the boundary point as done in the literature (Dondelinger et al., 2013). The obtained time-series was binarized in such a way that a 1 indicates that the gene expression level is greater than or equal to its sample mean, while a 0 indicates the gene expression level is smaller than its sample mean. The obtained dataset was used to infer the saccharomyces cerevisiae networks associated with the switch on and switch off experimental conditions.

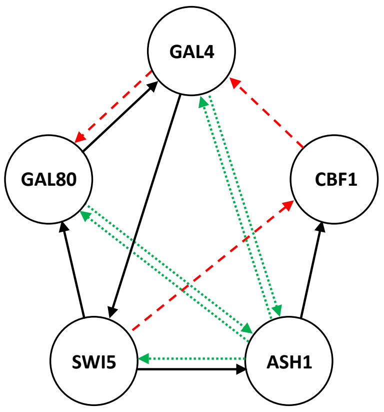
The network inference task was performed by learning a nsCTBN under the UNE setting with the following parameter values $\lambda_c = \{0.2, 0.4, 1, 2\}$ and $\lambda_e = \{0.2, 0.4, 1, 2\}$. Furthermore, we set the maximum number of parents to 4, the number of iterations to 1,000 and the number of runs to 100. Only arcs that occurred in more than 50 percent of the runs are included in the inferred nsCTBN model. Precision, recall and F_1 measure values achieved by nsCTBN are compared to those achieved by the state-of-the-art algorithms (i.e. TSNI, nsDBN and TVDBN) in Table 8.

The result of the performed numerical experiment shows that nsCTBN is competitive with respect to state-of-the-art algorithms, while it achieves non-optimal results only for precision associated with the switch on experimental condition. Under this condition, nsCTBN achieves a precision equal to 0.5 ($\frac{5}{10}$), while the optimal value achieved by TSNI and TVDBN is 0.8 ($\frac{4}{5}$). On the contrary, nsCTBN achieves the best recall value, which is equal to 0.63 ($\frac{5}{8}$). Under the switch off experimental condition, nsCTBN achieves the best value for both precision, which is equal to 0.67 ($\frac{6}{9}$), and recall, which is equal to 0.75 ($\frac{6}{8}$).

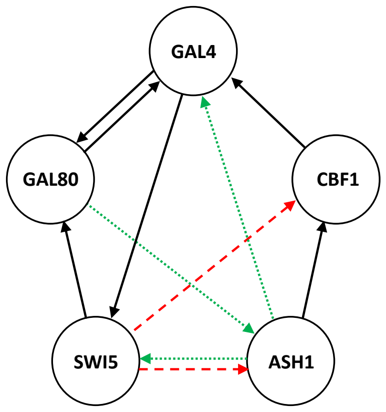
We also computed the overall performance of the structural learning algorithms. In this case, focusing the attention on the F_1 measure, we can conclude that nsCTBN (0.63) is comparable to TVDBN (0.60), which is considered to be the state-of-the-art algorithm for the structural learning task applied to the saccharomyces cerevisiae dataset. The networks inferred by the nsCTBN model under the switch on and switch off experimental conditions, using $\lambda_c = 0.2$ and $\lambda_c = 2$, are depicted in Figure 6.

Table 8: nsCTBN compared to TSNI, nsDBN, and TVDBN when learning from the saccharomyces cerevisiae dataset. nsCTBN is learned under the UNE setting ($\lambda_c = 0.2$, $\lambda_e = 2$); time-point 17 is used as the transition time between the switch on and the switch off experimental conditions. TSNI, nsDBN and TVDBN networks are described in the specialized literature. Precision, recall and F_1 measure are reported for the switch on and switch off experimental conditions. The number of true positive arcs (superscript) and the sum of true and false positive arcs (subscript) are reported for precision, while the number of true positive arcs (superscript) and the sum of true positive and false negative arcs (subscript) are reported for recall. Performance values achieved by aggregating the inferred networks over the two epochs are also reported.

	Switch on			Switch off			Aggregated		
	Prec	*Rec*	F_1	*Prec*	*Rec*	F_1	*Prec*	*Rec*	F_1
TSNI	0.80^{4}_{5}	0.50^{4}_{8}	0.62	0.60^{3}_{5}	0.38^{3}_{8}	0.46	0.70^{7}_{10}	0.44^{7}_{16}	0.54
nsDBN	0.33^{2}_{6}	0.25^{2}_{8}	0.29	0.60^{3}_{5}	0.38^{3}_{8}	0.46	0.45^{5}_{11}	0.31^{5}_{16}	0.37
TVDBN	0.80^{4}_{5}	0.50^{4}_{8}	0.62	0.56^{5}_{9}	0.63^{5}_{8}	0.59	0.64^{9}_{14}	0.56^{9}_{16}	0.60
nsCTBN	0.50^{5}_{10}	0.63^{5}_{8}	0.56	0.67^{6}_{9}	0.75^{6}_{8}	0.71	0.58^{11}_{19}	0.69^{11}_{16}	0.63



(a) Switch on network.



(b) Switch off network.

Figure 6: Switch on (a) and switch off (b) networks inferred with nsCTBN from the saccharomyces cerevisiae dataset under the UNE setting when $\lambda_c = 0.2$, $\lambda_e = 2$. The two pictures report the positive arcs (black continuous), the false negative arcs (red dashed) and the false positive arcs (green dotted) of the inferred networks.

Figure 7 shows the posterior distribution of the number of epochs (left) together with the distribution of the transition times (right) for the nsCTBN learned with $\lambda_c = 0.2$ and $\lambda_e = 2$. The transition between the switch on and switch off experimental conditions is known to occur at time-point 17 (i.e. the switch off epoch starts at time-point 18). It is worthwhile to notice that the small number of arcs, associated with the synthetic regulatory network of saccharomyces cerevisiae, suggests that one should be very careful when evaluating the result of the performed numerical experiment. In particular, we think that overstatements on the effectiveness and/or superiority of different structural learning algorithms for the learning task on the saccharomyces cerevisiae dataset should be avoided.

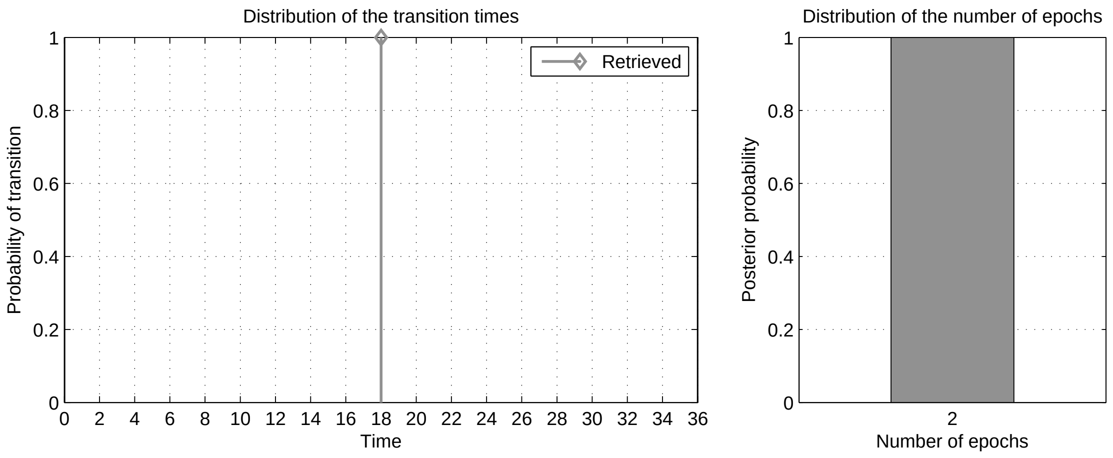


Figure 7: Transition time graph (left) and posterior probability over the number of epochs (right) associated with the nsCTBN inferred from the saccharomyces cerevisiae dataset under the UNE setting when $\lambda_c = 0.2$ and $\lambda_e = 2$. The maximum aposteriori estimate over the number of epochs is associated with $E = 2$ epochs: epoch 1 starts at time-point 1 and ends at time-point 17, while epoch 2 starts at time-point 18 and ends at time-point 36.

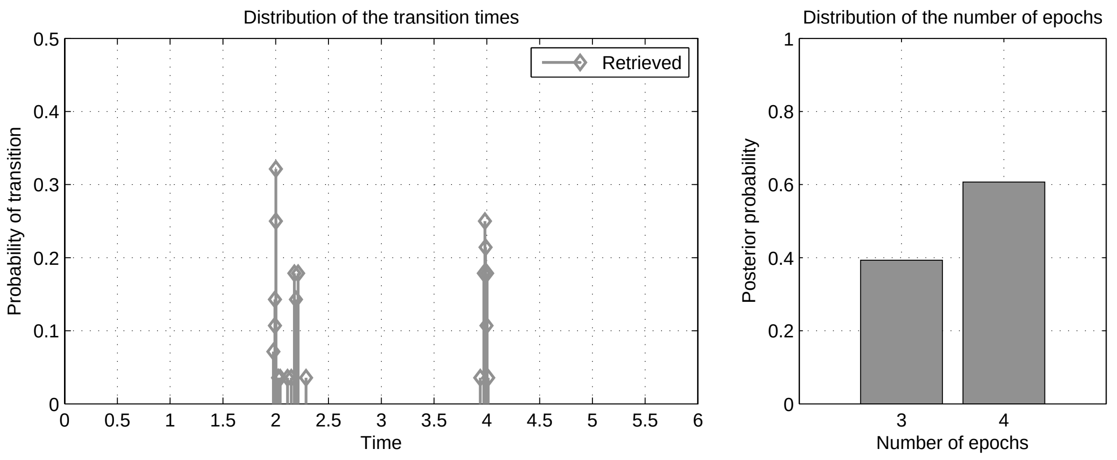
5.2.3 Songbird

The songbird dataset was collected with eight electrodes placed into the *vocal nuclei* of six female zebra finches (Smith et al., 2006). Voltage changes were recorded from populations of neurons while the birds were provided with four different two-second auditory stimuli, each presented from 18 to 20 times. Voltages were post-processed with a root mean square transformation and binned to 5 ms (Robinson & Hartemink, 2010).

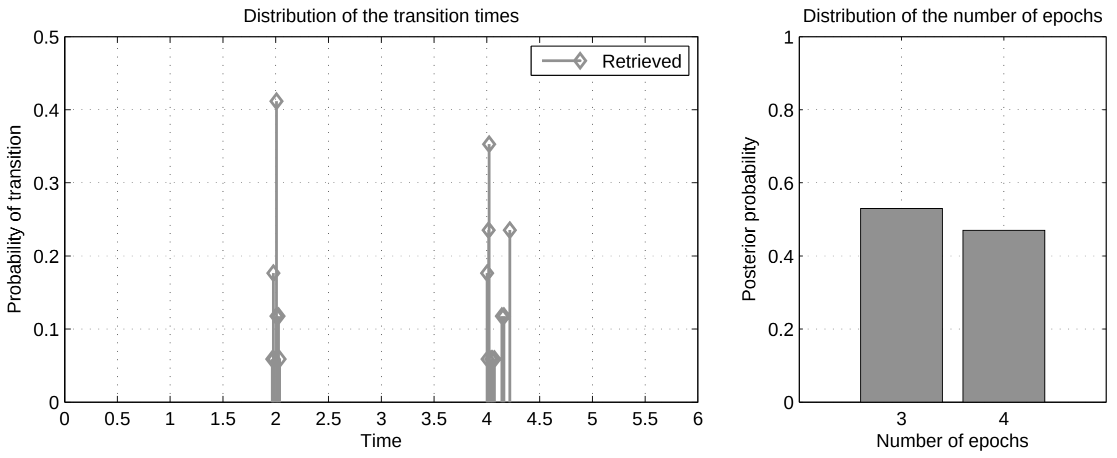
The songbird dataset is used to learn neural information flow networks, i.e. the networks that represent the transmission of information between different regions of the songbird brain. A neural information flow network represents the dynamic utilization of the potential pathways along which information can travel. The identification of the neural information flow networks in songbirds during auditory stimuli allows you to understand how sounds are stored and processed in the songbird's brain. The songbird dataset consists of data of 8 variables recorded from electrodes for two seconds pre-stimulus, two seconds during stimulus and two seconds post-stimulus for six birds. The stimuli are *hear-song*, i.e. the bird hears another bird singing, and *white-noise*, i.e. the bird hears a white noise stimulus.

We show the results of the nsCTBN learned on two out of the six birds of the songbird dataset, namely *bird 648* and *bird 841*. The results obtained for the other four birds are similar. Given that nsCTBNs are based on discrete variables, the values of the 8 variables were discretized into three bins using uniform quantiles $(0, \frac{1}{3}, \frac{2}{3}, 1)$ according to the literature (Robinson & Hartemink, 2010). The inference task of the neural information flow networks was performed by learning a nsCTBN under the UNE setting with the following parameter values $\lambda_c = \{0.25, 0.5, 1, 2, 5, 10\}$ and $\lambda_e = \{0.25, 0.5, 1, 2, 5, 10\}$. We set the maximum number of parents to 3, the number of iterations to 500 and the number of runs to 10.

Figure 8 (a) and (b) show the probability of transition (left) and the posterior probability over the number of epochs (right) for bird 648 and bird 841 under the white-noise stimulus. Figure 9 (a) and (b) show the probability of transition (left) and the posterior probability over the number of epochs (right) for bird 648 and bird 841 under the hear-song stimulus.



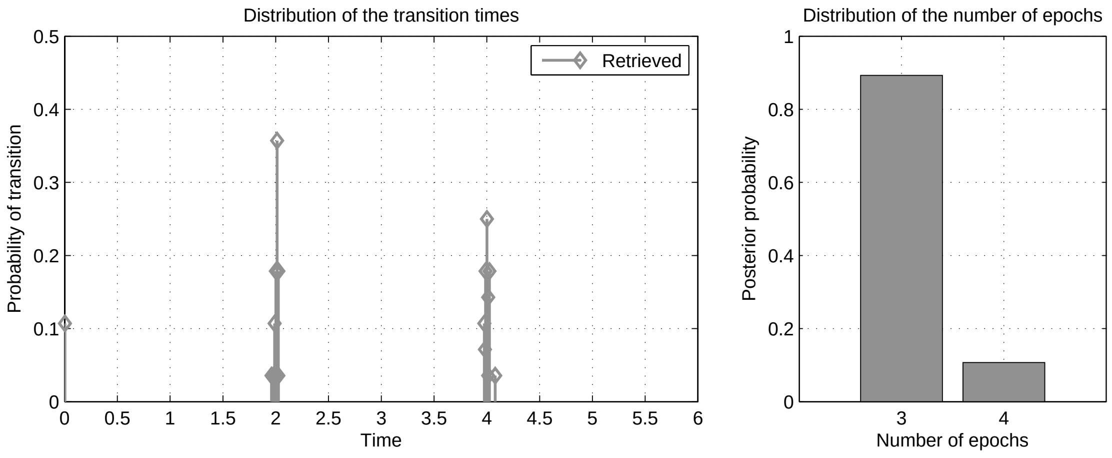
(a) white-noise stimulus for bird 648: learned model results.



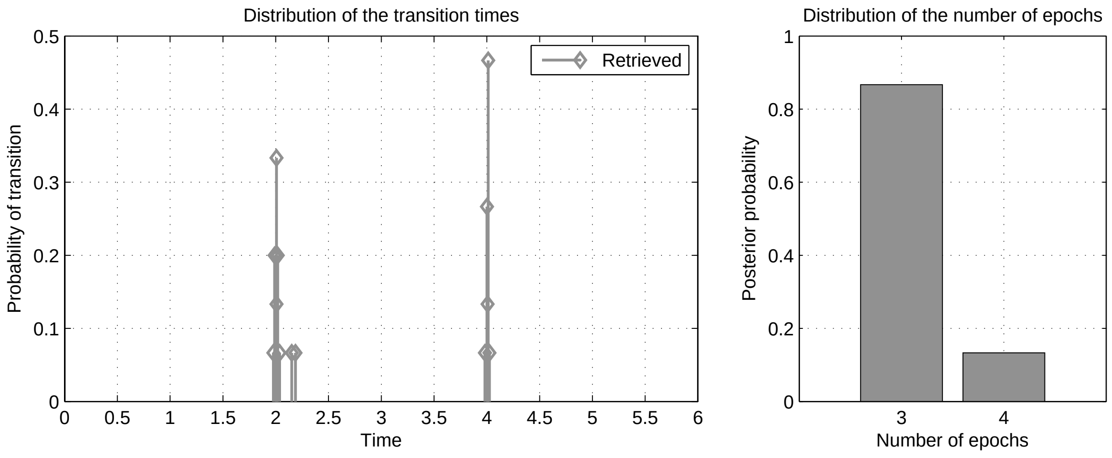
(b) white-noise stimulus for bird 841: learned model results.

Figure 8: Distribution of the transition times and posterior distribution over epochs for nsCTBN under the UNE setting on the songbird dataset for the white-noise stimulus.

The location of the transition time-points under the white-noise stimulus and the hear-song stimulus are accurately inferred for bird 648 and bird 841. The posterior distribution over the number of epochs for birds 648 and 841 under the white-noise stimulus is nearly equally split between 3 and 4 epochs, while under the hear-song stimulus it is peaked over 3 epochs. Therefore, both the number of epochs and the location of the transition time-points are reliably recovered by the nsCTBN learned under the UNE setting. Unfortunately, we were not able to find any additional information to validate the learned nsCTBNs for this dataset. Moreover, a comparison across different birds to eventually develop a consensus network is not possible due to the songbird data collection settings. Indeed, each of the six birds is characterized by its own electrodes, which make difficult to obtain a correspondence map across different birds.



(a) hear-song stimulus for bird 648: learned model results.



(b) hear-song stimulus for bird 841: learned model results.

Figure 9: Distribution of the transition times and posterior distribution over epochs for nsCTBN under the UNE setting on the songbird dataset for the hear-song stimulus.

5.2.4 Macroeconomics

The macroeconomics dataset consists of 17 financial/economic time-series pertaining to the economy of the United States. Time-series have different time granularity and span from 1st January 1986 to 31st March 2015. More specifically, five time-series have daily granularity, namely Crude oil (OIL), USD to EUR spot exchange rate (USDEUR), Gold (GOLD), S&P500 equity index (S&P500) and the 10-years treasury bond yield rate (US10yrsNote). Eleven time-series have monthly granularity, namely production of total industry (PTI), real manufacturing and trade industries sales (RMTIS), personal income (PI), unemployment (UN), consumer price index (CPI), federal funds rate (RATE), producer price index (PPI), non-farm payrolls (NFP), new one-family houses sold (NHSold), new houses for sale (NHSale) and new private house permits (NHPermit). Finally, the gross domestic product (GDP) time-series has quarterly granularity.

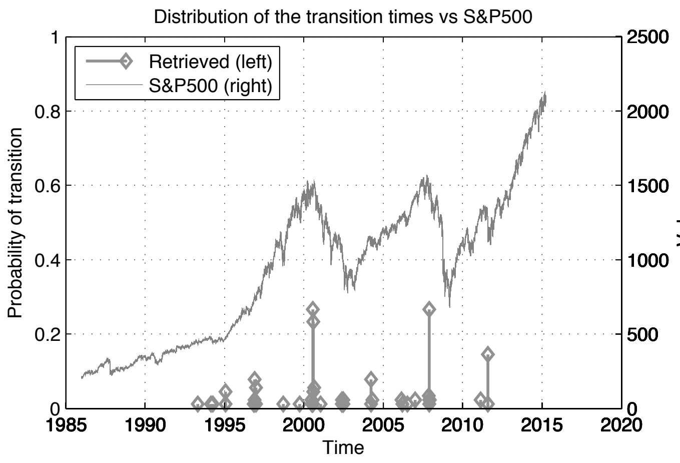
The goal of this study is to discover how the financial and economic environment evolves over time. In particular, we focused the attention to detect *business cycles*[12] and the associated change of relationships among financial and economic variables. Given that the duration of a business cycle is highly variable, the ability to identify the turning point of a cycle (i.e. when a recession starts) is of considerable importance to policymakers, financial companies as well as to individuals. A substantial literature is available about the business cycle turning points detection generally relying on Markov-switching models (Hamilton & Raj, 2005). However, these models are not able to represent some important features such as the dependence structure among variables in each business cycle.

In order to use the nsCTBN model in such a context, we applied a binary discretization to the variable associated with each time-series. Discretization was performed using a look-back period of 1 year, i.e. if the current value is greater than the past one, then the binary variable is set to 1 otherwise, it is set to 0. The approach of looking back into the past is widely used in finance (Moskowitz, Ooi, & Pedersen, 2012). nsCTBNs learning was performed under the UNE setting using the following parameter values: $\lambda_c = \{0.5, 1, 2\}$, $\lambda_e = \{0.1, 1, 10\}$, 2 maximum parents per node, 300 iterations and 10 runs.

Figure 10 shows the probability of transition (left side, left axis) versus the S&P500 equity index used as a reference (left side, right axis) and the posterior probability over the number of epochs (right side). The nsCTBN consists of three epochs with transition times close to the end of July 2000 and the end of November 2007. If we compare these dates to the turning points of the US business cycle reported by the National Bureau of Economic Research[13], then we see that we are not far from the turning point of March 2001 and very close to the one of December 2007, while we missed the turning point which occurred in July 1990, probably because of the limited length of the dataset.

Figure 11 shows the structure of the nsCTBN model corresponding to the most probable number of epochs, i.e. $E = 3$. An arc is included in the nsCTBN model when it occurs in more than 75% of the performed runs in each epoch. The retrieved networks correspond to the following time periods: from January 1986 to July 2000 (epoch 1), from August 2000 to November 2007 (epoch 2) and from December 2007 to March 2015 (epoch 3).

12. Business cycles are fluctuations in aggregate economic activity, they are recurrent (i.e. it is possible to identify expansion-recession cycles), persistent and not periodic (i.e. they differ in length and severity).
13. The official business cycle turning points and dates are available at http://www.nber.org/cycles.html



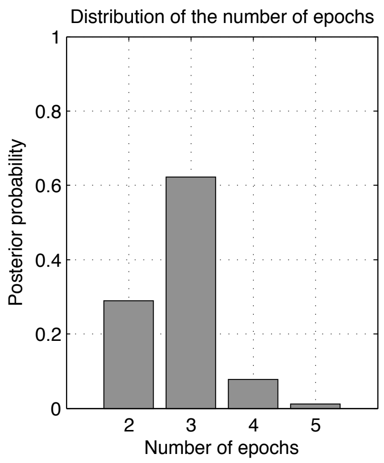
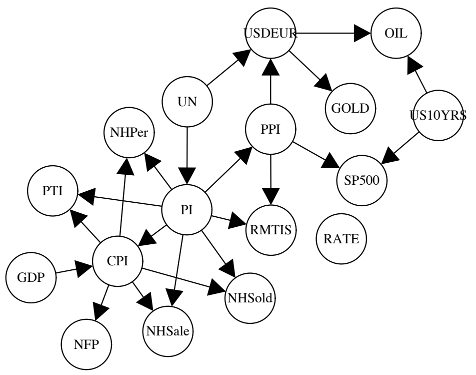
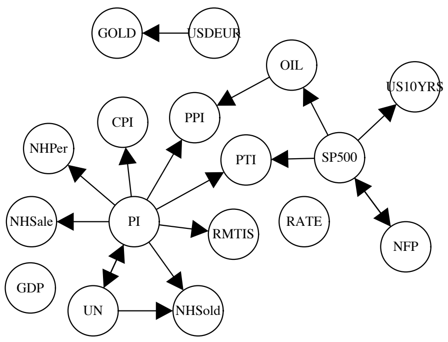


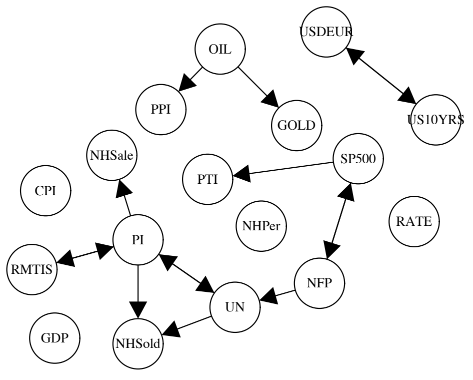
Figure 10: Distribution of the transition times and S&P500 behavior over time (left). Posterior probability over epochs (right) for the learned nsCTBN under the UNE setting.



(a) Epoch 1 (Jan 1986 - Jul 2000).



(b) Epoch 2 (Aug 2000 - Nov 2007).



(c) Epoch 3 (Dec 2007 - Mar 2015).

Figure 11: nsCTBN learned on the macroeconomics dataset under the UNE setting. nsCTBN corresponds to the most probable number of epochs ($E = 3$). An arc is included in the nsCTBN model when it occurs in more than 75% of the runs in each epoch.

The novelty of this approach to the economic analysis opens the door to many considerations and new speculations about the economic variables during business cycles. In this paper, we highlight two patterns emerging from the learned nsCTBN model: the well known relevant role of the personal income (PI) and its relation to the unemployment (UN) (Mankiw, 2014) and the less known relation of the non-farm payrolls (NFP) to the S&P500 equity index (S&P500) (Miao, Ramchander, & Zumwalt, 2014).

6. Conclusions

We introduced non-stationary continuous time Bayesian networks and developed three structural learning algorithms to be used under different knowledge settings (i.e. KTT, KNE and UNE) for the problem to be analyzed. The structural learning algorithm in the known transition times case is exact and it exploits graph theory to infer the optimal nsCTBN's structure. It has a polynomial time complexity under the assumption that the maximum number of parents for each node is fixed. All the nsCTBN's structural learning algorithms are competitive to state-of-the-art algorithms when synthetic and real-world datasets are considered. This statement is proved by a rich set of numerical experiments.

nsCTBNs can be adapted to use different score metrics, as far as the considered score metrics integrates over the non-structural parameters. nsCTBNs exploit an interesting property of CTBNs and offer the possibility to learn the optimal nsCTBN's structure for each single variable. This could be extremely useful in the case when the non-stationary behavior of the analyzed system is not synchronous, and thus it may be the case that each node changes its parents independently from how other nodes change their parents set.

However, two main limitations exist with nsCTBNs: *i*) the variables are assumed to be discrete; specifically each variable of the dataset must take value over a countable number of states and *ii*) finding the optimal value for the λ_c and λ_e hyperparameters can be extremely difficult (the same is true for nsDBNs). Concerning *i*), the problem of discretizing continuous variables has been studied for a long time and robust solutions have been described in the specialized literature. Discretizing continuous variables whose value is measured over time has not been studied intensively and many issues still remain. The problem *ii*) of selecting the optimal value of hyperparameters is known in the specialized literature and much can be done when experts provide their valuable apriori knowledge. However, when such apriori knowledge is poor or not available at all, selecting optimal hyperparameter values can be extremely difficult. It is important to note that one of the strong limitations to studying and comparing non-stationary models is the lack of ground truth models.

Possible directions for further research include the application of nsCTBN's structural learning algorithms to other datasets, such as the arabidopsis thaliana dataset (Grzegorczyk, Aderhold, & Husmeier, 2015) as well as other financial datasets supported by in-depth economic analyses. Another interesting perspective is the study and development of a modeling approach, going towards the direction of allowing each node to change its parents set asynchronously. Furthermore, we think that to increase the applicability to real-world time-series data of the proposed nsCTBN's structural learning algorithms the issue of time-series discretization must be addressed. In particular, we think that this issue must be addressed in an integrated manner with the nsCTBN's structural learning algorithm.

Finally, it could be interesting to apply the framework of nsCTBNs to address the task of classification of objects in a streaming context when using a probabilistic graphical model based approach (Borchani, Martinez, Masegosa, Langseth, Nielsen, Salmerón, Fernández, Madsen, & Sáez, 2015a; Borchani, Martínez, Masegosa, Langseth, Nielsen, Salmerón, Fernández, Madsen, & Sáez, 2015b).

Acknowledgments

The authors wish to thank Alexander Hartemink for having kindly provided the nsDBN jar executable and the associated datasets. A special thank goes to Marco Grzegorczyk for providing the arabidopsis thaliana dataset together with fundamental information to analyze it. The authors are greatly indebted to anonymous referees for their constructive comments and their extremely helpful suggestions, which contributed to significantly improve the quality of the paper. A special thank goes to the Associate Editor Manfred Jaeger.

Fabio Stella is the corresponding author of this article.

References

Acerbi, E., & Stella, F. (2014). Continuous time bayesian networks for gene network reconstruction: a comparative study on time course data. In *The 10th International Symposium on Bioinformatics Research and Applications, Zhangjiajie, China, 2014, 10*.

Acerbi, E., Viganò, E., Poidinger, M., Mortellaro, A., Zelante, T., & Stella, F. (2016). Continuous time bayesian networks identify prdm1 as a negative regulator of th17 cell differentiation in humans. *Scientific Reports*, *6*, 23128.

Acerbi, E., Zelante, T., Narang, V., & Stella, F. (2014). Gene network inference using continuous time bayesian networks: a comparative study and application to th17 cell differentiation. *BMC Bioinformatics*, *15*(1).

Ahmed, A., & Xing, E. P. (2009). Recovering time-varying networks of dependencies in social and biological studies. *Proceedings of the National Academy of Sciences*, *106*(29), 11878–11883.

Bertsimas, D., & Tsitsiklis, J. (1993). Simulated annealing. *Statistical Science*, *8*(1), 10–15.

Borchani, H., Martinez, A. M., Masegosa, A., Langseth, H., Nielsen, T. D., Salmerón, A., Fernández, A., Madsen, A. L., & Sáez, R. (2015a). Dynamic Bayesian modeling for risk prediction in credit operations. In *The 13th Scandinavian Conference on Artificial Intelligence (SCAI 2015), Halmstad, Sweden*.

Borchani, H., Martínez, A. M., Masegosa, A. R., Langseth, H., Nielsen, T. D., Salmerón, A., Fernández, A., Madsen, A. L., & Sáez, R. (2015b). Modeling concept drift: A probabilistic graphical model based approach. In *The 14th International Symposium on Intelligent Data Analysis (IDA 2015), Saint-Etienne, France*.

Boudali, H., & Dugan, J. B. (2006). A continuous-time bayesian network reliability modeling, and analysis framework. *IEEE Transactions on Reliability*, *55*(1), 86–97.

Burge, J., Lane, T., Link, H., Qiu, S., & Clark, V. P. (2009). Discrete dynamic bayesian network analysis of fmri data. *Human brain mapping*, *30*(1), 122–137.

Cantone, I., Marucci, L., Iorio, F., Ricci, M. A., Belcastro, V., Bansal, M., Santini, S., di Bernardo, M., di Bernardo, D., & Cosma, M. P. (2009). A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches. *Cell*, *137*(1), 172 – 181.

Dean, T., & Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Comput. Intell.*, *5*(3), 142–150.

Dondelinger, F., Lebre, S., & Husmeier, D. (2013). Non-homogeneous dynamic bayesian networks with bayesian regularization for inferring gene regulatory networks with gradually time-varying structure. *Machine Learning*, *90*(2), 191–230.

Durante, D., & Dunson, D. B. (2014). Bayesian dynamic financial networks with time-varying predictors. *Statistics & Probability Letters*, *93*, 19–26.

Fan, Y., & Shelton, C. R. (2009). Learning continuous-time social network dynamics. In *The 25th Conference on Uncertainty in Artificial Intelligence (UAI 2009), Montreal, Canada*.

Friedman, N., & Koller, D. (2000). Being bayesian about bayesian network structure: A bayesian approach to structure discovery in bayesian networks. *Machine Learning*, *50*, 95–125.

Gatti, E., Luciani, D., & Stella, F. (2011). A continuous time bayesian network model for cardiogenic heart failure. *Flexible Services and Manufacturing Journal*, *24*(2), 496–515.

Geiger, D., & Heckerman, D. (1997). A characterization of dirchlet distributions through local and global independence. *Annals of Statistics*, *25*, 1344–1368.

Grzegorczyk, M., Aderhold, A., & Husmeier, D. (2015). Inferring bi-directional interactions between circadian clock genes and metabolism with model ensembles. *Statistical Applications in Genetics and Molecular Biology*, *14*(2), 143–167.

Guo, F., Hanneke, S., Fu, W., & Xing, E. P. (2007). Recovering temporally rewiring networks: a model-based approach. In *Machine Learning, Proceedings of the 24th International Conference (ICML 2007), Corvallis, USA, June 20-24, 2007*, pp. 321–328.

Hamilton, J. D., & Raj, B. (Eds.). (2005). *Advances in Markov-Switching Models: Applications in Business Cycle Research and Finance*. Studies in Empirical Economics. Springer-Verlag.

Herbrich, R., Graepel, T., & Murphy, B. (2007). Structure from failure. In *The 2nd USENIX workshop on Tackling computer systems problems with machine learning techniques (SYSML 07), Cambridge, USA*, pp. 1–6.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680.

Lebre, S., Becq, J., Devaux, F., Stumpf, M., & Lelandais, G. (2010). Statistical inference of the time-varying structure of gene regulation networks. *BMC Systems Biology*, *4*(1), 130+.

Liu, M., Hommersom, A., van der Heijden, M., & Lucas, P. J. (2016). Hybrid time bayesian networks. *International Journal of Approximate Reasoning*, –.

Mankiw, N. G. (2014). *Principles of Macroeconomics* (7th edition). South-Western College Pub.

Marini, S., Trifoglio, E., Barbarini, N., Sambo, F., Camillo, B. D., Malovini, A., Manfrini, M., Cobelli, C., & Bellazzi, R. (2015). A dynamic bayesian network model for long-term simulation of clinical complications in type 1 diabetes. *Journal of Biomedical Informatics*, *57*, 369 – 376.

Miao, H., Ramchander, S., & Zumwalt, J. K. (2014). S&p 500 index-futures price jumps and macroeconomic news. *Journal of Futures Markets*, *34*(10), 980–1001.

Moskowitz, T. J., Ooi, Y. H., & Pedersen, L. H. (2012). Time series momentum. *Journal of Financial Economics*, *104*(2), 228–250.

Mumford, J. A., & Ramsey, J. D. (2014). Bayesian networks for fmri: A primer. *Neuroimage*, *86*, 573–582.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Nodelman, U. (2007). *Continuous Time Bayesian Networks*. Ph.D. thesis, Stanford University.

Nodelman, U., & Horvitz, E. (2003). Continuous time bayesian networks for inferring users' presence and activities with extensions for modeling and evaluation. Tech. rep. MSR-TR-2003-97, Microsoft Research.

Nodelman, U., Shelton, C. R., & Koller, D. (2002). Continuous time bayesian networks. In *The 18th Conference on Uncertainty in Artificial Intelligence (UAI 2002), Edmonton, Canada*, pp. 378–387.

Nodelman, U., Shelton, C., & Koller, D. (2003). Learning continuous time bayesian networks. In *The 19th Conference on Uncertainty in Artificial Intelligence (UAI 2003), Acapulco, Mexico*, pp. 451–458.

Pearl, J. (1989). *Probabilistic reasoning in intelligent systems - networks of plausible inference*. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann.

Robinson, J. W., & Hartemink, A. J. (2010). Learning non-stationary dynamic bayesian networks. *Journal of Machine Learning Research*, *11*, 3647–3680.

Scutari, M., & Denis, J.-B. (2014). *Bayesian Networks with Examples in R*. Chapman and Hall, Boca Raton. ISBN 978-1482225587.

Segal, E., Pe'er, D., Regev, A., Koller, D., & Friedman, N. (2005). Learning module networks. *Journal of Machine Learning Research*, *6*, 557–588.

Smith, A. V., Yu, J., Smulders, T. V., Hartemink, A. J., & Jarvis, E. D. (2006). Computational Inference of Neural Information Flow Networks. *PLoS Computational Biology*, *2*(11), e161+.

Spiegelhalter, D. J., & Lauritzen, S. L. (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks*, *20*(5), 579–605.

Sturlaugson, L., & Sheppard, J. W. (2014). Inference complexity in continuous time bayesian networks. In *The 30th Conference on Uncertainty in Artificial Intelligence (UAI 2014), Quebec City, Canada*, pp. 772–779.

Vinh, N. X., Chetty, M., Coppel, R., & Wangikar, P. P. (2012). Gene regulatory network modeling via global optimization of high-order dynamic bayesian network. *BMC Bioinformatics*, *13*, 131.

Xu, J., & Shelton, C. R. (2008). Continuous time bayesian networks for host level network intrusion detection. In *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2008), Antwerp, Belgium*, pp. 613–627.

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

Zou, M., & Conzen, S. D. (2005). A new dynamic bayesian network (dbn) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, *21*(1), 71–79.