Learning Temporal Bayesian Networks for Power Plant Diagnosis

Pablo Hernandez-Leal[1], L. Enrique Sucar[1], Jesus A. Gonzalez[1], Eduardo F. Morales[1], and Pablo H. Ibarguengoytia[2]

[1] National Institute of Astrophysics, Optics and Electronics
Tonantzintla, Puebla, Mexico
[2] Electrical Research Institute
Cuernavaca, Morelos, Mexico

Abstract. Diagnosis in industrial domains is a complex problem, because it includes uncertainty management and temporal reasoning. Dynamic Bayesian Networks (DBN) can deal with this type of problem. However, they usually lead to complex models. Temporal Nodes Bayesian Networks (TNBNs) are an alternative to DBNs for temporal reasoning, that result in much simpler and efficient models in certain 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 interval approximation, (ii) learn the network structure based on the intervals, and (iii) refine the intervals for each temporal node. The number of possible sets of intervals is obtained for each temporal node based on a clustering algorithm and the set of intervals that maximizes the prediction accuracy is selected. We applied the algorithm to learn a TNBN for predicting errors in a combined cycle power plant. The proposed algorithm obtains a simple model with high predictive accuracy.

1 Introduction

Power plants and their effective operation are vital to the development of industries, schools, and even for our houses, for this reason they maintain strict regulations and quality standards. However, problems may appear and when these happen, human operators have to take decisions relying mostly on their experience to determine the best recovery action with very limited help from the system. In order to provide useful information to the operator, different models have been developed that can deal with the industrial diagnosis. These models must manage uncertainty because real world information is usually imprecise, incomplete, and with errors (noisy). Furthermore, they must manage temporal reasoning, since the timing of occurrence of the events is an important piece of information.

Bayesian Networks [9] are an alternative to deal with uncertainty that has proven to be successful in various domains. Nevertheless, these models cannot

deal with temporal information. An extension of BNs called Dynamic Bayesian Networks, can deal with temporal information. 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) [1] are another extension of Bayesian Networks. They belong to a class of temporal models known as *Event Bayesian Networks* [5]. 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 [1]. However, one problem that appears when using TNBNs is that no learning algorithm exists, so, the model has to be obtained from external sources (i.e., a domain expert). 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, and apply it to the diagnosis of a combined cycle power plant.

The learning algorithm consists of three phases. In the first phase, we obtain an approximation of the intervals. F this, we apply a clustering algorithm. T we convert these clusters into initial intervals. For the second phase, the BN structure is obtained with a structure learning algorithm [2]. The last step is performed to refine the intervals for each Temporal Node. Our algorithm obtains the 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 applied the proposed method to fault diagnosis in a subsystem of a power plant. The data was obtained from a power plant simulator. The structure and intervals obtained by the proposed algorithm are compared to a uniform discretization and k-means clustering algorithm; the results show that our approach creates a simpler TNBN with high predictive accuracy.

2 Related Work

Bayesian Networks (BN) have been applied to industrial diagnosis [6] However, *static* BNs are not suited to deal with temporal information. For this reason Dynamic Bayesian Networks [3] were created. In a DBN, a copy of a base model is done for each time stage. These copies are linked via a transition network which is usually connected through links only allowing connections between consecutive stages (Markov property). The problem is that DBNs can become very complex; and 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 a fixed time interval between stages.

In TNBN, each variable represents an event or state change. S only one or a few instances of each variable are required. Assuming there is one or a few changes of a variable state in the temporal range of interest, n copies of the model are needed, and no assumption about the Markovian nature of the process is done. 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 [8]. Unfortunately, the algorithms used to learn BNs cannot deal with the problem of learning temporal intervals. S 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. [7] 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). 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, which represents the structure of the TNBN. The previous work assumes 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. In contrast, our approach constructs the TNBN directly from data, which in many applications is readily available or can be generated, for instance, using a simulator.

3 Temporal Nodes Bayesian Networks

A Temporal Nodes Bayesian Network (TNBN) [1, 5] 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 $s = (\text{'no change'}, \emptyset)$, which has no interval associated. If a Node has no intervals defined for all its states then it receives the name of Instantaneous Node.

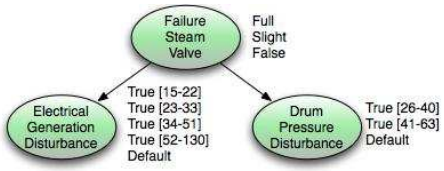

Fig. 1. The TNBN for Example 1. Each oval represents a node. The Failure Steam Valve is an Instantaneous Node, so it does not have temporal intervals. The Electrical Generation Disturbance and Drum Pressure Disturbance are Temporal Nodes. Therefore, they have temporal intervals associated with their values.

Example 1. Assume that at time $t = 0$, a Failure in a Steam Valve occurs. This kind of failure can be classified as *Full*, *Slight* and *False*. To simplify the model we will consider only two immediate consequences in the plant process, the Disturbance in Electrical Generation (DEG) and the Disturbance in the Drum Pressure (DDP). These events are not immediate. They depend on the severity of the accident. Therefore, they have temporal intervals associated. For the DEG node four intervals are defined $[15-22], [23-33], [34-51], [52-130]$, for the DDP node two intervals are defined $[26-40], [41-63]$. These intervals represent that the state of the node was assigned during that period of time. A TNBN for this simple example is shown in Figure 1.

4 Learning algorithm

First, we present the interval learning algorithm for a TN. We will assume that we have a defined structure, and 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 with the temporal data. E cluster corresponds, in principle, to a temporal interval. The algorithm is presented first by ignoring the values of the parent nodes (first approximation). L 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 [4]. 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 by: $[\mu - \sigma, \mu + \sigma]$.

Now we will 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 $\ell = 3$).

To select the best set of intervals, an evaluation is done over the network, which is 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: $RBS\ \ (in\ \ \%) = (1 - \frac{BS}{BS_{max}}) \times 100$. This RBS is used to evaluate the TNBN 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 precise approximation. For this, we use the configurations of the parent nodes. The number of configurations of each node i is $q_i = \prod_{Xt \in Pa(i)} |s_t|$ (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. For p_i and p_j, we apply the first approximation and obtain ℓ sets of intervals for each partition. In the last step, we obtain the combination of these sets of intervals, that yield to ℓ^2 sets of final intervals for each p_i,p_j. For example, if a node has parents: X (with states a, b) and Y (with states c, d), there are 4 partitions in total. We select two out of those four (six combinations) and apply the first approximation to each of them to obtain intervals. Finally, we combine those sets of intervals. After this process is applied, we have different sets of intervals, that we need to adjust. This adjustment is described in Algorithm 1.

Algorithm 1 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**

Algorithm 1 performs two nested loops. For each set of final intervals, we sort the intervals by their starting point. T we 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.

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 3), the state of the parent nodes is not initially known. So, we cannot 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 [1]). Once these are defined, we know the values of the parents of the nodes in the 3rd level, so 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 per node is in $O(q^2\ell^2)$, for this reason we used two pruning techniques for each TN to reduce the computation time.

The first pruning technique discriminates the partitions that provide little information to the model. 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 final 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 Structural Learning

Algorithm 2 Algorithm to obtain the initial intervals

Require: Sorted Points Pi obtained by k-means algorithm, An array of continuous values $data$ from Node n, min minimum value of $data$, max maximum value of $data$.

Ensure: Array of intervals for a Node n.

```
Interval[0].start=min,Interval[0].end=average(point[0],point[1])
for i=0 to size(Pi)-1 do
    Interval[i+1].start=average(point[i],point[i+1]) ,
    Interval[i+1].end=average(point[i+1],point[i+2])
end for
Interval[i].start=average(point[i],point[i+1]),Interval[i].end=max
```

Now we present the complete algorithm that learns the structure and the intervals of the TNBN. First we perform an initial discretization of the temporal variables based on a clustering algorithm (k-means), the obtained clusters are converted into intervals according to the process shown in Algorithm 2. 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 [2]. T algorithm has as a parameter an ordering of the nodes. For learning TNBN, we can exploit this parameter and define an order based on domain information.

When a structure is obtained, we can apply the interval learning algorithm described in Section 4.1. Moreover, this process of alternating interval learning and structure learning may be iterated until convergence. (With interval learning we obtain intervals for each TN, and we can apply the K2 structure algorithm, then with the structure, we can perform the interval learning again.)

5 Application to Power Plant Diagnosis

The proposed algorithm was tested on a subsystem of a combined cycle power plant. A simplified diagram is shown in Figure 2. In the process, a signal exceeding its specified limit of normal functioning is called an event.

5.1 Application Domain

A power plant mainly consists of three equipments: the steam generator (HRSG), the steam turbine and the electric generator. The steam generator, with the operation of burners, produces steam from the feed-water system. After the steam is superheated, it is introduced to the steam turbine to convert the energy carried out by the steam in work and finally, in electricity through the corresponding steam generator.

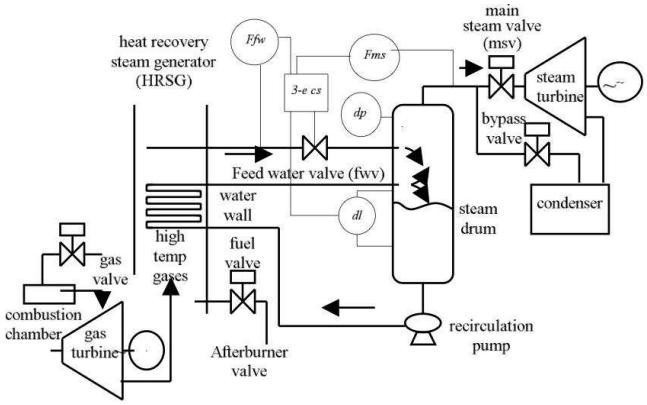

Fig. 2. Schematic description of a power plant showing the feedwater and main steam subsystems. Ffw refers to feedwater flow, Fms refers to main stream flow, dp refers to drum pressure, dl refers to drum level.

The HRSG consists of a huge boiler with an array of tubes, the drum and the circulation pump. The feed-water flows through the tubes receiving the heat provided by the gases from the gas turbine and the burners. Part of the water mass becomes steam and is separated from the water in a special tank called the drum. Here, water is stored to provide the boiler tubes with t appropriate volume of liquid that will be evaporated and steam is stored to be sent to the steam turbine.

From the drum, water is supplied to the rising water tubes called water walls by means of the water recirculation pump, where it will be evaporated, and water-steam mixture reaches the drum. From here, steam is supplied to the steam turbine. The conversion of liquid water to steam is carried out at a specific saturation condition of pressure and temperature. In this condition, water and saturated steam are at the same temperature. This must be the stable condition where the volume of water supply is commanded by the feed-water control system. Furthermore, the valves that allow the steam supply to the turbine are controlled in order to manipulate the values of pressure in the drum. The level of the drum is one of the most important variables in the generation process. A decrease of the level may cause that not enough water is supplied to the rising tubes and the excess of heat and lack of cooling water may destroy the tubes. On the contrary, an excess of level in the drum may drag water as humidity in the steam provided to the turbine and cause a severe damage in the blades. In both cases, a degradation of the performance of the generation cycle is observed.

Even with a very well calibrated instrument, controlling the level of the drum is one of the most complicated and uncertain processes of the whole generation system. This is because the mixture of steam and water makes very difficult the reading of the exact level of mass.

5.2 Experiments and Results

For obtaining the data used in the experiments, we used a full scale simulator of the plant, then we simulate two failures randomly: failure in the Water Valve and failure in the Steam Valve. These types of failures are important because, they may cause disturbances in the generation capacity and the drum.

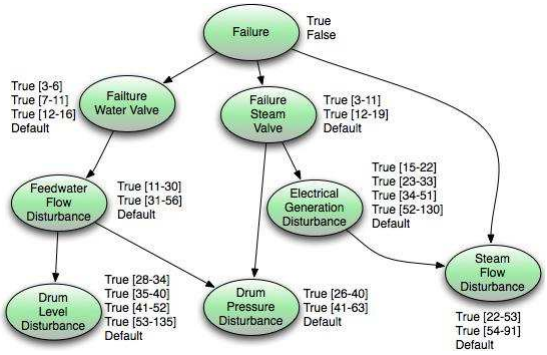

Fig. 3. The learned TNBN for a subsystem of a combined cycle power plant. For each node the obtained temporal intervals are shown.

In order to evaluate our algorithm, we obtained the structure and the intervals for each Temporal Node with the proposed algorithm. In this case, we do not have a reference network, so to compare our method, we used as baselines an equal-width discretization (EWD) and a K-means clustering algorithm to obtain the intervals for each TN. We evaluated the model using three measures: (i) the predictive accuracy using RBS, (ii) the error in time defined as the difference between the real event and the expected mean of the interval, and (iii) the number of intervals in the network. The best network should have high predictive RBS, low error in time and low complexity (reduced number of intervals).

We performed three experiments, varying the number of cases. The experiments are performed as follows. First, we generate the data with the simulator, then we learned the structure and the intervals. Finally, we used the learned network to compare the results with the original data. The results are presented in Table 1. The network obtained with the proposed algorithm with higher accuracy is presented in Figure 3.

The following observations can be obtained from these results. In all the experiments, our algorithm obtained the best RBS score and the lowest number of intervals. The K-means and EW discretization obtained the best score in time error. However, this happens because they produced a high number of intervals of smaller size, which decreases the difference between the mean of an interval and the real event. Even though our algorithm does not obtain the best time error, it is not far from the other algorithms. It is important to note that our algorithm obtains higher accuracy with a simpler model.

Table 1. Evaluation of the algorithm by inference in terms of RBS, time error and number of intervals

Num. of Cases	Algorithm	RBS (Max 100)	Time Error	Average num. intervals
50	Prop.	**93.26**	18.02	**16.25**
50	K-means	83.57	**15.6**	24.5
50	EWD	85.3	16.5	24.5
75	Prop.	**93.7**	17.8	**16**
75	K-means	85.7	**16.3**	24.5
75	EWD	86.9	17.2	24.5
100	Prop.	**93.37**	17.7	**17**
100	K-Means	90.4	17.1	24.5
100	EW D	91.9	**15.29**	24.5

6 Conclusions and Future Research

We have developed a method for learning both the structure and the temporal intervals for a TNBN from data. The method generates initially a set of candidate intervals for each Temporal Node based on a Gaussian clustering algorithm, and then the best intervals are selected based on predictive accuracy. We evaluated our method with data generated from a power plant simulator. The proposed method produces a simpler (low number of intervals) and better (high predictive accuracy) model than EWD and K-means clustering. As future work we propose to evaluate our model with a larger industrial case and apply the algorithm in other domains such as a medical case.

References

1. Arroyo-Figueroa, G., Sucar, L.: A temporal Bayesian network for diagnosis and prediction. In: Proceedings of the 15th UAI Conference. pp. 13–22 (1999)
2. Cooper, G., Herskovits, E.: A bayesian method for the induction of probabilistic networks from data. Machine learning 9(4), 309–347 (1992)
3. Dagum, P., Galper, A., Horvitz, E.: Dynamic network models for forecasting. In: Proc. of the 8th Workshop UAI. pp. 41–48 (1992)
4. Dempster, A., Laird, et al.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society 39(1), 1–38 (1977)
5. Galán, S.F., Arroyo-Figueroa, G., Díez, F.J., Sucar, L.E.: Comparison of two types of event bayesian networks: A case study. Appl. Artif. Intell. 21(3), 185–209 (2007)
6. Knox, W., Mengshoel, O.: Diagnosis and reconfiguration using bayesian networks: An electrical power system case study. SAS-09 p. 67 (2009)
7. Liu, W., Song, N., Yao, H.: Temporal functional dependencies and temporal nodes bayesian networks. The Computer Journal 48(1), 30–41 (2005)
8. Neapolitan, R.: Learning bayesian networks. Pearson Prentice Hall (2004)
9. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann (1988)