SCUOLA DI DOTTORATO
UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA

Department of
Informatics, Systems and Communication

PhD program in Computer Science, Cycle XXXVI

Structure learning and knowledge extraction with Continuous Time Bayesian Network

Bregoli Alessandro
Registration Number 780711

Tutor: Prof. Paolo Napoletano
Supervisor: Prof. Fabio Stella
Coorinator: Prof. Leonardo Mariani

ACADEMIC YEAR 2023-2024

Contents

1 Introduction **11**
1.1 Motivation 11
1.2 Main Contributions 12
1.3 Overview 13

2 Continuous Time Markov Processes **15**
2.1 Formal Definition 15
2.2 Parameter Learning 16
2.3 Inference 19
2.4 Reward Function 21
2.5 Discussion 22

3 Continuous Time Bayesian Networks **23**
3.1 Introduction 23
3.2 Conditional Continuous Time Markov Processes 24
3.3 Continuous Time Bayesian Network 25
3.4 Formal Definition 26
3.4.1 Generative Semantic 26
3.5 Parameter Learning 28
3.6 Structure Learning 28
3.7 Reward Factorization 29
3.8 Multidimensional CTBN Classifier 30
3.9 Discussion 31

4 A Constraint-Based Algorithm for Structure Learning **33**
4.1 Constraint-Based Algorithms for BNs 33
4.2 The CTPC Structure Learning Algorithm 34
4.3 CTPC computational complexity 37
4.4 Numerical Experiments 37
4.5 Discussion 41

5 Constraint-based structure learning of Multi-CTBNC **43**
5.1 Constraint Based Algorithm - naive adaptation 43
5.2 Markov Blanket CTPC 44
5.3 Hybrid structure learning algorithm 45
5.4 Experiments 46
5.4.1 Performance measures 46
5.4.2 Experimental results on synthetic data 47
5.4.3 Experimental results on real data 48
5.5 Discussion 49

6 Sentry State for CTBN **51**
6.1 Naive Approach . . . 51
6.2 Sentry State Identification . . . 51
6.3 Monte Carlo Algorithm . . . 52
6.4 Synthetic experiments . . . 53
6.5 ESS Data Set . . . 57
6.6 Discussion . . . 58

7 Conclusions **59**
7.1 Main Contributions . . . 59
7.2 Future works . . . 60

Appendices **67**

A CTPC computational complexity **69**
A.1 Compute the CIM . . . 70
A.1.1 Test Independence . . . 70
A.2 Learn the parent set (one node) . . . 71
A.2.1 Learn the network's structure . . . 72

List of Figures

2.1 Weather condition: partial trajectory for the area of Brescia (North-east Italy). 18
2.2 These figures depict how the probability of each possible state of the weather condition changes in continuous time for the city of Brescia, as modeled by using the CTMP presented in the Example 2.2. Specifically, Figure 2.2a shows how the probability for each possible state of the weather condition changes in continuous time when the initial ($t = 0$) weather condition is sunny, i.e., the initial state of the weather is assumed to be *sun*. Figures 2.2b and 2.2c do the same as Figure 2.2a for those cases where the initial ($t = 0$) weather condition are assumed to be *rain* and *storm* respectively. 20

3.1 24
3.2 CTBN Example - Eating? Hungry? Full Stomach? 26
3.3 State Space CTBN Example - Eating? Hungry? Full Stomach? - Nodes represent states of the CTBN, two states are adjacent if a transition from one to the other, and vice versa, is possible. 27

4.1 Each of the plots on this figure represents the average F1 score and the standard deviation of the constraint-based algorithm and the CTSS one, against the number of nodes for a specific combination of network density and node cardinality. 38
4.2 Each of the plots on this figure represents the average $\Delta BIC\%$ and the standard deviation against the number of nodes for a specific combination of network density and node cardinality. 39
4.3 Each of the plots on this figure represents the average execution time in seconds and the standard deviation of the constraint-based algorithm and the CTSS one, against the number of nodes for a specific combination of network density and node cardinality. . . 39
4.4 Each of the plots on this figure represents the average Precision, Recall and their respective standard deviations of the constraint-based algorithm and the CTSS one, against the number of nodes for a specific combination of network density and node cardinality. 40

6.1 (a) depicts the graph $\mathcal{G}$ of a CTBN. Its CIMs are in Table 6.1. Slow processes are represented by solid line nodes, while fast processes are represented by dashed line nodes. Colors and filling describe the most likely sentry state, $s = (s_1, s_2, s_3)$, in this system: If a node has green border and it is filled, the corresponding alarm is 1 (*on*) in s. If a node has red border and it is not filled, the corresponding alarm is 0 (*off*) in s. (b) shows an example trajectory from the CTBN represented in Figure 6.1a. Each function in the plot represents the evolution of one of the three binary processes, A, B, and C. 55
6.2 (a) Graph $\mathcal{G}$ of a CTBN model consisting of six processes. The graph $\mathcal{G}$ contains the cycle (A, B, C) as well as the chain (D, E, F). See the caption of Figure 6.1 for an explanation of the node colors. 55
6.4 Graph $\mathcal{G}$ of a CTBN model consisting of a chain of five processes. 55
6.5 Graph $\mathcal{G}$ of a CTBN model consisting of five processes, including a cycle. 55

6.3 This figure reports the Jaccard similarity@k between the REDNT and the naive approach. The x-axis represents the number of states taken into account from the ordered lists generated by the two methods. The structures of the networks used for the experiments, in order of appearance in the plot, are depicted in Figures 6.1a, 6.4, 6.5, 6.6, 6.2a, and 6.7. . . . 56
6.6 Graph $\mathcal{G}$ of a CTBN model consisting of five processes. The graph $\mathcal{G}$ contains a bifurcation after the root node A. . . . 56
6.7 Graph $\mathcal{G}$ of a CTBN model consisting of nine processes and with a more complex structure. The sentry state has three active alarms. . . . 56
6.8 Graph $\mathcal{G}$ of the CTBN learned from the ESS data set. Nodes with green border and background represent active alarms, while nodes with red border and no background indicate inactive alarms. . . . 57

A.1 Conceptual model of the CTPC algorithm. . . . 69

List of Tables

4.1 Execution time of the CTPC on big, sparse networks 41

5.1 Percentage of datasets in which structure learning algorithms achieve better, worse or identical results accordingly to the Wilcoxon signed-rank test with significant level of 0.05. 47
5.2 Overall performances of the structure learning algorithms. 48
5.3 Overall performances of the structure learning algorithms on BHPS dataset. 49

6.1 Conditional Intensity Matrices used for the example in Figure 6.1a. Process A has no parents and therefore its transition rate only depends on its own state: If A is in state 0 (*off*), then its transition rate (to state 1 (*on*)) is 1.0. Process B has a single parent, process A. The states of processes A and B determine the transition rate of process B. If A is in state 0 (*off*) and B is in state 0 (*off*), then B transitions to state 1 (*on*) with rate 0.1. A CTBN is defined from its CIMs and its initial distribution. Its graph illustrates the dependence structure in the CIMs. 54
6.2 Values of EDNT, REDNT, Naive Score, and Naive Count for the CTBN depicted in Figure 6.1a. Higher values of REDNT indicate CTBN states that are more likely to be sentry states. One should note that high-scoring states with few alarms (bold rows) are more interesting in our application as they correspond to states that occur before strong cascading behavior. 54
6.3 Values of EDNT, REDNT, Naive Score, and Naive Count of the states with at most one active alarm for the CTBN depicted in Figure 6.2a. 56

Notation

1. $\mathcal{G} = (\mathbf{V}, \mathbf{E})$: graph.
2. $\mathbf{V}$: set of vertices of the graph $\mathcal{G}$.
3. $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$: directed edges between the vertices of the graph $\mathcal{G}$.
4. $dom(\cdot)$: identify the domain of a variable.
5. $(a, b) \in \mathbf{E}$: directed edge from a to b.
6. $pa(v)\ v \in \mathbf{V} : \forall v' \in pa(v)\ \exists\ (v', v) \in \mathbf{E}$: parent-set of v.
7. $ch(v)\ v \in \mathbf{V} : \forall v' \in ch(v)\ \exists\ (v, v') \in \mathbf{E}$: children-set of v.
8. $\varphi =< v_1, \ldots, v_n >$: $v_i \in \mathbf{V}, v_i \in pa(v_{i+1})$: directed path.
9. $an(v_j) \subseteq \mathbf{V}\ s.t.\ \forall\ v_i \in an(v_j)\ \exists\ \varphi =< v_i, \ldots, v_j >$: Ancestral set of v_j.
10. $de(v_j) \subseteq \mathbf{V}\ s.t.\ \forall\ v_i \in de(v_j)\ \exists\ \varphi =< v_j, \ldots, v_i >$: Descendant set of v_j.
11. $a \perp\!\!\!\perp b|c$: denotes the conditional independence between a and b given c.
12. $\boldsymbol{\mathcal{X}}$: a set of discrete static random variables.
13. $\mathcal{X} \in \boldsymbol{\mathcal{X}}$: a discrete static random variable.
14. $\mathcal{P}$: set of conditional probability distributions.
15. $\mathcal{U}$: utility function over a static **PGM**.
16. X: a continuous time stochastic process.
17. $\mathbf{X}$: a set of continuous time stochastic processes.
18. Q: a (conditional) intensity matrix.
19. $\boldsymbol{Q}$: a set of (conditional) intensity matrices.
20. q: parameter for an exponential distribution.
21. $\mathbf{q}$: parameters for a set of exponential distribution.
22. θ: parameters for a multinomial distribution.
23. $\boldsymbol{\theta}$: parameters for a set of multinomial distributions.
24. $\mathbf{I}$: identity matrix. A bi-dimensional matrix where the main diagonal elements assume value 1 and the off-diagonal elements assume value 0.
25. $\mathcal{R}(x)$: instantaneous component of a reward function.
26. $\mathcal{C}(x, x')$: Lump sum component of a reward function.
27. $\mathcal{N}$: Continuous Time Bayesian Network.

28. $\mathcal{M}$: Multidimensional Continuous Time Bayesian Network Classifier.
29. $\mathbf{x} \in \times_{X_j \in \mathbf{X}} dom(X_j)$: a specific configuration for a Continuous Time Bayesian Network.
30. σ: Trajectory.
31. $\boldsymbol{\sigma}$: set of trajectories.
32. **BHPS**: British Household Panel survey
33. **BN**: Bayesian Network.
34. **CIM**: Conditional Intensity Matrix.
35. **CKD**: Chronic Kidney Disease.
36. **CPT**: Conditional Probability Table.
37. **CTBN**: Continuous Time Bayesian Network.
38. **CTBNC**: Continuous Time Bayesian Network Classifier.
39. **CTMDP**: Continuous Time Markov Decision Process.
40. **CTMP**: Continuous Time Markov Process.
41. **DAG**: Directed Acyclic Graph.
42. **DBN**: Dynamic Bayesian Network.
43. **EU**: Expected Utility.
44. **ID**: Influence Diagram.
45. **MAP**: Maximum A Posteriori
46. **MEU**: Maximum Expected Utility.
47. **MLE**: Maximum Likelihood Estimation.
48. **Multi-CTBNC**: Multidimensional Continuous Time Bayesian Network Classifier.
49. **PE**: Physical Examination.
50. **PGM**: Probabilistic Graphical Model.
51. **Qa**: Access blood flow.
52. **VA**: Vascular Access.

Chapter 1

Introduction

1.1 Motivation

Healthcare, finance, telecommunications, social networks, e-commerce and homeland security, are a few instances of real world domains where the system to be studied involves several variables whose value changes over time. Studying such systems consists in understanding how they work, in making accurate predictions about their evolution over time, and consequently in making effective decisions. To this extent, huge amount of data are typically collected by measuring the value of several variables over time, with the aim of modeling the *underlying process*, i.e., the process which rules the evolution of the system under study.

In healthcare, the increasing use of digital machines and devices, combined with the availability of Electronic Health Records (EHR), enable to measure, at different points in time, the value of numerous variables related to patients. The huge amount of data collected can then be used to study, analyze and better understand the underlying process which governs the temporal evolution of the patient disease. This ambitious goal is pursued by feeding collected data into powerful artificial intelligence and machine learning algorithms which are used to develop and deploy models for making accurate predictions about the temporal evolution of the patient disease, and for making effective decisions such that the decision maker is aware of their impact on the patient [1, 2, 3, 4]. The same applies to the industrial sector, where machines and tools are increasingly equipped with sensors that monitor their operation by generating data streams [5, 6] with the aim to accomplish diagnostic and/or prognostic tasks on them. Modeling these type of systems, understanding how they work, i.e., understanding how they evolve over time, and making proactive and effective decisions, are challenging tasks of paramount relevance for several reasons.

Many theoretical frameworks have been developed with these goals in mind. For instance, the evolution of a process over discrete time is modeled by the Markov Chain framework [7], while the Markov Decision Process expands on the Markov Chain by incorporating decision capabilities. To capture time as a continuous quantity, researchers have developed a continuous time version of the Markov Chain [8] and Markov Decision Process [9]. If the process being analyzed can be decomposed into a set of components (variables) that describe its evolution over time, other frameworks such as dynamic Bayesian networks (DBNs) [10] or Continuous Time Bayesian Networks (CTBNs) [11] can be employed. The Point Process framework [12] is useful when events are the main focus and time matters, but the duration of the event is meaningless or not that relevant to describe and study the underlying process. If the sequence of events or decisions to be made are the main issue to be addressed and solved, the influence diagram (ID) framework [13] offers an effective and efficient tool for modelling the problem and finding the optimal sequence of decisions to be made.

Artificial intelligence, machine learning and in particular deep learning have achieved impressive results in the last few years. However, it is increasingly understood and agreed on that a prediction needs to be explained and understood, before being accepted and deployed. In particular, for the domains mentioned above, we are asked to develop efficient algorithms for making accurate predictions, about the temporal evolution of the system under study, which at the same time must be understood. Furthermore, it commonly agreed upon that the decision-making task substantially differs from the prediction task. Indeed, while predictive algorithms substantially solve the problem of approximating a function, decision-making algorithms, to be reliable, credible, and explainable, must share a mechanistic model of the underlying process.

DBNs, CTBNs, and IDs, are powerful artificial intelligence and machine learning models which belong to the class of probabilistic graphical models (PGMs). They have been recognized to offer an excellent degree of interpretability and explainability, thus helping the domain expert to accomplish the complex task of understanding the underlying process. Furthermore, these PGMs models allow combining the available data and the domain expert's knowledge, which is very often of paramount importance for discovering/approximating the probabilistic structure of the underlying process. DBNs, CTBNs, and IDs can be used for performing both prediction and decision-making. Indeed, they allow making predictions about the future evolution of the underlying process, as well as they allow estimating the impact of a given decision or a sequence of decisions on the system under study. They typically allow for a deeper exploration of the model to gain a better understanding of the reasons why a given response is obtained by the system when making a given decision or sequence of decisions. In other terms, these models can be exploited to understand the underlying process, by grasping the mechanics governing its evolution and by identifying situations of particular relevance when the system under study is evolving over time. Therefore, we made the decision to study CTBNs for modeling the underlying process when the system under study evolves over time and when we are asked to understand and explain predictions, as well as to ensure that a decision or a sequence of decisions can be properly assessed.

1.2 Main Contributions

This dissertation is focused on probabilistic graphical models, with specific reference to: Continuous Time Bayesian Networks and Multidimensional Continuous Time Bayesian Network Classifiers. All the contributions presented in this document have been published on peer-reviewed journals or international conferences. In particular, during my Ph.D. I published the followings manuscripts:

- *Constraint-Based Learning for Continuous Time Bayesian Networks* [14]: describes the first constraint based structure learning algorithm for continuous time Bayesian networks. This work, starting from the definition of conditional independence for a continuous time Bayesian network, identifies suitable hypothesis tests and proposes a modified version of the PC algorithm, called Continuous Time PC (CTPC). The new algorithm is capable of learning the structure by exploiting the peculiarities of continuous time Bayesian networks. This paper also presents a preliminary set of synthetic experiments to empirically highlight strengths and weaknesses of the new algorithm when compared to the current state of the art.

- *A constraint-based algorithm for the structure learning of continuous time Bayesian networks* [15]: this manuscript extends the previous work in [14] by studying effectiveness and efficiency of the CTPC constraint-based structure learning algorithm. In particular, in this paper, we propose a parallelized version of the plain vanilla CTPC algorithm which also exploits a cache mechanism. A rich set of numerical experiments, using synthetic data generated by large continuous time Bayesian network models, are performed to investigate how the new implementation of the CTPC algorithm compares to its plain-vanilla version. The results obtained confirm the new implementation of the CTPC algorithm scales well with respect to both the number of nodes and the amount of data available for learning when compared to the plain vanilla versions. In particular, results of numerical experiments confirm that the use of a cache greatly reduces the time needed by the learning algorithm when compared to the plan vanilla CTPC. This work also provides a formal analysis of the time complexity of the CTPC algorithm, which shows that its time complexity is the same as that of the score based algorithm.

- *Structure learning algorithms for multidimensional continuous time Bayesian network classifiers* [16]: in this work, I served as second author by helping to extend the constraint-based algorithm for CTBNs [14] to learn multidimensional continuous time Bayesian network classifiers. This work introduces two new algorithms: a naive adaptation of the CTPC algorithm and a new algorithm called Markov-Blanket CTPC, which exploits the peculiarities of the classifier to greatly reduce the execution time at the cost of slightly reducing the accuracy of the recovered multidimensional continuous time Bayesian network classifier.

- *Constraint-based and hybrid structure learning of multidimensional continuous time Bayesian network classifiers*: this is an extension of the previous work [16]. This paper presents an extensive set of numerical experiments as well as a real word dataset, to better understand strengths and weaknesses of the proposed algorithms. It also introduced the first hybrid structure learning algorithm for continuous time Bayesian networks and multidimensional continuous time Bayesian network classifiers. This new algorithm combines the strengths and weaknesses of score based and constraint based algorithms while achieving good results both in terms of performance and execution time.

- *Analyzing Complex Systems with Cascades Using Continuous Time Bayesian Networks* [5]: this manuscript introduces the concept of *sentry state* for a continuous time Bayesian network. A sentry state is a critical configuration of the system under study that triggers a series of significant events, which in turn lead to a ripple effect. This work presents two algorithms for identifying a sentry state and describes the results of a set of numerical experiments which confirm their effectiveness.

- *Personalized Arteriovenous Fistula Management through Utility Maximization with Influence Diagram* [4]: in this paper we present a practical application of an Influence Diagram to solve the problem of early identification of stenosis in vascular access which should be used for the hemodialysis treatment of chronic kidney disease patients. The model developed allows to make a decision whether to intervene or not, thus reducing the risk of being too late. The main goal of this work was to evaluate the influence diagram in a setting where the only available data are domain expert knowledge and scientific literature. It is worthwhile to mention that, even if this work was developed during the Ph.D., it will not be further discussed in this dissertation because it does not introduce any methodological results, and it is not based on the framework of the CTBNs.

1.3 Overview

The rest of the dissertation consists of six chapters, two giving the main definitions and notation used in this manuscript, three devoted to present and describe the main contributions of my research activity, and a final chapter which draws conclusions about what has been achieved by my research activity. More precisely, the dissertation is organized as follows:

- Chapter 2 introduces *continuous time Markov processes,* giving the main definitions and the notation needed to understand continuous time Bayesian networks. In particular, this chapter gives a formal definition of *continuous time Markov process*, and shows how to perform parameter learning and inference for this mathematical model.

- Chapter 3 presents the *continuous time Bayesian network* framework. This chapter gives a formal definition of continuous time Bayesian network, shows how to use available data to learn a continuous time Bayesian network model and how a continuous time Bayesian network can be extended to include a reward function. In this chapter, a formal definition of *multidimensional continuous time Bayesian network classifier* is also given.

- Chapter 4 introduces and describes the main contributions presented in [14] and [15]. In particular, this chapter is about the first, and up to date the only, constraint based structure learning algorithm which has been made available in the specialized literature on continuous time Bayesian networks.

- Chapter 5 is about the main contributions presented in [16] and [17]. In particular, this chapter introduces and describes the first constraint based structure learning algorithm for multidimensional continuous time Bayesian network classifiers and the first hybrid structure learning algorithm for both continuous time Bayesian networks and multidimensional continuous time Bayesian network classifiers.

- Chapter 6 introduces and describes the contribution in [5]. In particular, this chapter introduces, in the continuous time Bayesian networks' framework, the new concept of *sentry state*. Furthermore, the chapter also describes a new algorithm to identify a sentry state from a given set of data when a continuous time Bayesian network model has been previously learnt.

- Chapter 7: this chapter, starts by commenting on the main contributions of this dissertation, then it describes and comments their main limitations in theoretical and practical terms. This chapter closes by illustrating future works and listing potentially interesting extensions of the continuous time Bayesian networks' framework.

Chapter 2

Continuous Time Markov Processes

A *Continuous Time Markov Process* (CTMP) is a probabilistic model which allows describing the evolution of a discrete state variable over continuous time. The CTMP framework exploit the Markov property; specifically, in this dissertation, we only consider CTMPs that satisfy the first order Markov property, i.e., the state of the variable in the future is independent of the past given the state at present time. CTMPs are the continuous-time counterpart of Markov chains. By modeling time as a continuous quantity, CTMPs can effectively describe all those processes that evolve asynchronously over time. Indeed, unlike Markov chains, in CTMPs it is not necessary to specify a sampling frequency. The main motivation behind the exploration of these probabilistic models is that CTMPs are the basis for the Continuous Time Bayesian Network Framework that will be presented in Chapter 3.

The rest of the chapter is organized as follows:

- Section 2.1 gives a formal definition of CTMP.
- Section 2.2 presents some methodologies for learning the parameters of a CTMP.
- Section 2.3 introduces and describes a basic algorithm for performing inference on CTMPs.
- Section 2.4 gives the concept of reward function for a CTMP.
- Section 2.5 closes the chapter with a short discussion on CTMPs.

2.1 Formal Definition

A CTMP [18] is a continuous time stochastic process $X = \{X(t) : t \in [0, \infty)\}$ which satisfies the following Markov property:

$$X(t_1) \perp\!\!\!\perp X(t_3)|X(t_2), \quad \forall\, t_1 < t_2 < t_3, \tag{2.1}$$

where $\cdot \perp\!\!\!\perp \cdot \mid \cdot$ denotes conditional independence. The state of the process X changes in continuous-time and takes values in the domain, $dom(X)$ which we assume to be a finite set of discrete values.

A CTMP can be formally described as: CTMP $= (X, \mathcal{P}_X, Q_X)$ where:

- X: is a continuous time stochastic process over a finite domain $dom(X)$.
- $\mathcal{P}_X$: is the initial distribution at time $t = 0$ modeled as a static discrete distribution.
- Q_X: is an intensity matrix modeling the evolution of the process over time.

In particular, Q_X is a square matrix where the number of rows and columns equals the number of possible values (states) that the continuous time stochastic process X can take, i.e., $|dom(X)|$. Each row of the intensity matrix Q_X sums up to 0 and models two different processes:

1. The time when the CTMP abandons the current state $x \in dom(X)$, which is assumed to be a random variable following an *exponential distribution* with parameter $q_x \in \mathbb{R}^+$.
2. The state where X transitions when abandoning the current state $x \in dom(X)$, which is assumed to be a random variable following a *multinomial distribution* with parameters $\theta_{xy} = \frac{q_{xy}}{q_x}$, $x, y \in dom(X)$, $x \neq y$.

An instance of the intensity matrix Q_X, when X can take three possible values (states) x_1, x_2, x_3, i.e., $dom(X)=\{x_1, x_2, x_3\}$, is as follows:

$$Q_X = \begin{bmatrix} -q_{x_1} & q_{x_1 x_2} & q_{x_1 x_3} \\ q_{x_2 x_1} & -q_{x_2} & q_{x_2 x_3} \\ q_{x_3 x_1} & q_{x_3 x_2} & -q_{x_3} \end{bmatrix} \qquad q_{x_i} = \sum_{i \neq j} q_{x_i x_j}, \;\; q_{x_i x_j} \geqslant 0 \; \forall \, i, j. \tag{2.2}$$

To better understand the nature of Q_X, it is possible to explore its relationship with the probability distribution of the process at time t. Accordingly to [18], if we let $p(t)$ be the marginal distribution of the process X at time t; the infinitesimal rate semantics of the intensity matrix Q_X can be stated as follows:

$$p(t+\epsilon) = p(t)(\mathbf{I} + \epsilon Q_X) + o(\epsilon). \tag{2.3}$$

If we let $\epsilon \to 0$ in the previous equation we get:

$$\lim_{\epsilon \to 0} (p(t+\epsilon) = p(t)(\mathbf{I} + \epsilon Q_X) + o(\epsilon)) \tag{2.4}$$

$$\lim_{\epsilon \to 0} \frac{p(t+\epsilon) - p(t)}{\epsilon} = p(t) Q_X \tag{2.5}$$

$$\frac{dp(t)}{dt} = p(t) Q_X \tag{2.6}$$

The solution to the first-order homogeneous differential equation (Equation 2.6) is:

$$p(t+s) = p(t) e^{s Q_X} \tag{2.7}$$

with $s > 0$.

Example 2.1 (Evolution of weather condition in continuous time)**.** Assume we want to model the evolution of the weather condition in continuous time for a given location, and that the weather's condition can take one of the following three values (states):

- sun
- rain
- storm

A possible CTMP for describing the evolution of the weather condition in continuous time consists of: *i*) a stochastic process X which can take one of the following values; *sun*, *rain*, *storm*, i.e., where $dom(X) = \{sun, rain, storm\}$, *ii*) an initial distribution $\mathcal{P}_X$, giving the probability of observing each of the following values (states); *sun*, *rain*, *storm* at time 0, and *iii*) an intensity matrix Q_X which is used to specify:

- the parameters of three exponential distributions, where each exponential distribution is associated with one of the following states; *sun*, *rain*, *storm*, and models the time to transition from the current state to a different one, i.e., the time for the weather condition to abandon the current state to transition to a different one,
- the parameters of three multinomial distributions, each one associated with a specific state of the weather's condition, i.e., $dom(X) = \{sun, rain, storm\}$, and describing how the weather condition transitions from one state to another one.

2.2 Parameter Learning

The representation of a phenomenon modeled by a CTMP consists of a sequence of time-indexed events, one for each transition from one state to another. More formally, we say that a realization of a CTMPs is a *trajectory*, being a right-continuous, and piece-wise constant function of time:

$$\sigma = \{\langle t_0, s_0 \rangle, \langle t_1, s_1 \rangle, ..., \langle t_I, s_I \rangle\}, \qquad t_0 < t_1 < \cdots < t_I \text{ and } s_i \in dom(X) \;\; \forall i \in (0, I). \tag{2.8}$$

Accordingly to [11] the *likelihood of a trajectory* is defined as the product of the conditional probabilities of each event, formally as follows:

$$p(\sigma) = \overbrace{\mathcal{P}(X(t_0) = s_0)}^{\text{init distribution}} \prod_{i=0}^{I-1} \left(\overbrace{q_{s_i} e^{-q_{s_i}(t_{i+1}-t_i)}}^{\text{density of duration}} \; \overbrace{\frac{q_{s_i,s_{i+1}}}{q_{s_i}}}^{\text{pr of trans}} \right) \tag{2.9}$$

$$= \mathcal{P}(X(t_0) = s_0) \prod_{i=0}^{I-1} e^{-q_{s_i}(t_{i+1}-t_i)} \prod_{i=0}^{I-1} q_{s_i,s_{i+1}} \tag{2.10}$$

while the corresponding *log-likelihood of a trajectory* is defined as follows:

$$\ln p(\sigma) = \ln \mathcal{P}(X(t_0) = s_0) - \sum_{i=0}^{I-1} q_{s_i}(t_{i+1} - t_i) + \sum_{i=0}^{I-1} \ln q_{s_i,s_{i+1}} \tag{2.11}$$

Looking at Equation 2.11 we can identify the *sufficient statistics of a trajectory*:

- $T[s]$: the total time spent in state s.
- $N[s, s']$: the number of transitions from state s to s'

We can now rewrite Equation 2.11 as follows:

$$\ln p(\sigma) = \ln \mathcal{P}(X(t_0) = s_0) - \sum_{s} T[s] q_s + \sum_{s \neq s'} N[s, s'] \ln q_{s,s'} \qquad s_0, s, s' \in dom(X) \tag{2.12}$$

In case there is more than one trajectory, the Equation 2.12 still holds true, and the sufficient statistics are given by the sums of the sufficient statistics over the individual trajectories.

After giving the concepts of trajectory, likelihood, log-likelihood and sufficient statistic, we now introduce the process of *parameter estimation*. The *Maximum Likelihood Estimation* (MLE) can be obtained by differentiating Equation 2.12 as follows:

$$\frac{d \ln p(\sigma)}{d q_{s,s'}} = \frac{N[s, s']}{q_{s,s'}} - T[s] \qquad \forall s \neq s' \tag{2.13}$$

to obtain the following parameters:

$$\hat{q}_{s,s'} = \frac{N[s, s']}{T[s]} \qquad \forall s \neq s' \tag{2.14}$$

$$\hat{q}_s = \sum_{s \neq s} \frac{N[s, s']}{T[s]} \qquad \forall s \tag{2.15}$$

The literature also proposes a *Maximum A Posteriori* (MAP) approach to estimate the parameters [18]. This approach is particularly useful when the availability of data is limited. Indeed, in such a condition, MAP allows exploiting a prior distribution over the parameters. For the CTMPs, it is used an independent gamma distribution for each parameter $q_{s,s'}, \forall s \neq s'$. given the two hyperparameters $\alpha_{s,s'}$, $\tau_{s,s'}$ for each gamma distribution. In particular, the MAP estimation of parameters turn out to be as follows:

$$\hat{q}_{s,s'} = \frac{N[s, s'] + \alpha_{s,s'}}{T[s] + \tau_{s,s'}} \qquad \forall s \neq s' \tag{2.16}$$

$$\hat{q}_s = \sum_{s \neq s'} \frac{N[s, s'] + \alpha_{s,s'}}{T[s] + \tau_{s,s'}} \qquad \forall s \tag{2.17}$$

An alternative approach for the MAP estimation of the parameters is discussed in [11]. Considering that the intensity matrix of a CTMP can be decomposed as follows:

$$\boldsymbol{\theta}_X = \begin{bmatrix} 0 & \theta_{x_1x_2} & \theta_{x_1x_3} \\ \theta_{x_2x_1} & 0 & \theta_{x_2x_3} \\ \theta_{x_3x_1} & \theta_{x_3x_2} & 0 \end{bmatrix} \qquad \mathbf{q}_X = [q_{x_1}, q_{x_2}, q_{x_3}] \tag{2.18}$$

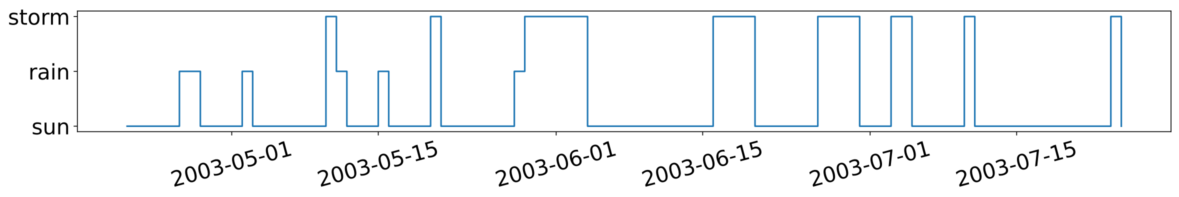


Figure 2.1: Weather condition: partial trajectory for the area of Brescia (North-east Italy).

where $\mathbf{q}_X$ is the main diagonal of the intensity matrix Q_X and models the set of exponential distributions governing when a transition occurs, while $\boldsymbol{\theta}_X$ describes the set of multinomial distributions governing where the CTMP transition when abandoning the current state. The main diagonal of $\boldsymbol{\theta}_X$ is filled with 0 while each off-diagonal element is computed starting from Q_X: $\theta_{xy} = \frac{q_{xy}}{q_x}$, $x, y \in dom(X)$, $x \neq y$. Starting from this decomposition of the parameters [11] identify two classes of prior: the gamma distribution for $\mathbf{q}_X$ and the Dirichlet distribution for $\boldsymbol{\theta}_X$. Specifically:

- For each exponential parameter $q_s \in \mathbf{q}_X$ a gamma distribution with parameters α_s and τ_s is used.
- For each multinomial distribution (one for each row of $\boldsymbol{\theta}_X$) a Dirichlet distribution with parameters $\alpha_{s,s'} \forall s \neq s'$ *s.t.* $\sum_{s'} \alpha_{s,s'} = \alpha_s$

Using this formulation of parameters and prior distributions for the MAP the parameters can be estimated as follows:

$$\hat{\boldsymbol{\theta}}_{s,s'} = \frac{\alpha_{s,s'} + N[s,s']}{\alpha_s + \sum_{s \neq s''} N[s,s'']} \qquad \forall s \neq s' \tag{2.19}$$

$$\hat{q}_s = \frac{\alpha_s + \sum_{s \neq s'} N[s,s']}{\tau_s + T[s]} \qquad \forall s \tag{2.20}$$

Example 2.2 (Evolution of weather condition in continuous time - Area of Brescia)**.** The meteorological website *ilmeteo* has a section dedicated to day by day meteorological history of most Italian cities [1]. For this example, we are interested in the data from the city of Brescia (North east Italy) for the time frame 2003-2022. Given the trajectory of the weather condition for the area of Brescia (Figure 2.1), we first compute the corresponding sufficient statistics:

N	*Sun*	*Rain*	*Storm*
Sun	0	640	336
Rain	657	0	129
Storm	319	146	0

T	*Sun*	*Rain*	*Storm*
	5060	1583	661

The sufficient statistics described by the N matrix count how many times the weather condition transitions from one state to another, according to the available data. For example, by the first row of the above matrix, we can state that the weather condition transitioned 640 times from the state *sun* to the state *rain* and transitioned 336 times from the state *sun* to the state *storm*. The sufficient statistic described by T shows the number of days that a given weather condition, i.e., *sun*, *rain*, *storm*, has been observed.

The second and last step consists of computing the intensity matrix by using the MLE approach (Equations 2.14, 2.15):

$$Q_{\text{weather's-condition}} = \begin{bmatrix} -0.193 & 0.126 & 0.067 \\ 0.415 & -0.497 & 0.082 \\ 0.483 & 0.221 & -0.704 \end{bmatrix}$$

From this matrix we can extract different information. For example, the main diagonal represents the average frequency of transitions per time unit; consequently, its reciprocal represents the average

[1] www.ilmeteo.it/portale/archivio-meteo- September 04Th, 2023

residence time in any specific state. As a result, we can conclude that the average duration of a period without precipitation is $\frac{1}{0.193} = 5.18$ days.

It is worthwhile to mention that for this example, the static distribution $\mathcal{P}$, i.e., the distribution at time zero ($t = 0$), has been deliberately ignored. The main motivations behind this decision are as follows:

1. Since only one trajectory (2003-2022) was used for this example, then only one observation for the time t_0 January 1, 2003 is available.
2. It is almost always the case that the only component of interest is the dynamic one. As regards the distribution at time t_0, it is common practice to use a uniform distribution.

2.3 Inference

The most basic kind of inference that can be made over a CTMP is the inference where no evidence, except for the starting time $t = 0$, is available for the considered stochastic process. Exact inference can be made using Equation 2.7. However, this approach suffers the following issues:

- The matrix exponential operation cannot be exactly computed.
- Accordingly to [19], all the approximation present in literature have limitations.
- The matrix exponential makes inference intractable even for CTMPs of modest size.

A different approach, based on Monte Carlo (MC) simulation, can be exploited to overcome the above issues. Indeed, MC simulation overcomes all the issues related to matrix exponentiation, but this is achieved at the expenses of the exactness of obtained inference.

Algorithm 1 Forward Sampling for CTMP.

```
procedure CTMP-SAMPLE($X(0)$, $t_{end}$)
    $t \leftarrow 0$, $\sigma \leftarrow \{\langle 0, X(0) \rangle\}$                                    ▷ Initialization
    loop
        $\Delta t \leftarrow$ draw a sample from an exponential with rate $q_{X(t)}$      ▷ Time to transition sampling
        if $t + \Delta t > t_{end}$ then
            Add $\langle t_{end}, X(t) \rangle$ to $\sigma$
            return $\sigma$
        end if
        $X(t + \Delta t) \leftarrow$ draw a sample from a multinomial with $\theta_{X(t)}$    ▷ Next state
        $t \leftarrow t + \Delta t$
        Add $\langle t, X(t) \rangle$ to $\sigma$
    end loop
end procedure
```

The Algorithm 1 allows generating a trajectory σ starting from a CTMP, an initial state $X(0)$ and an ending time t_{end}. The sampling procedure exploits the decomposability of the intensity matrix. First, it uses the parameter $q_X(t)$ to determine when the next transition occurs (*Time to transition sampling*); then it draws a sample from the multinomial distribution with parameter $\theta_{X(t)}$.

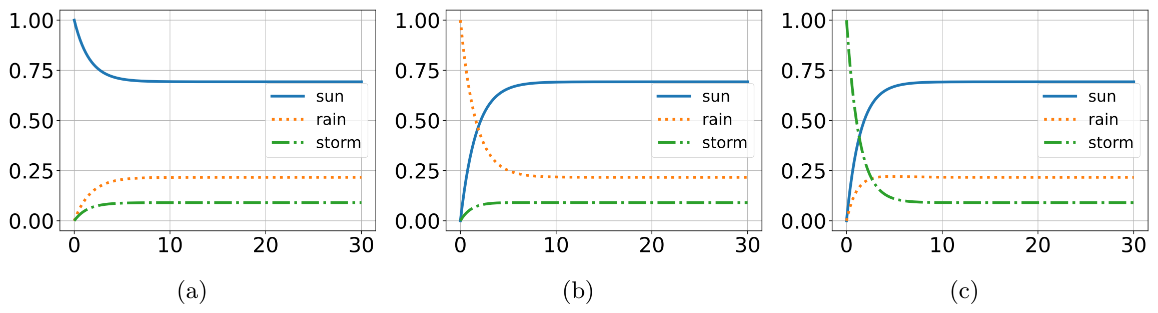


Figure 2.2: These figures depict how the probability of each possible state of the weather condition changes in continuous time for the city of Brescia, as modeled by using the CTMP presented in the Example 2.2. Specifically, Figure 2.2a shows how the probability for each possible state of the weather condition changes in continuous time when the initial ($t = 0$) weather condition is sunny, i.e., the initial state of the weather is assumed to be *sun*. Figures 2.2b and 2.2c do the same as Figure 2.2a for those cases where the initial ($t = 0$) weather condition are assumed to be *rain* and *storm* respectively.

Algorithm 2 Approximated inference without evidence for a CTMP

```
procedure CTMP-INFERENCE-APPROX($X(0)$, $t_{inference}$, $iterations$)
    for all $x \in dom(X)$ do                                   ▷ Initialization
        $C[x] \leftarrow 0$
    end for
    for $i \leftarrow 1$ to $iterations$ do
        $\sigma \leftarrow$ CTMP-SAMPLE($X(0), t_{inference}$)          ▷ Generate trajectory
        $X(t_{inference}) \leftarrow$ last state of $\sigma$                ▷ Extract last state
        $C[X(t_{inference})] += 1$                                  ▷ Update the counted
    end for
    return $C/iterations$      ▷ Normalization of the counter vector to obtain the probabilities
end procedure
```

Algorithm 2 implements the approximated inference based on MC simulation. The algorithm returns the probability distribution at time $t_{inference}$ starting from the state $X(0)$. The variable *iterations* determines the number of trajectories generated to estimate the distribution of interest; the more trajectories are used, the more accurate the result will be.

Example 2.3 (Inference on weather conditions - City of Brescia)**.** Assume we are interested in estimating the probability of a specific weather condition for the next day. Given the parameters value estimated in the Example 2.2, we can use an inference algorithm to estimate the probability distribution of the weather condition. We start by distinguishing among three different cases: today is *i*) a sunny day, *ii*) a rainy day or *iii*) a stormy day. Then we have two options, i.e., we can apply the exact algorithm (Equation 2.7) or the approximate algorithm based on MC simulation (Algorithm 2). In this case, we are faced with a few states, thus we can compute the desired probability distribution by applying the exact algorithm to obtain the following:

	Sun	*Rain*	*Storm*
Sun	0.86	0.10	0.04
Rain	0.31	0.63	0.06
Storm	0.35	0.15	0.5

Figure 2.2 depicts the evolution, in continuous time, of the weather condition over the course of a month, when assuming the first day ($t = 0$) was a sunny day, a rainy day or a stormy day. Observing the 3 figures we can conclude that the probability values associated with the three states of the weather condition converge to the same values just after few days. Informally, we can understand that a sunny day 10 days ago has little to do with the weather of today. Formally we say that the model reaches steady state.

There are more sophisticated inference algorithms that deal with the presence of evidence [11], [20]. However, these algorithms do not fall within the scope of this thesis, and consequently they will not be discussed.

2.4 Reward Function

Accordingly to [9], a *reward function* is a function that maps the states of a CTMP onto a real number. A reward function consists of two quantities,

- $\mathcal{R}(x) : dom(X) \to \mathbb{R}$, the *instantaneous reward* for staying in state x and
- $\mathcal{C}(x, x') : dom(X) \times dom(X) \to \mathbb{R}$, the *lump sum reward* when the CTMP transitions from state x to state x'.

The estimation of a utility function for a CTMP can be performed in two ways: *i*) the *Finite-horizon expected reward* and *ii*) the *Infinite-horizon expected discounted reward* [9].

Following the *Finite-horizon expected reward* approach we evaluate the reward function up to a specific point in time:

$$V_{t_{end}}(x) = \mathbb{E}_x\left[\sum_{i=0}^{t_{i+1}=t_{end}} \mathcal{C}(X(t_i), X(t_{i+1})) + \int_{t_i}^{t_{i+1}} \mathcal{R}(X(t_i))dt\right] : t_i < t_{i+1} \tag{2.21}$$

where t_{end} is the ending time and $\mathbb{E}_x(\cdot)$ is the expectation when conditioning on $X(0) = x$, and the t_i's are the transition times.

The *Infinite-horizon expected discounted reward* approach does not define any time horizon, but to ensure convergence, it relies on a discount factor:

$$V_\gamma(x) = \mathbb{E}_x\left[\sum_{i=0}^{\infty} e^{-\gamma t_{i+1}}\mathcal{C}(X(t_i), X(t_{i+1})) + \int_{t_i}^{t_{i+1}} e^{-\gamma t}\mathcal{R}(X(t_i))dt\right] : t_i < t_{i+1} \tag{2.22}$$

where $\gamma > 0$ is referred to as the *discounting factor*.

Regardless of the approach used, it is always possible to compute the reward function for a trajectory. Thus, it is straightforward to apply the Monte Carlo method to estimate an approximate expected reward. The Algorithm 3 shows the code for the approximated estimation of the infinite horizon expected reward. In order to estimate the expected reward, we need to set the values of the following hyperparameters:

1. γ: the discounting factor;
2. t_{end}: the ending time for each trajectory;
3. *iterations*: the number of trajectories to be generated.

Choosing the discounting factor γ and the ending time t_{end}, we implicitly decide on the importance of the *distant future* and the appropriate values depend on the specific application. On the other hand, the variable *iterations* controls the trade-off between the quality of the approximation and its computational cost; it is possible to choose its value by using a stopping-rule approach based on variance following what proposed and described in [21].

Also, the finite horizon expected reward can be obtained from Algorithm 3 by setting the discount factor γ to 0.

Example 2.4 (Reward on weather condition - City of Brescia)**.** In the previous section we asked how likely a specific weather condition would happen in the near future. Now we ask a slightly different question: *"How many sunny days will there be in the next week?"*

Answering this question can be done by exploiting the instantaneous component of a given reward function. Since we are not interested in the weather change but only in the number of days we can ignore the lump sum component. The instantaneous component can be conveniently set as follows:

$$\mathcal{R}(x) = \begin{cases} 1, & \text{if } x = sun \\ 0, & \text{otherwise} \end{cases} \qquad \mathcal{C}(x, x') = 0 \qquad \forall x, x' \in dom(X)$$

Algorithm 3 Approximated estimation of the infinite horizon expected reward for a CTMP

```
procedure CTMP-INFINITE-HORIZON-REWARD-APPROX($X(0)$, $\gamma$, $t_{end}$, $iterations$)
    $reward \leftarrow 0$
    for $i \leftarrow 1$ to $iterations$ do
        $\sigma \leftarrow$ CTMP-SAMPLE($X(0), t_{end}$)                ▷ Generate trajectory
        $r \leftarrow \sum_{i=0}^{t_{i+1}=t_{end}} e^{-\gamma t_{i+1}} \mathcal{C}(X(t_i), X(t_{i+1})) + \int_{t_i}^{t_{i+1}} e^{-\gamma t} \mathcal{R}(X(t_i))dt$    ▷ Estimate the reward of $\sigma$
        $reward+ = r$                ▷ Update the total reward
    end for
    return $reward/iterations$                ▷ Estimate the expected reward
end procedure
```

Now, since the lump sum reward is set to 0, and the instantaneous reward is set to 1 only for $x = sun$, by using the finite horizon approach the expected reward will turn out to be exactly the expected number of sunny days. Applying Algorithm 3 with $\gamma = 0.0$, $t_{end} = 7$, $iterations = 5000$ for all the possible meteorological conditions we obtain the following results, depending on the initial weather condition $X(0)$, i.e., the weather condition at time 0:

$X(0)$	*Expected number of sunny days*
Sun	5.29
Rain	3.69
Storm	3.86

2.5 Discussion

In this chapter we introduced and described the CTMPs framework, together with how to learn a CTMP from data, and how to make inference for a CTMP in the case where no evidence is available. We also introduced the concept of utility function and a Monte Carlo simulation algorithm for estimating the expected reward on a given CTMP. The main strength of CTMPs is their ability to model time as a continuous variable. Thanks to this feature, CTMPs are able to effectively model phenomena that evolve at very different speeds. Furthermore, the creation of a CTMP is event-based; consequently, the resulting trajectory turns out to be extremely compact. The discrete-time counterpart of CTMPs, i.e., Markov Chains, struggle with the discretization issue because they require to sample the process accordingly to the fastest transition. Consequently, in the best case the discretization generates an extremely less compact trajectory and in the worst case it could significantly distort the process modeled.

The main limitation of CTMPs is that they become unmanageable, especially concerning inference, already for domains consisting of small number of states. However, it is important to mention that the following chapter will introduce and describe a specialization of CTMPs which overcomes this limitation.

Chapter 3

Continuous Time Bayesian Networks

3.1 Introduction

Probabilistic graphical models (PGMs) are a class of powerful and expressive models based on the concept of declarative representation. Accordingly to [22], the key property of a declarative representation is the separation between knowledge and reasoning. In this way, we can disentangle the task of model elicitation from the reasoning engine. Such a disentanglement allows accomplishing the model elicitation task by exploiting domain expert knowledge, as well as by using the available data or by combining domain expert knowledge together with the available data. Furthermore, the disentanglement of the reasoning engine, from the model elicitation task, brings the advantage that we can design and develop a suite of general purpose algorithms.

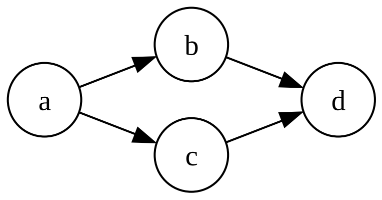
When working with complex systems, the ability to deal with the uncertainty, arising both from simplifications in modeling, and from a natural non-determinism of the studied process, is fundamental. Declarative representations (model-based methods) cover a wider area than PGMs. Indeed, PGMs specialize the concept of declarative representation to model complex systems dealing with uncertainty.

The PGMs framework also provides a tool for taking advantage of the structure in complex probability distributions. There are two equivalent views for interpreting the graph structure of PGMs. The first interpretation highlights the compact representation of a set of independence that hold true for the underlying probability distribution. The second one identifies in the graph the definition of a skeleton for compactly representing the underlying probability distribution. Instead of explicitly modeling each possible assignment for all the considered variables, it is possible to decompose the probability distribution into smaller factors, each one being defined on a smaller number of possible configurations.

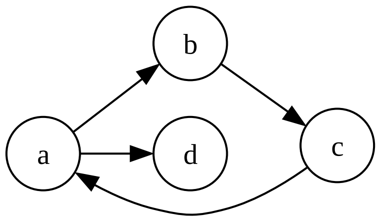
In a probabilistic graphical model (PGM), we can distinguish between the graphical aspect also known as *qualitative* aspect and the probabilistic or *quantitative* aspect. As a matter of fact, the qualitative aspect of a PGM is a graph representing the interactions among a set of variables, where each variable is represented as a vertex in the graph, and the interactions are represented as oriented edges between vertices. The quantitative aspect describes the strength of such interactions.

In this dissertation, we will focus on two kinds of graph, **D**irected **A**cyclic **G**raphs (*DAG*) and **D**irected possibly cyclic **G**raphs.

Graphs Graphs have been used as an intuitive way of representing a set of dependencies and independencies in order to allow an effective communication and discussion. One of the most common class of graph used to describe such set of (in)dependencies is the directed acyclic graph (DAG) (Figure 3.1a). This kind of graph can be seen as a factorized representation of a joint probability distribution, and it is used by many PGMs such as: Bayesian Networks (BN) [23], Influence Diagrams (ID) [24] and Dynamic Bayesian Networks (DBN) [25]. Another common class of graphs that will be explored in this thesis is the one of directed, possibly cyclic graphs (Figure 3.1b). This kind of graph is used as a factorized representation of a joint probability distribution for the model of Continuous Time Bayesian Networks (CTBN)[11].



(a) Directed Acyclic Graph.



(b) Directed Graph with a cycle.

Figure 3.1

Formally, a graph is a pair $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ where $\mathbf{V}$ is a finite set of distinct vertices and $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ is a set of edges. An ordered pair $(a, b) \in \mathbf{E}$ is a directed edge from vertex a to vertex b; a is called *parent* of b and b is called *child* of a. In general, the set of parents of a node v is denoted as $pa(v)$ and the set of children is denoted as $ch(v)$. A directed *path* $\varphi =< v_1, \ldots, v_n >: v_i \in \mathbf{V}$ is a sequence of distinct vertices such that $v_i \in pa(v_{i+1})$ or equivalently $v_i \in ch(v_{i-1})$; given the above definitions, we can introduce the concept of *ancestral set* $an(v) \subseteq \mathbf{V}$ and *descendent set* $de(v) \subseteq \mathbf{V}$. A node v_i is an ancestor of v_j if a path directed from v_i to v_j exists. Similarly, a node v_j is a descendent of v_i if a path directed from v_i to v_j exists. A cycle is a directed path $\varphi =< v_1, \ldots, v_1 >$ where the first node of the path is also the last node of the path. A directed graph with no cycles is called *Directed Acyclic Graph* (DAG).

Probabilities The quantitative aspect of PGMs consists of the probability distribution which models the underlying process of the system under study. While the qualitative aspect (the graph) is similar among the different models, the quantitative aspect can be significantly different, especially between static/discrete time models i.e. BN, DBN, ID and continuous time models i.e. CTBN, Continuous Time Markov Process (CTMP) [26]. If we take into account only discrete variables over a finite domain, we can clearly see the difference between static/discrete time models and continuous time models. The quantitative aspect of BNs and IDs for discrete variables is based on conditional probability tables (CPTs) which are used to describe the joint probability of observing a specific configuration of the network. Also DBNs exploit the concept of CPTs. However, in DBNs and especially in 2TDBNs, the variables can be represented more than one time, and each representation is associated with different times. The quantitative aspect of CTMPs and CTBNs, instead of using CPTs, relies on conditional intensity matrices (CIMs) that describe the evolution of the process over time by modeling when a transition occurs and what will be the new state after the transition occurred.

3.2 Conditional Continuous Time Markov Processes

A CTMP is a homogeneous Markov process; this means that the next state only depends on the current state. Unfortunately, such process does not exploit the factorization property of Bayesian Networks, and thus it does not enjoy a synthetic and expressive state space representation. However, conditional Markov processes allow overcoming such a limitation [11].

A *conditional Markov process* is an inhomogeneous Markov process in which, for any given random variable, the intensities are a function of the current state of a particular set of other variables, which also evolve as Markov processes. Therefore, intensities vary over time, but not as a function of time.

To clarify how a conditional Markov process is described, let X be a random process whose domain is $dom(X) = \{x_1, \ldots, x_m\}$ and assume that it evolves as a Markov process $X(t)$. Furthermore, assume that the dynamics of $X(t)$ are conditionally dependent from a set $\mathbf{U}$ of random variables evolving over time. Then the dynamics of $X(t)$ can be fully described by means of a conditional intensity matrix

(CIM), which can be written as follows:

$$Q_{X\,|\,\mathbf{u}} = \begin{bmatrix} -q_{x_1\,|\,\mathbf{u}} & q_{x_1x_2\,|\,\mathbf{u}} & \cdots & q_{x_1x_m\,|\,\mathbf{u}} \\ q_{x_2x_1\,|\,\mathbf{u}} & -q_{x_2\,|\,\mathbf{u}} & \cdots & q_{x_2x_m\,|\,\mathbf{u}} \\ \vdots & \vdots & \ddots & \vdots \\ q_{x_mx_1\,|\,\mathbf{u}} & q_{x_mx_2\,|\,\mathbf{u}} & \cdots & -q_{x_m\,|\,\mathbf{u}} \end{bmatrix}.$$

A CIM is a set of intensity matrices, one intensity matrix for each instance of values $\mathbf{u}$ to the set of variables $\mathbf{U}$.

The diagonal elements of Q_X are such that $q_{x_i} = \sum_{x_j \neq x_i} q_{x_ix_j}$, where q_{x_i} is the parameter of the exponential distribution associated with the state x_i of the variable X. Therefore, $1/q_{x_i}$ is the expected amount of time that variable X stays in state x_i before transitioning to a different state x_j when $\mathbf{U} = \mathbf{u}$. The off-diagonal elements $q_{x_ix_j}$ are proportional to the probability that X transitions from state x_i to state x_j when $\mathbf{U} = \mathbf{u}$.

The process X can be equivalently summarized with two independent sets of parameters:

- $\mathbf{q}_X = \{q_{x_i|\mathbf{u}}, \forall x_i \in dom(X)\}$, the set of intensities of the exponential distributions of the *waits until the next transition.*
- $\boldsymbol{\theta}_X = \{\theta_{x_ix_j|\mathbf{u}} = q_{x_ix_j|\mathbf{u}}/q_{x_i|\mathbf{u}}, \forall x_i, x_j \in dom(X_k), x_i \neq x_j\}$, the *probabilities of transitioning to specific states.*

The *log-likelihood* for a conditional CTMP can be derived from Equation 2.12. as follows:

$$LL_X = \sum_{\mathbf{u} \in \mathbf{U}} \left(-\sum_s T[s|\mathbf{u}] q_{X_s|\mathbf{u}} + \sum_{s \neq s'} N[s, s'|\mathbf{u}] \ln q_{s,s'} \right) \qquad s, s' \in dom(X) \tag{3.1}$$

where T and N are the conditional sufficient statistics:

- $T[s|\mathbf{u}]$ represents the residence time of the process in the state $s \in dom(X)$ when $\mathbf{u} \in \mathbf{U}$.
- $N[s, s'|\mathbf{u}]$ represents the number of transitions from state $s \in dom(X)$ to state $s' \in dom(X)$ with $s \neq s'$ when $\mathbf{u} \in \mathbf{U}$.

3.3 Continuous Time Bayesian Network

Continuous Time Bayesian Networks (CTBNs) are a class of probabilistic graphical models combining Bayesian networks [22] and CTMPs to model discrete-state continuous-time dynamical systems [11]. Just as CTMPs overcome the problems related to the discretization of time for MCs, similarly CTBNs solve the problem of having to discretize time in Dynamic Bayesian Networks. Furthermore, CTBNs exploit the natural factorization of many phenomena, thus allowing for a more compact representation with respect to the CTMPs counterpart.

Other models that can represent discrete-state continuous time processes include Poisson networks [27], cascade of Poisson process model [28], piecewise-constant intensity models [29], forest-based point processes [30], and graphical models for marked point processes [31]. The main advantages of CTBNs, when compared to the models above, are:

- their graphical structure makes it possible to understand and explain the underlying stochastic process they model,
- prior knowledge from domain experts can be integrated in structure learning by setting a list of forbidden arcs and a list of included arcs.

However, CTBNs have a number of limitations: they do not allow modeling continuous state variables; they do not model point events very well, particularly if there are non-temporal values associated with the events; structure learning and inference are computationally challenging.

Example 3.1 (Hungry? - Eating? - Full Stomach?)**.** This example shows how to use a CTBN to model the process that governs being hungry. Figure 3.2 depicts the structure of a simple CTBN, which models the effect that eating has on the content of the stomach of an individual. Furthermore, the content of the stomach has an effect on whether the individual is hungry or not, which in turn has an effect on the individual to start eating.

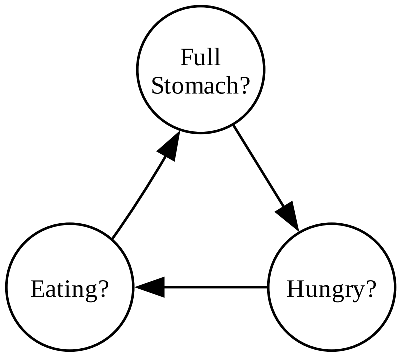


Figure 3.2: CTBN Example - Eating? Hungry? Full Stomach?

3.4 Formal Definition

A CTBN is a tuple $\mathcal{N} = \langle \mathcal{P}_0, \mathbf{X}, \mathcal{G}, \boldsymbol{Q} \rangle$ specified by:

- An initial probability distribution $\mathcal{P}_0$, given as a Bayesian network over $\mathbf{X}$.
- A set of L local variables $\mathbf{X} = \{X_1, \ldots, X_L\}$ such that each variable $X_j \in \mathbf{X}$ takes value on a finite domain $dom(X_j)$.
- A continuous-time transition model, specified as:
  - a directed (possibly cyclic) graph $\mathcal{G}$ with node set $\mathbf{X}$;
  - a set of conditional intensity matrices $\boldsymbol{Q}$ for each process $X_j \in \mathbf{X}$.

Differently from classical BNs, a CTBNs allow for cycles in the graph $\mathcal{G}$. Each edge of the graph $\mathcal{G}$ has a temporal connotation, therefore the evolution of the state of a variable X_i can be directly influenced by the state of a variable X_j and, at the same time, the state of the variable X_i can directly influence the evolution of the state of the variable X_j.

Since CTBNs are a probabilistic graphical model designed to represent the evolution of a process over time, by far the most important component is the dynamic one. On the contrary, the BN describing the initial probability distribution $\mathcal{P}_0$ can be in general safely ignored and replaced with a non-informative distribution, such as the uniform one.

3.4.1 Generative Semantic

The fundamental assumption made by CTBNs, to exploit the factorization of the state space, is that two variables cannot transition at the same time. This assumption is justified because the variables in a CTBN represent distinct aspects of the world.

Similarly to a CTMP the realization of a CTBN is a trajectory and can be formally described as follows:

$$\sigma = \{\langle t_0, \mathbf{s}_0 \rangle, \langle t_1, \mathbf{s}_1 \rangle, ..., \langle t_I, \mathbf{s}_I \rangle\}, \quad t_0 < t_1 < \cdots < t_I \text{ and } \mathbf{s}_i \in \bigtimes_{X_j \in \mathbf{X}} dom(X_j) \;\; \forall i \in (0, I). \tag{3.2}$$

where $\times$ denotes the Cartesian product.

Starting from a CTBN $\mathcal{N}$, and exploiting the assumption just stated, it is possible to specify a generative process through a procedure that takes the initial state of the CTBN and generates a trajectory.

Algorithm 4 represents for CTBNs what Algorithm 1 represents for CTMPs. The main difference is that the algorithm for CTBNs (Algorithm 4) exploits the factorization of the process by enforcing that only a single variable at a time can transition.

Figure 3.3 depicts the state space for the Example 3.1 where the nodes represent the possible configurations of the CTBN and the edges represent the allowed transitions (only one variable at a time can change its state).

Algorithm 4 Forward Sampling for CTBN.

```
procedure CTBN-SAMPLE($\mathbf{X}(0)$, $t_{end}$)
    $t \leftarrow 0$, $\sigma \leftarrow \{\langle 0, \mathbf{X}(0) \rangle\}$                               ▷ Initialization
    loop
        for all $X_i \in \mathbf{X}$ s.t. $Time(X_i)$ is undefined do          ▷ Time to transition sampling
            $\Delta t \leftarrow$ draw a sample from an exponential with rate $q_{X_i(t)|pa(X_i(t))}$
            $Time(X_i) \leftarrow t + \Delta t$
        end for
        $j = \arg\min_{X_i \in \mathbf{X}} [Time(X_i)]$                          ▷ Transitioning variable
        if $Time(X_j) > t_{end}$ then
            Add $\langle t_{end}, X(t) \rangle$ to $\sigma$
            return $\sigma$
        end if
        $x_j \leftarrow$ draw a sample from a multinomial with $\theta_{X_j(t)|pa(X_j(t))}$     ▷ Next state
        $t \leftarrow Time(X_j)$
        Add $\langle t, X(t) \rangle$ to $\sigma$
        Undefine $Time(X_j)$ and $Time(X_i)$ $\forall X_i \in pa(X_j)$          ▷ Reset the time to transition
    end loop
    return $\sigma$
end procedure
```

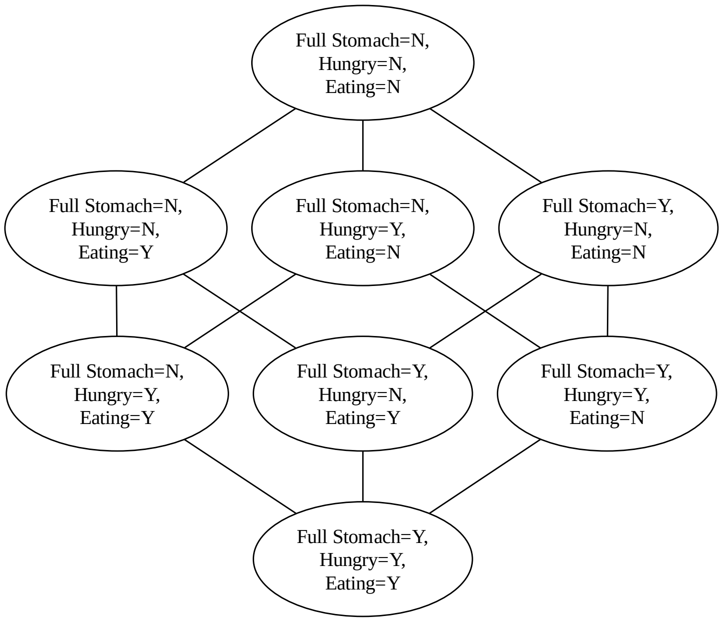

Full Stomach=N, Hungry=N, Eating=N
Full Stomach=N, Hungry=N, Eating=Y
Full Stomach=N, Hungry=Y, Eating=N
Full Stomach=Y, Hungry=N, Eating=N
Full Stomach=N, Hungry=Y, Eating=Y
Full Stomach=Y, Hungry=N, Eating=Y
Full Stomach=Y, Hungry=Y, Eating=N
Full Stomach=Y, Hungry=Y, Eating=Y

Figure 3.3: State Space CTBN Example - Eating? Hungry? Full Stomach? - Nodes represent states of the CTBN, two states are adjacent if a transition from one to the other, and vice versa, is possible.

3.5 Parameter Learning

Since a CTBN is composed by L conditional CTMPs, we can compute its *log-likelihood* by combining the log-likelihoods of different CTMPs as follows:

$$LL_{\mathcal{N}} = \sum_{X \in \mathbf{X}} LL_X \tag{3.3}$$

Analogously to the approach described in Section 2.2 for CTMP, it is possible to learn in closed form the parameters of a CTBN when a set of trajectories are available. Accordingly to what presented in [11], the log-likelihood of a CTBN (Equation 3.3) is maximized by the following MLE approach for the parameters associated with each variable $X_j \in \mathbf{X}$.

$$\hat{q}_{x,x'|\mathbf{u}} = \frac{N[x,x'|\mathbf{u}]}{T[x|\mathbf{u}]} \quad \hat{q}_{x|\mathbf{u}} = \sum_{x \neq x'} \frac{N[x,x'|\mathbf{u}]}{T[x|\mathbf{u}]} \qquad \forall x,x' \in dom(X_j),\ x \neq x',\ \mathbf{u} \in dom(\mathbf{U}_j) \tag{3.4}$$

where $\mathbf{U}_j = pa(X_j)$ is the parentset of X_j. Alternatively, we can learn q and θ parameters separately, according to the following:

$$\hat{q}_{x|\mathbf{u}} = \sum_{x \neq x'} \frac{N[x,x'|\mathbf{u}]}{T[x|\mathbf{u}]} \quad \hat{\theta}_{x,x'|\mathbf{u}} = \frac{N[x,x'|\mathbf{u}]}{\sum_{x \neq x''} N[x,x''|\mathbf{u}]} \qquad \forall x,x',x'' \in dom(X_j),\ x \neq x',\ \mathbf{u} \in dom(\mathbf{U}_j) \tag{3.5}$$

The literature also presents a MAP or Bayesian approach to learn the parameters [18]. This approach is useful when the availability of data is limited and allows to define the Bayesian score required in the score based structure learning. The prior distribution introduced is a Gamma distribution for each parameter $q_{x,x'|\mathbf{u}}$. The hyper-parameters for the prior distribution can be interpreted as follows:

- $\alpha_{xx'|\mathbf{u}}$: is an imaginary count of transitions from x to x' when the parentset is such that $\mathbf{u} \in dom(\mathbf{U})$.
- $\tau_{x|\mathbf{u}}$: is an imaginary residence time for the state x when the parentset is such that $\mathbf{u} \in dom(\mathbf{U})$.

The parameters for each $X_j \in \mathbf{X}$ can be estimated following the MAP approach by using the following equations:

$$\begin{aligned} \hat{q}_{x,x'|\mathbf{u}} &= \tfrac{N[x,x'|\mathbf{u}]+\alpha_{xx'|\mathbf{u}}}{T[x|\mathbf{u}]+\tau_{x|\mathbf{u}}} \quad \forall x,x' \in dom(X_j),\ x \neq x',\ \mathbf{u} \in dom(\mathbf{U}_j) \\ \hat{q}_{x|\mathbf{u}} &= \textstyle\sum_{x' \neq x} \tfrac{N[x,x'|\mathbf{u}]+\alpha_{xx'|\mathbf{u}}}{T[x|\mathbf{u}]+\tau_{x|\mathbf{u}}} \qquad \forall x \in dom(X_j),\ \mathbf{u} \in dom(\mathbf{U}_j) \end{aligned} \tag{3.6}$$

As it happens for CTMPs, MAP estimation for CTBNs can be obtained by an alternative approach, i.e., by decomposing the intensity matrix according to the two processes modelled by $\mathbf{q}$ and $\boldsymbol{\theta}$. For each variable $X_i \in \mathbf{X}$ we use two classes of prior: the gamma distribution for $\mathbf{q}_{X_i}$ and the Dirichlet distribution for $\boldsymbol{\theta}_X$. Specifically:

- For each exponential parameter $q_x \in \mathbf{q}$ a gamma distribution with parameter $\alpha_{x|\mathbf{u}}$ and $\tau_{x|\mathbf{u}}$ is used.
- For each multinomial distribution (one for each row of $\boldsymbol{\theta}_{X_i}$) a Dirichlet distribution with parameters $\alpha_{x,x'} \forall x \neq x'$ *s.t.* $\sum_{x'} \alpha_{x,x'} = \alpha_x$

Using this formulation the parameters can be estimated as follows:

$$\begin{aligned} \hat{q}_{x|\mathbf{u}} &= \tfrac{\alpha_{x|\mathbf{u}}+\sum_{x \neq x'} N[x,x'|\mathbf{u}]}{\tau_{x|\mathbf{u}}+T[x|\mathbf{u}]} \quad \forall x,x' \in dom(X_j),\ x \neq x',\ \mathbf{u} \in dom(\mathbf{U}_j) \\ \hat{\theta}_{x,x'|\mathbf{u}} &= \tfrac{\alpha_{x,x'|\mathbf{u}}+N[x,x'|\mathbf{u}]}{\alpha_{x|\mathbf{u}}+\sum_{x \neq x''} N[x,x''|\mathbf{u}]} \qquad \forall x,x',x'' \in dom(X_j),\ x \neq x',\ \mathbf{u} \in dom(\mathbf{U}_j) \end{aligned} \tag{3.7}$$

3.6 Structure Learning

The structure of a CTBN, i.e., the graph $\mathcal{G}$, it is almost always unknown, unless we perfectly know the system to represent and/or we can resort to domain expert knowledge. Starting from the correct graph $\mathcal{G}$ is a fundamental assumption for the CTBN to be effective when studying a given system. Indeed, an incorrect CTBN's structure leads, under the most favorable setting, to an inefficient representation of the system under study, i.e., one where more arcs and parameters have to be used, but almost always it results into the wrong model of the system we want to study.

The above mentioned reasons make the problem of recovering the correct structure $\mathcal{G}$ of the CTBN to be an extremely challenging problem which, accordingly to the specialized literature, can be tackled by using one of the following three approaches:

1. Knowledge elicitation; using prior knowledge made available by a domain expert, i.e., an expert of the domain to be modeled and analyzed.
2. structure learning algorithm; learning happens by only using the available data, i.e., the available trajectories.
3. Combining the above approaches; using a structure learning algorithm which allow to combine the available data with the knowledge elicited from the domain expert.

In this subsection we focus on the second approach, i.e., the one which learns the graph $\mathcal{G}$ by using only the available trajectories while ignoring any prior knowledge, by assuming such knowledge is not made available to the learning algorithm. According to the specialized literature, the task of learning the graph $\mathcal{G}$ using only the available data can be accomplished by two main types of learning algorithms:

- Score Based.
- Constraint Based.

The score based algorithm presented in [11] casts the problem of learning the graph $\mathcal{G}$ of a CTBN in an optimization problem, i.e., a problem where we must find the graph $\mathcal{G}^*$ which, given a trajectory σ, maximizes the following posterior:

$$\ln P(\mathcal{G} \mid \sigma) \propto \ln P(\mathcal{G}) + \ln P(\sigma \mid \mathcal{G}) \tag{3.8}$$

where $P(\mathcal{G})$ is the *prior distribution over the space of graphs* spanning $\mathbf{X}$ and $P(\sigma \mid \mathcal{G})$ is the *marginal likelihood* of the data, i.e., the trajectory, given the graph $\mathcal{G}$ averaged over all possible parameter values. The simplest algorithm which is typically used to solve the learning problem is the hill climbing algorithm.

On the other hand, the class of constraint based algorithms (extensively presented in Chapters 4 and 5) is based on hypothesis testing with the aim to identify conditional dependencies and conditional independencies among the variables $\mathbf{X}$.

3.7 Reward Factorization

In order to exploit the factorization of a CTBN also with the introduction of a reward function, it is fundamental that also the reward function is factorized. In [32], the authors proposed to introduce an instantaneous reward function for each node. The authors of [33] present a factorization function capable of modeling both instantaneous and lump-sum rewards. Accordingly to these works, for each node $X_j \in \mathbf{X}$ of a CTBN the following quantities are introduced:

- $\mathcal{R}_{X_j}(x) : dom(X_j) \to \mathbb{R}$, the *instantaneous reward* for staying in state x and
- $\mathcal{C}_{X_j}(x, x') : dom(X_j) \times dom(X_j) \to \mathbb{R}$, the *lump sum reward* when X_j transitions from state x to state x'.

The authors of [33] allows for interactions among the variables introducing the synergy nodes.

However, in this work, we assumed that there is no interaction between the reward functions of the variables. Therefore, we can derive the equations for the estimation of the expected reward from Section 2.4 as follows. In the *Finite-horizon expected reward* approach, we evaluate the reward function up to a specific point in time:

$$V_{t_{end}}(\mathbf{x}) = \mathbb{E}_{\mathcal{N}_\mathbf{x}}\left[\sum_{i=0}^{t_{i+1}=t_{end}} \sum_{X_j \in \mathbf{X}} \left(\mathcal{C}(X_j(t_i), X_j(t_{i+1})) + \int_{t_i}^{t_{i+1}} \mathcal{R}(X_j(t_i))dt\right)\right] : t_i < t_{i+1} \tag{3.9}$$

where t_{end} is the ending time and $\mathbb{E}_{\mathcal{N}_\mathbf{x}}(\cdot)$ is the expectation when conditioning on $\mathbf{X}(0) = \mathbf{x}$, with $\mathbf{x} \in \times_{X_j \in \mathbf{X}} dom(X_j)$ and where the t_i's are the transition times.

The *Infinite-horizon expected discounted reward* approach does not define any time horizon, but to ensure convergence, it relies on a discount factor:

$$V_\gamma(\mathbf{x}) = \mathbb{E}_{\mathcal{N}_\mathbf{x}}\left[\sum_{i=0}^{\infty} \sum_{X_j \in \mathbf{X}} \left(e^{-\gamma t_{i+1}}\mathcal{C}(X_j(t_i), X_j(t_{i+1})) + \int_{t_i}^{t_{i+1}} e^{-\gamma t}\mathcal{R}(X_j(t_i))dt\right)\right] : t_i < t_{i+1} \tag{3.10}$$

where $\gamma > 0$ is referred to as the *discounting factor.*

It is worthwhile to mention that the lump sum reward can be used as an indicator of transition. In order to achieve such goal we and let the instantaneous reward be zero while the lump sum reward is configured as follows:

$$\mathcal{C}_{X_j}(x, x') = 1, \quad \forall x \neq x' \in dom(X_j)\ \forall X_j \in \mathbf{X} \tag{3.11}$$

This uses of the lump sum reward will be further discussed in Chapter 6.

3.8 Multidimensional CTBN Classifier

The work [34] presents an extension of the CTBN, *Continuous Time Bayesian Network Classifier* (CTBNC), a class of supervised classification models. With a CTBNC, it is possible to classify a trajectory introducing a new variable Y (the class node) such that the new model can be described as a touple $<\mathcal{N}, P(Y)>$ and satisfy the following characteristics:

- $\mathcal{G}$ is connected
- $Pa(Y) = \varnothing$: the class variable is associated with a root node in the graph $\mathcal{G}$.
- Y does not depend on time.

The paper [35] generalize the CTBNC developing a *Multidimensional CTBNC* (Multi-CTBNC). A Multi-CTBNC is a tuple $\mathcal{M} =< \boldsymbol{\mathcal{X}}, \mathbf{X}, \mathcal{G}, \mathcal{P}, \boldsymbol{Q} >$ where:

- $\boldsymbol{\mathcal{X}}$: is the set of discrete class variables. These variables do not depend on time.
- $\mathbf{X}$: is the set of continuous time discrete random variables modelling the process over time.
- $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a directed possibly cyclic graph where:
  - $\mathbf{V} = \boldsymbol{\mathcal{X}} \cup \mathbf{X}$ is the set of the graph.
  - $\mathbf{E}_{\boldsymbol{\mathcal{X}}} \subset \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{X}}$ is the set of edges representing the relations among the class variables.
  - $\mathbf{E}_\mathbf{X} \subseteq \mathbf{X} \times \mathbf{X}$ is the set of edges representing the relations among the continuous time discrete random variables.
  - $\mathbf{E}_\mathbf{B} \subseteq \boldsymbol{\mathcal{X}} \times \mathbf{X}$: is the set of edges representing the relation from the class variables $\boldsymbol{\mathcal{X}}$ to the continuous time variables $\mathbf{X}$.
  - $\mathbf{E} = \mathbf{E}_{\boldsymbol{\mathcal{X}}} \cup \mathbf{E}_\mathbf{X} \cup \mathbf{E}_\mathbf{B}$
- $\mathcal{P}$: is the set of conditional probability tables for the class variables $\boldsymbol{\mathcal{X}}$.
- $\boldsymbol{Q}$: is the set of CIM for the continuous time variables $\mathbf{X}$.

The class variables $\boldsymbol{\mathcal{X}}$ do not depend on time, thus the Multi-CTBNC models deals with them as a classical BN. Therefore, we can split a Multi-CTBNC in three components:

- The *class subgraph* $\mathcal{G}_{\boldsymbol{\mathcal{X}}} = (\boldsymbol{\mathcal{X}}, \mathbf{E}_{\boldsymbol{\mathcal{X}}})$ is the component modelled by a BN describing the relation among the class variables. As a result, the graph must be a DAG.
- The *feature subgraph* $\mathcal{G}_{\mathbf{X}} = (\mathbf{X}, \mathbf{E}_{\mathbf{X}})$ is defined by a CTBN that model the relations between feature variables. Since it is a CTBN, cycles may appear.
- The *bridge subgraph* $\mathcal{G}_{\mathbf{B}} = (\mathbf{V}, \mathbf{E}_{\mathbf{B}})$ represents the dependencies of the features variables from the class variables. Since all the edges in start from a variable $c \in \boldsymbol{\mathcal{X}}$ and end in a variable $x \in \mathbf{X}$, it is a bipartite graph.

3.9 Discussion

This chapter, in combination with Chapter 2, introduces a set of minimal notions required to better understand the contributions presented in Chapter 4, 5 and 6. Specifically, in this chapter we presented the CTBN model, which belongs to the class of PGMs and is capable to model the evolution of a process over time by exploiting the factorization of the associated state space. We also introduced the concept of reward function of a CTBN and the possibility of using a CTBN as a multidimensional classifier.

Similarly to the CTMP, the CTBN can also model the evolution of a process without the need to choose a temporal granularity. The structure of the CTBN, that allows the factorization while enforcing the conditional independence among the variables, results into a directed possibly cyclic graph. This allows us to have an insight about the behavior of the model while allowing expert knowledge to be easily incorporated.

Chapter 4

A Constraint-Based Algorithm for Structure Learning

Learning the structure of a BN is a problem that is well explored in the literature. Several approaches have been proposed, spanning score-based, constraint-based and hybrid algorithms; recent reviews are available from [36], [37]. Score-based algorithms find the BN structure that maximizes a given score function, while constraint-based algorithms use statistical tests to learn conditional independence relationships (called *constraints*) from the data and infer the presence or absence of particular arcs. Hybrid algorithms combine aspects of both score-based and constraint-based algorithms. It is worthwhile to note that CTBNs are a special case of the local independence model [31],[38],[39] for which a general structure learning algorithm has been made available. In particular, [40] proposed a structure learning algorithm for local independence graphs. In their work, the authors studied independence models induced by directed graphs (DGs) by formalizing the properties of abstract graphoids that ensure that the global Markov property holds for a given directed graph. [40] applied their theoretical arguments to the Ito diffusion as well as the event process, which are both related to CTBNs. Therefore, the algorithm we propose in [15] can be considered as an instance of that presented by [40], even if we independently formulated and developed our structure learning algorithm that is specifically designed for CTBNs. After a brief introduction to constraint-based algorithms for BNs, we propose such an algorithm for CTBNs.

4.1 Constraint-Based Algorithms for BNs

Constraint-based algorithms for BN structure learning originate from the *Inductive Causation* (IC) algorithm from [41] for learning causal networks. IC starts (step 1) by finding pairs of nodes connected by an undirected arc, as those are not independent given any other subset of variables. The second step (step 2) identifies the v-structures $\mathcal{X}_i \rightarrow \mathcal{X}_k \leftarrow \mathcal{X}_j$ among all pairs $\mathcal{X}_i$ and $\mathcal{X}_j$ of non-adjacent nodes which share a common neighbor $\mathcal{X}_k$. Finally, step 3 and 4 of IC identify compelled arcs and orient them to build the completed partially oriented DAG (CPDAG) that describes the *equivalence class* the BN falls into.

However, steps 1 and 2 of the IC algorithm are computationally unfeasible for non-trivial problems due to the exponential number of conditional independence relationships to be tested.

The *PC algorithm*, which is briefly illustrated in Algorithm 5, was the first proposal addressing this issue; its modern incarnation is described in [42], and we will use it as the foundation for CTBN structure learning below. PC starts from a fully-connected undirected graph. Then, for each pair of variables $\mathcal{X}_i$, $\mathcal{X}_j$ it proceeds by gradually increasing the cardinality of the set of conditioning nodes $\mathbf{U}_{\mathcal{X}_i\mathcal{X}_j}$ until $\mathcal{X}_i$ and $\mathcal{X}_j$ are found to be independent or $\mathbf{U}_{\mathcal{X}_i\mathcal{X}_j} = \boldsymbol{\mathcal{X}} \setminus \{\mathcal{X}_i, \mathcal{X}_j\}$. The remaining steps are identical to those of IC.

Neither IC nor PC (or other constraint-based algorithms, for that matter) require a specific test statistic to test conditional independence, making them independent from the distributional assumptions we make on the data.

Algorithm 5 PC Algorithm

```
1. Form the complete undirected graph $\mathcal{G}$ on the vertex set $\boldsymbol{\mathcal{X}}$.
2. For each pair of variables $\mathcal{X}_i, \mathcal{X}_j \in \boldsymbol{\mathcal{X}}$, consider all the possible separating set, from the smallest ($\mathbf{U}_{\mathcal{X}_i\mathcal{X}_j} = \varnothing$) to the largest ($\mathbf{U}_{\mathcal{X}_i\mathcal{X}_j} = \boldsymbol{\mathcal{X}}\backslash\{\mathcal{X}_i, \mathcal{X}_j\}$). If there is not any set $\mathbf{U}_{\mathcal{X}_i\mathcal{X}_j}$ such that, $\mathcal{X}_i \perp\!\!\!\perp \mathcal{X}_j|\mathbf{U}_{\mathcal{X}_i\mathcal{X}_j}$ then the edge $\mathcal{X}_i - \mathcal{X}_j$ is removed from $\mathcal{G}$.
3. For each triple $\mathcal{X}_i, \mathcal{X}_j, \mathcal{X}_k \in \mathcal{G}$ such that $\mathcal{X}_i - \mathcal{X}_j$, $\mathcal{X}_j - \mathcal{X}_k$, and $\mathcal{X}_i$, $\mathcal{X}_j$ are not connected, orient the edges into $\mathcal{X}_i \rightarrow \mathcal{X}_j \leftarrow \mathcal{X}_k$ if and only if $\mathcal{X}_j \notin \mathbf{U}_{\mathcal{X}_i\mathcal{X}_j}$ for every $\mathbf{U}_{\mathcal{X}_i\mathcal{X}_j}$ that makes $\mathcal{X}_i$ and $\mathcal{X}_k$ independent.
4. The algorithm identifies the compelled directed arcs by iteratively applying the following two rules:
    (a) if $\mathcal{X}_i$ is adjacent to $\mathcal{X}_j$ and there is a strictly directed path from $\mathcal{X}_i$ to $\mathcal{X}_j$ then replace $\mathcal{X}_i - \mathcal{X}_j$ with $\mathcal{X}_i \rightarrow \mathcal{X}_j$ (to avoid introducing cycles);
    (b) if $\mathcal{X}_i$ and $\mathcal{X}_j$ are not adjacent but $\mathcal{X}_i \rightarrow \mathcal{X}_k$ and $\mathcal{X}_k - \mathcal{X}_j$, then replace the latter with $\mathcal{X}_k \rightarrow \mathcal{X}_j$ (to avoid introducing new v-structures).
5. Return the resulting CPDAG $\mathcal{G}$.
```

4.2 The CTPC Structure Learning Algorithm

CTBNs differ from BNs in three fundamental ways: BNs do not model time, while CTBNs do; BNs are based on DAGs, while CTBNs allow cycles; and BNs model the dependence of a node on its parents using a conditional probability distribution, while CTBNs model it using a CIM. These differences make structure learning a simpler problem for CTBNs than it is for BNs.

Firstly, learning arc directions is an issue in BNs but not in CTBNs, where arcs are required to follow the arrow of time. Unlike BNs, which can be grouped into equivalence classes that are probabilistically indistinguishable, each CTBN has a unique minimal graphical representation [43]. For instance, let a CTBN $\mathcal{N}$ have graph $\mathcal{G} = \{X \rightarrow X'\}$: unless trivially X and X' are marginally independent, $\mathcal{G}$ cannot generate the same transition probabilities as any CTBN $\mathcal{N}'$ with graph $\mathcal{G}' = \{X \leftarrow X'\}$.

Secondly, in CTBNs we can learn each parent set $pa(X_k)$ in isolation, thus making any structure learning algorithm embarrassingly parallel. Acyclicity imposes a global constraint on $\mathcal{G}$ that makes it impossible to do the same in BNs.

Thirdly, each variable X_k is modelled conditional on a given function of its parent set $pa(X_k)$: a conditional probability table for (discrete) BNs, a CIM for CTBNs.

However, a CIM $Q_{X_k|\mathbf{U}}$ describes the temporal evolution of the state of variable X_k conditionally on the state of its parent set $\mathbf{U}$. Hence, we can not test conditional independence by using classical test statistics like the mutual information or Pearson's χ^2 that assume observations are independent [22]. Instead, we need to adapt our definition of conditional independence to CTBNs in order to design a constraint-based algorithm for structure learning.

Definition 4.1. Conditional Independence in a CTBN

Let $\mathbf{X}$ be a set of variables modelled as conditional CTMP. We say that X_i is conditionally independent from X_j given $\mathbf{U}_{\mathcal{X}_i\mathcal{X}_j} \subseteq \mathbf{X}\backslash\{X_i, X_j\}$ if

$$Q_{X_i\,|\,x,\mathbf{u}} = Q_{X_i\,|\,\mathbf{u}} \qquad \forall\, x \in dom(X_j), \forall\, \mathbf{u} \in dom(\mathbf{U}_{X_iX_j}). \tag{4.1}$$

where $Q_{X_i\,|\,X_j,\mathbf{U}_{\mathcal{X}_i\mathcal{X}_j}}$ is the CIM for the variable X_i given the separation set $\mathbf{U}_{\mathcal{X}_i\mathcal{X}_j} \cup \{X_j\}$ and $Q_{X_i\,|\,\mathbf{U}_{\mathcal{X}_i\mathcal{X}_j}}$ is the CIM for the variable X_i given the separation set $\mathbf{U}_{\mathcal{X}_i\mathcal{X}_j}$. If $\mathbf{U}_{X_iX_j} = \varnothing$, then X_i is said to be marginally independent from X_j.

It is important to note that Definition 4.1 is not symmetric: it is perfectly possible for X_i to be conditionally or marginally independent from X_j, while X_j is not independent from X_i. This discrepancy is, however, not a practical or theoretical concern because arcs are already non-symmetric (they must follow the direction of time) and therefore we only test whether X_i depends on X_j if X_j precedes X_i and not the other way round.

As for the test statistics, we can test for conditional independence using $q_{X_k|\mathbf{u}}$ (the waiting times) and, if we do not reject the null hypothesis of conditional independence, we can perform a further test using $\theta_{X_k|\mathbf{u}}$ (the transitions).

Note that conditional independence can be established by testing only the waiting times $q_{X_k|\mathbf{u}}$ if the CTBN contains only binary variables because the transition is deterministic for a binary node. Instead, in the general case in which variables can take more than two values, testing for conditional independence involves both waiting times and transitions.

Since we consider that rates are the most important characteristic to assess in a stochastic process, we decide without loss of generality to test $q_{X_k|\mathbf{u}}$ first, and then $\theta_{X_k|\mathbf{u}}$.

For $q_{X_k|\mathbf{u}}$, we define the *null time to transition hypothesis* as follows.

Definition 4.2. Null Time To Transition Hypothesis
Given X_i, X_j and the conditioning set $\mathbf{U}_{X_iX_j} \subseteq \mathbf{X}\backslash\{X_i, X_j\}$, the null time to transition hypothesis of X_j over X_i is

$$q_{x|x',\mathbf{u}} = q_{x|\mathbf{u}} \qquad \forall\, x \in dom(X_i), \forall\, x' \in dom(X_j), \forall\, \mathbf{u} \in dom(\mathbf{U}_{X_iX_j}).$$

For $\theta_{X_k|\mathbf{u}}$, we define the *null state-to-state transition hypothesis* as follows.

Definition 4.3. Null State-To-State Transition Hypothesis
Given X_i, X_j and the conditioning set $\mathbf{U}_{X_iX_j} \subseteq \mathbf{X}\backslash\{X_i, X_j\}$, the null state-to-state transition hypothesis of X_j over X_i is

$$\theta_{x\cdot|x',\mathbf{u}} = \theta_{x\cdot|\mathbf{u}} \qquad \forall\, x \in dom(X_i), \forall\, x' \in dom(X_j), \forall\, \mathbf{u} \in dom(\mathbf{U}_{X_iX_j})$$

Definition 4.2 characterizes conditional independence for the times to transition for variable X_i when adding (or not) X_j to its parents; Definition 4.3 characterizes conditional independence for the transitions of X_i when adding (or not) X_j to its parents.

To test the *null time to transition hypothesis*, we use the F test to compare two exponential distributions from [44]. In the case of CTBNs, the test statistic and the degrees of freedom take form

$$F_{r_1,r_2} = \frac{q_{x\ \mathbf{u}}}{q_{x|x',\mathbf{u}}}, \qquad \text{with} \qquad r_1 = \sum_{y\in dom(X_i)} N[x, y|x', \mathbf{u}], \qquad r_2 = \sum_{y\in dom(X_i)} N[x, y|\mathbf{u}]. \tag{4.2}$$

To test the *null state-to-state transition hypothesis*, we investigated the use of the *two-sample chi-square* and *Kolmogorov-Smirnov* tests [45]. For CTBNs the former takes form:

$$\chi^2 = \sum_{x'\in dom(X_i)} \frac{(K \cdot N[x, y|x'\mathbf{u}] - L \cdot N[x, y|\mathbf{u}])^2}{N[x, y|x', \mathbf{u}] + N[x, y|\mathbf{u}]}, \tag{4.3}$$

$$K = \sqrt{\frac{\sum_{y\in dom(X_i)} N[x, y|\mathbf{u}]}{\sum_{y\in dom(X_i)} N[x, y|x', \mathbf{u}]}}, L = \frac{1}{K}\ , \tag{4.4}$$

and is asymptotically distributed as a $\chi^2_{|dom(X_i)|-1}$. The latter is defined as

$$D_{r_1,r_2} = \sup_{y\in dom(X_i)} \left|\Theta_{xy|\mathbf{u}} - \Theta_{xy|x',\mathbf{u}}\right|, \qquad \Theta_{xy} = \sum_{\substack{z\in dom(X_i)\\ z\leqslant y}} \theta_{xz}. \tag{4.5}$$

After characterizing conditional independence, we can now introduce our constraint-based algorithm for structure learning in CTBNs. The algorithm, which we call *Continuous-Time PC* (CTPC), is shown in Algorithm 6.

The first step is the same as the corresponding step of the PC algorithm in that it determines the same pattern of conditional independence tests. However, as discussed above, the hypotheses being tested are the *null time to transition hypothesis* and the *null state-to-state transition hypothesis*.

Algorithm 6 Continuous-time PC Algorithm

```
1. Form the complete directed graph $\mathcal{G}$ on the vertex set $\mathbf{X}$.
2. For each variable $X_i \in \mathbf{X}$:
   2.1 Set $\mathbf{U} := \{X_j \in \mathbf{X} : X_j \to X_i\}$, the current parent set.
   2.2 Set $b := 0$
   2.3 While $b < |\mathbf{U}|$:
      2.3.1 For each $X_j \in \mathbf{U}$, test $X_i \perp\!\!\!\perp X_j | \mathbf{U}_{X_i X_j}$ for all possible subsets of size $b$ of $\mathbf{U} \backslash X_j$.
      2.3.2 As soon as $X_i \perp\!\!\!\perp X_j \,|\, \mathbf{U}_{X_i X_j}$ for some $\mathbf{U}_{X_i X_j}$, remove $X_j \to X_i$ from $\mathcal{G}$ and $X_j$ from
            $\mathbf{U}$.
      2.3.3 Set $b := b + 1$
3. Return directed graph $\mathcal{G}$.
```

The second step of CTPC differs from that in the PC algorithm. Since independence relationships are not symmetric in CTBNs, we can find the graph $\mathcal{G}$ of the CTBN without identifying and then refining a CPDAG representing an equivalence class. Therefore, steps 3 and 4 of the PC algorithm are not needed in the case of CTBNs.

CTPC starts by initializing the complete directed graph $\mathcal{G}$ without self loops (step 1). Note that while loops (that is, arcs like $X_i \to X_i$) are not included, cycles of length two (that is, $X_i \to X_j$ and $X_j \to X_i$) are, as well as cycles of length three or more.

Step 2 iterates over the X_i to identify their parents $\mathbf{U}$. This is achieved in step 2.3.1 by first testing for unconditional independence, then by testing for conditional independence, gradually increasing the cardinality b of the considered separating sets.

Each time Algorithm 6 concludes that X_i is independent from X_j given some separating set, we remove the arc from node X_j to node X_i in step 2.3.2. At the same time, we also remove X_j from the current parent set $\mathbf{U}$. The iteration for X_i, X_j terminates either when X_j is found to be independent from X_i or when there are no more larger separating sets to try because $b = |\mathbf{U}|$; and the iteration over X_i terminates when there are no more X_j to test.

CTPC checks the *null time to transition hypothesis* (Definition 4.2) by applying the *test for two exponential means* in (4.2). On the contrary, the *null state- to-state transition hypothesis* (Definition 4.3) can be tested using two different tests: the *two sample chi-square test* in (4.4) and the *two sample Kolmogorov- Smirnov test* in (4.5). We call these two options CTPC_{χ^2} and CTPC_{KS}, respectively.

The proposed algorithm is able to recover the true graph under the standard assumptions of the PC algorithm:

1. The faithfulness assumption.
2. The database consists of a set of independent and identically distributed cases.
3. The database of cases is infinitely large.
4. The causal sufficiency assumption, that is, no hidden (latent) variables are involved.
5. The statistical tests make no type-I or type-II errors [46].

CTPC could be extended to account for hidden variables. However, this extension would not be straightforward and would have a high computational complexity. In the case of score-based structure learning, [47] used the Structural Expectation Maximization algorithm for this purpose, while [48] developed a novel gradient-based approach to structure learning which makes it possible to learn structures of previously inaccessible sizes. However, to the best of our knowledge, no constraint-based algorithm for learning the structure of CTBNs with latent variables has been presented in the literature. The results presented and discussed in [40] may allow the CTPC to be extended to handle hidden variables.

4.3 CTPC computational complexity

The computational complexity of estimating the network structure of a CTBN from data using the CTPC algorithm depends on the following quantities: the number of nodes, the number of parents of each node and the number of transitions.

In Appendix A we presented the computational complexity of learning the stucture of a CTBN using CTPC summarized in the following table:

Case	**Complexity**
Best Case	$O(n^2 \cdot (2 \cdot \eta^3 + 2 \cdot \psi))$
Worst case	$O(n \cdot 2^n \cdot (\eta^n + n \cdot \psi))$
General case	$O\left(\frac{n^{\rho+2}}{(\rho+1)!} \cdot (\eta^{\rho+3} + (\rho+2) \cdot \psi)\right)$

where n is the number of nodes, η is the maximum node cardinality present in the network, ψ is the number of transitions occurring in the dataset and ρ is the maximum parent set size present in the network.

Nodelman [11] states that the CTSS algorithm for CTBNs is polynomial in the number of variables (n) and in the size of the dataset (ψ) when a maximum number of parents is given. The general case of the CTPC, for a constant value of ψ and n, is also polynomial. However, in the CTPC algorithm it is natural to bound the size of the separating sets in the conditional independence tests but not the size of the parent sets because CTPC does not operate on parent sets explicitly. Bounding the size of the separating sets also results in a time complexity similar to the CTSS, but has the drawback of producing denser networks because it potentially makes some independence relationships impossible to establish.

4.4 Numerical Experiments

We now assess the performance of CTPC against that of the CTSS algorithm from [11] using synthetic data. In particular, we generate random CTBNs as the combinations of directed graphs and the associated CIMs; and we generate random trajectories from each CTBN.

Note that we only generate connected networks, hence the number of edges in a network is bounded below by $n - 1$.

We measure the performance of the learning algorithms using the F1 score over the arcs, which is defined as

$$F_1 = 2 \cdot \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} .$$

Since there is no score equivalence in CTBNs, nor are networks constrained to be acyclic, comparing graphs is equivalent to evaluating a binary classification problem.

Furthermore, we compare the two algorithms by their scaled difference in Bayesian Information Criterion (BIC), the latter defined as in [22]:

$$\Delta BIC\% = \frac{BIC_{CTSS} - BIC_{CTCP}}{BIC_{CTSS}} \cdot 100$$

with

$$BIC = \ln(\hat{L}) - \frac{1}{2} k \ln(\psi) \tag{4.6}$$

and where k is the number of parameters in the learned CTBN model, ψ is the number of transitions while $\hat{L}$ is the corresponding data likelihood.

After a first set of experiments presented in a preliminary work [14] we found that the CTPC$_{\chi^2}$ performs marginally better than CTPC$_{KS}$; thus, we concentrated the esperiments only on CTPC$_{\chi^2}$. Then, in the extended version [15] we set up a full factorial experimental design over different numbers of nodes $n = \{5, 10, 15, 20\}$, network densities[1] $\{0.1, 0.2, 0.3, 0.4\}$, number of states for the nodes $|dom(X_i)| = \{2, 3, 4\}$. For each network, we generate 300 trajectories that last on average 100 units of

[1]network density $= \frac{\text{Number of edges}}{n \cdot (n-1)}$

time each. We perform 10 replicates for each simulation configuration except for the networks with 20 ternary nodes and network density equal to 0.4, for which we only perform 3 replicates. We did not consider quaternary networks with 20 nodes because it is unfeasible to learn them on the available hardware.

We perform the experiments using new, implementations of the CTSS algorithm and of the CTPC$_{\chi^2}$ algorithm[2] that can handle larger networks and that can learn the parent set of each node in parallel. Those implementations are parallelized but use more memory, requiring a more powerful machine with 8 cores and 64GB of memory.

The results of our simulation study are summarized in Figure 4.1, in Figure 4.2, in Figure 4.3 and in Figure 4.4.

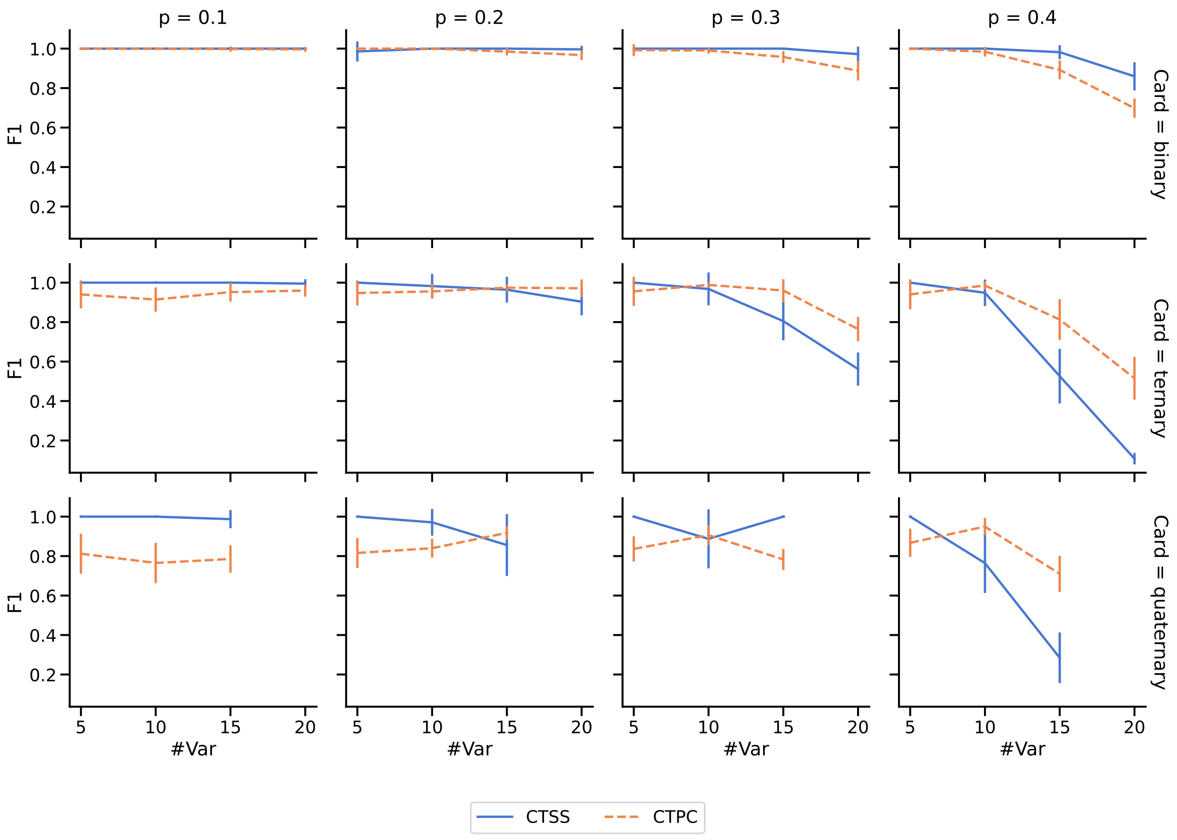


Figure 4.1: Each of the plots on this figure represents the average F1 score and the standard deviation of the constraint-based algorithm and the CTSS one, against the number of nodes for a specific combination of network density and node cardinality.

[2]The new implementations provided by Filippo Martini and Luca Moretti are available at https://github.com/madlabunimib/PyCTBN.

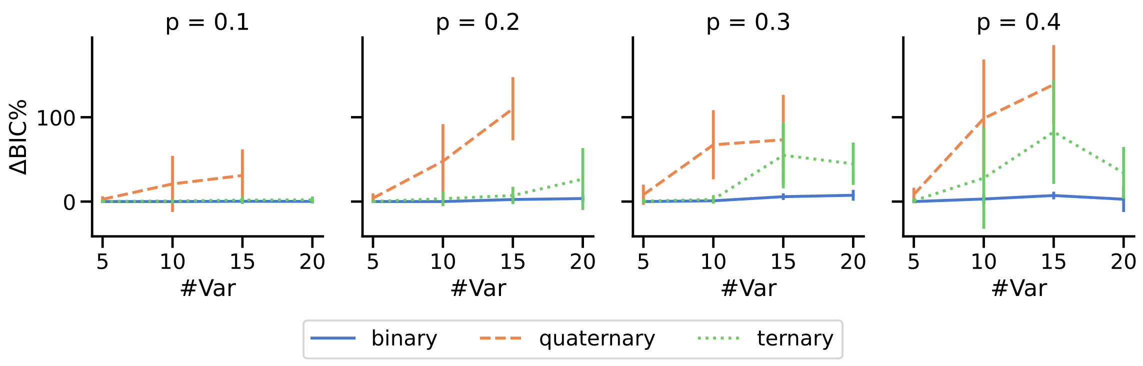


Figure 4.2: Each of the plots on this figure represents the average $\Delta BIC\%$ and the standard deviation against the number of nodes for a specific combination of network density and node cardinality.

Figure 4.1 shows that the CTSS algorithm is the best choice for networks consisting of binary nodes. However, the CTPC$_{\chi^2}$ and the CTSS perform similarly for networks with ternary and quaternary nodes. Furthermore, the performance of the CTPC algorithm, if compared with the CTSS one, seems to improve with the increase in the number of nodes and in the network density.

We can see from Figure 4.1 and Figure 4.3 that the execution time increases when the network density increases, and that both algorithms perform poorly for dense networks. This behavior may be attributed to the method used to generate the trajectories that does not increase their size when the

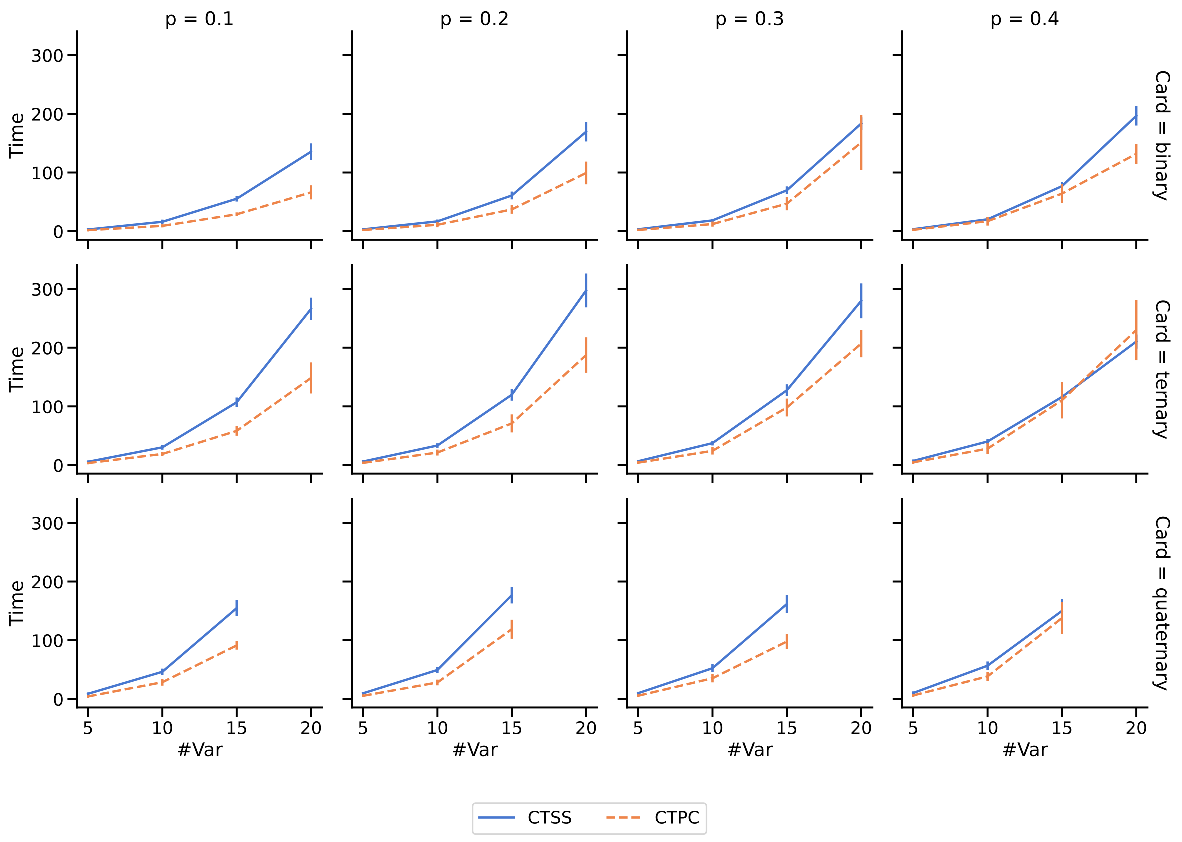


Figure 4.3: Each of the plots on this figure represents the average execution time in seconds and the standard deviation of the constraint-based algorithm and the CTSS one, against the number of nodes for a specific combination of network density and node cardinality.

network density increases.

Figure 4.2 shows something different. Indeed, if we consider the $\Delta BIC\%$ as an evaluation metric, we note the CTSS algorithm to be at least as good as the CTPC algorithm in almost every experiment.

Figure 4.4 shows an important difference between the two algorithms; the CTSS algorithm achieves a precision of 1 in almost all simulations. However, the constraint-based algorithm has a better recall for the networks with ternary and quaternary nodes.

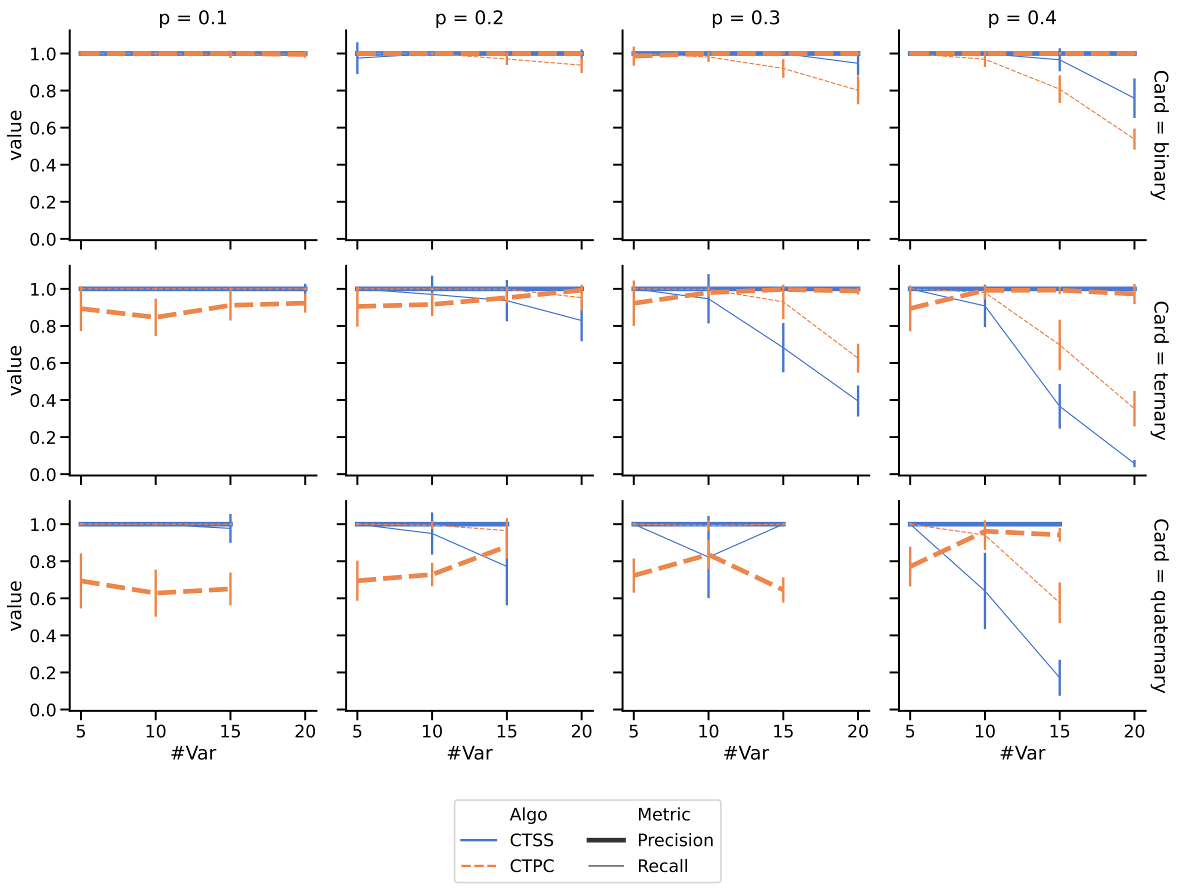


Figure 4.4: Each of the plots on this figure represents the average Precision, Recall and their respective standard deviations of the constraint-based algorithm and the CTSS one, against the number of nodes for a specific combination of network density and node cardinality.

Until now, we tested the CTPC algorithm with small to medium networks consisting of up to 20 nodes. To evaluate the execution time of CTPC with larger networks, we combined up to 5 networks with 20 binary nodes and network density 0.1. The results presented in Table 4.1 show that CTPC is able to learn networks with up to 100 nodes in a reasonable amount of time. It is important to note that the tested networks are *scattered*. [3] It was also necessary to use a machine with 256GB of memory to carry out the experiments. The huge memory requirement is closely related to our implementation. In fact, due to the python Global Lock Interpreter, we couldn't use a multi-thread architecture but we had to use a multi-process one. Since each process is independent, it requires a copy of the dataset to be loaded into the ram. Furthermore, the space complexity is closely related to the network density.[4]

[3] Networks composed by disjoint sub-networks that are collated together in such a way to maintain sparsity.

[4] A CIM for a binary node with 99 binary parents would require learning and storing 2^{101} values that would occupy about 10^{19}TB.

[5] This value was estimated as it was not possible to perform the experiment with the amount of memory available to us.

Cardinality	Execution Time (single core)	Execution Time (two cores)
20	4m	4m
40	50m	50m
60	4h15m	3h40
80	13h30m	8h47
100	30h28m	21h19m [5]

Table 4.1: Execution time of the CTPC on big, sparse networks

4.5 Discussion

In this chapter we introduced the first implementation of a constraint-based algorithm for structure learning in CTBNs, which we called CTPC, comprising both a suitable set of statistics for testing conditional independence and a heuristic algorithm based on PC. We also derived the complexity of this new algorithm, finding that it is similar to the CTSS one.

CTPC has better structural reconstruction accuracy, compared to the only CTSS algorithm previously available in the literature [11], when variables in the CTBN can assume more than two values. For binary variables, that CTSS algorithm performs well, but its performance rapidly degrades as variables are allowed to have increasingly more states. However, if we compare the two algorithms in terms of BIC the CTSS is the best option in all the performed experiments. A major limitation of the proposed constraint-based algorithm is the computational cost, which becomes problematic in domains with more than 20 variables. However, the CTSS algorithm has the same limitation.

Chapter 5

Constraint-based and hybrid structure learning of multidimensional continuous-time Bayesian network classifiers

When the continuous time Bayesian network classifier (CTBNC) was first presented [34], the only structure learning algorithm by that time was the score based one, i.e., the structure learning algorithm introduced and described in [11]. The same was true for the paper generalizing the CTBNC to the Multi-CTBNC [35], where structure learning was performed by using the same score based algorithm as the one used for learning the structure of CTBNC.

In this chapter, we introduce and describe a new set of constraint based structure learning algorithms [17], which have been developed by collaborating with the authors of CTBNC and Multi-CTBNC. These algorithms exploit the specific structure of the Multi-CTBNC, which consist of the following three main components; *i*) class subgraph, *ii*) feature subgraph, and *iii*) bridge subgraph, to extend the constraint based CTPC algorithm, originally designed for learning the structure of CTBNs, to learn the structure of Multi-CBTNC.

A rich set of numerical experiments has found that the new structure learning algorithms are more efficient and empirically faster, although less robust than the original CTPC algorithm, which is not specialized to leverage on the particular structure of Multi-CTBNC. Furhtermore, we carreid out a set of experiments over the British Household Panel survey [49] where the constraint based algorithms obtained the best performaces. However, as we will see in Section 5.4, execution time performance improvement is traded for prediction performances.

5.1 Constraint Based Algorithm - naive adaptation

In order to adapt the original constraint based algorithm, it is fundamental to take into account the main components of the Multi-CTBNC, i.e., *i*) the class subgraph, *ii*) the feature subgraph, and *iii*) the bridge subgraph. The class subgraph $\mathcal{G}_{\boldsymbol{\mathcal{X}}}$ consists of a Bayesian network (BN), thus its structure can be learnt by applying a standard constraint based algorithm for BNs (i.e. Algorithm 5), while the feature subgraph and the bridge subgraph can be learned together by adapting the CTPC algorithm (Algorithm 6).

A naive adaptation of the CTPC algorithm to learn the structure of a Multi-CTBNC (Algorithm 7) presents the following differences when compared to the original version designed to learn the structure of CTBNs:

- The initial graph $\mathcal{G}$ for learning the structure of a Multi-CTBNC is not the complete graph, as it happens with the initial graph for learning the structure of a CTBN. Indeed, the initial graph $\mathcal{G}$ of a Multi-CTBNC is set in such a manner that only those edges which are allowed in a Multi-CTBNC are included. Consequently, no edges directed from a feature node to a class node are allowed. Furthermore, we are not interested to learn edges between class nodes, because those edges are learned by the PC algorithm for BNs. More formally, the complete feature subgraph $\mathcal{G}_{\mathbf{X}}$ consists in all the directed edges (X_i, X_j) where $X_i, X_j \in \mathbf{X}$ and $X_i \neq X_j$; while the complete bridge subgraph $\mathcal{G}_{\mathbf{B}}$ consists in all the directed edges $(\mathcal{X}_i, X_j)$ where $\mathcal{X}_i \in \boldsymbol{\mathcal{X}}$ and $X_j \in \mathbf{X}$.

- Since we need to learn only the edges of feature and bridge subgraphs but not those of the class subgraph, the loop at line 3 of Algorithm 7 iterates only over the feature nodes of the Multi-CTBNC.

Algorithm 7 Naive adaptation of the CTPC Algorithm for Multi-CTBNC

```
1. Add the edges of the complete directed graph $\mathcal{G}_{\mathbf{X}}$ to $\mathcal{G}$
2. Add the edges of the complete directed graph $\mathcal{G}_{\mathbf{B}}$ to $\mathcal{G}$
3. For each variable $X_i \in \mathbf{X}$:
   3.1 Set $\mathbf{U} := \{X_j \in \mathbf{X} \cup \boldsymbol{\mathcal{X}} : X_j \rightarrow X_i\}$, the current parent set.
   3.2 Set $b := 0$
   3.3 While $b < |\mathbf{U}|$:
      3.3.1 For each $X_j \in \mathbf{U}$, test $X_i \perp\!\!\!\perp X_j | \mathbf{U}_{X_i X_j}$ for all possible subsets of size $b$ of $\mathbf{U} \backslash X_j$.
      3.3.2 As soon as $X_i \perp\!\!\!\perp X_j \,|\, \mathbf{U}_{X_i X_j}$ for some $\mathbf{U}_{X_i X_j}$, remove $X_j \rightarrow X_i$ from $\mathcal{G}$ and $X_j$ from $\mathbf{U}$.
      3.3.3 Set $b := b + 1$
4. Return directed graph $\mathcal{G}$.
```

5.2 Markov Blanket CTPC

The naive approach presented in the previous section is an elementary adaptation of the CTPC algorithm, which does not exploit in any manner the peculiar characteristics of the Multi-CTBNC.

In this section we introduce and describe a novel and original constraint based structure learning algorithm [17] which goes under the name of *Markov blanket-based continuous time PC* (MB-CTPC). This algorithm exploits the fact that a Multi-CTBNC comprises a bipartite graph, namely the bridge subgraph, and takes into account the ultimate objective of such a model, i.e. multidimensional classification.

The MB-CTPC algorithm leverages on the fact that the evidence on the nodes belonging to the Markov Blanket of the class nodes is sufficient to perform the classification task. In practice, the MB-CTPC algorithm implements a set of rules which avoid testing for those independencies that are irrelevant to accomplish the multidimensional classification task.

The pseudocode presented in Algorithm 8 describes the MB-CTPC algorithm. The first steps of the algorithm consists in learning the structure of the class subgraph and fill the feature and bridge subgraphs. These steps are exactly the same used for the naive adaptation of the CTPC algorithm. Step 4 identifies all the descendents of the class variables. This step provides a superset of the bridge subgraph and also gives us valuable information which can be exploited to prune the feature subgraph, thus avoiding testing for many irrelevant conditional independencies. This is made possible because, if a pair of feature nodes do not share the same parent class variables, information does not flow at least in one direction, and thus at least one edge can be safely removed from the graph of the Multi-CTBNC. Based on these intuitions, Step 5 applies the following 3 rules to prune the feature subgraph, and thus to reduce the computational effort:

- **Rule 1**: if a feature node is not a descendent of a class node, then all the edges directed from other feature nodes to the given feature node can be safely removed.

Algorithm 8 MB-CTPC Algorithm for Multi-CTBNC

```
1. $\mathcal{G} \leftarrow$ Learn the class subgraph using the classical PC algorithm over the class variables $\boldsymbol{\mathcal{X}}$.
2. Add the edges of the complete directed graph $\mathcal{G}_{\mathbf{X}}$ to $\mathcal{G}$
3. Add the edges of the complete directed graph $\mathcal{G}_{\mathbf{B}}$ to $\mathcal{G}$
4. $\mathcal{G} \leftarrow$ Apply the CTPC algorithm only for the edges in the bridge subgraph.
5. For each $X_i, X_j \in \mathbf{X}$ $s.t.$ $X_i \neq X_j$
    5.1 If $\boldsymbol{\mathcal{X}} \cap pa(X_i) = \emptyset$
        5.1.1 Remove $X_j \rightarrow X_i$ from $\mathcal{G}$
    5.2 Else If $\boldsymbol{\mathcal{X}} \cap pa(X_i) \cap pa(X_j) = \emptyset$ and $\boldsymbol{\mathcal{X}} \cap pa(X_j) \neq \emptyset$
        5.2.1 Remove $X_j \rightarrow X_i$ and $X_i \rightarrow X_j$ from $\mathcal{G}$
    5.3 Else If $(pa(X_i) \backslash pa(X_j)) \cap \boldsymbol{\mathcal{X}} \neq \emptyset$ and $pa(X_j) \neq \emptyset$
        5.3.1 Remove $X_i \rightarrow X_j$ from $\mathcal{G}$
6. $\mathcal{G} \leftarrow$ apply the CTPC algorithm starting from $\mathcal{G}$ instead of the complete graph.
7. Return directed graph $\mathcal{G}$.
```

- **Rule 2**: if two feature nodes are descendent of the class subgraph, but they do not share any class variable in their parent set, we can safely remove all the edges connecting these two feature nodes.
- **Rule 3**: if two feature nodes X_i and X_j are descendent of the class subgraph and only a subset of the class variables in X_i are shared with the parent set of X_j, then we can remove the edge pointing from X_i to X_j, i.e., we can remove the edge $X_i \rightarrow X_j$.

Finally, Step 6 applies the CTPC algorithm to the pruned graph $\mathcal{G}$.

It is worthwhile to mention that, since at Step 1 of Algorithm 8 the PC algorithm is used and at Step 4 the CTPC algorithm is used, the MB-CTPC algorithm inherit the complexity of both the PC and the CTPC algorithms.

5.3 Hybrid structure learning algorithm

The strengths and weaknesses of score-based and constraint-based algorithms are specific to each of them, and the selection of the algorithm is usually based on the given context. However, hybrid algorithms, combining the best aspects of both score-based and constraint-based approaches [50], [51], [52], [53] are made available by the PGMs literature. In this section, we introduce and describe a hybrid algorithm which has been specifically designed for learning the structure of Multi-CTBNC. The algorithm consists of two pruning phases, where the first one exploits conditional independence tests to find the initial structure of the Multi-CTBNC, while the second phase consists of a maximization step to refine the initial structure of the Multi-CTBNC. Depending on the considered subgraph, we have the following:

- **Class subgraph**: the PC algorithm for BNs is used to recover the undirected graph (skeleton). Then, the hill climbing algorithm (score based) is used to learn the optimal directed graph by starting from the empty graph, while allowing the insertion of only those edges that are contained in the found skeleton.
- **Bridge and feature subgraph**: in the pruning phase, the CTPC algorithm finds an initial structure. In the maximization phase, the hill-climbing optimizes the likelihood to identify the possibly optimal subset of edges to be included from the initial structure found by the CTPC algorithm. In the pruning phase, the maximum size of the separation set determines how much the initial graph has to be ”refined”.

5.4 Experiments

This experimental section aims to compare the classification performance of a Multi-CTBNC learned with different structure learning algorithms over several synthetic datasets. Furthermore, we tested the algorithm on a real dataset. The compared algorithms are the following:

Score based algorithms The score based algorithms we consider are the hill climbing and the tabu search with tabu list of size 5. The used scores are BIC and BDe.

Constraint based algorithms The constraint based algorithms we consider are the naive adaptation of the CTPC and the MB-CTPC.

Hybrid Algorithms The considered hybrid algorithm combines CTPC, implementing the pruning step, and the hill climbing using the BIC score implementing the maximization step. We tested the hybrid algorithm with the separation set size set equal to zero and to one.

To obtain a fair comparison of different algorithms, we evaluated the learned models by using several performance measures by means of a 5-fold cross-validation scheme. The experiments were run on a 4.20 GHz Intel Core i7-7700 K with 32 GB of RAM using Windows 10. The structure learning algorithms were developed in Java, while software and datasets are made freely available on GitHub [1].

5.4.1 Performance measures

It is worthwhile to mention that, since the final goal of a Multi-CTBNC is to solve a supervised classification task, all the performance measures evaluate and compare the learned models based on their capability to correctly classify previously unseen instances. The performance measures we use are the following ones:

- **Global accuracy** [54]: the ratio of trajectories correctly classified for all class variables. A trajectory is considered to be correctly classified only if all the class variables are correctly classified, thus such performance measure is formally defined as follows

$$Acc = \frac{1}{N}\sum_{l=1}^{N}\delta(\hat{\boldsymbol{x}}_l, \boldsymbol{x}_l), \tag{5.1}$$

  where $\hat{\boldsymbol{x}}_l$ represents the predicted classes associated with the l-th trajectory, while $\boldsymbol{x}_l$ represents the real classes, while δ is the Kronecker's delta function which returns 1 if $\hat{\boldsymbol{x}}_l = \boldsymbol{x}_l$, 0 otherwise.

- **Mean accuracy** [54]: mean of the accuracies obtained for each class variable separately and formally defined as follows:

$$\overline{Acc} = \frac{1}{|\boldsymbol{\mathcal{X}}|}\sum_{i=1}^{|\boldsymbol{\mathcal{X}}|}\frac{1}{N}\sum_{l=1}^{N}\delta(\hat{x}_{li}, x_{li}), \tag{5.2}$$

  where $\hat{x}_{li}$ and x_{li} are respectively the predicted and real class values for the l-th trajectory and i-th class variable.

- **Global Brier**[55]: it evaluates the probability distribution of the predicted variables with respect to the real classes, and it is formally defined as follows:

$$Bs = \frac{1}{N}\sum_{l=1}^{N}\sum_{g=1}^{|\mathcal{I}|}\left(p(\boldsymbol{\mathcal{X}} = x_g|\mathbf{X} = \mathbf{x}_l) - \delta(\boldsymbol{x}_g, \boldsymbol{x})\right)^2, \tag{5.3}$$

  where $\mathcal{I} = \times_{\mathcal{X}\in\boldsymbol{\mathcal{X}}} dom(\mathcal{X})$ is the space of joint configurations of the class variables. Smaller values of Global Brier score correspond to better classifications.

Performace measure	Results of ... are	Better	Worse	Same	Than
Global accuracy	Hill climbing [BIC]	10.56%	5.93%	83.52%	Hill climbing [BDe]
	Hill climbing [BIC]	1.20%	0.83%	97.76%	Tabu search [BIC]
	CTPC	17.69%	42.31%	40.00%	Hill climbing [BIC]
	MB-CTPC	1.76%	39.26%	58.98%	CTPC
	Hybrid (separation set = 0)	1.76%	0.00%	98.24%	Hybrid (separation set = 1)
	Hybrid (separation set = 0)	5.56%	34.54%	59.91%	Hill climbing [BIC]
	Hybrid (separation set = 0)	22.96%	20.56%	56.48%	CTPC
Mean accuracy	Hill climbing [BIC]	11.39%	6.48%	82.13%	Hill climbing [BDe]
	Hill climbing [BIC]	1.20%	1.20%	97.59%	Tabu search [BIC]
	CTPC	22.39%	39.44%	39.17%	Hill climbing [BIC]
	MB-CTPC	1.67%	40.28%	58.06%	CTPC
	Hybrid (separation set = 0)	1.85%	0.00%	98.15%	Hybrid (separation set = 1)
	Hybrid (separation set = 0)	9.35%	31.85%	58.80%	Hill climbing [BIC]
	Hybrid (separation set = 0)	22.59%	21.76%	55.65%	CTPC
Global Brier score	Hill climbing [BIC]	24.81%	18.89%	56.30%	Hill climbing [BDe]
	Hill climbing [BIC]	5.15%	5.28%	89.54%	Tabu search [BIC]
	CTPC	19.91%	59.81%	20.28%	Hill climbing [BIC]
	MB-CTPC	11.85%	53.52%	34.63%	CTPC
	Hybrid (separation set = 0)	3.52%	2.59%	93.89%	Hybrid (separation set = 1)
	Hybrid (separation set = 0)	7.41%	49.26%	43.33%	Hill climbing [BIC]
	Hybrid (separation set = 0)	34.54%	23.80%	41.67%	CTPC
Learning time	Hill climbing [BIC]	37.04%	62.96%	0.00%	Hill climbing [BDe]
	Hill climbing [BIC]	37.96%	62.04%	0.00%	Tabu search [BIC]
	CTPC	99.54%	0.46%	0.00%	Hill climbing [BIC]
	MB-CTPC	98.15%	1.85%	0.00%	CTPC
	Hybrid (separation set = 0)	36.11%	63.89%	0.00%	Hybrid (separation set = 1)
	Hybrid (separation set = 0)	90.28%	9.72%	0.00%	Hill climbing [BIC]
	Hybrid (separation set = 0)	4.63%	95.37%	0.00%	CTPC

Table 5.1: Percentage of datasets in which structure learning algorithms achieve better, worse or identical results accordingly to the Wilcoxon signed-rank test with significant level of 0.05.

5.4.2 Experimental results on synthetic data

In this section, we briefly summarize the performances achieved by different structure learning algorithms when applied to different synthetic datasets.

During the synthetic experiments the hyperparameters of the constraint based algorithms have been setted as follows:

- Significance Level (class subgraph): 0.05
- Significance level (bridge and feature subgraph): 10^{-5}
- Separation set size for the hybrid algorithm: 0, 1

The generation of the synthetic datasets has been performed by considering the following parameters and parameters values:

- Number of feature variables: 5, 10, 20
- Cardinality of feature variables: 2, 3, 4, 8
- Number of class variables: 4
- Cardinality of class variables: 2, 3
- Density of class subgraph: 30%
- Density of bridge subgraph: 5%, 10%, 20%
- Density of feature subgraph: 5%, 10%, 20%

[1] https://github.com/carlvilla/Multi-CTBNCs

Algorithm	Acc	$\overline{Acc}$	Br	Learning time (s)
Hill climbing [BIC]	0.63 ± 0.27	0.87 ± 0.11	0.46 ± 0.31	108 ± 138
Tabu search [BIC]	0.63 ± 0.27	0.87 ± 0.11	0.46 ± 0.31	107 ± 137
Hill climbing [BDe]	0.62 ± 0.27	0.87 ± 0.12	0.47 ± 0.31	106 ± 136
CTPC	0.64 ± 0.27	0.87 ± 0.11	0.45 ± 0.31	42 ± 47
MB-CTPC	0.61 ± 0.26	0.86 ± 0.11	0.48 ± 0.29	22 ± 23
Hybrid (separation set = 0)	0.62 ± 0.27	0.87 ± 0.12	0.47 ± 0.31	50 ± 57
Hybrid (separation set = 1)	0.62 ± 0.27	0.87 ± 0.12	0.47 ± 0.31	45 ± 51

Table 5.2: Overall performances of the structure learning algorithms.

- Number of trajectories for each dataset: 5000
- Ending time for each trajectory: 20

For each combination of the above parameters, 5 datasets have been generated, thus a total of 1080 datasets have been generated. The Wilcoxon signed-rank test with a significance level of 0.05 has been used to compare the performance of different algorithms applied to different datasets. Specifically, for each pair of algorithms and for each performance measure we computed the number of dataset where the two algorithms performed the same and the number of datasets where one algorithm performed statistically better than the other one. The most relevant findings of such results are summarized in Table 5.1. First of all, we compared the score based algorithms available finding that the Hill climbing, which optimizes the BIC score, performs statistically better than itself when using the BDe score as well as when Tabu search is used to optimize the BIC score. This finding holds true also when comparing Hill Climbing, when optimizing the BIC score, to both the constraint based and the hybrid algorithms. To further explore the performance of the algorithms under comparison we computed mean and standard deviations (summarized in Table 5.2). This table shows that, even if the score based is statistically better than both the constraint based and the hybrid algorithms, the overall performance are really close. Furthermore, both Table 5.1 and Table 5.2 show that the constraint based algorithm and the hybrid one require a statistically smaller amount of execution time to learn the structure of the network. We are aware of the fact that the execution time is strongly entangled with the implementation. However, we directly implemented all the algorithms as described in the literature. Furthermore, the ”heaviest” component of the learning algorithm is the computation of the sufficient statistics and this component is shared by all the implementations. For this reason, even if we are aware of the fact that the execution time does not share the same level of objectivity with the other metrics, we think that it is still a metric worthy to mention.

It is worthwhile to mention that, similarly to what was observed for CTBNs, also for the Multi-CTBNC, the constraint based approach outperforms the score based one when the cardinality of the nodes increases. This limitation of the CTPC algorithm is partially overcome by the hybrid algorithm which, by combining the score based approach with the constraint based approach, manages to obtain good performances even with low cardinalities while maintaining an acceptable execution time.

5.4.3 Experimental results on real data

In the previous subsection we compared the performances of the different structure learning algorithms for Multi-CTPC using synthetic data that allowed for complete control over the experimental setting. However, the experiments with synthetic data do not answer to the question: is there any real-world scenario were the constraint based approach can be effectively applied? For this reason, we decided to conduct a new set of experiments over the British Household Panel survey (BHPS) [49]. The BHPS dataset was collected in 18 waves during the period 1991-2009 with the goal of understanding the evolution of social and economic change at individual level and household level in Britain. The survey collects information from 29702 individuals on more than 1300 variables.

The task we carried out over the BHPS dataset was a multi-label classification over the following variables:

- *Dental check-up*: a binary variable that denotes if the individual has had a dental check-up in the last year.

Algorithm	Acc	$\overline{Acc}$	Br	Learning time (s)
Hill climbing [BIC]	0.203 ± 0.001	0.777 ± 0.001	0.899 ± 0.001	11.5 ± 0.5
Tabu search [BIC]	0.203 ± 0.001	0.777 ± 0.001	0.899 ± 0.001	11.0 ± 0.4
Hill climbing [BDe]	0.218 ± 0.002	0.790 ± 0.001	0.886 ± 0.002	10.8 ± 0.2
CTPC	0.380 ± 0.002	0.869 ± 0.001	0.793 ± 0.001	47.5 ± 7.8
MB-CTPC	0.385 ± 0.001	0.861 ± 0.001	0.787 ± 0.001	23.5 ± 3.0
Hybrid (separation set = 0)	-	-	-	-
Hybrid (separation set = 1)	0.157 ± 0.002	0.765 ± 0.001	0.921 ± 0.001	57.7 ± 17.5
Hybrid (separation set = 2)	0.160 ± 0.002	0.769 ± 0.001	0.919 ± 0.001	17.8 ± 0.6

Table 5.3: Overall performances of the structure learning algorithms on BHPS dataset.

- *Employment status*: a categorical variable describing the current employment status of the individual. It can assume one of the following values: self-employed, in paid employment, unemployed, retired, maternity leave, looking after family or home, full-time student, long-term sick or disabled, on a government training scheme, others.
- *Limb, back or neck problems*: a binary variable that denotes if the individual has any disability or problems related to arms, legs, hands, feet, back or neck.
- *Lives with partner*: a binary variable that denotes if the individual lives with her/his spouse or partner.
- *Responsible adult for child*: a binary variable that denotes if the individual is responsible for a child under 16 years old.
- *Sex*: sex of the individual
- *Smoker*: a binary variable that specifies if the individual smokes.

For the feature variables, we selected a subset of discrete-state variables related to health, work, household, and other personal information. The BHPS dataset contains a high level of missing values. Therefore, during the selection of the features, we discarded all the variables with more than 3% of missing data. Specifically, we considered: — Employee or self-employed — Region or metropolitan area — Had paid work last week — Number of people in household — looked for work in last 4 weeks — Hospital inpatient last year — Alcohol or drugs — Household type — Health hinders dressing up — Working age — How far health limits work — Health limits daily activities — Marital status — Is dependent child — No health problems — Health limits work — Vision problems — Stomach, liver, kidneys or digestive problems — Migraine — Anxiety or depression — Diabetes — Hearth, blood pressure or blood circulation problems — Hearing problems — Breathing problems — Epilepsy.

Each sequence extracted from the BHPS dataset contains the survey results of a single individual. For each sequence we generated as many trajectories as the number of changes in the class variables plus one.

We designed the experiment as 10-fold cross-validation, and we compared the performances of the different algorithms in the classification task with the same metrics used in Section 5.4.2. The average results of this experiment are reported in Table 5.3. On this specific dataset, the constraint based algorithms (CTPC and MB-CTPC) performed better than score based and hybrid algorithms with the drawback of having a higher execution time. Furthermore, unlike the results obtained with synthetic data, in this case the score based algorithm using BDe score performed better than the one using BIC score. Finally, the worst results were achieved by the hybrid algorithms and, in the case of separation set equal to 0, the computational resources used were not enough to learn the structure. This behavior can be traced back to the fact that, in the pruning phase, the hybrid algorithm removes too few edges, requiring the maximization phase to deal with an extremely dense structure.

5.5 Discussion

In this chapter we introduced the MB-CTPC, i.e. the first constraint-based algorithm specifically designed for learning the structure of a Multi-CTBNC. We also introduced the first hybrid structure learning algorithm for both CTBN and CTBNC. For the purpose of evaluating the structure learning algorithms, we performed a set of synthetic experiments for a total of 1080 datasets. The score based structure learning algorithm obtaining statistically better results in more synthetic dataset at the cost of a higher execution time.

We also performed an experiment on real data using the BHPS dataset and, in this particular case, the constraint based algorithm achieved better performance at the cost of a higher execution time.

In conclusion, we want to stress out the fact that the there is not an overall best structure learning algorithm. Consequently, it is essential to take into account all of them since, the optimality of one algorithm over the others largely depends on the specific context.

Chapter 6

Sentry State for CTBN

Interacting systems of events may exhibit *cascading* behavior, where events tend to be temporally clustered. While the cascades themselves may be obvious from the data, it is important to understand which are the conditions that may trigger them.. For this purpose, we propose a modeling framework based on CTBNs to analyze cascading behavior in complex systems [5]. This framework allows us to describe how events propagate through the system and to identify likely *sentry states*, that is, system states that may lead to imminent cascading behavior. Moreover, CTBNs have a simple graphical representation and provide interpretable outputs, both of which are important when communicating with domain experts.

The evaluation of the sentry state concept was carried out with a set of synthetics experiment. Furthermore, we used the dataset provided by the European Spallation Source (ESS), a large research facility, to evaluate the sentry state approach on real data. Specifically, in Section 6.5, we analyzed the logs produced by the facility's monitoring/alarm system to assist operators in pinpointing the most crucial alarms that could potentially trigger a chain reaction of failures.

6.1 Naive Approach

Informally, a cascade of events is a fast sequence of transitions; where fast is relative to the rest of the transitions that are observed during the evolution of the process. Starting from this informal definition, we can develop a naive approach to identifying such cascades in a trajectory. Firstly, we need to identify two quantities: - λ_{ft}: the *fast threshold* determines when two consecutive transitions are considered to occur *fast*. - λ_{mcl}: the *minimum cascade length* determines the minimum number of fast consecutive events to be considered a cascade.

Given the two parameters, the identification procedure consists of iterating over the entire trajectory and identifying subsets of consecutive transitions with length at least λ_{mcl} and with a transition time between each pair of consecutive events of less than λ_{ft} (Algorithm 9). This approach can also be used to identify a *sentry state*. Once a cascade of events has been identified, the sentry state is the state from which the cascade begins.

The main limitation of this approach is the difficulty of identifying the correct parameters, as it requires knowing in advance common durations and sizes of event cascades.

In addition, we define two simple quantities: *Naive Count* - the number of times a state starts a cascade, and *Naive Score* - the fraction of times that observing a specific state coincides with the start of a cascade.

6.2 Sentry State Identification

Given a CTBN, we are interested in understanding its cascading behavior. We are in particular interested in identifying what we will call *sentry states*. A sentry state is a state which may trigger a *ripple effect*, that is, a sequence of fast transitions. It is important to stress that we are interested in states that *start* a cascade of events. Intuitively, this means that we are assuming the existence of at least one state in the state space graph which is directly connected to the sentry state and which has a much smaller expected number of transitions than the sentry state itself.

Algorithm 9 Naive cascade identification algorithm

```
procedure NAIVE-CASCADE-IDENTIFICATION($\sigma$, $\lambda_{ft}$, $\lambda_{mcl}$)
    for all event $\sigma_i \in \sigma$ do
        $dt_i = Time(\sigma_i) - Time(\sigma_{i-1})$    ▷ Compute the delta time between an event and its predecessor.
    end for
    Identify the set $\mathbf{s}$ such that:
      • $s \in \mathbf{s}$ is a sequence of contiguous events in $\sigma$
      • each event in $\sigma_i \in s$ has $dt_i < \lambda_{ft}$
      • $|s| \geqslant \lambda_{mcl}$
      • $s \not\subset \mathbf{s}$
    return $\mathbf{s}$
end procedure
```

In order to identify a sentry state, we need to take a further step from the heuristic definition of a sentry state we have just given and formalize the concept. For this purpose, we first compute the expected (discounted) number of transitions for each state of the CTBN. This can be achieved by using the lump sum reward in (3.11) to obtain the *Expected Discounted Number of Transitions* (EDNT) of each state $\mathbf{x}$.

$$EDNT_\gamma(\mathbf{x}) = \mathbb{E}_{\mathcal{N}_\mathbf{x}}\left[\sum_{i=0}^{\infty}\sum_{X_j \in \mathbf{X}} e^{-\gamma t_{i+1}}\mathcal{C}(X_j(t_i), X_j(t_{i+1}))\right] : \mathcal{C}(x, x') = 1. \tag{6.1}$$

There is no guarantee that a state with high EDNT is often the starting point of a cascade. States that tend to occur in the middle of a cascade may easily have a high EDNT if the cascade tends to continue after reaching that state. We are interested in early detection of cascades, and the solution we propose is to take into account the number of transitions in the neighborhood $\mathrm{Ne}_{\mathcal{G}_s}(x)$ of the state $x \in S$. For this purpose, we define a new quantity called *Relative Expected Discounted Number of Transitions* (REDNT),

$$REDNT_\gamma(\mathbf{x}) = \max_{\mathbf{x}' \in \mathrm{Ne}_{\mathcal{G}_s}(x)} \frac{EDNT_\gamma(x)}{EDNT_\gamma(x')} \tag{6.2}$$

where γ in (6.1) and (6.2) is the discounting factor as in (3.10), and the neighborhood in (6.2) refers to the undirected state space graph (see Figure 3.3). The central idea is that a large ratio between two adjacent states implies that transition from one to the other leads to a significant change in the expected discounted number of transitions. We will use REDNT to identify potential sentry states (states with high values of REDNT are likely sentry states).

One could propose other ways to aggregate EDNT across different states. We focus on REDNT as defined above in the interest of brevity.

An approach involving the use of the RDNT would require exploring the entire state space. However, if we have an insight on the specific problem that we want to address this may not be necessary. For example in the case in which all the nodes describe alarms (binary events); we are interested to identify which is the combination of alarms responsible to start a cascade. Specifically, If we let $\bar{s}$ denote the number of alarm that are *on* in the state $s = (s_1, s_2, \ldots, s_L)$, $\bar{s} = \sum_{i=1}^{L} s_i$; in the alarm data application, we are mostly interested in sentry states such that $\bar{s}$ is fairly small. States with large $\bar{s}$ may also have large REDNT values; however, these are states that occur when a cascade is already happening. As we want early detection, we should focus on sentry states such that $\bar{s}$ is small.

6.3 Monte Carlo Algorithm

We are now left with the problem of estimating the EDNT of each state from which we can compute the REDNT. We propose a Monte Carlo approach based on Algorithm 4 from [11]. This sampling algorithm starts from an initial state $X(0)$ and generates a single trajectory σ ending at time t_{end}. After the *initialization* phase, the algorithm enters into a loop. At each iteration, the algorithm samples

Algorithm 10 Approximated estimation of the EDNT

```
procedure EDNT-APPROX($X(0)$, $\gamma$, $t_{end}$, $iterations$)
    $reward \leftarrow 0$
    for $i \leftarrow 1$ to $iterations$ do
        $\sigma \leftarrow$ CTBN-SAMPLE($X(0), t_{end}$)                ▻ Generate trajectory (Algorithm 4)
        $r \leftarrow \sum_{i=0}^{|\sigma|-1} e^{-\gamma t_{i+1}} \mathcal{C}(X(t_i), X(t_{i+1}))$     ▻ Estimate the EDNT of $\sigma$
        $reward + = r$                                            ▻ Update the total reward
    end for
    return $reward/iterations$                                    ▻ Estimate the expected reward
end procedure
```

a time to transition for each of the variables, identifies the next *transitioning variable*, generates the *next state*, and *resets the time to transition* for the transitioned variable and all its children. The Algorithm 10, combines Algorithm 4 with (6.1) to compute

$$\widehat{EDNT}_{\gamma}(\mathbf{x}) = \frac{1}{|\boldsymbol{\sigma}|} \sum_{\sigma \in \boldsymbol{\sigma}} \sum_{i=0}^{|\sigma|} e^{-\gamma t_i} C(x(t_i), x(t_{i+1})) \tag{6.3}$$

where $|\boldsymbol{\sigma}|$ is the number of trajectories generated by Algorithm 4 and $|\sigma|$ represents the number of events in the trajectory σ.

In order to compute $\widehat{EDNT}_{\gamma}$, we need to set the values of the following hyperparameters:

1. γ, the discounting factor;
2. t_{end}, the ending time for each trajectory;
3. $|\boldsymbol{\sigma}|$, the number of trajectories to be generated.

Choosing the discounting factor γ and the ending time t_{end}, we identify the time horizon of interest and appropriate values, depend on the application. On the other hand, $|\boldsymbol{\sigma}|$ controls the trade-off between the quality of the approximation and its computational cost: We can choose its value using a stopping-rule approach based on variance, as proposed in [21].

6.4 Synthetic experiments

We now study the performance of the proposed approach. We generate synthetic data from CTBNs such that the sentry states are known. Data is generated from different CTBNs. In all of them,

- each process, $\mathbf{X}_j \in \mathbf{X}$, has a binary state space.
- the CTBN consists of *slow processes* and *fast processes*.
- each process, $\mathbf{X}_j \in \mathbf{X}$, replicates the state of its parent processes, $pa(\mathbf{X}_j)$.
- if a process, $\mathbf{X}_j \in \mathbf{X}$, has more than one parent, it stays in state 0 with high probability if at least one of its parents is in state 0.

Experiment 1 The first synthetic experiment is based on a CTBN model whose graph $\mathcal{G}$ is a chain consisting of three nodes A, B, and C (Figure 6.1a). The corresponding CIMs for the processes A, B, and C are shown in Table 6.1. This CTBN describes a structured stochastic process such that the root process, A, changes slowly from the state no-alarm (0) to the state alarm (1) and vice versa. This can be seen from the CIM corresponding to the process A. The CIMs associated with processes B and C make these two processes replicate the state of their parent process, and this happens at a faster rate. Therefore, starting from $(0, 0, 0)$, if process A changes its state, process B quickly changes its state to match that of its parent A. The same holds true for the process C. For this reason, we expect $\{A = 1, B = 0, C = 0\}$ to be a sentry state because as soon as the process A transitions from state 0 to state 1, a fast sequence of transitions (a cascade of events) makes the processes B and C transition from state 0 to state 1. This behavior is shown in Figure 6.1b. Estimates of the REDNT quantity are shown in Table 6.2 and they confirm that $\{A = 1, B = 0, C = 0\}$ is a sentry state.

A	0	1
0	-1.0	1.0
1	5.0	-5.0

A	B	0	1
0	0	-0.1	0.1
	1	15.0	-15.0
1	0	15.0	-15.0
	1	0.1	-0.1

B	C	0	1
0	0	-0.1	0.1
	1	15.0	-15.0
1	0	-15.0	15.0
	1	0.1	-0.1

Table 6.1: Conditional Intensity Matrices used for the example in Figure 6.1a. Process A has no parents and therefore its transition rate only depends on its own state: If A is in state 0 (*off*), then its transition rate (to state 1 (*on*)) is 1.0. Process B has a single parent, process A. The states of processes A and B determine the transition rate of process B. If A is in state 0 (*off*) and B is in state 0 (*off*), then B transitions to state 1 (*on*) with rate 0.1. A CTBN is defined from its CIMs and its initial distribution. Its graph illustrates the dependence structure in the CIMs.

A	B	C	EDNT	REDNT	Naive Score	Naive Count
1	**0**	**0**	**6.316**	**1.589**	**0.35**	**304**
1	0	1	6.444	1.359	0.21	13
0	**1**	**0**	**5.394**	**1.357**	**0.16**	**41**
0	**0**	**1**	**4.740**	**1.192**	**0.03**	**26**
0	1	1	5.511	1.163	0.22	153
1	1	0	6.173	1.145	0.08	57
1	1	1	5.455	1.0	0.03	19
0	**0**	**0**	**3.976**	**1.0**	**0.02**	**24**

Table 6.2: Values of EDNT, REDNT, Naive Score, and Naive Count for the CTBN depicted in Figure 6.1a. Higher values of REDNT indicate CTBN states that are more likely to be sentry states. One should note that high-scoring states with few alarms (bold rows) are more interesting in our application as they correspond to states that occur before strong cascading behavior.

Experiment 2 The second synthetic experiment is based on the CTBN shown in Figure 6.2a which consists of a slow cycle (A, B, C) and a fast chain (D, E, F). In this CTBN, the sentry state is expected to be $\{A = 0, B = 0, C = 1, D = 0, E = 0, F = 0\}$. Figure 6.2b shows that this state triggers a fast sequence of alarms in the chain (D, E, F) and a slow sequence of alarms in the cycle (A, B, C). Estimates of the REDNT quantity are shown in Table 6.3 and they confirm that $\{A = 0, B = 0, C = 1, D = 0, E = 0, F = 0\}$ is a sentry state.

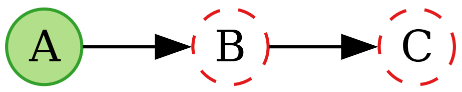
Comparison We compare the REDNT method to the naive approach proposed in Section 6.2. In synthetic data it is easier to identify the two parameters of the naive approach. Each synthetic experiment has only two transition rates and we can let the parameter λ_{ft} [1] be the median elapsed time between two consecutive events when combining events of all types. The parameter λ_{mcl} [2] can be determined based on the structure of the network. For instance, in the example in Figure 6.1a we expect a cascade to have at least two transitions,

$$\{A = 1, B = 0, C = 0\} \rightarrow \{A = 1, B = 1, C = 0\} \rightarrow \{A = 1, B = 1, C = 1\}.$$

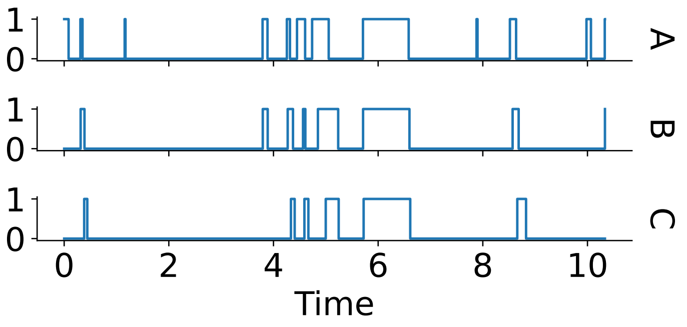
To identify sentry states using the naive approach we should simply identify the cascades of events and compute the fraction of times that observing a specific state coincides with the start of a cascade. We are interested in sentry states with a low number of active alarms. For this reason, we consider only states such that the number of active alarms is less than or equal to the size of the largest parent set in the true graph. The naive approach and the REDNT both produce a list where states are ordered from the most likely sentry state to the least likely. We compare the two approaches with the *Jaccard similarity* [56] using the K most likely sentry states. We tested our approach on the 6 different structures with different numbers of nodes depicted in Figures 6.1a, 6.4, 6.5, 6.6, 6.2a, and 6.7. Results are reported in Figure 6.3. In every experiment, the two methods share at least one state in their top-two lists. It is important to emphasize that the parameters of the naive method have been

[1]Threshold between a slow and a fast transition.

[2]It determines the minimum number of fast consecutive events to be considered a cascade.

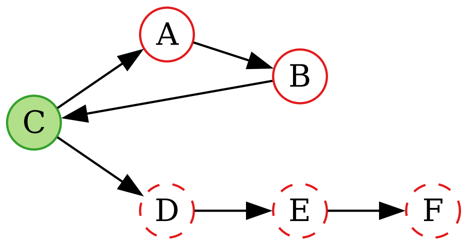


(a)

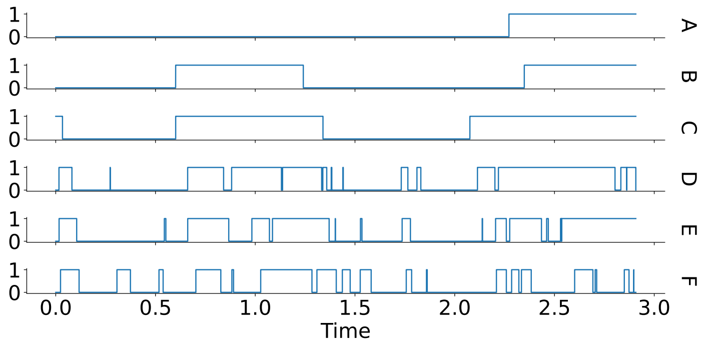


(b)

Figure 6.1: (a) depicts the graph $\mathcal{G}$ of a CTBN. Its CIMs are in Table 6.1. Slow processes are represented by solid line nodes, while fast processes are represented by dashed line nodes. Colors and filling describe the most likely sentry state, $s = (s_1, s_2, s_3)$, in this system: If a node has green border and it is filled, the corresponding alarm is 1 (*on*) in s. If a node has red border and it is not filled, the corresponding alarm is 0 (*off*) in s. (b) shows an example trajectory from the CTBN represented in Figure 6.1a. Each function in the plot represents the evolution of one of the three binary processes, A, B, and C.



(a)



(b)

Figure 6.2: (a) Graph $\mathcal{G}$ of a CTBN model consisting of six processes. The graph $\mathcal{G}$ contains the cycle (A, B, C) as well as the chain (D, E, F). See the caption of Figure 6.1 for an explanation of the node colors.

set knowing the length of cascades. Conversely, the REDNT method does not require this knowledge in order to identify sentry states.

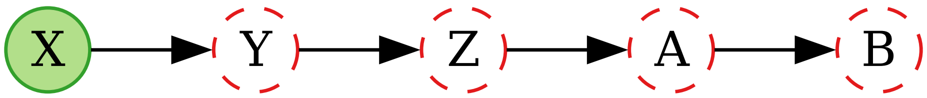


Figure 6.4: Graph $\mathcal{G}$ of a CTBN model consisting of a chain of five processes.

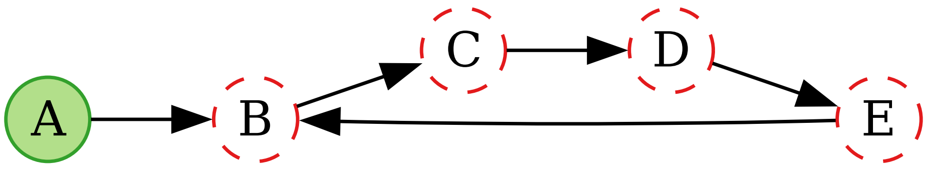


Figure 6.5: Graph $\mathcal{G}$ of a CTBN model consisting of five processes, including a cycle.

A	B	C	D	E	F	EDNT	REDNT	Naive Score	Naive Count
0	0	1	0	0	0	12.98	1.46	0.24	2172
0	0	0	1	0	0	11.33	1.28	0.25	2156
0	1	0	0	0	0	10.76	1.21	0.04	848
0	0	0	0	1	0	10.75	1.21	0.14	1533
1	0	0	0	0	0	10.24	1.15	0.02	341
0	0	0	0	0	1	9.90	1.12	0.03	651
0	0	0	0	0	0	8.87	1.0	0.01	426

Table 6.3: Values of EDNT, REDNT, Naive Score, and Naive Count of the states with at most one active alarm for the CTBN depicted in Figure 6.2a.

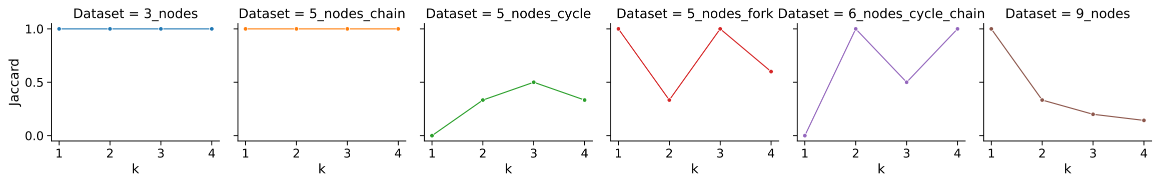


Figure 6.3: This figure reports the Jaccard similarity@k between the REDNT and the naive approach. The x-axis represents the number of states taken into account from the ordered lists generated by the two methods. The structures of the networks used for the experiments, in order of appearance in the plot, are depicted in Figures 6.1a, 6.4, 6.5, 6.6, 6.2a, and 6.7.

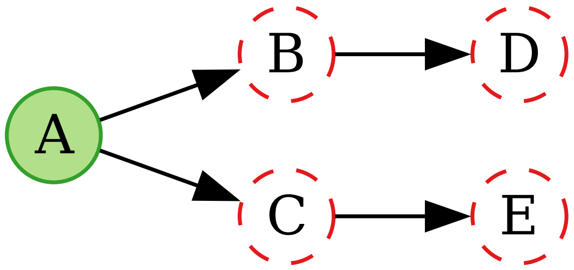


Figure 6.6: Graph $\mathcal{G}$ of a CTBN model consisting of five processes. The graph $\mathcal{G}$ contains a bifurcation after the root node A.

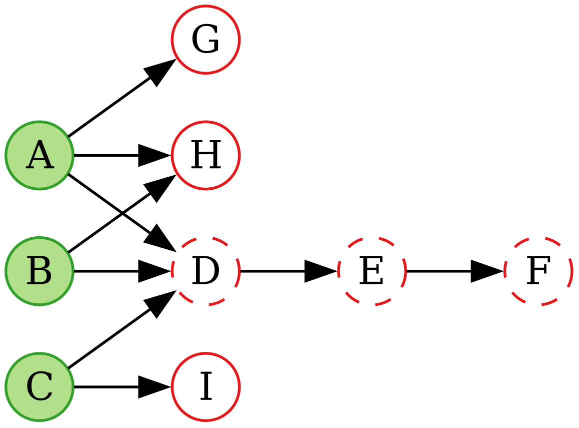


Figure 6.7: Graph $\mathcal{G}$ of a CTBN model consisting of nine processes and with a more complex structure. The sentry state has three active alarms.

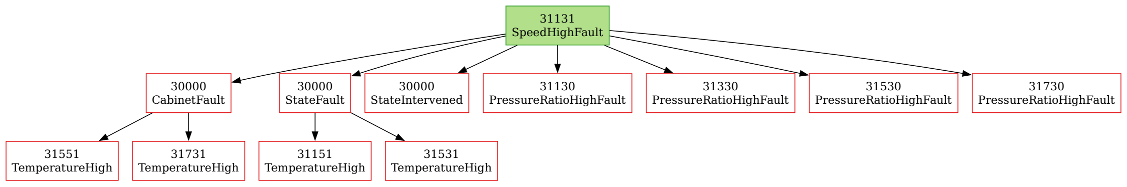


Figure 6.8: Graph $\mathcal{G}$ of the CTBN learned from the ESS data set. Nodes with green border and background represent active alarms, while nodes with red border and no background indicate inactive alarms.

6.5 ESS Data Set

The European Spallation Source ERIC (ESS) is a large research facility which is being built in Lund, Sweden. Its main components include a linear proton accelerator, a tungsten target, and a collection of neutron instruments [57]. It comprises numerous systems, including an integrated control system [58]. The facility has a goal of 95% availability, and state-of-the-art alarm handling may contribute to reaching this goal.

Operators of large facilities are often facing large quantities of data in real time, and good tools may help system understanding and support decision-making. Operators rely on alarm systems to warn them about unexpected behavior. However, alarm problems are common [59]. One example is that of *cascading alarms*. In large facilities, different alarms monitor different processes and when an issue occurs this may result in a large number of alarms that occur within a short time frame due to the interconnectedness of the different processes. Operators will often find it difficult to respond to such cascades, as hundreds or thousands of alarms may sound, making it difficult to identify the underlying issue.

The alarm system has two purposes. One, it should help operators foresee and mitigate fault situations. Two, it should help operators understand a fault situation. In this chapter, we illustrate how the methods we propose can help achieve these goals using data from the accelerator cryogenics plant at ESS, which has been in operation since 2020.

The data set consists of observations of 138 alarm processes from January 2020 to March 2023. No structure was provided, thus we use the score-based structure learning algorithm presented in [11]. We chose not to use the constraint-based algorithm [15] because, in the case of binary variables, it has been shown to be outperformed by the score-based algorithm. The score-based algorithm penalizes the size of the parent sets, leading to sparsity in the graphical structure. For this data, the learned graph is composed of disconnected components. We only present the results of applying the REDNT method for one of them. The most likely sentry state has the alarm *SpeedHighFault* set to *on* and everything else set to *off* (see Figure 6.8). We observe that the connected component which contains *SpeedHighFault* is a rooted directed tree, and that *SpeedHighFault* is the root. This means that an alarm in *SpeedHighFault* propagates along the directed paths in the tree. A CTBN assumes that at most one event occurs at any point in time. This is reasonable in this application because of the high sampling rate.

The four *PressureRatioHighFault* alarms in Figure 6.8 could be verified as consequences of the root cause *SpeedHighFault*, both from documentation and by an experienced operator. On the other hand, the alarms *StateIntervened*, *CabinetFault*, and *StateFault* were not evident from the documentation and were not expected to be related to the root alarm *SpeedHighFault*. If these connections are real, this information is relevant to operators, as they may look for reasons for this connection and enhance their process understanding. Moreover, the identified root alarm *SpeedHighFault* can be given a high priority to ensure that the operators will be made aware of a potential cascade starting from this alarm.

The CTBN was learned with no structural input, but using a prior distribution one can provide engineering knowledge to a Bayesian learning method. We believe that the graphical output can be shown to and interpreted by engineers; however, user studies are needed to validate this. Moreover, the sentry states that are identified can be presented to the operators. In the example shown in figure 6.8 the sentry state is covered by an already existing alarm, but if a more complex sentry state had been identified then an additional alarm would be needed. Using domain expertise, one can assign appropriate instructions to sentry states and these can be presented to operators when the machine reaches a fault state. When we have learned a CTBN from data, we may also compute the probability of reaching each of the sentry states from the current state. This can be provided to operators in real time to facilitate early mitigation. This should be studied in detail in future work.

6.6 Discussion

In this work we defined the concept of sentry states in CTBNs, and we presented a naive approach and a heuristic (REDNT) for identifying such sentry states. The synthetic experiments showed that REDNT can identify the configuration of the network from which a fast sequence of events starts. A key limitation is the fact that the REDNT heuristic is computed for each state, and the number of states is exponential in the number of nodes. However, the simplicity of its implementation and the effectiveness showed in the synthetic experiments make the REDNT heuristic attractive. Moreover, only states with few active alarms may be of interest, and this reduces the computational cost. The proposed heuristic assigns a score to each state in the state space of a CTBN; a possible extension of this work is the identification of the contribution of each process to the REDNT.

This work laid the theoretical groundwork for the implementation of online early warning systems based on the identification of sentry states. In practical implementations, a list of sentry states can be provided to domain experts for them to formulate appropriate actions in order to mitigate alarm cascades. This is left for future work. Moreover, the graph representing a learned CTBN indicates how the behavior of each alarm process depends on the states of the other alarm processes. This graph also represents conditional independence in the system. In future work, we hope to demonstrate that the intended end users, engineers and system operators, also find this graphical tool useful.

Chapter 7

Conclusions

This chapter summarizes the main contributions of our research activity during the last three years, while also clarifying their main limitations. Directions for further research activities, as well as potentially interesting extensions to the framework of continuous time Bayesian networks, are presented and discussed.

The main goals pursued during the PhD can be summarized as follows: *i*) to design and develop new algorithms for learning the structure of a continuous time Bayesian network, when temporal data is available, *ii*) to design and develop new algorithms that, starting from a learnt continuous time Bayesian network model, are capable to extract knowledge about the underlying data-generating process of the system under study.

7.1 Main Contributions

The main contributions which we developed during the PhD and that are presented in this dissertation are as follows:

- **Chapter 4 - CTPC Algorithm**: presents and describes the first constraint based structure learning algorithm for continuous time Bayesian networks. In this work, we identified three independence tests suitable for the specific characterization of the continuous time Bayesian network model while exploiting the conditional independence definition provided by [11]. In detail, we modified the PC algorithm, initially proposed for Bayesian networks, to fit the continuous time Bayesian networks' framework. A study of the time complexity of the proposed structure learning algorithm has been carried out. This study proved that the complexity of this new algorithm is comparable to the complexity of the score based algorithm, which is the only other structure learning algorithm made available in the specialized literature for learning the structure of a continuous time Bayesian network. The chapter also includes the results of a rich set of synthetic numerical experiments carried out to evaluate the new algorithm and to compare its performance to that of the CTSS algorithm. According to the results of these synthetic numerical experiments, it is possible to conclude that usually, the new algorithm (CTPC) performs better than the score based (CTSS) counterpart in the case of variables which can take more values. However, the CTPC algorithm suffers the following main limitations:
  - Although, as demonstrated in [14] and [15], CTPC and CTSS share the same computational complexity, the CTSS algorithm allows limiting the size of the parent set, thus reducing its computational complexity at the cost of a sparser learned network, while the same does not hold true for the CTPC algorithm.
  - The high number of sequential hypothesis tests which have to be carried out to identify a conditional (in)dependence between two variables, makes the CTPC algorithm to suffer the well-known multiple comparisons' problem.
- **Chapter 5 - structure learning for Multi-CTBNC**: adapts the CTPC algorithm to multidimensional continuous time Bayesian network classifiers. New structure learning algorithms,

specifically designed for solving the supervised classification problem, are developed. In particular, three new structure learning algorithms, for the multidimensional continuous time Bayesian network classifiers, are developed:

- A naive adaptation of the CTPC to solve a supervised classification problem.
- The MB-CTPC algorithm: an adaptation of the CTPC algorithm, specifically developed for the multidimensional continuous time Bayesian network classifiers. This algorithm exploits the information gathered from learning the bridge-subgraph, to drastically reduce the number of conditional independence tests to be performed, at the cost of a slight decrease in classification performance.
- A set of hybrid algorithms that combine the strengths of constraint based and score based algorithms. These are also the first hybrid algorithms presented for both multidimensional continuous time Bayesian network classifiers and for continuous time Bayesian networks as well.

The CTPC naive adaptation for multidimensional continuous time Bayesian network classifiers shares the same limitations of the original CTPC algorithm. Those limitations are shared also by the MB-CTBN algorithm. The hybrid algorithm manages to overcome at least in part the problem of multiple hypothesis tests, since, it limits the size of the parent set, and thus as a consequence it also limits the number of tests to be carried out.

- **Chapter 6 - Sentry State for CTBN**: defines the concept of *sentry state* for continuous time Bayesian networks and describes two algorithms to identify these states. This contribution extracts useful information from a given continuous time Bayesian network with the goal to better understand the underlying data generating process and thus to provide the domain expert with useful insights. More precisely, this contribution aims to identify the states from which a ripple effect develops. This information is extremely useful for the alarm network by helping to identify those states which represent the most dangerous alarm configurations. The main drawback of this contribution is that, even if the REDNT concept strogly relay on the factorization of the CTBN, neither the definition nor the algorithm take any advantage from the structure of the CTBN. As a result, the proposed algorithm is exponential in the number of nodes, thus making even medium-sized networks computationally intractable.

7.2 Future works

During the development of the contributions presented in this dissertation, I identified a set of possible extension and future works that I will now list briefly.

Multiple comparisons' problem in CTPC. Statistics is well aware of the multiple comparison problem. This problem occurs whenever simultaneously testing multiple hypotheses. In the case of the CTPC algorithm, to identify the conditional independence between two nodes given a separation set, a batch of hypothesis tests are performed and all of them must not reject the null hypothesis. Thus, the significant level selected for one test does not directly reflect the Type-I error for the identification of a conditional independence. Consequently, the multiple comparison problem, significantly affects the algorithm by greatly increasing the associated Type-I error. Various solutions have been proposed in the specialized literature [60] which can be investigated to address the multiple comparison problem for the specific case of the CTPC algorithm.

New hypothesis tests for CTPC. The current version of the CTPC algorithm consists of three hypothesis tests: two are related to the multinomial distribution and the third one relates to the exponential distribution. However, different kinds of hypothesis tests and also different definitions of conditional independence for CTBNs could be explored.

Adaptation of HITON-PC to CTBN. The HITON-PC algorithm presented in [61] introduces a new algorithm for identifying the Markov Blanket of a given variable. Although this algorithm was designed for the variable selection task, the insights presented in the article could be transferred to structure learning by decreasing the number of hypothesis tests to be performed, thus bringing to a more efficient version of the CTPC algorithm.

New algorithms for Sentry State identification. For the CTMP framework, closed form algorithms for evaluating a reward function [9] are available. However, the only implementation currently available for sentry state identification is based on Monte Carlo Simulation. New approaches could be explored by using an adapted version of the closed-form algorithm for exploiting the CTBN structure.

Explore new definitions for the Sentry State identification problem. The current definition of sentry state requires identifying a specific state for all the variables in the CTBN. A different approach worth exploring might be to identify the node or set of nodes having the greatest impact to start a chain of events.

Continuous Variables for CTBN. The continuous time Markov processes and the continuous time Bayesian networks frameworks both model the evolution of a finite state process over continuous time. This is a strong limitation, because many real world phenomena deal with continuous quantities evolving in continuous time. The easiest solution consists in performing a discretization of the continuous quantities, and then in applying standard algorithms for continuous time Markov processes and continuous time Bayesian networks. However, the discretization process is extremely critical and can strongly bias the real distribution. According to the specialized literature on Bayesian networks, the following two approaches can be pursued to overcome such a limitation:

- *Introducing continuous nodes.* This approach is applied by conditional linear Gaussian Bayesian networks [62]. An adaptation of this approach to continuous time Bayesian networks requires to identify an appropriate distribution for describing continuous variables and to rethink many inference algorithms.
- *Introducing the discretization in the learning process.* This approach is well studied for standard Bayesian networks [63]. An adaptation of this approach requires only to rethink the learning phase of continuous time Markov processes and continuous time Bayesian networks.

Finally, the specialized literature on state space models and CTMPs addresses this issue. The papers [64], [65] addressed the problem of continuous variables for state space model by applying an automatic discretization. Even if this approach is promising, it is still a discretization, and furthermore, it requires the discretized function to be bounded. Another interesting paper is [66], where the authors pointed out the strong link between stochastic differential equations and CTMP, showing that, for a specific subset of problems (i.e., the SIR model), it is possible to describe the phenomena both with CTMP and stochastic differential equations.

Bibliography

[1] Tyler Forrester, Mark Harris, Jacob Senecal, and John Sheppard. "Continuous Time Bayesian Networks in prognosis and health management of centrifugal pumps". In: *Proceedings of the Annual Conference of the PHM Society*. Vol. 11. 1. 2019.

[2] Manxia Liu, Fabio Stella, Arjen Hommersom, Peter JF Lucas, Lonneke Boer, and Erik Bischoff. "A comparison between discrete and continuous time Bayesian networks in learning from clinical time series data with irregularity". In: *Artificial intelligence in medicine* 95 (2019), pp. 104–117.

[3] J Ion Titapiccolo, Manuela Ferrario, Sergio Cerutti, Maria G Signorini, Carlo Barbieri, Flavio Mari, and Emanuele Gatti. "Mining medical data to develop clinical decision making tools in hemodialysis". In: *2012 IEEE 12th international conference on data mining workshops*. IEEE. 2012, pp. 99–106.

[4] Alessandro Bregoli, Luca Neri, Max Botler, Erik Schumacher, Ricardo Peralta, Pedro Ponce, Stefano Stuard, and Francesco Bellocchio. "Personalized Arterovenous Fistula Management through Utility Maximization with Influence Diagram." In: *SMARTERCARE@ AI* IA*. 2021, pp. 61–66.

[5] Alessandro Bregoli, Karin Rathsman, Marco Scutari, Fabio Stella, and Søren Wengel Mogensen. "Analyzing Complex Systems with Cascades Using Continuous-Time Bayesian Networks". In: *arXiv preprint arXiv:2308.10606* (2023).

[6] Mahshid Rahnamay-Naeini, Zhuoyao Wang, Nasir Ghani, Andrea Mammoli, and Majeed M Hayat. "Stochastic analysis of cascading-failure dynamics in power grids". In: *IEEE Transactions on Power Systems* 29.4 (2014), pp. 1767–1779.

[7] Paul A Gagniuc. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

[8] Arthur Albert. "Estimating the infinitesimal generator of a continuous time, finite state Markov process". In: *The Annals of Mathematical Statistics* (1962), pp. 727–753.

[9] Xianping Guo and Onésimo Hernández-Lerma. "Continuous-time Markov decision processes". In: *Continuous-Time Markov Decision Processes*. Springer, 2009, pp. 9–18.

[10] Kevin Patrick Murphy. *Dynamic bayesian networks: representation, inference and learning*. University of California, Berkeley, 2002.

[11] Uri D Nodelman. "Continuous time Bayesian networks". PhD thesis. Stanford University, 2007.

[12] Daryl J Daley, David Vere-Jones, et al. *An introduction to the theory of point processes: volume I: elementary theory and methods*. Springer, 2003.

[13] Ronald A Howard and James E Matheson. "Influence diagrams". In: *Decision Analysis* 2.3 (2005), pp. 127–143.

[14] Alessandro Bregoli, Marco Scutari, and Fabio Stella. "Constraing-Based Learning for Continous-Time Bayesian Networks". In: *International Conference on Probabilistic Graphical Models*. PMLR. 2020, pp. 41–52.

[15] Alessandro Bregoli, Marco Scutari, and Fabio Stella. "A constraint-based algorithm for the structural learning of continuous-time Bayesian networks". In: *International Journal of Approximate Reasoning* 138 (2021), pp. 105–122.

[16] Carlos Villa-Blanco, Alessandro Bregoli, Concha Bielza, Pedro Larranaga, and Fabio Stella. "Structure learning algorithms for multidimensional continuous-time Bayesian network classifiers". In: *International Conference on Probabilistic Graphical Models*. PMLR. 2022, pp. 313–324.

[17] Carlos Villa-Blanco, Alessandro Bregoli, Concha Bielza, Pedro Larrañaga, and Fabio Stella. "Constraint-based and hybrid structure learning of multidimensional continuous-time Bayesian network classifiers". In: *International Journal of Approximate Reasoning* 159 (2023), p. 108945.

[18] Christian R Shelton and Gianfranco Ciardo. "Tutorial on structured continuous-time Markov processes". In: *Journal of Artificial Intelligence Research* 51 (2014), pp. 725–778.

[19] Cleve Moler and Charles Van Loan. "Nineteen dubious ways to compute the exponential of a matrix". In: *SIAM review* 20.4 (1978), pp. 801–836.

[20] Liessman Sturlaugson and John W. Sheppard. "Uncertain and negative evidence in continuous time Bayesian networks". In: *International Journal of Approximate Reasoning* 70 (Mar. 2016), pp. 99–122. ISSN: 0888-613X. DOI: 10.1016/j.ijar.2015.12.013.

[21] Martin Bicher, Matthias Wastian, Dominik Brunmeir, and Niki Popper. "Review on Monte Carlo Simulation Stopping Rules: How Many Samples Are Really Enough?" In: *Simul. Notes Eur.* 32.1 (2022), pp. 1–8.

[22] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

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

[24] Judea Pearl. "Influence diagrams—Historical and personal perspectives". In: *Decision Analysis* 2.4 (2005), pp. 232–234.

[25] Paul Dagum, Adam Galper, and Eric Horvitz. "Dynamic network models for forecasting". In: *Uncertainty in artificial intelligence*. Elsevier. 1992, pp. 41–48.

[26] William J Anderson. *Continuous-time Markov chains: An applications-oriented approach*. Springer Science & Business Media, 2012.

[27] Shyamsundar Rajaram, Thore Graepel, and Ralf Herbrich. "Poisson-Networks: A Model for Structured Poisson Processes". In: *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*. Ed. by Robert G. Cowell and Zoubin Ghahramani. Vol. R5. Proceedings of Machine Learning Research. Reissued by PMLR on 30 March 2021. PMLR, June 2005, pp. 277–284. URL: http://proceedings.mlr.press/r5/rajaram05a.html.

[28] Aleksandr Simma and Michael I. Jordan. "Modeling Events with Cascades of Poisson Processes". In: *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*. UAI'10. Catalina Island, CA: AUAI Press, 2010, pp. 546–555. ISBN: 9780974903965.

[29] Asela Gunawardana, Christopher Meek, and Puyang Xu. "A Model for Temporal Dependencies in Event Streams". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper/2011/file/c9f95a0a5af052bffce5c89917335f67-Paper.pdf.

[30] Jeremy C. Weiss and David Page. "Forest-Based Point Process for Event Prediction from Electronic Health Records". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 547–562. ISBN: 978-3-642-40994-3.

[31] Vanessa Didelez. "Graphical models for marked point processes based on local independence". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 70.1 (2008), pp. 245–264.

[32] Kin Fai Kan and Christian R Shelton. "Solving Structured Continuous-Time Markov Decision Processes." In: *ISAIM*. 2008.

[33] Liessman Sturlaugson, Logan Perreault, and John W Sheppard. "Factored performance functions and decision making in continuous time Bayesian networks". In: *Journal of Applied Logic* 22 (July 2017), pp. 28–45. ISSN: 1570-8683. DOI: 10.1016/j.jal.2016.11.030.

[34] Fabio Stella and Y Amer. "Continuous time Bayesian network classifiers". In: *Journal of biomedical informatics* 45.6 (2012), pp. 1108–1119.

[35] Carlos Villa-Blanco, Pedro Larrañaga, and Concha Bielza. “Multidimensional continuous time Bayesian network classifiers”. In: *International Journal of Intelligent Systems* 36.12 (2021), pp. 7839–7866.

[36] M. Scutari, C. E. Graafland, and J. M. Gutiérrez. “Who Learns Better Bayesian Network Structures: Accuracy and Speed of Structure Learning Algorithms”. In: *International Journal of Approximate Reasoning* 115 (2019), pp. 235–253.

[37] Mauro Scanagatta, A. Salmerón, and F. Stella. “A survey on Bayesian network structure learning from data”. In: *Progress in Artificial Intelligence* (2019), pp. 1–15.

[38] Vanessa Didelez. “Asymmetric separation for local independence graphs”. In: *arXiv preprint arXiv:1206.6841* (2012).

[39] Christopher Meek. “Toward Learning Graphical and Causal Process Models.” In: *CI@ UAI*. 2014, pp. 43–48.

[40] Søren Wengel Mogensen, Daniel Malinsky, and Niels Richard Hansen. “Causal Learning for Partially Observed Stochastic Dynamical Systems.” In: *UAI*. 2018, pp. 350–360.

[41] J. Pearl and T. Verma. “A Theory of Inferred Causation”. In: *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers Inc., 1991, pp. 441–452.

[42] D. Colombo and M. H. Maathuis. “Order-Independent Constraint-Based Causal Structure Learning”. In: *Journal of Machine Learning Research* 15 (2014), pp. 3921–3962.

[43] U. Nodelman, C.R. Shelton, and D. Koller. “Learning Continuous Time Bayesian Networks”. In: *19th Conference on Uncertainty in AI (UAI)*. 2003, pp. 451–458.

[44] Elisa T Lee and John Wang. *Statistical methods for survival data analysis*. Vol. 476. John Wiley & Sons, 2003.

[45] B. Mitchell. “A Comparison of Chi-Square and Kolmogorov-Smirnov Tests”. In: *The Royal Geographical Society* 3.4 (1971), pp. 237–241.

[46] Uffe Bro Kjærulff and Anders Læsø Madsen. *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*. Dansk. 2. Information Science and Statistics. Springer VS, 2013. ISBN: 978-1-4614-5103-7.

[47] Uri Nodelman, Christian R Shelton, and Daphne Koller. “Expectation maximization and complex duration distributions for continuous time Bayesian networks”. In: *arXiv preprint arXiv:1207.1402* (2012).

[48] Dominik Linzner, Michael Schmidt, and Heinz Koeppl. “Scalable Structure Learning of Continuous-Time Bayesian Networks from Incomplete Data”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 3741–3751. URL: https://proceedings.neurips.cc/paper/2019/hash/d8330f857a17c53d217014ee776bfd50-Abstract.html.

[49] Institute For Social University Of Essex. *British Household Panel Survey: Waves 1-18, 1991-2009*. 2021. DOI: 10.5255/UKDA-SN-5151-2.

[50] Silvia Acid and Luis M de Campos. “A hybrid methodology for learning belief networks: BENEDICT”. In: *International Journal of Approximate Reasoning* 27.3 (2001), pp. 235–262.

[51] Hui Liu, Shuigeng Zhou, Wai Lam, and Jihong Guan. “A new hybrid method for learning bayesian networks: Separation and reunion”. In: *Knowledge-Based Systems* 121 (2017), pp. 185–197.

[52] Ghada Trabelsi, Philippe Leray, Mounir Ben Ayed, and Adel Mohamed Alimi. “Dynamic MMHC: A local search algorithm for dynamic Bayesian network structure learning”. In: *Advances in Intelligent Data Analysis XII: 12th International Symposium, IDA 2013, London, UK, October 17-19, 2013. Proceedings 12*. Springer. 2013, pp. 392–403.

[53] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. “The max-min hill-climbing Bayesian network structure learning algorithm”. In: *Machine learning* 65 (2006), pp. 31–78.

[54] Concha Bielza, Guangdi Li, and Pedro Larranaga. "Multi-dimensional classification with Bayesian networks". In: *International Journal of Approximate Reasoning* 52.6 (2011), pp. 705–727.

[55] Jose A Fernandes, Jose A Lozano, Iñaki Inza, Xabier Irigoien, Aritz Pérez, and Juan D Rodríguez. "Supervised pre-processing approaches in multiple class variables classification for fish recruitment forecasting". In: *Environmental modelling & software* 40 (2013), pp. 245–254.

[56] Allan H Murphy. "The Finley affair: A signal event in the history of forecast verification". In: *Weather and forecasting* 11.1 (1996), pp. 3–20.

[57] ESS. *European Spallation Source.* Last accessed 2023-04-25. URL: https://europeanspallationsource.se/about.

[58] EPICS. *The Experimental Physics and Industrial Control System.* Last accessed 2023-04-25. URL: https://epics-controls.org/about-epics/.

[59] B.R. Hollifield and E. Habibi. *Alarm Management: A Comprehensive Guide : Practical and Proven Methods to Optimize the Performance of Alarm Management Systems.* International Society of Automation, 2011. ISBN: 9781628700312. URL: https://books.google.se/books?id=UuSMswEACAAJ.

[60] Juliet Popper Shaffer. "Multiple hypothesis testing". In: *Annual review of psychology* 46.1 (1995), pp. 561–584.

[61] Constantin F Aliferis, Ioannis Tsamardinos, and Alexander Statnikov. "HITON: a novel Markov Blanket algorithm for optimal variable selection". In: *AMIA annual symposium proceedings.* Vol. 2003. American Medical Informatics Association. 2003, p. 21.

[62] Steffen L Lauritzen. "Propagation of probabilities, means, and variances in mixed graphical association models". In: *Journal of the American Statistical Association* 87.420 (1992), pp. 1098–1108.

[63] Yi-Chun Chen, Tim A Wheeler, and Mykel J Kochenderfer. "Learning discrete Bayesian networks from continuous data". In: *Journal of Artificial Intelligence Research* 59 (2017), pp. 103–132.

[64] Leland E. Farmer. "The Discretization Filter: A Simple Way to Estimate Nonlinear State Space Models". In: *SSRN Electronic Journal* (2017). ISSN: 1556-5068. DOI: 10.2139/ssrn.2780166.

[65] Leland E. Farmer and Alexis Akira Toda. "Discretizing nonlinear, non-Gaussian Markov processes with exact conditional moments: Discretizing Markov processes". In: *Quantitative Economics* 8.2 (July 2017), pp. 651–683. ISSN: 1759-7323. DOI: 10.3982/qe737.

[66] Amy J. Ekanayake and Linda J. S. Allen. "Comparison of Markov Chain and Stochastic Differential Equation Population Models Under Higher-Order Moment Closure Approximations". In: *Stochastic Analysis and Applications* 28.6 (Oct. 2010), pp. 907–927. ISSN: 0736-2994. DOI: 10.1080/07362990903415882.

Appendices

Appendix A

CTPC computational complexity

To compute the time complexity of CTPC, first recall that:

- ψ is the number of transitions occurring in the data set;
- $\mathbf{X}$ is the set of nodes of the CTBN;
- n is the number of nodes of the CTBN;
- $X_k \in \mathbf{X}$ is the node associated with the $k-th$ random variable;
- $|X_k|$ is the cardinality of the node X_k;
- $\eta = max(\{|X_k| : X_k \in \mathbf{X}\})$ is the maximum node cardinality present in the network;
- $pa(X_k) \subset \mathbf{X}$ is the parent set of node X_k;
- $|pa(X_k)|$ is the size of the parent set $pa(X_k)$;
- $\rho = max(\{|pa(X_k)| : X_k \in \mathbf{X}\})$ is the maximum parent set size present in the network;
- $T,\ N$: are the sets of sufficient statistics.

The structure of the CTPC algorithm can be represented with a series of blocks nested within each other (Figure A.1):

1. **Compute the CIM**: learn the CIM of a node from a data set given a separating set.
2. **Test Independence**: test the independence between two nodes given a separating set.
3. **Learn the parent set (one node)**: learn the parent set of a node.

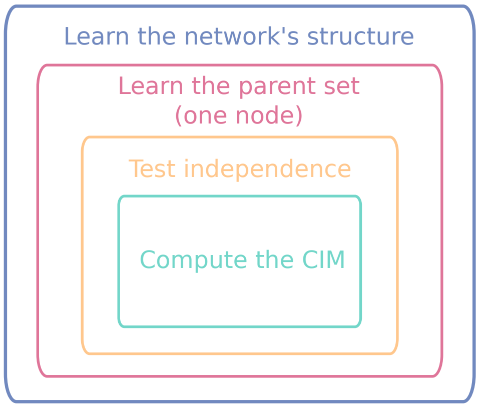


Figure A.1: Conceptual model of the CTPC algorithm.

4. **Learn the network's structure**: learn the parent set for each node in the network.

We derive the complexity of the CTPC algorithm in a bottom-up fashion, that is, by using the complexity of the inner blocks to derive the complexity of the outer blocks.

A.1 Compute the CIM

This is the innermost box in Figure A.1. The tasks requiring the largest number of operations, which determine the leading terms of the computational complexity, are the following:

- Computing the sufficient statistics T and M over the subset of the dataset containing the analyzed node and its parent set: $O((\rho + 1) \cdot \psi)$.

  This complexity rise from the fact that, to learn the sufficient statistics, we need to scroll to the entire dataset composed by ψ transitions and check if the variable for which we are learning the CIM or one of its parents (at most ρ parents) transitioned to a new state.

- Computing the CIM Q, given the sufficient statistics T and N. For this task we need to compute an intensity matrix of dimension at most η^2 (where η is the biggest domain for a variable in the CTBN under analysis) for each combination of possible realizations of the parent set $O(\eta^\rho)$. The resulting complexity is: $O(\eta^\rho \cdot \eta^2) = O(\eta^{\rho+2})$

The overall time complexity of this box can be computed as a summation of the two componets and it is equal to: $O((\rho + 1) \cdot \psi) + \eta^{\rho+2})$.

A.1.1 Test Independence

For this box, we characterize separately the best case and the worst case computational complexity. Recall that, in order to access the conditional independence of the variable $X_i \in \mathbf{X}$ given the variable $X_j \in \mathbf{X}$ and a separation set $\mathbf{S} \subset \mathbf{X}$, we need to learn the CIM of the variable X_i given the separation set and the potetial parent, then we need to perform two classes of hypothesis tests: one to verify the *null time to transition hypothesis* and one to verify the *null state-to-state transition hypothesis*. If all the test accept the null hypothesis, we can say that the two variables are independent given the separation set.

- **Best Case**: when the first hypothesis test fails. The most complex operations are:
  - The computation of the CIM: $O((\rho + 1) \cdot \psi + \eta^{\rho+2})$.
  - The computation of one test: $O(1)$.

  Since the computational complexity of one test is $O(1)$, we can ignore it in the overall complexity of this case.

- **Worst Case**: when no hypothesis test fails. The most complex operations are:
  - The computation of the CIM: $O((\rho + 1) \cdot \psi + \eta^{\rho+2})$.
  - Performing all the hypothesis tests, η t-tests (each with complexity $O(1)$) and η chi-square tests (each with complexity $O(\eta)$) for each possible value of the parent set η^ρ: $O(\eta^{\rho+1} + \eta^{\rho+2}) = O(\eta^{\rho+2})$.

  These two operations are performed in sequence. It follows that to identify the overall complexity of this case it is sufficient to add the complexities of the two operations.

Therefore, the complexity of the independence test can be summarized as:

Case	**Complexity**
Best Case	$O((\rho + 1) \cdot \psi + \eta^{\rho+2})$
Worst case	$O((\rho + 1) \cdot \psi + 2 \cdot \eta^{\rho+2})$

A.2 Learn the parent set (one node)

For this box of the algorithm, we distinguish three cases:

- **Best Case**: the target node has no parents and the hypothesis tests correctly identify it, thus the most complex operation is:
  - The independence test (worst case): $O(2 \cdot \eta^{\rho+2} + (\rho + 1) \cdot \psi)$.

    If the node under analysis has no parents, it follows that, for this case, $\rho = 0$ because ρ identify the number of parents. We need to test all the possible independences with empty separation set, hence we perform $n - 1 \rightarrow O(n)$ tests (one for each variable in the CTBN).

    The overall complexity of this case can be computed by multiplying the complexity of the worst case for the hypothesis test by the number of test performed and it is equal to: $O(n \cdot (2 \cdot \eta^3 + 2 \cdot \psi))$.
- **Worst case**: all nodes are parents of the current node, thus the most complex operation is:
  - The independence test (best case): $O((\rho + 1) \cdot \tau + \eta^{\rho+2})$. The target node depends on all other nodes ($\rho = n - 1$). Indeed, even if there is a dependence, it is possible that an hypothesis test does not identify it by chance (Type II error) or because that specific exponential distribution or multinomial distibution is not affected by the state of the parent under analysis. However, for simpliciy, we assume that the first independence test always fail rejecting the independece hypothesis. The algorithm evaluates all possible parent sets, the number of which can be calculated as follows:

    $$O\left(\sum_{r=1}^{n-1} \frac{(n-1)!}{r! \cdot (n-1-r)!} \cdot (\eta^{r+2} + (r+1) \cdot \psi)\right)$$

    which simplifies to $O(2^n \cdot (\eta^{n+1} + n \cdot \psi))$ since

    $$O(\eta^{r+2} + r \cdot \psi) \sim O(\eta^n + n \cdot \psi).$$

    and

    $$\sum_{r=1}^{n-1} \frac{(n-1)!}{r! \cdot (n-1-r)!} \rightarrow O(2^n)$$
- **General case**: The General case is similar to the worst case with one important difference: ρ can assumes any value less than $n - 1$ instead of being equal to $n - 1$. The most complex operations are:
  - Independence test (best case): $O((\rho + 1) \cdot \psi + \eta^{\rho+2})$. All the tests until that conditional on the true parent set of size ρ fail. Therefore, the algorithm evaluates all possible parent sets of size $\leqslant \rho + 1$, resulting in the following complexity:

    $$O\left(\sum_{r=1}^{\rho+1} \frac{(n-1)!}{r! \cdot (n-1-r)!} \cdot (\eta^{r+2} + (r+1) \cdot \psi)\right). \tag{A.1}$$

  Noting that:

  $$O(\eta^{r+2} + (r+1) \cdot \psi) \rightarrow O(\eta^{\rho+3} + (\rho+2) \cdot \psi),$$

  $$O\left(\sum_{r=1}^{\rho+1} \frac{(n-1)!}{r! \cdot (n-1-r)!}\right) \rightarrow O\left(\frac{n^{\rho+1}}{(\rho+1)!}\right).$$

  the computational complexity in (A.1) simplifies to:

  $$O\left(\frac{n^{\rho+1}}{(\rho+1)!} \cdot (\eta^{\rho+3} + (\rho+2) \cdot \psi)\right)$$

In conclusion, the complexity of the three cases can be summarized as follows:

Case	**Complexity**
Best Case	$O(n \cdot (2 \cdot \eta^3 + 2 \cdot \psi))$
Worst case	$O(2^n \cdot (\eta^n + n \cdot \psi))$
General case	$O\left(\frac{n^{\rho+1}}{(\rho+1)!} \cdot (\eta^{\rho+3} + (\rho+2) \cdot \psi)\right)$

A.2.1 Learn the network's structure

In Section A.2 we presented the computational complexity of learning the parent set of a single node. In order to have the complexity of the CTPC algorithm we just need to multiply those equations by n. The computational complexity of the CTPC algorithm can be summarized as follows:

Case	**Complexity**
Best Case	$O(n^2 \cdot (2 \cdot \eta^3 + 2 \cdot \psi))$
Worst case	$O(n \cdot 2^n \cdot (\eta^n + n \cdot \psi))$
General case	$O\left(\frac{n^{\rho+2}}{(\rho+1)!} \cdot (\eta^{\rho+3} + (\rho+2) \cdot \psi)\right)$