Learning Temporal Nodes Bayesian Networks

Pablo Hernandez-Leal, L. Enrique Sucar and **Jesus A. Gonzalez**
National Institute of Astrophysics, Optics and Electronics
Tonanzintla, Puebla, Mexico

Abstract

Temporal Nodes Bayesian Networks (TNBNs) are an alternative to Dynamic Bayesian Networks for temporal reasoning, that result in much simpler and efficient models in some domains. However, methods for learning this type of models from data have not been developed. In this paper we propose a learning algorithm to obtain the structure and temporal intervals for TNBNs from data. The method has three phases: (i) obtain an initial approximation of the intervals, (ii) obtain a structure using a standard algorithm and (iii) refine the intervals for each temporal node based on a clustering algorithm. We evaluated the method with synthetic data. Our method obtains the best score in terms of the structure and a competitive predictive accuracy.

1 Introduction

Bayesian Networks (Pearl 1988) are an alternative to deal with uncertainty. They have proven to be successful in various domains such as medicine (Pang et al. 2004). However, these models cannot deal with temporal information. For this, an extension called Dynamic Bayesian Networks (DBNs) was introduced. DBNs can be seen as multiple slices of a *static* BN over time, with links between adjacent slices. Nonetheless, these models can become quite complex, in particular, when only a few important events occur over time.

Temporal Nodes Bayesian Networks (TNBNs) (Arroyo-Figueroa and Sucar 1999) are another extension of Bayesian Networks. They belong to a class of temporal models known as *Event Bayesian Networks* (Galán et al. 2007). TNBNs were proposed to manage uncertainty and temporal reasoning. In a TNBN, each Temporal Node has intervals associated to it. Each node represents an event or state change of a variable. An arc between two Temporal Nodes corresponds to a causal-temporal relation. One interesting property of this class of models, in contrast to Dynamic Bayesian Networks, is that the temporal intervals can differ in number and size.

TNBNs had been used in diagnosis and prediction of temporal faults in a steam generator of a fossil power plant (Arroyo-Figueroa and Sucar 1999). The problem with this approach is that there does not exist an algorithm to learn TNBNs. Then the model has to be obtained from external sources such as domain experts. This can be a hard and prone to error task. In this paper, we propose a learning algorithm to obtain the structure and the temporal intervals for TNBNs from data.

The learning algorithm consists of three phases. In the first phase, we obtain an approximation of the intervals using an algorithm such as Equal-Width discretization (EWD) or K-means clustering. For the second phase, the BN structure is obtained with the structure learning algorithm introduced in (Cooper and Herskovits 1992). The last step is performed to refine the intervals for each Temporal Node. Our algorithm obtains a number of possible sets of intervals for each configuration of the parents by clustering the data based on a Gaussian mixture model. It then selects the set of intervals that maximizes the prediction accuracy. We apply our algorithm to a synthetic medical case. The data was generated with different distributions. We compare our algorithm with two baselines: K-means and Equal-Width discretization. We also compare it to the algorithm proposed in (Friedman and Goldszmidt 1996). In the experiments, our algorithm obtains the best score in terms of structure and a competitive predictive accuracy.

2 Related Work

Bayesian Networks (BN) are a successful model for dealing with uncertainty. However, *static* BNs are not suited to deal with temporal information. For this reason, Dynamic Bayesian Networks (Dagum, Galper, and Horvitz 1992) were introduced. In a DBN, a copy of a base model is made for each time stage. These copies are linked via a transition network. In this transition network is common that only links between consecutive stages are allowed (Markov property). The problem is that DBNs can become very complex. This is unnecessary when dealing with problems for which there are only a few changes for each variable in the model. Moreover, DBNs are not capable of managing different levels of time granularity. They usually have a fixed time interval between stages.

In TNBNs, each variable represents an event or state change. So, only one (or a few) instance(s) of each variable is required, assuming there is one (or a few) change(s) of a variable state in the temporal range of interest. No copies of the model are needed, and no assumption about the Marko-

Algorithm 1 Algorithm introduced in (Dagum, Galper, and Horvitz 1992).

Require: An initial discretization.
Ensure: A discretization policy.
1: Push all continuous variables onto a queue Q
2: **while** Q is not empty **do**
3: Remove first element X from Q
4: Compute discretization policy for X
5: **if** score with new discretization $<$ score with old discretization **then**
6: Use new discretization
7: For all Y interacting with X, if $Y \notin Q$, push Y onto Q.
8: **end if**
9: **end while**
10: **return** Discretization policy

vian nature of the process is made. TNBNs can deal with multiple granularity, because the number and the size of the intervals for each node can be different.

There are several methods to learn BNs from data (Neapolitan 2004). Unfortunately, the algorithms used to learn BNs cannot deal with the problem of learning temporal intervals. Then, these cannot be applied directly to learn TNBNs.

To the best of our knowledge, there is only one previous work that attempts to learn a TNBN. Liu et al. (Liu, Song, and Yao 2005) proposed a method to build a TNBN from a *temporal probabilistic database*. The method obtains the structure from a set of temporal dependencies in a *probabilistic* temporal relational model (PTRM). A Temporal Relational Model (TRM) is a relational model with temporal attributes. Let $R(U)$ be a TRM, then a PTRM is a TRM extended by adding a probabilistic attribute p, therefore $R(U, p)$ is a PTRM. In order to build the TNBN, they obtain a variable ordering that maximizes the set of conditional independence relations implied by a dependency graph obtained from the PTRM. Based on this order, a directed acyclic graph corresponding to the implied independence relations is obtained. This graph represents the structure of the TNBN. They assume a known probabilistic temporal relational model from the domain of interest, which is not always the case. Building this PTRM could be as difficult as building a TNBN. They do not learn the intervals of each node. In contrast, our approach constructs the TNBN directly from data, it also learns the intervals for each node.

Another related work is (Friedman and Goldszmidt 1996). Here the idea is to learn the structure of a BN while discretizing the continuous variables. For this, an score based on the Minimum Description Length (Lam and Bacchus 1994)) is proposed. The score takes into account the parameters, the structure, and the discretization policy. With this score, a search for the best discretization is done for each continuous variable. The algorithm scores the N_i midpoints of each variable. Then, a top-down refinement is performed using a greedy search strategy. Depending on the network structure, this approach can become complex if the continuous variables interact with each other (if a variable does not interact with other variables it can be discretized separately). For this reason, the approach is to discretize one variable at a time and leave the rest as discrete and fixed. Therefore, it is not guaranteed to find an optimal discretization. In order to learn the structure of the network, all the nodes are first discretized and later, the structure learning algorithm is applied. This alternating process is performed until convergence. The algorithm for discretizing several variables is shown in Algorithm 1. This algorithm may become too complex in cases where there exist a lot of midpoints in the variables. This may also happen if the network has many edges (several nodes interacting with each other).

3 Temporal Nodes Bayesian Networks

A Temporal Nodes Bayesian Network (TNBN) (Arroyo-Figueroa and Sucar 1999; Galán et al. 2007) is composed by a set of Temporal Nodes (TNs). TNs are connected by edges, each edge represents a causal-temporal relationship between TNs. There is at most one state change for each variable (TN) in the temporal range of interest. The value taken by the variable represents the interval in which the event occurs. Time is discretized in a finite number of intervals, allowing a different number and duration of intervals for each node (multiple granularity). Each interval defined for a child node represents the possible delays between the occurrence of one of its parent events (cause) and the corresponding child event (effect). Some Temporal Nodes do not have temporal intervals, these correspond to Instantaneous Nodes. Formally,

Definition 1 *A TNBN is defined as a pair $B = (G, \Theta)$. G is a Directed Acyclic Graph, $G = (\mathbf{V}, \mathbf{E})$. G is composed of* $\mathbf{V}$*, a set of Temporal and Instantaneous Nodes;* $\mathbf{E}$ *a set of edges between Nodes. The Θ component corresponds to the set of parameters that quantifies the network. Θ contains the values $\Theta_{v_i} = P(v_i|Pa(v_i))$ for each $v_i \in \mathbf{V}$; where $Pa(v_i)$ represents the set of parents of v_i in G.*

Definition 2 *A Temporal Node, v_i, is defined by a set of states* $\mathbf{S}$*, each state is defined by an ordered pair $S = (\lambda, \tau)$, where λ is the value of a random variable and $\tau = [a, b]$ is the interval associated, with initial value a and final value b, that corresponds to the time interval in which the state change occurs. In addition, each Temporal Node contains an extra default state $s =$ ('no change', $\emptyset$), which has no interval associated. If a Node has no intervals defined for all its states then it receives the name of Instantaneous Node.*

The following is an example of a TNBN based on (Arroyo-Figueroa and Sucar 1999), its corresponding graphical representation is shown in Figure 1.

Example 1 *Assume that at time $t = 0$, an accident occurs, a Collision. This kind of accident can be classified as* severe*,* moderate *and* mild*. To simplify the model we will consider only two immediate consequences for the person involved in the collision, Head Injury (HI) and Internal Bleeding (IB). HI can take two values* true *or* false*, IB can be* gross*,* slight *or* false*. These three are instantaneous events, no interval*

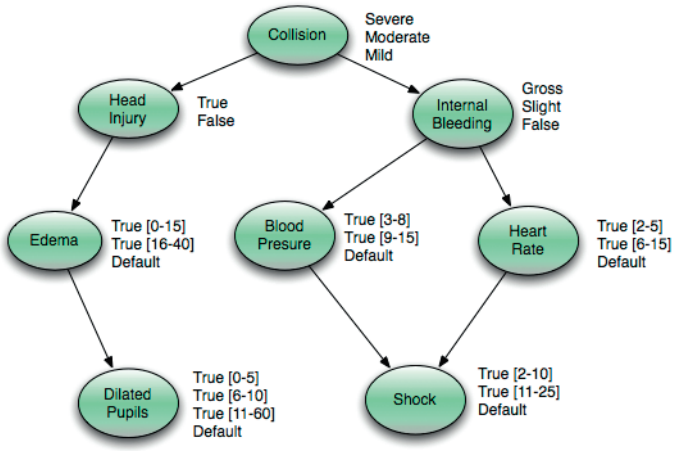

Figure 1: The TNBN for Example 1. Each oval represents a Temporal Node. The three upper nodes (Collision, Head Injury and Internal Bleeding) are Instantaneous Nodes, so they do not have temporal intervals. The other five nodes are Temporal Nodes with intervals associated to them. The Default state corresponds to the initial value of no change.

appears in these nodes. These events will generate subsequent changes, for example the HI event might generate an Edema and later as a consequence it may produce Dilated Pupils. These events are not immediate, they depend on the severity of the accident, therefore, they have temporal intervals associated. For the IB node two events may occur: an increase in Blood Pressure and an increase in Heart Rate, those will lead to Shock, these are also Temporal Nodes. For this example the intervals represent minutes and a doctor would appreciate not only the information of the occurrence of the events but also the time they appear, in order to obtain a better evaluation/diagnosis of the person.

4 Learning Algorithm

First, we present the interval learning algorithm for a TN. We initially assume that we have a defined structure. Later, we present the whole learning algorithm.

4.1 Interval Learning

Initially, we will assume that the events follow a known distribution. With this idea, we can use a clustering algorithm for the temporal data. Each cluster corresponds, in principle, to a temporal interval. The algorithm is presented first by ignoring the values of the parent nodes (first approximation). Later, we refine the method by incorporating the parent nodes configurations.

4.2 First Approximation: Independent Variables

Our approach uses a Gaussian mixture model (GMM) to perform an approximation of the data, therefore we can use the Expectation-Maximization (EM) algorithm (Dempster, Laird, and Rubin 1977). EM works iteratively using two steps: (i) The E-step tries to *guess* the parameters of the Gaussian distributions, (ii) the M-step updates the parameters of the model based on the previous step. By applying EM, we obtain a number of Gaussians (clusters), specified by their mean and variance. For now, assume that the number of temporal intervals (Gaussians), k, is given. For each TN we have a dataset of points over time, and these are clustered using GMM, to obtain k Gaussian distributions. Based on the parameters of each Gaussian, each temporal interval is initially defined as: $[\mu - \sigma, \mu + \sigma]$.

Now we deal with the problem of finding the number of intervals. The ideal solution has to fulfill two conditions: (i) the number of intervals must be small in order to reduce the complexity of the network, and (ii) the intervals should yield good estimations when performing inference over the network. Based on the above, our approach uses the EM algorithm with the parameter for the number of clusters in the range from 1 to ℓ, where ℓ is the highest value (for the experiments in this paper we used $\ell = 3$).

In order to select the best set of intervals, we evaluate the network. This corresponds to an indirect measure of the quality of the intervals. In particular, we used the *Relative Brier Score* to measure the predictive accuracy of the network. The selected set of intervals for each TN, are those that maximize the Relative Brier Score. The Brier Score is defined as $BS = \sum_{i=1}^{n} (1 - P_i)^2$, where P_i is the marginal posterior probability of the correct value of each node given the evidence. The maximum brier score is $BS_{max} = \sum_n 1^2$. The Relative Brier Score (RBS) is defined as: $RBS\ \ (in\ \ \%) = (1 - \frac{BS}{BS_{max}}) \times 100$. The RBS, is obtained by instantiating a random subset of variables in the model, predicting the unseen variables, and obtaining the RBS for these predictions.

4.3 Second Approximation: Considering the Network Topology

Now we will construct a more accurate approximation. For this, we use the configurations of the parent nodes. The number of configurations of each node i is $q_i = \prod_{Xr \in Pa(i)} |s_r|$ (the product of the number of states of the parents nodes).

Formally, we construct partitions of the data (disjoint sets of values), one partition for each configuration. Then we get the combinations taking 2 partitions p_i,p_j from the total, this yields $q(q-1)/2$ different combinations of partitions (we used only binary combinations in order to reduce the computation). For p_i and p_j we apply the first approximation and obtain ℓ sets of intervals for each partition, the last step is to obtain the combination of this sets of intervals, that yield ℓ^2 sets of final intervals for each p_i,p_j. After this process we have different sets of intervals, that we need to adjust. This adjustment is described in Algorithm 2.

Algorithm 2 is described next. For each set of intervals sort them by their starting point, then check if there is an interval contained in another interval. While this is true, the algorithm obtains an average interval, taking the average of the start and end points of the intervals and replacing these two intervals with the new one. Next we refine the intervals to be continuous by taking the mean of two adjacent values.

An example of this algorithm is presented in Section 4.6.

Algorithm 2 Algorithm to adjust the intervals.

Require: Array of intervals sets
Ensure: Array of intervals adjusted
1: **for** Each set of intervals s **do**
2: sortIntervalsByStart(s)
3: **while** Interval i is contained in Interval j **do**
4: tmp=AverageInterval(i,j)
5: s.replaceInterval(i,j,tmp)
6: **end while**
7: **for** $k = 0$ to number of intervals in set s -1 **do**
8: Interval[k].end=(Interval[k].end + Interval[k+1].start)/2
9: **end for**
10: **end for**

As in the first approximation, the best set of intervals for each TN is selected based on the predictive accuracy in terms of the RBS. However, when a TN has as parents other Temporal Nodes (an example of this situation is illustrated in Figure 1), the state of the parent nodes is not initially known. So, we cannot apply directly the second approximation. In order to solve this problem, the intervals are selected sequentially in a top-down fashion according to the TNBN structure. That is, we first select the intervals for the nodes in the second level of the network (the root nodes are instantaneous by definition in a TNBN (Arroyo-Figueroa and Sucar 1999). Once these are defined, we know the values of the parents of the nodes in the 3rd level. Then, we can find their intervals; and so on, until the leaf nodes are reached.

4.4 Pruning

Taking the combinations and joining the intervals can become computationally too expensive, the number of sets of intervals for node is in $O(q^2\ell^2)$ where q is the number of configurations and ℓ is the maximum number of clusters for the GMM. For this reason we used two pruning techniques for each TN to reduce the computation time.

The first pruning technique discriminates the partitions that contain few instances. For this, we count the number of instances in each partition, and if it is greater than a value $\beta = \frac{\text{Number of instances}}{\text{Number of partitions}\times 2}$ the configuration is used, if not it is discarded. A second technique is applied when the intervals for each combination are being obtained. If the final set of intervals contains only one interval (no temporal information) or more than α (producing a complex network), the set of intervals is discarded. For our experiments we used $\alpha = 4$.

4.5 Structure Learning

Now we present the whole algorithm that learns the structure and the intervals of the TNBN. First, we perform an initial discretization of the temporal variables, for example using an Equal-Width discretization (EWD) or K-means clustering. With this process we obtain an initial approximation of the intervals for all the Temporal Nodes and we can perform a standard BN structural learning. We used the K2 algorithm (Cooper and Herskovits 1992). This algorithm has as a parameter an ordering of the nodes. For learning TNBNs we can exploit this parameter and define an order based on domain information. Once the network structure has been obtained, we apply the interval learning algorithm described in Section 4.1. Moreover, this process of alternating interval learning and structure learning may be iterated until achieving convergence.

4.6 An Example

We will illustrate the process of obtaining the intervals for the TN *Edema* of Figure 1. We can see that its parent node (Head Injury) has two values, *true* and *false*. Thus, we separate the data of this node in two partitions, one for each configuration of the parent node. Then for each partition we apply the first approximation of the algorithm. We apply the EM algorithm to get a Gaussian mixture with parameter 1, 2 and 3 as the number of clusters. That gives us six different sets of intervals, as we show in Table 1.

Table 1: Initial sets of intervals obtained for node *Edema*. There are 3 sets of intervals for each partition.

Partition	Intervals
HI=true	$[11-35]$ $[11-27][32-53]$ $[8-21][25-32][45-59]$
HI=false	$[3-48]$ $[0-19][39-62]$ $[0-14][28-40][47-65]$

Then we combine these sets of intervals for each partition with the other partitions. In this case, there could be $3 \times 3 =$ nine different sets of intervals, but some of these combinations are eliminated.

For example, we should merge the sets $[11-35]$ and $[3-48]$, if we concatenate then we get $[11-35][3-48]$. With this set we apply Algorithm 2. In this case when we sort we get $[3-48][11-35]$. The next thing to do is to check whether one interval is contained in another, this is true ($[11-35]$ in $[3-48]$), then we obtain the average interval $[7-41]$. Next take $[11-35]$ and $[0-19][39-62]$. If we concatenate these intervals we get $[11-35][0-19][39-62]$. Applying Algorithm 2 we sort the intervals to obtain $[0-19][11-35][39-62]$. Here, no interval is contained in another so we skip the code inside the *while* statement, then we refine the intervals to obtain $[0-15][16-37][38-62]$. The process continues and we obtain the sets of intervals shown in Table 2.

This process is applied to each Temporal Node. To select the set of intervals for each node we apply the inference tests described in Section 4.2.

5 Evaluation on a Synthetic Case

In order to evaluate our algorithm, we considered the synthetic case corresponding to the TNBN presented in Figure 1. It is a hypothetical example of the consequences

Table 2: Final sets of intervals obtained for node *Edema*.

Intervals Node Edema
$[0-15][16-37][38-62]$
$[0-12][13-31][32-43][44-65]$
$[7-37][38-53]$
$[15-39][40-59]$
$[0-13][14-23][24-38][39-60]$

of an automobile accident. Given that we know the original structure, parameters, and intervals, we can compare the results of our algorithm to the reference network. The idea is to sample the fully specified TNBN to generate data that reflects the model and the intervals. Then, we can use that data to reconstruct the model with the proposed algorithm. For the data that represents the intervals, we perform two experiments. In the first one, data was generated by a normal distribution over the intervals with parameters $\mu = \frac{Interval\_Start+Interval\_End}{2}$, $\sigma = \frac{Interval\_End-Interval\_Start}{2}$ For the second test, the data was generated with a uniform distribution over the intervals. The intervals used to generate the data are the same as those shown in Figure 1.

5.1 Experiments

In both experiments we compared the proposed algorithm (P) and the algorithm introduced in (Friedman and Goldszmidt 1996) (F). As baselines we used an Equal-Width discretization (E) and K-means (K) clustering for learning the initial intervals for the Temporal Nodes. For measuring the quality of the structure of the TNBN with respect to the reference network, three measures were used: (i) The structural similarity (S-S) (Wu et al. 2001), which is a score between 0 and 1 (maximum) that counts the similar edges with a reference network. (ii) The number of added edges (E +) and (iii) the number of deleted edges (E -). The best network should obtain score 1 in structure similarity and 0 for edges deleted and added. For evaluating the quality of the intervals, we used two measures: (i) the error in time (T-E) defined as the difference between the real event and the expected mean of the interval and (ii) the number of intervals (# I) in the network. For evaluating the complete network the RBS was used. The best network should obtain a low time error, a low number of intervals, and a high inference quality (RBS). In all the tables, C represents the number of instances used for training.

For the first set of experiments, we used a Gaussian distribution to generate the data. In each experiment, we generated a different number of cases varying the number of initial intervals using EWD from 2-5 intervals (the results are summarized in the upper part of Table 3, which shows the average of 10 repetitions). It can be noticed that our algorithm obtains the lowest time error and with 200 cases it obtains the maximum score for structure similarity. Friedman's algorithm obtains the worst time error but its accuracy and number of intervals are the best. EWD obtained low time errors because the number of intervals is the highest. There-

Table 3: Evaluation of different algorithms (Alg) with Gaussian Distribution with Equal-Width Discretization and K-means as initialization. C is the number of cases (instances of data) used. T-E is the time error. S-S the structural similarity score. E + and E - are the edges added and deleted. # I is the average number of intervals.

Equal-Width discretization							
C	Alg	T-E	S-S	E +	E -	RBS	# I
100	P	**7.05**	0.72	1.75	**2**	80.76	13
	F	8.99	**0.75**	1.25	**2**	**80.86**	**11**
	E	7.21	0.72	**0.75**	2.25	75.34	17.5
150	P	**5.82**	**0.81**	1.25	**1.5**	81.31	13.5
	F	8.84	0.78	**0.25**	1.75	**82.60**	**11**
	E	6.66	0.72	**0.25**	2.25	76.59	17.5
200	P	6.68	**1.00**	**1**	**0**	80.72	14.5
	F	7.97	0.88	**1**	1	**81.66**	**11**
	E	**6.36**	0.88	1.75	1	75.07	17.5
K-means							
C	Alg	T-E	S-S	E +	E -	RBS	# I
100	P	**6.51**	**0.81**	2.25	**1.5**	80.19	12.75
	F	7.88	0.78	**0.25**	1.75	**81.20**	**11**
	K	7.10	0.75	2.75	2	76.55	17.5
150	P	**5.88**	**0.81**	1	**1.5**	78.72	13.75
	F	8.11	**0.81**	1	**1.5**	**81.09**	**11**
	K	7.09	0.75	**0.75**	2	76.09	17.5
200	P	**6.35**	**0.94**	1	**0**	78.77	13.75
	F	8.19	0.78	**0.25**	1.75	**81.60**	**11**
	K	7.17	**0.94**	1.25	0.5	77.53	17.5

Table 4: Evaluation of the algorithm with Uniform Distribution with EWD and K-means as initialization.

Equal-Width discretization							
C	Alg	T-E	S-S	E +	E -	RBS	# I
100	P	7.54	0.78	1.25	1.75	79.82	14.75
	F	7.94	**0.81**	**0.75**	**1.5**	**81.40**	**11**
	E	**6.86**	0.75	**0.75**	2	75.38	17.5
150	P	6.71	0.75	**0**	2	80.61	13.25
	F	8.01	0.75	0.75	2	**81.21**	**11**
	E	**5.84**	**0.78**	0.5	**1.75**	74.20	17.5
200	P	**5.89**	**1.00**	1	**0**	**81.69**	12
	F	7.23	0.88	**0.5**	1	80.99	**11**
	E	6.42	0.97	1.25	0.25	75.73	17.5
K-means							
C	Alg	T-E	S-S	E +	E -	RBS	# I
100	P	7.04	**0.78**	**0**	1.75	**80.28**	16
	F	7.99	0.72	1	**1.5**	79.89	**11**
	K	**6.96**	0.69	1.5	2.5	77.05	17.5
150	P	**6.03**	0.78	**0.25**	**1.75**	78.58	14
	F	7.69	**0.81**	0.5	**1.75**	**81.62**	**11**
	K	6.76	0.75	**0.25**	2	76.55	17.5
200	P	**6.64**	**0.91**	**1**	**0.75**	79.45	12.25
	F	7.24	0.78	**1**	2	**82.05**	**11**
	K	7.42	**0.91**	1.5	**0.75**	77.16	17.5

Table 5: Average results for all the experiments

Init	A	T. E.	S-S	E +	E -	RBS	# I	T
Gaussian Distribution								
E	P	**6.58**	**0.83**	1.17	**1.29**	80.66	13.5	**9.0**
E	F	8.61	0.78	**0.75**	1.80	**84.28**	**11**	10.7
E		6.90	0.77	1.28	1.83	77.91	17.5	-
K	P	**6.41**	**0.85**	1.33	**1.13**	79.66	14	**6.4**
K	F	7.98	0.78	**0.85**	1.80	**84.60**	**11**	8.7
K		6.74	0.78	1.43	1.70	75.77	17.5	-
Uniform Distribution								
E	P	**6.76**	**0.88**	**0.71**	**1.00**	75.77	13.5	**7.2**
E	F	7.69	0.84	0.95	1.35	**81.61**	**11**	9.3
E		6.83	0.79	0.79	1.67	77.45	17	-
K	P	**6.65**	**0.84**	**0.71**	**1.29**	79.93	14	**4.8**
K	F	7.67	0.76	1.05	1.90	**81.94**	**11**	6.3
K		6.69	0.81	1.30	1.43	75.49	17.5	-

fore, the intervals are smaller yielding a lower time error.

In the lower part of the Table 3 we can see the results for the same experiments but using K-means for the initialization of the intervals. Again, we vary the number of initial clusters from 2-5. We can observe that our algorithm obtains the best results in terms of time error and structural similarity. The results for 150 cases are interesting because they show that our algorithm and Friedman's obtained the same structural scores. However, our algorithm obtains the best score in time error with a lower accuracy than Friedman's algorithm.

For the second set of experiments, we generated data from a Uniform distribution and applied the same tests as those presented for Gaussian distribution. In the upper part of Table 4 we present the results for the experiments using EWD as initialization. Here, we can observe that as the number of cases increases, the same happens with the scores of our algorithm. With 200 cases, it obtains the best score for time error, structural similarity, and accuracy. In the lower part of Table 4, we present the results for the experiments using K-means as initialization for the intervals. The results show that our algorithm obtains the best time error and structural scores. Friedman's algorithm obtained the best RBS and the lowest number of intervals.

5.2 Overall Results

In Table 5 we report the average results for all the experiments. We also compare the algorithms in runtime (T). Each row presents the average of the experiments varying the number of cases in the range from 50-250, and varying the initial number of intervals from 2-5 using EWD or K-means as initialization. We observe that the proposed algorithm obtained the lowest time error and the best scores in structural learning. The RBS and the number of intervals were not the best but were not far from the best result. Moreover, the results show that the proposed algorithm obtained the lowest runtimes in all cases.

6 Conclusions and Future Research

We developed a method for learning both, the structure and the temporal intervals for a TNBN from data. The method initially generates a set of candidate intervals for each Temporal Node based on a Gaussian clustering algorithm. Then, the best intervals are selected based on their predictive accuracy. We evaluated our method with synthetic data and compared it with (Friedman and Goldszmidt 1996). In general, both methods obtain similar results in terms of structure and predictive accuracy. However, the proposed method is better in terms of time error and is also more efficient. As future research we propose to evaluate our model with larger synthetic cases and a real medical application.

Acknowledgments

This research was partially supported by project FONCICYT 95185. The first author is supported by a grant 234507 from Conacyt.

References

Arroyo-Figueroa, G., and Sucar, L. E. 1999. A temporal Bayesian network for diagnosis and prediction. In *Proceedings of the 15th UAI Conference*, 13–22.

Cooper, G., and Herskovits, E. 1992. A bayesian method for the induction of probabilistic networks from data. *Machine learning* 9(4):309–347.

Dagum, P.; Galper, A.; and Horvitz, E. 1992. Dynamic network models for forecasting. In *Proceedings of the 8th Workshop UAI*, 41–48.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* 39(1):1–38.

Friedman, N., and Goldszmidt, M. 1996. Discretizing continuous attributes while learning bayesian networks. In *Machine Learning -Int. Workshop 10th Conf.*, 157–165.

Galán, S. F.; Arroyo-Figueroa, G.; Díez, F. J.; and Sucar, L. E. 2007. Comparison of two types of event bayesian networks: A case study. *Applied Artificial Intelligence* 21(3):185–209.

Lam, W., and Bacchus, F. 1994. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational intelligence* 10(4):269–293.

Liu, W.; Song, N.; and Yao, H. 2005. Temporal functional dependencies and temporal nodes bayesian networks. *The Computer Journal* 48(1).

Neapolitan, R. 2004. *Learning bayesian networks*. Pearson Prentice Hall.

Pang, B.; Zhang, D.; Li, N.; and Wang, K. 2004. Computerized tongue diagnosis based on bayesian networks. *Biomedical Engineering, IEEE Transactions on* 51(10):1803–1810.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Wu, X.; Lucas, P.; Kerr, S.; and Dijkhuizen, R. 2001. Learning bayesian-network topologies in realistic medical domains. *Medical Data Analysis* 302–307.