A Bayesian Network Approach to Explaining Time Series with Changing Structure

Allan Tucker and Xiaohui Liu
Department of Information Systems and Computing
Brunel University, Uxbridge,
Middlesex, UB8 3PH, UK
Tel: +44 (0)1895 816253
Fax: +44 (0)1895 251686
E-mail: allan.tucker@brunel.ac.uk

Abstract

Many examples exist of multivariate time series where dependencies between variables change over time. If these changing dependencies are not taken into account, any model that is learnt from the data will average over the different dependency structures. Paradigms that try to explain underlying processes and observed events in multivariate time series must explicitly model these changes in order to allow non-experts to analyse and understand such data. In this paper we have developed a method for generating explanations in multivariate time series that takes into account changing dependency structure. We make use of a dynamic Bayesian network model with hidden nodes. We introduce a representation and search technique for learning such models from data and test it on synthetic time series and real-world data from an oil refinery, both of which contain changing underlying structure. We compare our method to an existing EM-based method for learning structure. Results are very promising for our method and we include sample explanations, generated from models learnt from the refinery dataset.

1 Introduction

There are many examples of Multivariate Time Series (MTS) where dependencies between variables change over time. For example, variables in an oil refinery can be affected by the way operators control the processes and by the different products being refined at a particular time. If these changing dependencies are not taken into account, any model that is learnt from the data will average over the different dependency structures. This will not only apply to chemical processes but also to many other dynamic systems in general. There has previously been work in the modelling of time series with changing dependencies. Methods have been explored to model or cluster the hidden states of a system, which change over time [6, 11]. However, our focus is on making the underlying processes understood to non-statisticians through the automatic generation of explanations. Previously, we have developed methods for learning Dynamic Bayesian Networks (DBNs) from MTS [12]. In this paper we extend this work to handle changing dependencies and at the same time remain transparent so that the resulting models and explanations account for these changes.

In the next section, we introduce the dynamic cross correlation function and use it to analyse the changes in dependency structure between two variables within MTS. We then outline a method to learn DBNs from MTS with changing dependency structures, followed by details of the experiments and their results when applied to synthetic and real world MTS data from an oil refinery process. Finally, we generate explanations from the resulting structures and draw conclusions.

2 Methods

During the analysis of MTS we have found it useful to explore how the cross correlation function [2] between two variables changes over time. The cross correlation function is used to measure the correlation between two time series variables over varying time lags by time shifting the data before calculating the correlation coefficient. For the remainder of the paper we use the following notation: A MTS is denoted by A and $a_i(t)$ represents the ith variable at time t. Our analysis of MTS with changing dependencies has involved developing a Dynamic Cross Correlation Function (DCCF), $\rho_{a_i a_j}(l, t_s, t_f)$, whereby the cross correlation function is calculated for a window of data bounded by t_s and t_f over lags, l, between two variables, a_i and a_j, and moved over the MTS by incrementing the window position by set amounts. The DCCF is defined in equation 1 and 2 where $a_i(t_s, t_f) = a_i(t_s), a_i(t_s + 1), \ldots, a_i(t_f)$ and t_s and t_f are determined by the window position and length, and $Cov(a_i, a_j)$ returns the covariance function of a_i and a_j.

$$\rho_{a_i a_j}(l, t_s, t_f) = \frac{\gamma_{a_i a_j}(l, t_s, t_f)}{\sqrt{(\gamma_{a_i, a_i}(0, t_s, t_f)\gamma_{a_j, a_j}(0, t_s, t_f))}} \tag{1}$$

$$\gamma_{a_i a_j}(l, t_s, t_f) = Cov(a_i(t_s, t_f), a_j(t_s + l, t_f + l)) \tag{2}$$

The DCCF is generated by calculating the cross correlation function for all lags and window increments [13], which can be visualised as a surface plot such as the example in Figure 4a. Unlike the cross correlation function, the DCCF will not only measure correlation between two variables over various time lags but it will also measure the *change* in correlations over time lags. Whilst this DCCF will not be able to tell us about the more complex relationships within the MTS that involve more than two variables, it will assist us in making preliminary judgments about where likely dependency changes occur.

Bayesian Networks (BNs) are probabilistic models that can be used to combine expert knowledge and data [10]. They also facilitate the discovery of complex relationships in large datasets. A BN is a directed acyclic graph consisting of links between nodes that represent variables in the domain. The links are directed from a parent node to a child node, and with each node there is an associated set of conditional probability distributions. The Dynamic Bayesian Network (DBN) is an extension of the BN that models time series [4]. There has been much research into learning BNs from data, such as [3, 15]. We propose the use of a hidden node to marginalise over the different dependency distributions that exist in different portions of the MTS for each variable in a DBN. Previously, hidden nodes have been used in BNs for modelling missing data and unobserved variables [7], and Markov chains have been used to control dependencies in DBNs [1]. The challenge of learning models with changing dependencies is that we have to discover both the network structure and also the parameters related to the link between each variable and its hidden controller (as the actual states of the controller will be unknown).

2.1 Hidden Controller Hill Climb: Representation

We introduce a hill climb to search for DBNs with a specifically designed representation. By including a hidden controller node as a parent for each node at time t representing variable a_i, which we will refer to from now on as $Opstate_i$, the dependencies associated with that variable can be controlled (see Figure 1).

OpState nodes will determine how the variables in the MTS behave based upon their current state. The structure is represented using a list of triples of the form (a_i,a_j,l), which represents a link from a_i to a_j with a lag of l. We will call the list of triples $DBNList$. For learning models with changing dependencies, the *OpState* node is automatically inserted before evaluation. The changing of state of *OpState* is not likely to happen very often in process data because oil refinery data is recorded every minute whereas changes in operation are likely to be many minutes between. Therefore, the relative stability of these variables can be used to speed convergence of a hill climb. For this reason, the hill climb will use a list, *SegmentList*, of pairs (*state*, *position*) to represent the switches in each *OpState* variable, where *state* represents its new state and *position* represents the position of change in the MTS.

A heuristic-based hill climb procedure is employed by making small changes in $DBNList$ and the *SegmentList* for each *OpState* variable. In order to search over the DBN structure and the parameters of *OpState* for each vari-

able, the Hidden Controller Hill Climb (HCHC) algorithm uses the representation described above. HCHC takes as input the initial random $DBNList$ and a $SegmentList$ for each $OpState$ node, along with the MTS, A, the maximum time lag, $MaxT$, and the number of iterations for the structure search, $DBNIterations$. $DBNList$ and the N $SegmentLists$ are output and used to generate the final DBN structure and $OpState$ parameters. The $OpState$ variables can then be reconstructed using the $SegmentLists$. Once all $OpState$ variables have been constructed, DBN parameters can be learnt from the MTS.

2.2 Hidden Controller Hill Climb: Search

HCHC applies a standard hill climb search [9] to the $OpState$ parameters by making small changes to each $SegmentList$ and keeping any changes that result in an improved log likelihood score. The log likelihood function was first introduced in [3] and is commonly used to score potential network structures. It requires the DBN constructed from the $DBNList$ and the MTS with $OpState$ nodes constructed from the $SegmentLists$ (see Figure 2a). It is calculated as follows:

$$logp(D|bn_D) = \prod_{i=1}^{N} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(F_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} F_{ijk}! \tag{3}$$

where N is the number of variables in the domain, r_i denotes the number of states that a node x_i can take, q_i denotes the number of unique instantiations of the parents of node x_i, D is the dataset of observations, bn_D is the candidate structure of the BN, F_{ijk} is the number of cases in D, where x_i takes on its kth unique instantiation and, the parent set of i takes on its jth unique instantiation. $F_{ij} = \sum_{k=1}^{r} F_{ijk}$. Structure search is then carried out by repeatedly making small changes to $DBNList$ and keeping any changes that improve log likelihood. These changes involve adding and removing links or applying a lag mutation operator [13]. Note that the $Opstate_i$ nodes are fixed so that they cannot have parents and their only child is their respective variable, i. After structure search is repeated $DBNIterations$ times, the entire process is repeated, returning to the $OpState$ parameter search. The search ends when the function calls, FC, reach some pre-defined value, $Iterations$. HCHC is illustrated in Figure 2b and defined formally below.

```
Input A, MaxT, DBNIterations, Random DBNList, NSegmentLists
1     FC = 0
2     Use current DBNList and SegmentLists to construct DBN
3     BestScore = log likelihood of DBN
4     Repeat
5        For i=1 to N
6              Apply random change to the ith SegmentList
7              Use current DBNList and SegmentLists to construct DBN
8              If log likelihood of DBN > BestScore Then
9                     BestScore = log likelihood of DBN Else
10                    Undo change to ith SegmentList
11             End If
```

```
12        End For
13        For j=1 to DBNIterations
14              Apply random change to DBNList
15              Use current DBNList and SegmentLists to construct DBN
16              If log likelihood of DBN > BestScore Then
17                    BestScore = log likelihood of DBN Else
18                    Undo change to DBNList
19              End If
20        End For
21        FC = FC + 1
22    Until Convergence or FC>Iterations
OutputDBNList and N SegmentLists used to construct DBN
```

The HCHC Segmentation Algorithm

2.3 Structural Expectation Maximisation (SEM)

In [5] Structural Expectation Maximisation (SEM) was developed that searches for hidden variables whilst learning structure. The SEM algorithm involves an iterative process whereby the network (current model) is initialised with a random structure and a random set of parameters for the hidden variables (here the *OpStates*). The current model and training data are then used to generate the expected statistics for the hidden variables. In the context of Bayesian network parameters, this involves applying inference using the current model where the training data supplies the observations. The resulting posterior distributions over the hidden nodes are then used to determine their expected states given the training data. Given the expected statistics and the current model, a search engine is used to improve the log likelihood score of the network structure. This process of repeatedly calculating expected statistics and structure search continues until some stopping criteria is met such as minimal improvement in score or the maximum number of iterations is met.

```
Input Random initial structure and parameters for unobserved variables,
Iterations, MaxT
1     i = 0
2     Repeat until convergence or i > Iterations
            Expectation Step
3           Use inference to calculate the expected statistics for
            OpState given the current structure and values for OpState.
4           Use these statistics to reassign the values for
            OpState based upon the observed values in the MTS
            Maximum Likelihood Step
5           Search for structure that improves the expected score
            given the new values of OpState using a standard scoring
            metric such as log likelihood or DL
6           i = i + 1
7     End Repeat
Output Final Structure and Parameters
```

The SEM Algorithm for Learning DBNs with Changing Dependencies.

3 Results

3.1 Synthetic Data

Synthetic data has been generated from hand-coded DBNs using stochastic simulation [8]. The MTS datasets consisted of between five and ten variables with time lags of up to 60 time slices. In order to incorporate changing dependencies within the datasets, different DBN structures were used to generate different sections of the MTS. Essentially, several MTS were appended together, having been generated from DBNs with varying structures and parameters. For example, MTS 1 consists of three MTS with five variables appended together, each of length 1000. Characteristics of the datasets are described in Table 1. The experiments involved applying the HCHC search procedure to learn the dependency structure and segment the data for each variable according to its *OpState*. The resultant structures and segmentations at convergence were then compared to the original structures that generated each segment of the MTS by calculating the Structural Differences (SD) [15]. We also apply Friedman's SEM and compare results.

Having applied both HCHC and the SEM algorithm to the synthetic datasets, we now document the results. The average log likelihood at convergence is shown in Table 2. It is apparent that the log likelihoods of the DBNs constructed with the SEM method are much higher than those resulting from HCHC. This could be due to HCHC being limited in its number of segmentations whereas the SEM is not. However, as the true number of segmentations for each dataset was within the limits set for HCHC, this implies either an inefficient HCHC or overfitting in the SEM.

Table 3 shows the SD between the original and the discovered structures and the percentage of correct links discovered for both SEM and HCHC. For all datasets the HCHC generates networks with SDs that are relatively small and the percentage of correct links discovered is high. The SEM appears to have performed less well with consistently higher SD and smaller percentages of correct links found. This implies that the high scores in Table 2 are due to spurious correlations causing high SD.

Upon investigating the segmentation from SEM, it was found that the expected states of the *OpState* variables varied from segment to segment. For example, in one segment they may remain steadily in one state whilst in another they may fluctuate rapidly between 2 or more. See Figure 3 for an example of the fluctuating states discovered for $Opsate_4$ in MTS 3 over each of the 5 different segments.

It is apparent that the discovered segmentations do not neatly allocate a different state to different MTS segments and so statistics could not easily be used to measure the quality, suffice to say that there would have been hundreds of segmentations in each dataset. This could well be due to the model overfitting identified in the SD analysis. On the other hand, segmentation for the HCHC results was very good. We now discuss this with respect to MTS 3.

We make use of the DCCF in order to further analyse the results of the

HCHC on MTS 3. Whilst this is not really necessary for synthetic data where the original network structures are available, it will be useful when analysing real world data where these structures are unknown. Figure 4a shows the DCCF for variable a_3 and a_4 in MTS 3. The maximum time lag, $MaxT$, calculated for each CCF was 30. Note the varying peaks (in white) and troughs (in black), which signify stronger positive and negative correlations, respectively. Below are the discovered $DBNList$ and $SegmentList$ for $OpState_4$ as discovered by HCHC.

DBNList: ((1,0,8), (0,1,5), (0,1,9), (2,1,3), (2,1,6), (3,2,2), (1,2,7), (3,2,20), (4,3,3), (2,3,5), **(3,4,3), (3,4,25), (3,4,5)***)*

SegmentList for $OpState_4$: *((0,0),(498,1),(981,2),(1502,3),(1997,4))*

Figure 4b shows a graph of the most significant correlation (the maximum absolute correlation) for each window position in the DCCF. Firstly there is no correlation from a_3 to a_4 for window positions 1-10 (MTS position 1-500). Then an inverse correlation occurs (black trough denoted by $pos1$) with a lag of 25 in window positions 10-20 (MTS position 500-1000). This lag then switches to 3 for MTS positions 1000-1500 ($pos2$). Then a positive correlation can be seen (white area denoted by $pos3$) with a lag of 5 in MTS positions 1500-2000, and in the final sections of data, no significant correlation from a_3 to a_4 can be found. These correlations and lags correspond well to the links in $DBNList$ from a_3 to a_4 (in bold) and the $SegmentList$ for $OpState_4$, discovered for MTS 3.

3.2 Oil Refinery Data

The real world MTS in this paper is from an oil refinery process which contains 300 variables and 40 000 time points. We now look at an example of applying HCHC to an oil refinery variable and comparing the discovered segmentations with the DCCF between certain variables. Due to the poor results of SEM on both the synthetic and the oil refinery MTS, we focus on segmentations and explanations generated using HCHC. Figure 5 shows the resultant DBN structure discovered from a subset of 21 variables used to generate the following explanations.

In Figure 6, the most significant correlation graphs for each window position of the DCCF are shown for variable TGF with each of its three discovered parents. There is generally a fluctuation between no and strong negative correlations between TGF and A/MGB throughout the length of the MTS. However, at about window position 40, a positive correlation occurs. This continues until position 50 where the fluctuating returns until the end of the series. This closely follows the segmentation found using HCHC, which are marked in Figure 6 by dotted lines. The same applies for the other two parent variables with SOT also showing positive and negative correlations and T6T showing varying amounts of positive correlation. The segmentation appears to successfully separate out each of these regions. Similar results were discovered for all variables in the oil refinery data with most pair-wise relationships being successfully segmented.

Note that most of the segmentations that have been discovered occur where there are switches from positive to negative correlation rather than between regions of strong and weak correlation. Also some of the results appear to find

a number of segmentations that do not correspond with the correlation changes. This could be because the relationships that are changing are more complex than the pair-wise relationships identified in a DCCF. We explore this in the next section.

3.3 Explanations Incorporating Hidden Controllers

We generated a DBN structure for a subset of 21 oil refinery variables in order to explore some generated explanations. In general, the results from the process data have been very encouraging with many of the relationships identified using the DCCF being discovered. It must be noted that other more complex relationships may also have been discovered and we now look at some sample explanations that have been generated from the DBNs with *OpState* nodes included to identify these relationships. Given the discovered DBN structure and parameters for each node in the network including the set of *OpStates*, inference can be applied to generate explanations. This involves a process whereby certain observations are made about variables in the DBN and inference is used to generate posterior probability distributions over the unobserved variables. *OpStates* can also be included as part of the explanation.

Figure 7 shows some explanations that have been generated using the DBN discovered from the oil refinery data. Shaded nodes represent observations that have been entered into the DBN (hence the probability is 1.0). It can be seen in Figure 7a that SOT has a strong likelihood of being in state 3, 50 time points previously, given the instantiations and $OpState_{TGF}$ being 0. When $OpState_{TGF}$ changes to 3 as shown in Figure 7b the most probable state for each parent variable alters (SOT now most likely to be in state 0 and A/MGB in state 2). This shows how *OpStates* can affect explanation.

In Figure 8a the effect of adding new evidence to Figure 7b is shown, which changes all of the variable states again. In Figure 8b TGF and its set of parents are instantiated resulting in the controller variable $OpState_{TGF}$ being most likely in state 0, showing how an *OpState* node can be included as part of an explanation. These sample explanations indicate how complex relationships involving 3 or more variables can be modelled, illustrating how the segmentation of the MTS in Figure 6 may represent more complex relationships than those in its DCCF.

4 Conclusions

In this paper, we have used hidden nodes in a dynamic Bayesian network to model changing dependencies within a Multivariate Time Series (MTS). These hidden nodes can be interpreted as 'the current state of the system' and, therefore, be used within explanations of events. We introduce an algorithm, HCHC, which reduces the search space drastically through the use of a specific representation. It performs far better than EM methods for learning hidden nodes with BN structure. HCHC has allowed good models to be learnt from oil refin-

ery MTS, reflecting expected dependency structures and changes in the data. Future work will involve trying to improve the EM results using deterministic annealing [14] or some method of *smoothing* the control transition states by fixing their transition probabilities. We also intend to look at other datasets such as gene expression and visual field data where dependency structure is based upon experimental conditions. We will use HCHC to classify these MTS into such operating states.

5 Acknowledgements

We would like to thank BP-Amoco for supplying the oil refinery data and Andrew Ogden-Swift and Donald Campbell-Brown for their control engineer expertise.

References

[1] J.A. Bilnes: Dynamic Bayesian Multinets Proceedings of the 16th Annual Conference on Uncertainty in AI, (2000) 38-45

[2] C. Chatfield: The Analysis of Time Series - An Introduction Chapman and Hall, 4th edition, (1989)

[3] G.F. Cooper, E. Herskovitz: A Bayesian Method for the Induction of Probabilistic Networks from Data Machine Learning, (1992) Vol. 9, 309-347.

[4] N. Friedman: Learning the Structure of Dynamic Probabilistic Networks Proceedings of the 14th Annual Conference on Uncertainty in AI, (1998) 139-147

[5] N. Friedman: The Bayesian Structural EM Algorithm Proceedings of the 14th Annual Conference on Uncertainty in AI, (1998) 129-138

[6] Z. Ghahramani, G.E. Hinton: Variational Learning for Switching State-Space Models Neural Computation, (1999), Vol. 12, No. 4, 963-996

[7] D. Heckerman: A Tutorial on Learning with Bayesian Networks Technical Report, MSR-TR-95-06, Microsoft Research, (1996)

[8] M. Henrion: Propagating uncertainty in Bayesian networks by probabilistic logic sampling Proceedings of the 2nd Annual Conference on Uncertainty in AI, (1988) 149-163

[9] Z. Michalewicz, D.B. Fogel: How To Solve It: Modern Heuristics,Springer, 1998

[10] J. Pearl: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference Morgan Kaufmann, (1988)

[11] M. Ramoni, P. Sebstiani, P. Cohen: Bayesian Clustering by Dynamics, Machine Learning, (2002), Vol. 47, No. 1 91-121

[12] A. Tucker, X. Liu, A. Ogden-Swift: Evolutionary Learning of Dynamic Probabilistic Models with Large Time Lags. International Journal of Intelligent Systems **16**, Wiley, (2001) 621-645

[13] A. Tucker: The Automatic Explanation of Multivariate Time Series PhD Thesis, Birkbeck College, University of London, (2001)

[14] N. Ueda, R. Nakano: Deterministic Annealing Variant of the SEM algorithm Advances in Neural Information Processing Systems 7, (1995) 545-552

[15] M. Wong, W. Lam, S. Leung: Using Evolutionary Programming and Minimum Description Length Principle for Data Mining of Bayesian Networks IEEE Transactions on Pattern Analysis and Machine Intelligence, (1999), Vol. 21, No.2, 174-178

Table 1: Details of the Synthetic Data with Changing Dependencies

-	Num of Variables	MTS Length	Segment Length	Maximum Lag	Num of Segments
MTS 1	5	3000	1000	5	3
MTS 2	10	3000	1000	60	3
MTS 3	5	2500	500	25	5

Table 2: Resulting Log Likelihoods on Synthetic Data

-	HCHC	SEM
MTS 1	-3115.665	-220.738
MTS 2	-15302.017	-2032.371
MTS 3	-4494.706	-239.346

Table 3: Structural Difference Results on Synthetic Data

Method	Dataset	Total SD	% Correct
-	**MTS 1**	5.6	0.916
HCHC	**MTS 2**	6.7	0.892
-	**MTS 3**	5.8	0.913
-	**MTS 1**	9.6	0.725
SEM	**MTS 2**	16.8	0.681
-	**MTS 3**	10.8	0.488

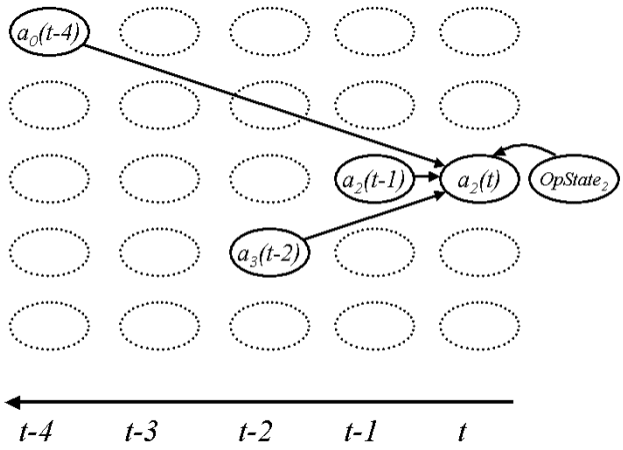

Figure 1: Using a Hidden Variable, $OpState_2$, to Control Variable a_2 at Time t

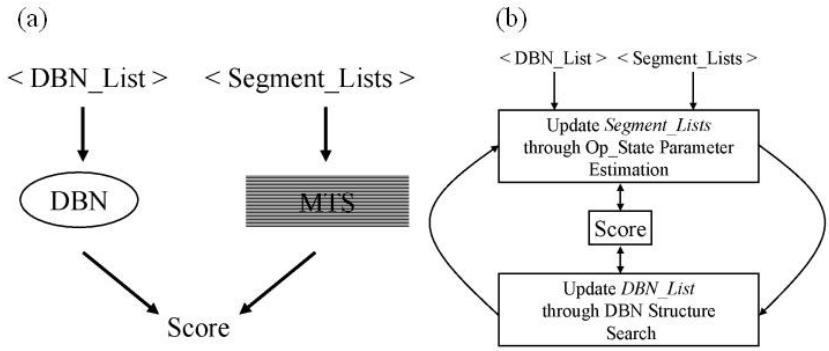

Figure 2: (a) Building a DBN from $SegmentLists$ and $DBNList$. (b) The HCHC procedure

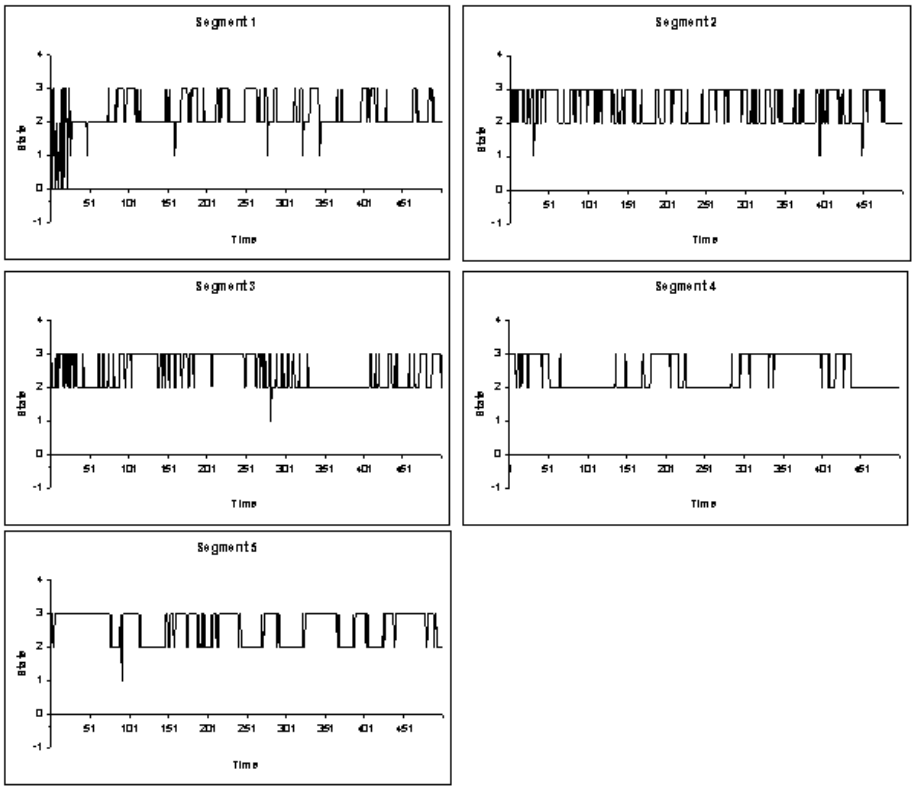

Figure 3: The Discovered States of the Hidden Node using SEM for Variable 4 in MTS 3, $Opstate_4$

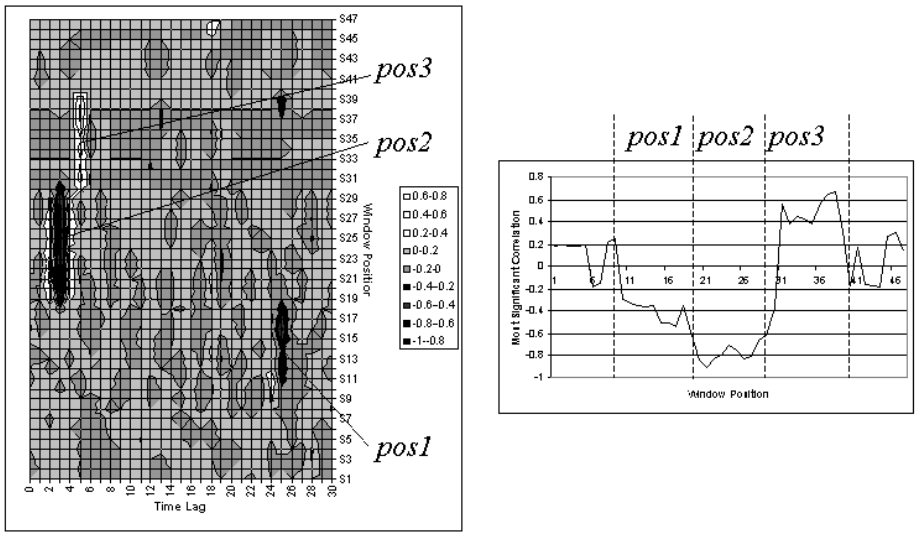

Figure 4: (a) The DCCF for Variable a_3 and a_4 in MTS 3. (b) The Most Significant Correlation for Each Window Position in (a)

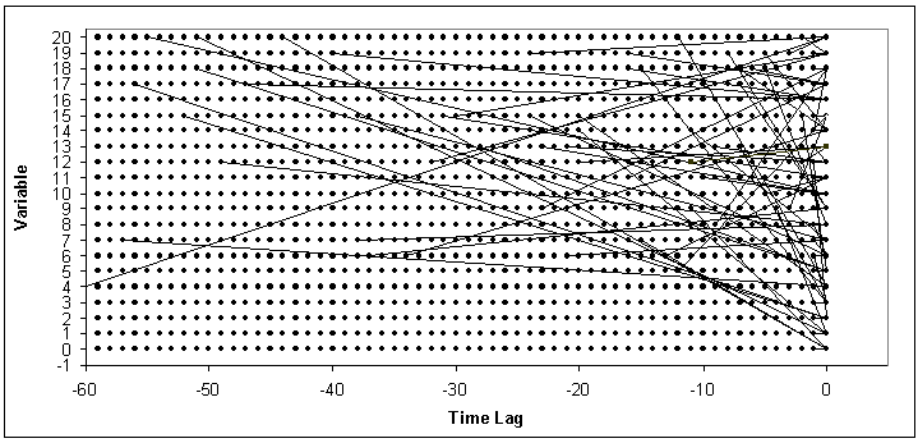

Figure 5: The DBN Structure learnt from the Oil Refinery Data

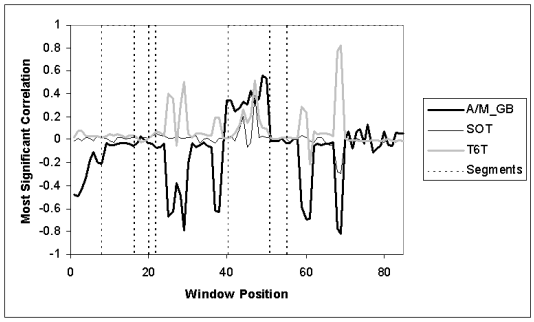

Figure 6: Most Significant Correlations for DCCFs from Oil Refinery Variables

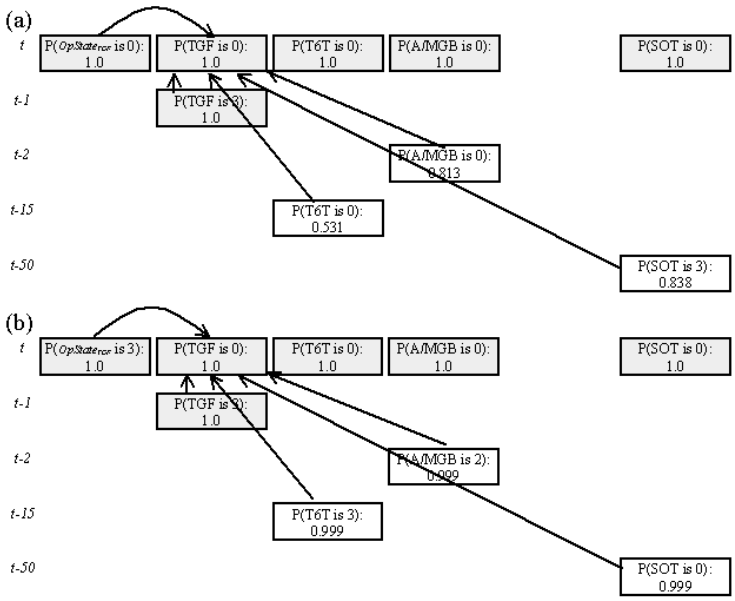

Figure 7: Sample of Generated Explanations from the Oil Refinery DBN

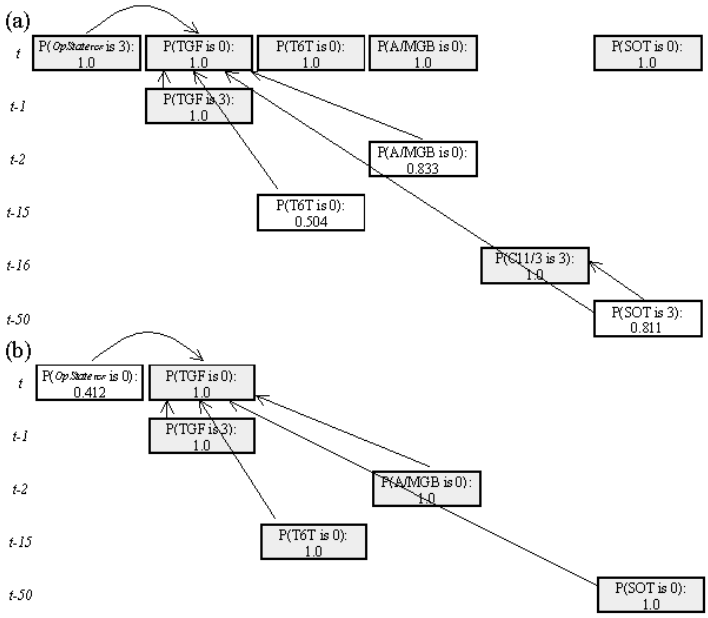

Figure 8: Sample of Generated Explanations from the Oil Refinery DBN