# A Framework for Modelling Virus Gene Expression Data

Paul Kellam[1], Xiaohui Liu[2], Nigel Martin[3], Christine Orengo[4], Stephen Swift[2], and Allan Tucker[2]

[1]Department of Immunology and Molecular Pathology,

University College London, Gower Street, London, WC1E 6BT, UK

P.Kellam@ucl.ac.uk

[2]Department of Information Systems and Computing, Brunel University,

Uxbridge, Middlesex, UB8 3PH, UK

{Xiaohui.Liu, Stephen.Swift, Allan.Tucker}@brunel.ac.uk

[3]Department of Computer Science, Birkbeck College,

Malet Street, London, WC1E 7HX, UK

Nigel@dcs.bbk.ac.uk

[4]Department of Biochemistry and Molecular Biology,

University College London, Gower Street, London, WC1E 6BT, UK

C.Orengo@biochemistry.ucl.ac.uk

**Abstract.** Short, high-dimensional, Multivariate Time Series (MTS) data are common in many fields such as medicine, finance and science, and any advance in modelling this kind of data would be beneficial. Nowhere is this truer than functional genomics where effective ways of analysing gene expression data are urgently needed. Progress in this area could help obtain a "global" view of biological processes, and ultimately lead to a great improvement in the quality of human life. We present a computational framework for modelling this type of data, and report experimental results of applying this framework to the analysis of gene expression data in the virology domain. The framework contains a three-step modelling strategy: correlation search, variable grouping, and short MTS modelling. Novel research is involved in each step which has been individually tested on different real-world datasets in engineering and medicine. This is the first attempt to integrate all these components into a coherent computational framework, and test the framework on a very challenging application area, producing promising results.

# 1 Introduction

Short, high-dimensional MTS data are common in many fields such as medicine, finance and science, and any advance in modelling this kind of data would be beneficial. One of the most exciting applications of this type of research is the analysis of gene expression data, often a short and high-dimensional MTS. For the first time, DNA microarray technology has enabled biologists to study all the genes within an entire organism to obtain a global view of gene interaction and regulation. This technology has great potential in providing a deep understanding of biological processes, and ultimately, could lead to a great improvement in the quality of human life. However, this potential will not be fully realised until effective ways of analysing gene expression data collected by DNA array technology have been found.

As gene expression data is essentially a high-dimensional but very short MTS, it makes sense to look into the possibility of using time series modelling techniques to analyse these data. There are many statistical MTS modelling methods such as the Vector Auto-Regressive (VAR) process, Vector Auto-Regressive Moving Average [Lutk93], non-linear and Bayesian systems [Casd92], but few of these methods have been applied to the modelling of gene expression data. This is perhaps not surprising since the very nature of these time series data means that a direct and fruitful application of these methods or indeed their counterparts in dynamic systems and signal processing literature will be extremely hard. For example, some methods such as the VAR process place constraints on the minimum number of time series observations in the dataset; also many methods require distribution assumptions to be made regarding the observed time series, e.g. the maximum likelihood method for parameter estimation.

Our recent work, based on the modelling of MTS data in medical and engineering domains, has demonstrated that it is possible to learn useful models from short MTS using a combination of statistical time series modelling, relevant domain knowledge and intelligent search techniques. In particular, we have developed a fast Evolutionary Program (EP) to locate variables that are highly correlated within high-dimensional MTS [Swif99], strategies for decomposing high-dimensional MTS into a number of lower-dimensional MTS [Tuck01a], a modelling method based on Genetic Algorithms (GA) and the VAR process which bypasses the size restrictions of the statistical methods [Swif02], and a method for learning Dynamic Bayesian Networks (DBN) structure and parameters for explanation from MTS [Tuck01b].

## 2 Methodology

We have examined the application of our framework to learning MTS models from gene expression data. A MTS is defined as a series of $n \times T$ observations, $x_{it}$, $i = 1, ..., n$, $t = 1, ..., T$, made sequentially through time where $t$ indexes the different measurements made at each time point, and $i$ indexes the number of variables in the time series. Figure 1 shows our methodology for modelling short, high-dimensional MTS, which reduces the dimensionality of the MTS before model building, and explicitly manages the temporal relationships between variables. The methodology involves three main stages.

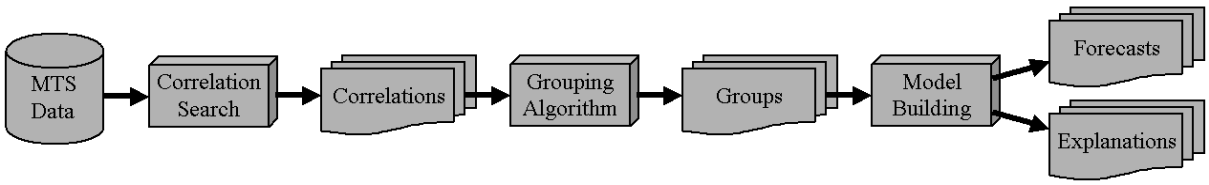



**Figure 1.** Methodology

Firstly, a **Correlation Search** is performed on the data, producing a set of strong correlations between variables over differing time lags. The correlations produced from the MTS data are

then fed into a **Grouping Algorithm** producing a set of groups of variables where there are a high number of correlations between members of the same group, but a low number of correlations between members of different groups. These groups are then used as a basis for the **Model Building** process. In this paper the models we build are DBNs, but our methodology can employ a variety of other techniques such as the VAR process. Forecasts and explanations are then produced using the discovered models.

### 2.1 Correlation Search

The first stage of the methodology constructs a list, $Q$, of correlations which contains pairs of highly correlated variables over all possible integer time lags from zero to some positive maximum. Each element of $Q$ is a *triple* made up of two variables and a time lag, *lag*. For example, the triple ($x_1$, $x_2$, 2) represents the correlation between $x_1$ and $x_2$ with a time lag of 2. The search can be either exhaustive or approximate in nature depending upon the application in question. For example, in [Swif99] evolutionary algorithms have been used for performing this search efficiently for large and time-limited domains. The construction of $Q$ is constrained so that $Q$ only contains one of the triples ($x_i$, $x_j$, *lag*) and ($x_j$, $x_i$, *lag*), namely the one with the highest correlation, and no triples of the form ($x_i$, $x_i$, *lag*). $Q$ is then used in conjunction with the grouping strategy described in Section 2.2.

In these experiments we use Pearson's Correlation Coefficient (PCC) [Sned67], defined in equation 1, but other metrics have been successfully employed in our methodology [Tuck01a]. PCC measures linear relationships between two variables, either discrete or continuous. We can calculate the PCC between two variables over differing time lags by shifting one variable in time.

$$PCC(x_1, x_2, lag) = \frac{\sum_{i=1}^{T-lag}(x_{1i} - \mu_{x_1})(x_{2(i+lag)} - \mu_{x_2})}{\sqrt{\sum_{i=1}^{T-lag}(x_{1i} - \mu_{x_1})^2 \cdot \sum_{i=1}^{T-lag}(x_{2(i+lag)} - \mu_{x_2})^2}} \quad (1)$$

where $\mu_x$ denotes the mean of the MTS variable $x$.

The correlation list, Q, is constructed using an Evolutionary Programming (EP) algorithm [Bäck93]. EP uses the notion of a population of chromosomes which represent a number of possible solutions to a particular problem. A *Mutation* operator is applied to these chromosomes according to some mutation rate. A selection process is applied to the population in order to preserve "good" solutions (e.g. chromosomes with strong correlations) and discard "poor" ones. The process is iterated over the chromosomes for a specified number of times. Our general EP algorithm for generating a list of high scoring triples is given below where the MTS, X is input and a list of $R$ triples, $Q$ is output and $c$ determines the number of calls to the scoring function (correlation):

**Algorithm 1.** Evolutionary Program for Correlation Search.

```
Input:  X (a n×T MTS), R, c
1.      Set Q = an empty list
2.      Generate R random triples consistent with X and insert into Q
3.      Set CallCount = R
4.      While CallCount < c
5.         Set Children to Q
6.         Apply Mutation operator to Children
7.         Insert valid Children into Q
8.         Update CallCount by the number of valid Children
9.         Sort Q
10.        Apply a survival operator to Q
11.     End While
Output: Q of length R
```

New individuals are considered invalid if they are already in $Q$. Traditionally, EP algorithms use *Tournament Selection* [Bäck93] during the survival of the fittest stage and the best chromosome out of the final population is the solution to the problem. However, it was decided that the entire population would be the solution for our EP method. That is, each individual chromosome would represent a single correlation (a triple) while the population

would represent the set of correlations found (population size = $R$). Hence the survival operator consists of keeping the best $R$ individuals. Although the entire population represents the solution, it should be noted that the fitness of each individual is still independent of the rest of the population. Each individual tries to maximise the magnitude of the correlation coefficient that it represents. This in turn maximises the population's fitness by improving the correlations represented by the population.

Within our EP method, a gene is $x_i$, $x_j$, or the lag. We have used the idea of *Self-Adapting Parameters* [Bäck96] in this context, with each gene, $gene_i$, in each chromosome being associated with a parameter, $\sigma_i$. Hence, every chromosome consists of the three $x_i$, $x_j$, and *lag* values and their corresponding $\sigma_i$ parameters. Each gene is then mutated according to a Normal distribution with mean 0 and standard deviation equal to the gene's corresponding standard deviation, $\sigma_i$, as specified in equation 2, while each $\sigma_i$ is itself mutated according to equation 3.

$$gene_i = gene_i + N(0, \sigma_i) \tag{2}$$

$$\sigma_i = \sigma_i \cdot \exp(N(0, \tau) + N(0, \tau_i)) \tag{3}$$

$$\tau = \frac{1}{\sqrt{2len}} \tag{4}$$

$$\tau_i = \frac{1}{\sqrt{2\sqrt{len}}} \tag{5}$$

Note that $\tau$ is constant for each gene in each chromosome but differs between chromosomes, while $\tau_i$ is different for all genes. Both parameters are generated each time mutation occurs. Each chromosome consisted of three parameters and their corresponding $s_i$ values. The value of *len* is the size of each chromosome, i.e. three. A check is required after mutation for any duplicates and for any invalid chromosomes. Any children that fall into this category are repeatedly mutated until they become valid.

The *Survival* operator involves removing triples from the population based on their fitness (i.e. their correlation magnitude irrespective of sign) until the population is of size *R* once again. Therefore, the *R* chromosomes with the highest magnitude of correlation are preserved for the next iteration.

### 2.2 Grouping

The grouping stage makes use of Falkenauer's Grouping Genetic Algorithm (GGA) [Falk98]. This uses a search procedure in conjunction with the results of the correlation search and a metric, *Partition Metric*, to score possible groupings. Note that as a result of this, the size of *Q* will substantially affect the final groupings discovered. Groupings are scored according to *Partition Metric* which groups variables together that have strong mutual dependency and separates them into different groups where the dependency is low.

Let $G = \{g_1, ..., g_m\}$ be a partition of the variables $\{x_1, ..., x_n\}$. That is, $\bigcup_{i=1}^{m} g_i = \{x_1,...,x_n\}$ and $g_i \cap g_j = \phi, 1 \le i \ne j \le m$. Given a list of correlations, *Q*, and $g \in G$, let

$g / Q = \{(x_i, x_j, lag) \in Q : x_i, x_j \in g\}, 1 \le i \ne j \le n.$

For any *g* in *G*, $| g / Q | \le ½| g | (| g | - 1)$ since there are no autocorrelations in *Q*. Hence, the number of non-correlations in *g* is $½| g | (| g | - 1) - | g / Q |$. We define the *Partition Metric* of *G*, denoted *f*(*G*), to be:

$$f(G) = \sum_{i=1}^{m}\left( | g_i / Q | -(\frac{1}{2} | g_i | (| g_i | -1)- | g_i / Q |\right) = \sum_{i=1}^{m}\left( 2 | g_i / Q | -\frac{1}{2} | g_i | (| g_i | -1)\right) \quad (6)$$

Intuitively, we sum the difference between the number of correlations and non-correlations over each group in *G*. Clearly we wish to maximise *Partition Metric*, which is discussed in detail in [Tuck01a].

The general GA [Holl95] is very much like an EP except that it makes use of a *recombination* operator as well as mutation which usually plays a lesser role in the search. Uniform crossover [Sysw89] is a common type of recombination but Falkenauer has developed a specific operator for grouping problems. The general GA is described below:

**Algorithm 2.** The General Genetic Algorithm.

```
Input:  Q , Popsize, CrossoverRate, MutationRate, Generations
        Fitness: The Partition Metric applied to a chromosome given Q
1.      Generate Popsize chromosomes
2.      For Loop = 1 to Generations
3.         Select Crossover Rate × Popsize chromosomes (with fitter chromosomes
           being chosen with higher probability and a chromosome can be chosen more
           than once)
4.         Randomly pair up selected chromosomes creating parent pairs
5.         Crossover parents to generate offspring
6.         Mutate offspring based on Mutation Rate
7.         Insert offspring into the population
8.         Sort the population according to their fitness
9.         Retain the Popsize fittest chromosomes
10.     End For
Output: G (a set of groups, constructed from the final fittest individual)
```

The representation for the GGA consists of a chromosome with each gene representing a variable in the domain. The value of the gene determines which group the variable is a member of. There is also an extra part on the chromosome that represents each group without any information about their contents. For example the following groupings:

**Group 0: 0 3 8** **Group 1: 2 7 4 1 5** **Group 2: 6 9**

would be represented by the following chromosome: **0 1 1 0 1 1 2 1 0 2 : 0 1 2**. The second part of the chromosome (after the colon) is simply a list of the existing groups that are found in the first part.

Crossover is only applied to the second part of the chromosome and is as follows:

1. Select two random crossing sites, delimiting the crossing section in each of the two parents denoted as [Start Position, End Position].
2. Inject the contents of the crossing section of the second parent at the first crossing site of the first parent.
3. Remove any elements that are repeated from the groups that were members of the first parent.
4. Remove any empty groups (groups that appear after the colon but not before) and reinsert any unassigned variables to existing groups.
5. Repeat (1) to (5) to produce the second offspring by reversing the roles of the first and second parent.

Example for first offspring:

**Parent 1: 0 1 1 0 0 2 1 2 : 0 1 2** **Parent 2: 4 5 3 4 5 6 3 6 : 3 4 5 6**

1. Starting with a copy of Parent 2 with all the first section undetermined and Cross Sites set as: Parent 1 = [0,1], Parent 2 = [1,3]

**? ? ? ? ? ? ? ? : 3 4 5 6**

2. Inject group 0 (determined from cross site limits [0,1] on parent 1) into position 1

**0 ? ? 0 0 ? ? ? : 3 0 4 5 6**

3. Remove group 4 and 5 due to repeats (the new group, 0, clashes with the old position of these two groups on the left part of the chromosome). Then fill in the remaining groups on the left part according to their old position (from parent 1).

**0 ? 3 0 0 6 3 6 : 3 0 6**

4. Reinsert variable 1 (which is at present unassigned) into random group (here 6)

**0 6 3 0 0 6 3 6 : 3 0 6**

where ? denotes an unallocated variable (adapted from [Falk98]).

Mutation involves randomly mutating groups (on the right side of the colon) according to the Mutation Rate. Each gene has Mutation Rate probability of being mutated so that the group is randomly split into two new groups or combined with another existing group. Therefore, for the offspring in the previous example, group 0 may be mutated by splitting the elements into two new groups (1 and 2) or combining it with another group (say 3).

### 2.3 Modelling

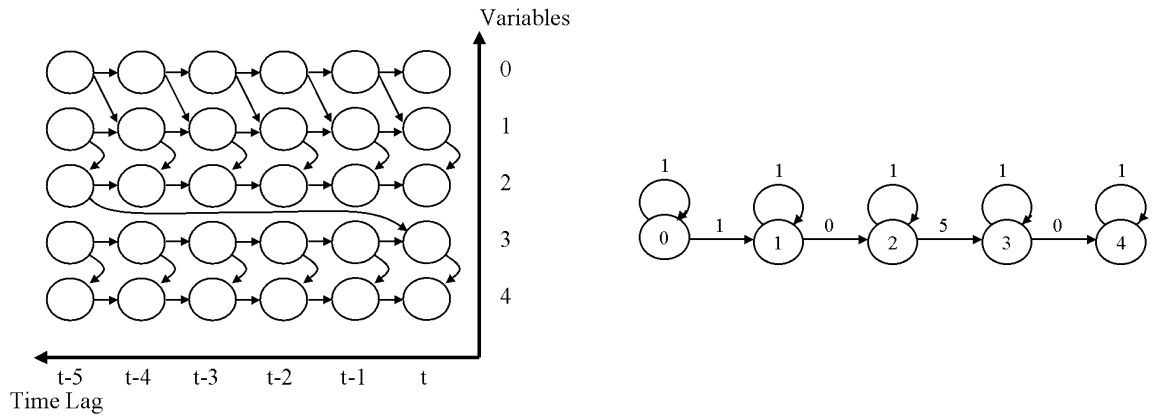
Within this framework, a modelling paradigm has been developed to explain and forecast MTS. This makes use of Dynamic Bayesian Networks (DBNs) which offer an ideal way to perform both explanation and forecasting. A Bayesian Network (BN) is a well-known model for performing probabilistic inference about a discrete system [Pear88], with its dynamic counterpart additionally modelling a system over time [Dagu95]. A DBN allows the testing of the effect of changing a biological process by manipulating key components of the system.

This is of particular interest to the pharmaceutical industry in trying to identify the best target in a modelled disease process for therapeutic intervention. A DBN consists of the following:

1. A set of $N$ nodes representing the $n$ variables in the domain at particular time slices and directed links between the nodes. Each node has a finite set of mutually exclusive states.
2. Associated to each node with a set of parents, there is an associated probability table.

Links can occur between nodes over different time lags (non-contemporaneous links) and within the same time lag (contemporaneous links). However, the topology of the network must not contain directed cycles: That is, the network must be a *Directed Acyclyic Graph* (DAG). Essentially a DBN can be considered as a Markov Model [Gilk96] where conditional independence assumptions are made. Figure 2(a) shows a DBN with five variables over six time lags where each node represents a variable at a certain time slice and each link represents a conditional dependency between two nodes. Given some evidence about a set of variables at time $t$, we can infer the most probable explanations for the current observations. Figure 2(b) shows the same network in a more compact representation where numbers within nodes represent the variable and numbers with links represent time lag.

When learning DBNs from high dimensional MTS, the search spaces can be huge. EP-Seeded GA is an algorithm for learning DBN models efficiently [Tuck01b] which does not suffer from local maxima due to the global nature of its search. It only searches for non-contemporaneous links assuming that all dependencies span at least one time slice. We are currently looking at extending this search using evolutionary methods to order the nodes so that contemporaneous links can also be found whilst ensuring the resultant structure is a DAG.




(a) Full Representation (b) Compact Representation.

**Figure 2.** A DBN with five variables over six time lags

Figure 3 illustrates the EP-Seeded GA process. It involves applying an EP in order to find a list of highly correlated triples. This is similar to the evolutionary correlation search [Swif99] as discussed in Section 2.1 but uses log likelihood [Coop92] on discretised data rather than a correlation coefficient and auto-correlations may be included. This list is then used to form the initial population of a GA which consists of subsets of the list, denoted $T_i$ in Figure 3. A form of uniform crossover is used to generate new candidate networks where the fitness function is the log likelihood of each network and a specific mutation is applied, *LagMutation*, which mutates only the lag portion of the triple based upon a uniform distribution.

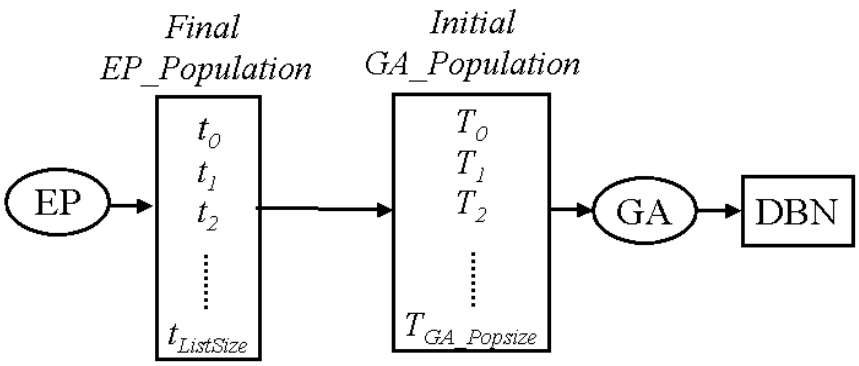



**Figure 3.** EP-Seeded GA

The algorithm is as follows:

**Algorithm 3.** EP-Seeded GA.

```
Input:  X (a n×T MTS), MaxT, EP_Generations, GA_ Generations,
        CrossoverRate, MutationRate, ListSize, GA_Popsize
1.      If the data is not discrete then apply some discretisation policy
        EP Seeding
2.      Initialise the population of the EP to a random selection of ListSize
        triples, t_i
3.      For Loop1 = 1 to EP_Generations
4.         Score all triples using log likelihood and sort according to score
5.         Add a copy of each link to the EP population and mutate according to
           equation (2)
6.         Cull the EP population to its original size by removing the lower ranking
           triples
7.      End For
        GA
8.      Initialise the population of the GA to GA_Popsize individuals, T_i, made up
        from a selection of triples, t_i, within the final EP population
9       For Loop2 = 1 to GA_Generations
10.        Score and sort each individual in the GA population by constructing the
           DBN represented by the list of triples and using the log likelihood of
           the network
11.        Apply Uniform Crossover based upon Crossover Rate to generate offspring
12.        Insert offspring into the GA population
13.        Apply LagMutation to the offspring and standard Mutation based Upon
           Mutation Rate
14.        Cull the GA population to its original size by removing lower ranking
           triple lists
15.     End For
Output: The best network structure with the largest log likelihood within the last
        generation of GA population
```

## 3 Application to Viral Gene Expression Data

Viruses have been studied extensively in molecular biology in order to understand cellular processes. Viruses also permit the study of complex multigene expression in the context of the human cell. Herpesviruses maintain an episomal genome of over 100 genes in the nucleus of infected cells. Under appropriate cellular stimulation a highly controlled cascade of viral transcription occurs with the clearly defined endpoint of new virus production. This therefore provides an excellent system for bioinformatic analysis of a transcriptome in the context of human cells. We have produced a DNA array of all known and putative open reading frames of human herpesvirus 8, consisting of 106 genes ($n$ = 106) expressed at eight time points ($T$ = 8).

The correlation search and grouping stages were applied to the expression data after pre-processing. This involved taking log ratios and then mean centring, both standard

techniques for normalising expression data. For the model building experiments in this paper, the gene expression data is discretised into 3 states based on whether a gene is under expressed (state 0), not expressed (state 1) or over expressed (state 2), taken from the work in [Jenn01]. Correlation search and grouping are carried out on the continuous data. Previous work has concentrated mostly on the clustering and classification of gene expression data. For example, in [Eise98, Jenn01] methods such as hierarchical clustering, K-means clustering and self-organising maps have been used to arrange genes into similar groups based on their gene expression profile. Some initial work has been performed on modelling gene expression profiles using BNs [Frie00]. However this work has not taken into account the time aspect of the expression data as we do here.

### 3.1 Correlation Search and Grouping Results

The correlation search used is an exhaustive method with maximum time lag of 3. The GGA was designed to look for between one and eight groups that maximised the number of correlations within a group. The number of correlations needed is 649, based on the calculations described in [Tuck01a]. The resulting number of correlations, using PCC, has an absolute average of 0.975 and an absolute standard deviation of 0.010. The groups formed are shown in Table 1. Biologically, these groups made sense to a varying degree. Some contained genes with phased inductions around similar time points (e.g. groups 0 and 1) and another contained mostly co-expressed host genes (e.g. group 7). However, some of these cellular genes were found in unlikely groups (e.g. group 5). A possible reason for this is that taking log ratios exaggerates the effects of negative expression values. Taking this into account, models that explain a state 1 to state 2 change are likely to have the more reliable biological meaning.

| Group (Size) | Member |
|---|---|
| 0 (10) | A1 F7 G2 G11 H3 H4 J5 K8 E3 I9 |
| 1 (15) | A2 A5 A6 A9 B1 G9 H5 C5 C9 C10 C12 J9 J12 D3 F1 |
| 2 (22) | A3 A4 A7 A8 A12 B5 B6 B8 B9 H8 H11 B12 K10 K11 C6 C7 J1 D11 D12 I5 E6 I6 |
| 3 (15) | K1 B2 B10 B11 G1 G5 G12 H10 C2 C3 C4 D1 D2 J6 D8 |
| 4 (12) | A10 A11 F11 K12 H2 H9 K5 C8 J7 J8 D9 E5 |
| 5 (11) | K4 F6 F9 F10 G4 G6 J10 D4 D10 K9 I2 |
| 6 (11) | F5 F8 F12 G3 H6 H7 J4 K6 I4 E7 I7 |
| 7 (10) | G8 J2 E1 I1 E2 I3 E4 E8 I8 E9 |

**Table 1.** GGA Results

### 3.2 EP-Seeded GA Results

The application of the EP-Seeded GA algorithm to learning DBNs from the eight grouped gene expression datasets has resulted in some highly connected networks. The fitness of these networks (their log likelihood) are documented in Table 2.

| Group (Size) | K2 Fitness | EP-Seeded GA Fitness | WK Forecast (1-Step Ahead) | WK Forecast (5-Step Ahead) |
|---|---|---|---|---|
| 0 (10) | 68.564 | 73.471 | 0.861 | 0.920 |
| 1 (15) | 79.934 | 92.273 | 0.796 | 0.753 |
| 2 (22) | 128.670 | 136.101 | 0.865 | 0.794 |
| 3 (15) | 95.184 | 104.434 | 0.848 | 0.761 |
| 4 (12) | 62.765 | 68.902 | 0.855 | 0.754 |
| 5 (11) | 74.857 | 81.276 | 0.824 | 0.882 |
| 6 (11) | 77.779 | 82.461 | 0.900 | 0.900 |
| 7 (10) | 68.904 | 71.746 | 0.946 | 0.918 |

**Table 2.** Final Fitness (Log Likelihhood) for K2 and EP Seeded GA. Also, the Weighted Kappa Scores (WK) for the Discrete Data Forecasts using the EP-Seeded GA DBNs.

This is in addition to the results of applying a well-known greedy algorithm, K2, introduced in [Coop92] for learning static networks, that has been adapted to learn DBNs in [Tuck01b] where we also compared EP-Seed GA to a number of other search methods. The fitness is clearly greater in the DBNs resulting from the EP-Seeded GA algorithm for all groups. Figure 4 shows the compact representation of two of the networks discovered by EP-Seeded GA (for groups 0 and 2).

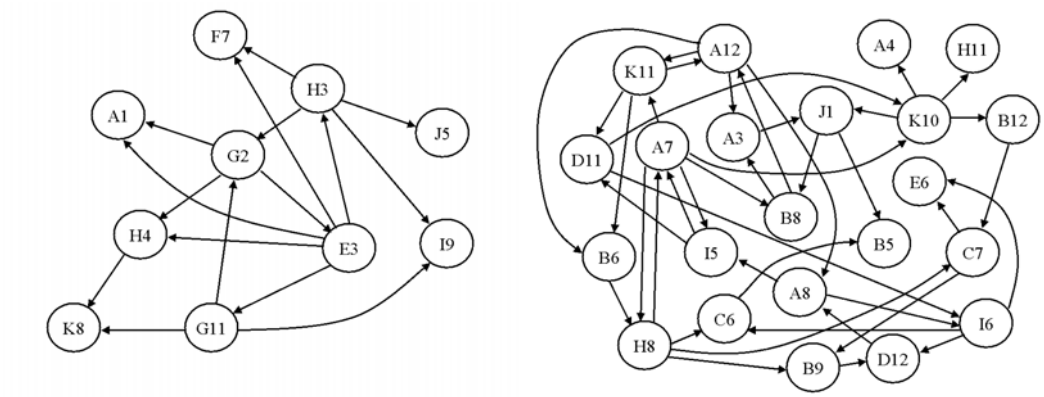



**Figure 4.** Sample DBNs learnt using EP-Seeded GA from group 0 and group 2 respectively (time lags and auto-regressive links have not been included).

Table 2 shows *Weighted-Kappa* (WK) values of the forecasts for each group given the first 3 values of each gene. WK is a metric used to rate agreement between the classification decisions made by two observers [Altm97], where a value of –1 indicates complete disagreement and +1 indicates full agreement. In this case observer 1 is the real data and observer 2 is the forecast data. For each group, the number of correct and incorrect forecasts is divided up into a 3 by 3 contingency table (for each state of the discrete variable). Once this table has been completed, WK can be computed using the procedure detailed in [Altm97].

For the forecasting experiments, all the data has been used for training, rather than testing the models on new data which is more common practice. This is because we want to obtain confidence in the discovered networks for generating explanations based upon the training data rather than generalising to the prediction of future observations. On the 5-step forecast, only the first 3 values of each gene were entered into the DBN. Based on this information, the DBN was used to forecast the next 5 values of each gene. On the 1-step ahead forecast the first 3 values of the genes were entered and the DBN was used to predict the next value for each gene. This was repeated for the next 5 time slices. Any WK value above 0.6 is considered *Good* and anything over 0.8 is considered *Very Good* [Altm97]. It can be seen that

all of the forecasts generated *Good* or *Very Good* WK values which implies a good model of the data has been learnt. As expected, the 1-step ahead forecast performed better in most cases than the 5-step forecast although the differences are small in each case.

One advantage of DBNs over other MTS models is that they are able to generate *transparent* explanations given certain observations based on the data. In this context, transparent means that the model and explanations are easily interpretable by users who are not experts in statistical and artificial intelligence models. Figure 5 illustrates two sample explanations generated from the networks in Figure 4.

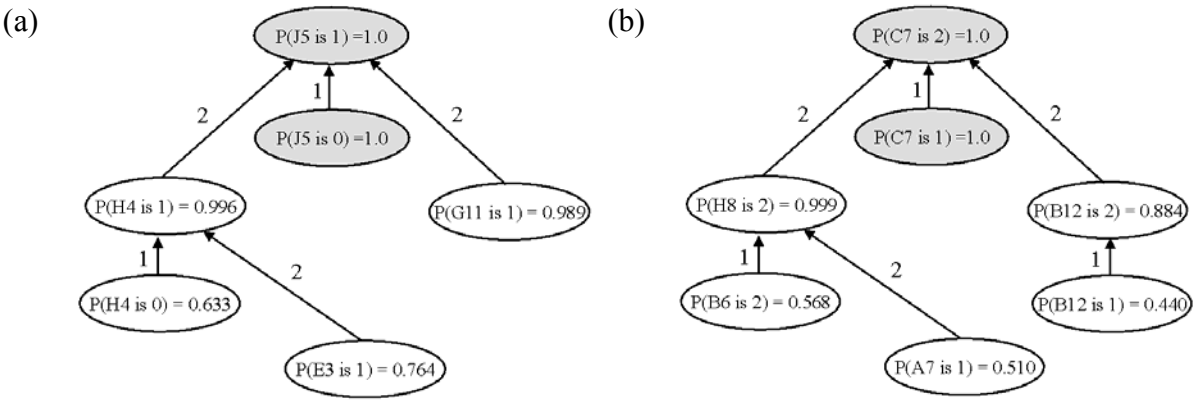



**Figure 5.** Sample Explanations generated using the DBNs in Figure 4

The explanations in Figure 5 show some of the nodes in the group 0 and group 2 networks relevant to the observed variables (in grey) for up to 4 time slices from the current time point. For example, Figure 5(a) contains the observations that gene J5 is currently in state 1 (not expressed) and was in state 0 (under expressed) one time slice previously. Some of the other related variables include H4, G11 and E3 which influence the observed variables over differing time lags and with differing probabilities.

One of the best tests for model building using this type of data is whether the model makes biological sense. This can be investigated by decoding the model and applying virology

domain knowledge. This was performed for the explanation in Figure 5(b). The model implies B12 (K-bZIP) and H8 (ORF 45) work in a network with, or to affect C7 (ORF 57). The induction of HHV8 genes as the virus begins its lytic replication cycle is controlled by the viral transactivator ORF 50. K-bZIP and ORF 57 are both up-regulated by ORF 50 but despite this the predictive model suggests subtle differential temporal regulation. Such differential gene expression by ORF 50 is known for other viral genes. Kb-ZIP is a basic leucine zipper protein suggesting a function as a transcription factor and/or DNA binding protein. Recent experimental data shows Kb-ZIP co-localises with the viral genome replication complex in specialised structures in the cell nucleus and may be involved with linking lytic genome replication with the cell cycle. ORF 45 has no known function but has an amino acid sequence reminiscent of a nuclear protein with transcription factor and/or DNA binding activity. This suggests both Kb-ZIP and ORF 45 are involved in similar processes and indicates obvious validating experiments. ORF 57 is a *key mediator* of virus lytic gene expression and specifically transports certain viral RNAs from the nucleus to the cytoplasm. Therefore, the fine temporal control of the HHV8 lytic cycle suggests that initial lytic genome replication events precede but are directly linked to a network involving ORF 57 mediated gene expression.

## 4. Concluding Remarks

Modelling short MTS data is a challenging task, particularly when the dataset is high-dimensional. Little work appears to have used time series methods to model this kind of data. Here we suggest a computational framework for modelling such data that explicitly manages the temporal relationships between variables within each step. We have obtained preliminary results when applying this method to the analysis of virus gene expression data. The groups found and their corresponding networks have varying degrees of biological

support. The forecasts produce very good WK statistics and some of the explanations have shown interesting biological connotations, indicating that DBNs can indeed model expression data with a high degree of accuracy.

The way we discretise the data will clearly have a large effect on the final DBNs and so future work will involve looking into ways to combine expert knowledge with existing techniques to minimise information loss. In addition we intend to investigate the use of continuous BNs to avoid the difficulties associated with discretisation. It would be useful to combine separate MTS data with common variables (e.g. different gene expression experiments) using a control node. We are currently investigating this so that more data can be used to learn the networks. We also intend to refine our framework in order to improve scalability and efficiency on both grouping variables and model building. It will be important to apply these methods to other datasets and we plan to do so on yeast expression data [Gasc00] and visual field data [Swif02]. In respect to the gene expression DBNs, we are working on collecting more detailed feedback from the biologists and carrying out follow-up experiments based upon the discovered networks to see if the discovered relationships can be confirmed.

**Acknowledgements**

This work is supported by the BBSRC, the EPSRC and the MRC in the UK. The authors would like to thank Jason Crampton for his constructive remarks on the *Partition Metric* and the manuscript in general, and Richard Jenner for preparing the gene expression data.

## References


[Altm97] Altman, D.G.: Practical Statistics for Medical Research, Chapman and Hall, London (1997).

[Bäck96] Bäck, T., Evolutionary Algorithms: Theory and Practice, Oxford University Press, (1996).

[Bäck93] Bäck, T., Rudolph, G., Schwefel, H.-P. Evolutionary Programming and Evolution Strategies: Similarities and Differences, Fogel, D.B. and Atmar, A., editor: Proceedings of the Second Annual Conference on Evolutionary Programming, 11-22, (1993).

[Casd92] Casdagli, M., Eubank, S.: Nonlinear Modeling and Forecasting, Addison Wesley (1992).

[Coop92] Cooper, G.F., Herskovitz, E.: A Bayesian Method for the Induction of Probabilistic Networks from Data, Machine Learning **9** (1992) 309-347.

[Dagu95] Dagum, P., Galper, A., Horvitz, E., Seiver, A.: Uncertain Reasoning and Forecasting, International Journal of Forecasting **11** (1995) 73-87.

[Eise98] Eisen, M.B., Spellman, P.T., Brown, P.O., Botstein, D.: Cluster Analysis and Display of Genome-Wide Expression Patterns, Proceedings of the National Academy of Science **95** (1998) 14863-14868.

[Falk98] Falkenauer, E.: Genetic Algorithms and Grouping Problems, Wiley (1998).

[Frie00] Friedman, N., Linial, M., Nachman, I., Pe'er, D.: Using Bayesian Networks to Analyze Expression Data, Journal of Computational Biology **7** (2000) 601-620.

[Gasc00] Gasch, A.P., Spellman, P.T., Kao, C.M., Carmel-Harel, O., Eisen, M.B., Storz, G., Botstein, D. and Brown, P.O.: Genomic Expression Programs in the Response of Yeast Cells to Environmental Changes, Molecular Biology of the Cell, **11**, (2000) 4241-57.

[Gilk96] Gilks, W.R., Richardson, S., Spiegelhalter, D.J.: Markov Chain Monte Carlo in Practice, Chapman and Hall, (1996).

[Holl95] Holland, J.H.: Adaptation in Natural and Artificial Systems, University of Michigan Press (1995).

[Jenn01] Jenner, R.G., Alba, M.M., Boshoff, C., Kellam, P.: Kaposi's Sarcoma-Associated Herpesvirus Latent and Lytic Gene Expression as Revealed by DNA Arrays, Journal of Virology **75** No. 2 (2001) 891-902.

[Lutk93] Lütkepohl, H.: Introduction to Multivariate Time Series Analysis, Springer-Verlag (1993).

[Pear88] Pearl, J.: Probabilistic Reasoning in Intelligent Systems, Networks of Plausible Inference, Morgan Kaufmann (1988).

[Sned67] Snedecor, G., Cochran, W.: Statistical Methods, Iowa State University Press, 6th edition (1967).

[Swif99] Swift, S., Tucker, A., Liu, X.: Evolutionary Computation to Search for Strongly Correlated Variables in High-Dimensional Time-Series. In: Hand, D.J., Kok, J.N., Berthold, M.R. (eds.): Advances in Intelligent Data Analysis 99, Springer-Verlag (1999) 51-62.

[Swif02] Swift, S., Liu, X.: Predicting Glaucomatous Visual Field Deterioration Through Short Multivariate Time Series Modelling, Artificial Intelligence in Medicine **24**, Elsevier (2002), 5-24.

[Sysw89] Syswerda, G.: Uniform Crossover in Genetic Algorithms, Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann (1989) 10-19.

[Tuck01a] Tucker, A., Swift, S., Liu, X.: Grouping Multivariate Time Series via Correlation, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, **31**, No. 2, (2001), 235-244.

[Tuck01b] Tucker, A., Liu, X., Ogden-Swift, A.: Evolutionary Learning of Dynamic Probabilistic Models with Large Time Lags, International Journal of Intelligent Systems, **16**, No. 5, Wiley (2001), 621-645.