

# Structure Learning Algorithms for Multidimensional Continuous-Time Bayesian Network Classifiers

**Carlos Villa-Blanco**<sup>1</sup>

CARLOS.VILLA@UPM.ES

**Alessandro Bregoli**<sup>2</sup>

A.BREGOLI1@CAMPUS.UNIMIB.IT

**Concha Bielza**<sup>1</sup>

MCBIELZA@FI.UPM.ES

**Pedro Larrañaga**<sup>1</sup>

PEDRO.LARRANAGA@FI.UPM.ES

**Fabio Stella**<sup>2</sup>

FABIO.STELLA@UNIMIB.IT

*Computational Intelligence Group, Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Madrid, Spain*<sup>1</sup>

*Models and Algorithms for Data and Text Mining, Department of Informatics, Systems and Communication, University of Milan-Bicocca, Milan, Italy*<sup>2</sup>

## Abstract

Learning the structure of continuous-time Bayesian networks directly from data has traditionally been performed using score-based structure learning algorithms. Only recently has a constraint-based method been proposed, proving to be more suitable under specific settings, as in modelling systems with variables having more than two states. As a result, studying diverse structure learning algorithms is essential to learn the most appropriate models according to data characteristics and task-related priorities, such as learning speed or accuracy. This article proposes several alternatives of such algorithms for learning multidimensional continuous-time Bayesian network classifiers, introducing for the first time constraint-based and hybrid algorithms for these models. Nevertheless, these contributions also apply to the simpler one-dimensional classification problem for which only score-based solutions exist in the literature. More specifically, the aforementioned constraint-based structure learning algorithm is first adapted to the supervised classification setting. Then, a novel algorithm of this kind, specifically tailored for the multidimensional classification problem, is presented to improve the learning times for the induction of multidimensional classifiers. Finally, a hybrid algorithm is introduced, attempting to combine the strengths of the score- and constraint-based approaches. Experiments with synthetic data are performed not only to validate the capabilities of the proposed algorithms but also to conduct a comparative study of the available structure learning algorithms.

**Keywords:** Continuous-time Bayesian networks; structure learning algorithms; Bayesian network classifiers; multidimensional classification; learning from data.

## 1. Introduction

Automated analysis and processing of time series data are present in practically any field of study, such as engineering, medicine, economics or signal processing, and they are essential to understand and automatise many of their processes. These data are characterised by their large size and high dimensionality (Fu, 2011), which are expected to keep growing as they become easier to collect and at higher granularity. Therefore, studying algorithms capable of modelling these data more accurately and efficiently is becoming more necessary.

In this article, we focus on learning continuous-time Bayesian network (CTBN) structures from data and, more specifically, structures of CTBNs that can be applied to the multidimensional classification of time series. CTBN classifiers have been successfully employed for problems such as post-stroke rehabilitation (Codecasa and Stella, 2014) or detecting energy consumption states (Villa-Blanco et al., 2021). Nevertheless, the study of structure learning algorithms for these classifiers remains largely unexplored in the literature.

The main contributions of this study are the following:

- (i) the development of the first constraint-based structure learning algorithm for CTBN classifiers, which is conceived to learn multidimensional continuous-time Bayesian network classifiers (Multi-CTBNCs);
- (ii) the introduction of the first hybrid structure learning algorithm for Multi-CTBNCs;
- (iii) a comprehensive comparative study to evaluate the strengths and weaknesses of the state-of-the-art algorithms and those proposed in this work;
- (iv) the development and integration of the presented algorithms into a software tool introduced in Villa-Blanco et al. (2021).

The remainder of this article is as follows. Section 2 reviews fundamental concepts. Sections 3 and 4 introduce novel constraint-based and hybrid algorithms for Multi-CTBNCs, respectively. Section 5 presents experiments and discusses the results of multiple structure learning algorithms. Section 6 concludes the article and discusses future research lines.

## 2. Fundamentals

A Bayesian network (BN) is a probabilistic graphical model (PGM) designed for reasoning about static processes (Pearl, 1988). Thus, such a model is unsuitable when a system exhibits temporal behaviour. For this reason, the CTBN has been proposed to represent the dynamics of continuous-time and discrete-state stochastic processes (Nodelman et al., 2002), which are described by finite state, continuous-time homogeneous Markov processes through intensity matrices. A factored representation, known as conditional Markov process, is used to be able to model more extensive networks.

**Definition 1** (*Conditional Markov process*). A conditional Markov process is a type of inhomogeneous Markov process whose intensity matrix changes over time as a function of some conditioning variables' state. Given a random variable  $X_i$  with sample space  $\Omega_{X_i} = \{x_1, \dots, x_k\}$ , a conditional intensity matrix (CIM)  $\mathcal{Q}_{X_i}^{\mathbf{pa}(X_i)}$  describes its temporal dynamics. A CIM is a set of homogeneous intensity matrices  $\mathcal{Q}_{X_i}^{\mathbf{pa}(X_i)}$ , each encoding the dynamics of  $X_i$  given a state  $\mathbf{pa}(X_i)$  of its parents  $\mathbf{Pa}(X_i)$  in a directed graph  $\mathcal{G}$ :

$$\mathcal{Q}_{X_i}^{\mathbf{pa}(X_i)} = \begin{bmatrix} -q_{x_1}^{\mathbf{pa}(X_i)} & q_{x_1x_2}^{\mathbf{pa}(X_i)} & \dots & q_{x_1x_k}^{\mathbf{pa}(X_i)} \\ q_{x_2x_1}^{\mathbf{pa}(X_i)} & -q_{x_2}^{\mathbf{pa}(X_i)} & \dots & q_{x_2x_k}^{\mathbf{pa}(X_i)} \\ \vdots & \dots & \ddots & \vdots \\ q_{x_kx_1}^{\mathbf{pa}(X_i)} & q_{x_kx_2}^{\mathbf{pa}(X_i)} & \dots & q_{x_kx_k}^{\mathbf{pa}(X_i)} \end{bmatrix},$$

where  $q_{x_ax_b}^{\mathbf{pa}(X_i)}$  is the intensity of leaving state  $x_a$  for  $x_b$  and  $q_{x_a}^{\mathbf{pa}(X_i)} = \sum_{b \neq a} q_{x_ax_b}^{\mathbf{pa}(X_i)}$ .

**Definition 2** (*Continuous-time Bayesian network*). A CTBN  $\mathcal{N} = (\mathcal{G}, \mathcal{Q}, P_{\mathcal{X}}^0)$  over a set of discrete random variables  $\mathcal{X} = \{X_1, \dots, X_m\}$  consists of: (1) a continuous transition

model specified by a directed (possible cyclic) graph  $\mathcal{G}$  and a CIM  $\mathcal{Q}_{X_i}^{\text{Pa}(X_i)}$  for each variable  $X_i$ , and (2) an initial distribution  $P_{\mathcal{X}}^0$ , specified as a BN over  $\mathcal{X}$ .

As with BNs, learning CTBNs is not only motivated by knowledge discovery but also to perform classification tasks. Continuous-time Bayesian network classifiers (CTBNCs) extend CTBNs to classify discrete-state temporal sequences  $\mathcal{S}_l = \{\mathbf{x}_l^{t_1}, \dots, \mathbf{x}_l^{t_{T_l}}, c_l\} (l = 1, \dots, N)^1$ , which describe transitions of, time-dependent, feature variables  $\mathcal{X}$  and the state of a, time-independent, class variable  $C$  (Stella and Amer, 2012). Nevertheless, some classification problems require predicting the state of multiple class variables  $\mathcal{C} = \{C_1, \dots, C_d\}$  simultaneously, i.e.,  $\mathcal{S}_l = \{\mathbf{x}_l^{t_1}, \dots, \mathbf{x}_l^{t_{T_l}}, \mathbf{c}_l\}$ , where  $\mathbf{c}_l = (c_{l1}, \dots, c_{ld})$ . The Multi-CTBNC models such a more complex setting, potentially introducing information relevant to classification by capturing probabilistic relationships of class variables (Villa-Blanco et al., 2021).

**Definition 3** (*Multidimensional continuous-time Bayesian network classifier*). A Multi-CTBNC  $\mathcal{M} = (\mathcal{G}, \mathbf{B}, \mathcal{Q}, P_{\mathcal{V}}^0)$  over a set of discrete variables  $\mathcal{V} = \{X_1, \dots, X_m, C_1, \dots, C_d\}$  is formed by:

- A directed (possibly cyclic) graph  $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ . Vertices  $\mathcal{V}$  are divided into those for feature and class variables, while arcs between class variables (class subgraph), feature variables (feature subgraph) and from class to feature variables (bridge subgraph).
- Class variable parameters  $\mathbf{B}$ , which form conditional probability tables (CPTs).
- A set of CIMs  $\mathcal{Q}$ , one for each feature variable  $X_i$ .
- An initial distribution  $P_{\mathcal{V}}^0$ , specified as a multidimensional BN classifier over  $\mathcal{V}$ .

## 2.1 Structure Learning Algorithms for CTBNs

Structure learning for CTBNs has been traditionally addressed as an optimisation problem (Nodelman et al., 2003), where a structure is selected from a candidate space by maximising a score (score-based algorithms). Only recently has a constraint-based algorithm been proposed, which reconstructs their structures by performing conditional independence tests. The continuous-time PC (CTPC) algorithm, introduced by Bregoli et al. (2021), is the first proposal of this kind, which adapts the classical PC algorithm, described in Algorithm 1, to CTBNs. As CIMs describe temporal dynamics, classical statistical tests cannot be applied. Thus, CTPC introduces a novel definition of conditional independence in CTBNs.

**Definition 4** (*Conditional independence in CTBNs*). Given a CTBN over a set of discrete random variables  $\mathcal{X} = \{X_1, \dots, X_m\}$ , a variable  $X_i$  is conditionally independent from  $X_j$  given a separating set  $\mathbf{S}_{X_i X_j} \subseteq \mathcal{X} \setminus \{X_i, X_j\}$  iff:

$$\mathbf{Q}_{X_i}^{x, \mathbf{s}} = \mathbf{Q}_{X_i}^{\mathbf{s}} \quad \forall x \in \Omega_{X_j}, \forall \mathbf{s} \in \Omega_{\mathbf{S}_{X_i X_j}}.$$

The CTPC algorithm is easily adapted to learn the bridge and feature subgraphs of Multi-CTBNCs since only the parent sets of nodes with CIMs are learned. The algorithm only has to meet the topology constraints of the model. Algorithm 2 shows the pseudocode of the CTPC algorithm used in this work, where  $\mathcal{V} = \mathcal{X} \cup \mathcal{C}$ . As for learning the class subgraph, traditional constraint-based solutions for discrete BNs can be used. We call this adaption naive since it tests all possible dependencies between feature variables, even those irrelevant for the classification task. This considerably increases the learning time, a problem aggravated in CTBNs since dependencies between feature variables are non-symmetric.

1. Sequences may have different timestamps; superscript  $l$  is omitted from  $t$  for simplicity.

---

**Algorithm 1** PC( $\mathcal{V}$ )

---

- 1: Build the complete undirected graph  $\mathcal{G}$  on node set  $\mathcal{V}$
  - 2: Find the skeleton and separating sets of  $\mathcal{G}$ . For each pair of adjacent nodes  $V_i, V_j \in \mathcal{V}$ , remove edge  $V_i - V_j$  from  $\mathcal{G}$  iff there is a separating set  $\mathcal{S}_{V_i V_j} \subseteq \mathcal{V} \setminus \{V_i, V_j\}$  such that  $V_i \perp\!\!\!\perp V_j | \mathcal{S}_{V_i V_j}$
  - 3: Orient the colliders using the separating sets. For any path  $V_i - V_j - V_k$  in  $\mathcal{G}$ , such that  $V_i \neq V_k$ , orient the edges as  $V_i \rightarrow V_j \leftarrow V_k$  iff  $V_i \perp\!\!\!\perp V_k | \mathcal{S}_{V_i V_k}$  and  $V_j \notin \mathcal{S}_{V_i V_k}$
  - 4: Orient all possible undirected edges:
    - 4.1: Given that  $V_i \rightarrow V_j - V_k$  and  $V_i \neq V_k$ , then orient the undirected edge into  $V_j \rightarrow V_k$  to avoid introducing a new v-structure
    - 4.2: Given that  $V_i - V_k$  and  $V_i \rightarrow V_j \rightarrow V_k$ , then orient  $V_i - V_k$  into  $V_i \rightarrow V_k$  to avoid introducing a cycle
    - 4.3: Given that  $V_i - V_k$ ,  $V_i - V_j \rightarrow V_k$ ,  $V_i - V_w \rightarrow V_k$  and  $V_i \neq V_w$ , then orient  $V_i - V_k$  into  $V_i \rightarrow V_k$  to avoid introducing a new v-structure or a cycle
  - 5: **return** partially directed acyclic graph  $\mathcal{G}$
- 

---

**Algorithm 2** CTPC( $\mathcal{X}, \mathcal{V}$ )

---

- 1: **for** each feature variable  $X_i \in \mathcal{X}$  **do**
  - 2:   Set  $\mathcal{U} = \{V_j \in \mathcal{V} | V_j \rightarrow X_i\}$
  - 3:   **for** increasing values  $s = 0, \dots, |\mathcal{V}|$ , until  $s = |\mathcal{U}| - 1$  **do**
  - 4:     **for** each variable  $V_j \in \mathcal{U}$  and subset  $\mathcal{S}_{X_i V_j} \subseteq \mathcal{U} \setminus \{V_j\}$ , where  $|\mathcal{S}_{X_i V_j}| = s$  **do**
  - 5:       **if**  $X_i \perp\!\!\!\perp V_j | \mathcal{S}_{X_i V_j}$  **then**
  - 6:          Remove arc  $V_j \rightarrow X_i$  from  $\mathcal{G}$  and  $V_j$  from  $\mathcal{U}$
  - 7:       **end if**
  - 8:     **end for**
  - 9:   **end for**
  - 10: **end for**
  - 11: **return** directed graph  $\mathcal{G}$
- 

### 3. Markov Blanket-based Continuous-Time PC Algorithm

This section introduces a novel constraint-based structure learning algorithm called Markov blanket-based continuous-time PC (MB-CTPC), specially designed to learn Multi-CTBNCs. This algorithm aims to evaluate only those relevant dependencies for the Markov blanket (parents, children and spouses) of class variables. To this end, MB-CTPC defines a set of rules to ignore irrelevant dependencies based on ancestor class variables of feature variables.

Algorithm 3 describes the pseudocode of MB-CTPC. The first step finds the probabilistic relationships between class variables using a traditional constraint-based algorithm, such as PC. Step 2 forms the complete bridge and feature subgraphs of  $\mathcal{G}$ . Then, Step 3 defines the descendants of the class variables via conditional independence tests between feature and class variables without considering other feature variables in the separating set. Thus, a dependency of a feature on a class variable might exist because it is its child or there is a flow of information through intermediate feature nodes. This step defines a preliminary bridge subgraph that provides valuable information to reduce the statistical tests for the

**Algorithm 3** MB-CTPC algorithm

---

```

1:  $\mathcal{G} \leftarrow \text{PC}(\mathcal{C})$ 
2: Build the complete bridge and feature subgraphs of  $\mathcal{G}$  on node set  $\mathcal{X} \cup \mathcal{C}$ 
3:  $\mathcal{G} \leftarrow \text{CTPC}(\mathcal{X}, \mathcal{C})$ 
4: for each feature variable  $X_i \in \mathcal{X}$  do
5:   for each feature variable  $X_j \in \mathcal{X}$ , where  $X_j \neq X_i$  do
6:     if  $\text{Pa}_{\mathcal{C}}(X_i) = \emptyset$  then
7:       Remove arc  $X_j \rightarrow X_i$  from  $\mathcal{G}$ 
8:     else if  $\text{Pa}_{\mathcal{C}}(X_i) \cap \text{Pa}_{\mathcal{C}}(X_j) = \emptyset$  AND  $\text{Pa}_{\mathcal{C}}(X_j) \neq \emptyset$  then
9:       Remove arcs  $X_i \rightarrow X_j$  and  $X_j \rightarrow X_i$  from  $\mathcal{G}$ 
10:    else if  $\text{Pa}_{\mathcal{C}}(X_i) \setminus \text{Pa}_{\mathcal{C}}(X_j) \neq \emptyset$  AND  $\text{Pa}_{\mathcal{C}}(X_j) \neq \emptyset$  then
11:      Remove arc  $X_i \rightarrow X_j$  from  $\mathcal{G}$ 
12:    end if
13:  end for
14: end for
15:  $\mathcal{G} \leftarrow \text{CTPC}(\mathcal{X}, \mathcal{C} \cup \mathcal{X})$ 
16: return directed graph  $\mathcal{G}$ 

```

---

feature subgraph. If a pair of feature variables do not share the same parent class variables, information is not flowing in at least one direction, and at least one dependency can be removed. Steps 4 to 14 use three rules to reduce the conditional independence tests:

**Rule 1 (Steps 6 and 7).** Given adjacent feature variables  $X_i$  and  $X_j$ , arc  $X_j \rightarrow X_i$  is removed iff  $\text{Pa}_{\mathcal{C}}(X_i) = \emptyset$ , where  $\text{Pa}_{\mathcal{C}}(X_i)$  denotes the parent class variables of  $X_i$ .

**Rule 2 (Steps 8 and 9).** Given adjacent feature variables  $X_i$  and  $X_j$ , arcs  $X_i \rightarrow X_j$  and  $X_j \rightarrow X_i$  are removed iff  $\text{Pa}_{\mathcal{C}}(X_i) \cap \text{Pa}_{\mathcal{C}}(X_j) = \emptyset$ ,  $\text{Pa}_{\mathcal{C}}(X_i) \neq \emptyset$  and  $\text{Pa}_{\mathcal{C}}(X_j) \neq \emptyset$ .

**Rule 3 (Steps 10 and 11).** Given adjacent feature variables  $X_i$  and  $X_j$ , an arc  $X_i \rightarrow X_j$  is removed iff  $\text{Pa}_{\mathcal{C}}(X_i) \setminus \text{Pa}_{\mathcal{C}}(X_j) \neq \emptyset$ ,  $\text{Pa}_{\mathcal{C}}(X_i) \neq \emptyset$  and  $\text{Pa}_{\mathcal{C}}(X_j) \neq \emptyset$ .

Finally, Step 15 further identifies conditional independence relationships that the previous rules could not discard.

**Example 1.** Given some data sampled from the Multi-CTBNC of Figure 1a, Figures 1b to 1f show the steps of MB-CTPC to learn the Markov blankets of class variables. First, conditional independence tests, which consider only class variables in the separating sets, find the descendant feature variables of the class variables. Figure 1b shows an arc from  $C_4$  to  $X_6$ , as a direct path exists through  $X_7$  in the original structure. Afterwards, Rule 1 removes incoming arcs of  $X_2$  and  $X_5$  in Figure 1c since they have no dependencies on class variables. Then, Rule 2 discards dependencies between feature variables not sharing parent class variables. That is the case for pairs like  $X_3$  and  $X_6$  or  $X_1$  and  $X_7$  in Figure 1d. In Figure 1e, Rule 3 removes a dependency from one feature variable to another if the former has parent class variables that the latter does not, such as  $X_3$  on  $X_1$  or  $X_7$  on  $X_6$ . These rules discard 27 out of 42 arcs of the feature subgraph without performing any conditional independence test. Finally, tests are performed on the remaining arcs in Figure 1f. The resulting structure shows that arcs from the original structure providing no information about the Markov blankets of class variables (those between  $X_2$  and  $X_5$ ) are discarded. ■

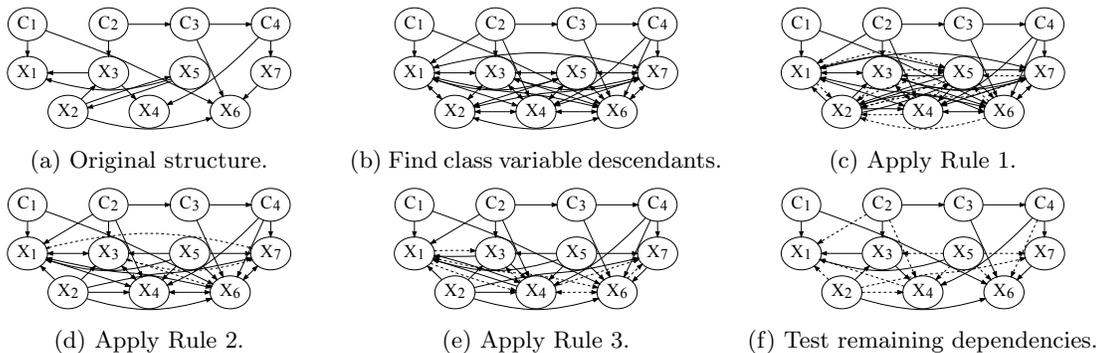


Figure 1: Steps of the MB-CTPC algorithm. Dash lines represent removed arcs.

### 4. Hybrid Structure Learning Algorithm

Score- and constraint-based algorithms have their own strengths and weaknesses, making them more or less useful in different classification contexts. Therefore, as has already been done for other PGMs (Liu et al., 2017; Trabelsi et al., 2013), we study hybrid algorithms for Multi-CTBNCs. These methods aim to combine the advantages of both approaches, such as faster learning speed of constraint-based algorithms and higher accuracy of score-based solutions (see Section 5.1.2). This section presents the first hybrid algorithm for learning both one-dimensional and multidimensional CTBNCs. The algorithm is divided into a restriction phase where conditional independence tests find an initial structure and a maximisation phase that refines it. Two variants are used depending on the subgraph:

- **Class subgraph:** the PC algorithm is used to reconstruct the skeleton of the class subgraph. Then, hill climbing searches for a solution, starting from the empty subgraph, but only allowing arcs that were included in the skeleton.
- **Bridge and feature subgraphs:** the CTPC algorithm defines an initial structure during the restriction phase, which serves as the initial solution for hill climbing in the maximisation phase. Two measures balance the influence of both algorithms. First, a maximum separating set size is established for conditional independence tests. Second, hill climbing only removes or adds arcs the restriction phase has not discarded. The separating set maximum size dictates each algorithm’s influence and range of action.

Although the scope of this article is learning classifiers, this hybrid algorithm can be applied to learn CTBNCs, making it their first hybrid proposal to the best of our knowledge.

### 5. Experiments

This section empirically compares the performance of Multi-CTBNCs learned with five different structure learning algorithms in a variety of contexts. Score-based algorithms are represented by hill climbing and tabu search (tabu list of size 5), whose scores (BIC or BDe) are indicated in square brackets, e.g., hill climbing [BDe]. Regarding constraint-based algorithms, CTPC and the presented MB-CTPC are evaluated. Significance levels of 0.05 (class subgraph) and  $1e-5$  (bridge and feature subgraphs) are used to test conditional independence. Finally, the proposed hybrid algorithm is studied using the BIC score and separating sets of zero (hybrid  $[|\mathcal{S}_{V_i V_j}| = 0]$ ) and one (hybrid  $[|\mathcal{S}_{V_i V_j}| = 1]$ ) maximum size.

Parameter	Studied values	Parameter	Studied values
Number of feature variables	5, 10, 20	Density class subgraph	30%
Cardinality of feature variables	2, 3, 4, 8	Density bridge subgraph	5%, 10%, 20%
Number of class variables	4	Density feature subgraph	5%, 10%, 20%
Cardinality of class variables	2, 3		

Table 1: Parameters used to generate the datasets for the experiments.

Synthetic datasets are sampled via probabilistic logic sampling from Multi-CTBNCs whose structures and parameters are randomly generated. Five datasets have been sampled from each combination of parameters’ values shown in Table 1 (1080 datasets), each with 5000 sequences that last 20 time units. The generated structures have at least one arc in the bridge subgraph, and feature variables are restricted to a maximum of three children to avoid memory problems. In order to guarantee an honest and fair comparison, the learned models are evaluated using several performance evaluation metrics and a 5-fold cross-validation scheme. The metrics in consideration are: global and mean accuracy, global Brier score, macro-averaged F1 score and learning time. The Wilcoxon signed-rank test is used with a significance level of 0.05 to verify that the results are significant. Regarding parameter learning, Bayesian estimation is used with hyperparameters for their prior distributions  $\lambda_{c_j} = 1$ ,  $\alpha_{x_a, x_b} = 1$  and  $\tau_{x_j} = 0.001$  (see Villa-Blanco et al. (2021) for more details).

The experiments were run on a 4.20GHz Intel Core i7-7700K with 32 GB of RAM using Windows 10. The structure learning algorithms were developed in Java, and the software and datasets are freely available at <https://github.com/carlvilla/Multi-CTBNCs>.

## 5.1 Experiment Results

Table 2 presents the results for all the datasets of certain structure learning algorithm comparisons that we found most relevant. In the following sections, we discuss some conclusions extracted from this table and perform a more exhaustive analysis.

### 5.1.1 HILL CLIMBING AND TABU SEARCH

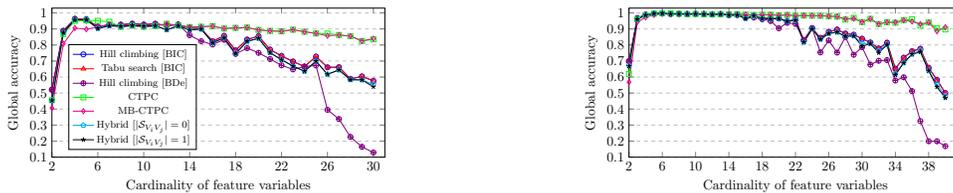
Table 2 shows that hill climbing optimising the BIC score obtains better results in all evaluation metrics, except learning time, for more datasets than the BDe score, improvements that were found statistically significant. As a result, subsequent comparisons will mainly consider the BIC score. Meanwhile, learning time differences between hill climbing [BIC] and tabu search [BIC] were significant, with the latter being faster in 66% of the datasets. However, these differences may not be very substantial, as tabu search achieves a sub-second improvement in 71% of the datasets where it outperforms hill climbing.

### 5.1.2 CTPC AND HILL CLIMBING

The CTPC algorithm performs significantly worse than hill climbing, optimising the BIC and BDe scores, for all evaluation metrics, except learning time, when considering all datasets. Nevertheless, this is the case only when feature variables’ cardinality is relatively low. Significant improvements with CTPC have been validated for all evaluation metrics when feature variables have eight possible states. To further study the influence of feature variables’ cardinalities, we have analysed the global accuracy results when the

Evaluation metric	Results of ... are	Better	Worse	Same	Same or better	Than
Global accuracy	Hill climbing [BIC]	10.09%	5.74%	<b>84.17</b> %	94.26 %	Hill climbing [BDe]
	Hill climbing [BIC]	0.93%	0.93%	<b>98.15</b> %	99.07 %	Tabu search [BIC]
	CTPC	17.41%	<b>42.31</b> %	40.28%	57.69 %	Hill climbing [BIC]
	MB-CTPC	1.48%	40.28%	<b>58.24</b> %	59.72 %	CTPC
	Hybrid $\ S_{V_i V_j}\  = 0$	1.67%	0.09%	<b>98.24</b> %	99.91 %	Hybrid $\ S_{V_i V_j}\  = 1$
	Hybrid $\ S_{V_i V_j}\  = 0$	7.96%	<b>46.39</b> %	45.65%	53.61 %	Hill climbing [BIC]
Mean accuracy	Hybrid $\ S_{V_i V_j}\  = 0$	28.15%	<b>37.69</b> %	34.17%	62.31 %	CTPC
	Hill climbing [BIC]	9.81%	7.22%	<b>82.96</b> %	92.78 %	Hill climbing [BDe]
	Hill climbing [BIC]	1.11%	0.93%	<b>97.96</b> %	99.07 %	Tabu search [BIC]
	CTPC	21.94%	<b>39.35</b> %	38.70%	60.65 %	Hill climbing [BIC]
	MB-CTPC	1.39%	40.93%	<b>57.69</b> %	59.07 %	CTPC
	Hybrid $\ S_{V_i V_j}\  = 0$	1.76%	0.09%	<b>98.15</b> %	99.91 %	Hybrid $\ S_{V_i V_j}\  = 1$
Macro-averaged F1 score	Hybrid $\ S_{V_i V_j}\  = 0$	10.09%	<b>45.74</b> %	44.17%	54.26 %	Hill climbing [BIC]
	Hybrid $\ S_{V_i V_j}\  = 0$	25.65%	<b>42.50</b> %	31.85%	57.50 %	CTPC
	Hill climbing [BIC]	11.39%	7.22%	<b>81.39</b> %	92.78 %	Hill climbing [BDe]
	Hill climbing [BIC]	0.93%	1.20%	<b>97.87</b> %	98.80 %	Tabu search [BIC]
	CTPC	21.57%	<b>41.76</b> %	36.67%	58.24 %	Hill climbing [BIC]
	MB-CTPC	2.50%	40.37%	<b>57.13</b> %	59.63 %	CTPC
Global Brier score	Hybrid $\ S_{V_i V_j}\  = 0$	1.76%	0.09%	<b>98.15</b> %	99.91 %	Hybrid $\ S_{V_i V_j}\  = 1$
	Hybrid $\ S_{V_i V_j}\  = 0$	20.09%	37.78%	<b>42.13</b> %	62.22 %	Hill climbing [BIC]
	Hybrid $\ S_{V_i V_j}\  = 0$	<b>36.11</b> %	33.33%	30.56%	66.67 %	CTPC
	Hill climbing [BIC]	24.72%	18.70%	<b>56.57</b> %	81.30 %	Hill climbing [BDe]
	Hill climbing [BIC]	4.72%	4.91%	<b>90.37</b> %	95.09 %	Tabu search [BIC]
	CTPC	20.93%	<b>60.09</b> %	18.98%	39.91 %	Hill climbing [BIC]
Learning time	MB-CTPC	19.07%	<b>61.02</b> %	19.91%	38.98 %	CTPC
	Hybrid $\ S_{V_i V_j}\  = 0$	3.06%	2.59%	<b>94.35</b> %	97.41 %	Hybrid $\ S_{V_i V_j}\  = 1$
	Hybrid $\ S_{V_i V_j}\  = 0$	8.98%	<b>67.13</b> %	23.89%	32.87 %	Hill climbing [BIC]
	Hybrid $\ S_{V_i V_j}\  = 0$	32.04%	<b>50.19</b> %	17.78%	49.81 %	CTPC
	Hill climbing [BIC]	32.78%	<b>67.13</b> %	0.09%	32.87 %	Hill climbing [BDe]
	Hill climbing [BIC]	33.89%	<b>65.93</b> %	0.19%	34.07 %	Tabu search [BIC]
Learning time	CTPC	<b>99.44</b> %	0.56%	0.00%	99.44 %	Hill climbing [BIC]
	MB-CTPC	<b>98.52</b> %	1.48%	0.00%	98.52 %	CTPC
	Hybrid $\ S_{V_i V_j}\  = 0$	36.67%	<b>63.24</b> %	0.09%	36.76 %	Hybrid $\ S_{V_i V_j}\  = 1$
	Hybrid $\ S_{V_i V_j}\  = 0$	<b>88.24</b> %	11.76%	0.00%	88.24 %	Hill climbing [BIC]
	Hybrid $\ S_{V_i V_j}\  = 0$	3.80%	<b>96.20</b> %	0.00%	3.80 %	CTPC

Table 2: Percentage of datasets where compared structure learning algorithms obtain better, worse or identical results.



(a) Datasets with sequences of 10 time units.

(b) Datasets with sequences of 20 time units.

Figure 2: Evolution of global accuracy as the cardinality of feature variables varies.

cardinality is increased from two to 30. Five datasets have been sampled for each possible cardinality (145 datasets) from a single randomly generated structure with 10 feature variables, four class variables, and bridge and feature subgraph densities of 10%. Figure 2a shows that classifiers learned with constraint-based algorithms (CTPC and MB-CTPC) are more robust than score-based and hybrid solutions as the cardinality is further increased.

The number of examples for each possible state transition declines as the cardinality of feature variables increases, making models learned with score-based algorithms less accurate. Figure 2b shows the results of the last experiment but using sequences with twice the duration. Increasing the sequence duration enables score-based algorithms to achieve better results with feature variables of a higher cardinality. Nevertheless, they still show worse robustness than constraint-based solutions, not only for global accuracy but also for mean accuracy, F1 score and global Brier score. At first glance, we thought BIC penalisation negatively influenced the models’ accuracies. However, this behaviour is even more severe for the BDe score. We can then conclude that for problems where feature variables have high cardinality and the sequence duration is relatively small, constraint-based algorithms

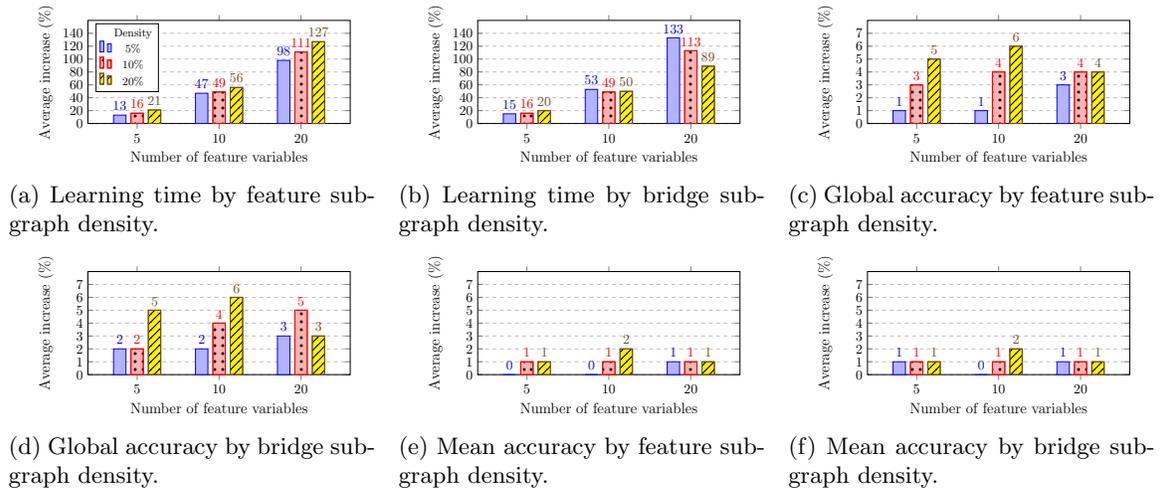


Figure 3: Average increase in learning time and global and mean accuracy of CTPC over MB-CTPC.

might be more convenient due to their robustness. This is consistent with the findings of Scutari et al. (2019) and Bregoli et al. (2021) for BNs and CTBNs, respectively.

Finally, it is worth noting that constraint-based algorithms achieve much shorter learning times since estimated parameters can be cached and quickly retrieved for future statistical tests. The usefulness of a cache is more limited for scored-based algorithms as they iteratively evaluate previously unseen parent set configurations.

### 5.1.3 MB-CTPC AND CTPC

The MB-CTPC algorithm, compared to CTPC, achieves the same or, in a few cases, better results in about 60% of the datasets for global and mean accuracy and F1 score. Simultaneously, it reduces the learning time on practically all datasets (99%), which was its main objective. Figures 3a and 3b show that time differences between the algorithms are very significant, as the mean time increase of CTPC reaches 133%. The differences become more profound as the number of feature variables and feature subgraph density increase since CTPC performs more tests irrelevant to the classification task. The bridge subgraph density also influences the results, as lower density implies more significant differences for higher dimensionality data. This is due to a decrease in the number of feature variables with class variables as ancestors, reducing the tests performed by MB-CTPC. Five datasets of higher dimensionality were sampled from randomly generated structures with bridge and feature subgraph densities of 10%. Due to memory limitations, 30 feature and five class variables with six and three states, respectively, were used. Table 3 shows that MB-CTPC drastically decreases learning time, being 2.3 times faster than CTPC, while other evaluation metrics, such as the global accuracy or F1 score, suffer relatively small differences.

The CTPC algorithm obtains better results than MB-CTPC on multiple datasets, significantly improving all evaluation metrics except learning time. Nevertheless, these differences may not be significant enough if our priority is to speed up the model learning. Figures 3c and 3d show that the mean improvement of the global accuracy can go from as little as 1%

Algorithm	Global accuracy	Mean accuracy	Macro-averaged F1 score	Global Brier score	Learning time
CTPC	<b>0.9874 ± 0.0149</b>	<b>0.9975 ± 0.0030</b>	<b>0.9597 ± 0.0851</b>	<b>0.0238 ± 0.0286</b>	256.9574 ± 8.4186
MB-CTPC	0.9590 ± 0.0403	0.9917 ± 0.0081	0.9546 ± 0.0792	0.0675 ± 0.0665	<b>112.4594 ± 14.1150</b>

Table 3: Estimated evaluation metrics (mean ± std. deviation) over synthetic datasets generated from Multi-CTBNCs with 30 and 5 feature and class variables, respectively.

to a maximum of 6% in the performed experiments. These differences are even lower for the mean accuracy (see Figures 3e and 3f) or F1 score. As for the global Brier score, differences in the percentages are more significant; however, they are less than 0.01 in 57% of the datasets. The slightly lower accuracy of classifiers learned with MB-CTPC arises from the incorrect definition of some class variable descendants (Step 3 of Algorithm 3). Possible causes behind this may include weak relationships between variables, training datasets not sufficiently representative of the underlying problem or the assumption that waiting times of feature variables conditional on a non-parent ancestor follow an exponential distribution.

We can conclude that the MB-CTPC algorithm is a good choice when learning time is a priority, especially when dealing with high dimensionality datasets. Nevertheless, we should also consider that a trade-off exists between assuring a better accuracy or significantly reducing the learning time, which has to be assessed depending on the particular problem.

#### 5.1.4 HYBRID [ $|\mathcal{S}_{V_i V_j}| = 0$ ] AND HYBRID [ $|\mathcal{S}_{V_i V_j}| = 1$ ]

Overall, varying the maximum separating set size of the hybrid algorithm from zero to one results in no change in most experiments except for the learning time. For example, the global accuracy improves in just 1.67% of the datasets when only testing for unconditional independence. However, 94.44% of these latter datasets have in common the presence of binary feature variables. As the maximum size increases, the constraint-based algorithm has more influence on the solution, which is less accurate than score-based approaches when feature variables are binary. For this reason, statistically significant improvements were obtained for most evaluation metrics with the hybrid [ $|\mathcal{S}_{V_i V_j}| = 0$ ] algorithm. The hybrid [ $|\mathcal{S}_{V_i V_j}| = 1$ ] solution only succeeded in reducing the learning time significantly. Nevertheless, differences between using both parameter values are generally negligible.

#### 5.1.5 HYBRID [ $|\mathcal{S}_{V_i V_j}| = 0$ ] VS. HILL CLIMBING [BIC] AND CTPC

The hybrid [ $|\mathcal{S}_{V_i V_j}| = 0$ ] algorithm performs significantly worse than CTPC and hill climbing [BIC] for most evaluation metrics. The exceptions are significantly improved learning time compared to hill climbing and no difference in F1 score against CTPC. Nevertheless, class variables' cardinalities significantly influence the differences between the hybrid and CTPC algorithms. If class variables are binary, the hybrid solution performs significantly worse for all evaluation metrics; however, if they are ternary, no statistically significant differences are found in global accuracy, and the F1 score is significantly improved. As a result, we decided to study further the influence of class variables' cardinalities on the results of the hybrid [ $|\mathcal{S}_{V_i V_j}| = 0$ ] algorithm to determine if it could improve the accuracy of CTPC for specific contexts while still reducing the learning time of hill climbing.

The hybrid and CTPC algorithms were evaluated on 210 datasets evenly divided according to whether four class variables were binary or had four to 10 states. As CTPC

Global accuracy	Mean accuracy	Macro-averaged F1 score	Global Brier score	Global accuracy	Mean accuracy	Macro-averaged F1 score	Global Brier score
<b>Binary feature variables / Binary class variables</b>				<b>Four states feature variables / Four to 10 states class variables</b>			
31.43%	25.71%	42.86%	40.0%	68.57%	68.57%	68.57%	77.14%
<b>Binary feature variables / Four to 10 states class variables</b>				<b>Six states feature variables / Binary class variables</b>			
80.0%	57.14%	74.29%	88.57%	11.43%	11.43%	14.29%	25.71%
<b>Four states feature variables / Binary class variables</b>				<b>Six states feature variables / Four to 10 states class variables</b>			
22.86%	17.14%	22.86%	17.14%	42.86%	40.0%	37.14%	51.43%

Table 4: Percentage of datasets where the hybrid  $[|\mathcal{S}_{V_i V_j}| = 0]$  algorithm outperforms CTPC.

performance differs depending on the cardinality of feature variables (see Section 5.1.2), datasets were further divided based on whether 10 of these variables had two, four or six states. Table 4 shows a substantial improvement when class variables’ cardinalities are above two, especially if feature variables are binary. In this latter case, the hybrid  $[|\mathcal{S}_{V_i V_j}| = 0]$  algorithm significantly outperforms CTPC in terms of global and mean accuracy, F1 score and global Brier score while reducing the learning time of hill climbing [BIC] by an average of 38%. Nevertheless, the results decline as the cardinality of feature variables increases.

## 6. Conclusions and Future Work

This article introduces for the first time constraint-based and hybrid structure learning algorithms for continuous-time Bayesian network classifiers, which were specially designed to learn Multi-CTBNCs. The novel constraint-based algorithm, named MB-CTPC, aims to learn the structure of these classifiers by performing conditional independence tests only on dependencies that could be relevant to the Markov blankets of class variables. Then, the hybrid algorithm, a solution not even studied for CTBNs, combines the strengths of score- and constraint-based methods.

Synthetic experiments show that MB-CTPC significantly improves the learning time of Multi-CTBNCs compared to other structure learning algorithms. The MB-CTPC algorithm is particularly convenient when learning with high dimensionality datasets. Nevertheless, significant improvements were obtained regardless of the number of variables, their cardinality or structure density. Finally, the hybrid algorithm provides an intermediate solution that significantly improves, in specific scenarios, the results of CTPC while drastically reducing the learning time of hill climbing techniques.

Multiple areas of open research were found while conducting this work:

- MB-CTPC may struggle to identify descendants of class variables. Using a phase distribution, such as Erlang, to model the transition times of feature variables conditioned on ancestor class variables may improve this task and the model’s accuracy.
- Discerning between descendants and children of class variables may improve MB-CTPC learning time, as arcs to feature nodes with no parent class variables are irrelevant for the classification task.
- Adaptation of Hiton algorithm for CTBNs. A novel association function is needed as there are two different data distributions and p-values for each state of the variables.
- A class-bridge decomposable Multi-CTBNC (Bielza et al., 2011) could improve classification times.
- Information from class variable relationships may be useful to improve the definition of their descendants and, therefore, the accuracy of MB-CTPC.

## Acknowledgments

This work has been partially supported by the Spanish Ministry of Science, Innovation and Universities through the PID2019-109247GB-I00 and RTC2019-006871-7 projects. C. Villa-Blanco is supported by a predoctoral contract for the formation of doctors from the Universidad Politécnic de Madrid.

## References

- C. Bielza, G. Li, and P. Larrañaga. Multi-dimensional classification with bayesian networks. *International Journal of Approximate Reasoning*, 52(6):705–727, 2011.
- A. Bregoli, M. Scutari, and F. Stella. A constraint-based algorithm for the structural learning of continuous-time Bayesian networks. *International Journal of Approximate Reasoning*, 138:105–122, 2021.
- D. Codecasa and F. Stella. Learning continuous time Bayesian network classifiers. *International Journal of Approximate Reasoning*, 55(8):1728–1746, 2014.
- T.-C. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.
- H. Liu, S. Zhou, W. Lam, and J. Guan. A new hybrid method for learning Bayesian networks: Separation and reunion. *Knowledge-Based Systems*, 121:185–197, 2017.
- U. Nodelman, C. R. Shelton, and D. Koller. Continuous time Bayesian networks. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 378–387, 2002.
- U. Nodelman, C. R. Shelton, and D. Koller. Learning continuous time Bayesian networks. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, pages 451–458, 2003.
- J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- M. Scutari, C. E. Graafland, and J. M. Gutiérrez. Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms. *International Journal of Approximate Reasoning*, 115:235–253, 2019.
- F. Stella and Y. Amer. Continuous time Bayesian network classifiers. *Journal of Biomedical Informatics*, 45(6):1108–1119, 2012.
- G. Trabelsi, P. Leray, M. Ben Ayed, and A. M. Alimi. Dynamic MMHC: A local search algorithm for dynamic Bayesian network structure learning. In *International Symposium on Intelligent Data Analysis*, pages 392–403. Springer, 2013.
- C. Villa-Blanco, P. Larrañaga, and C. Bielza. Multidimensional continuous time Bayesian network classifiers. *International Journal of Intelligent Systems*, 36(12):7839–7866, 2021.