Variable Grouping in Multivariate Time Series via Correlation

Allan Tucker, Stephen Swift, and Xiaohui Liu

Abstract--**The decomposition of high-dimensional multivariate time series (MTS) into a number of low dimensional MTS is a useful but challenging task because the number of possible dependencies between variables is likely to be huge. This paper is about a systematic study of the "variable groupings" problem in MTS. In particular, we investigate different methods of utilising the information regarding correlations among MTS variables. This type of method does not appear to have been studied before. In all 15 methods are suggested and applied to six datasets where there are identifiable mixed groupings of MTS variables. This paper describes the general methodology, reports extensive experimental results and concludes with useful insights on the strength and weakness of this type of grouping method.**

Index terms--**Multivariate Time Series, Grouping, Correlation, Genetic Algorithms, Evolutionary Programming.**

I. Introduction

There are many practical applications involving the partition of a set of objects into a number of mutually exclusive subsets. The objective is to optimise a metric defined over the set of all valid subsets, and the term grouping has been often used to refer to this type of problem. Examples of the grouping applications include bin packing, workshop layout design, and graph colouring [6]. Much research has been done on the grouping problem in different fields, and it was established that many, if not all grouping problems, are NP-hard [9]. Therefore, any algorithm that is guaranteed to find the global optimum will run in exponential time to the size of problem space, and a heuristic or approximate procedure is normally required to cope with most of the real world problems. A variety of techniques have been proposed to develop this procedure, including traditional clustering algorithms, hill-climbing and evolutionary algorithms. These techniques utilise a metric that takes relationships or dependencies between objects into account, and partition them into a number of mutually exclusive subsets [6].

When it comes to the problem of decomposing a high-dimensional multivariate time series (MTS) into a number of low dimensional MTS, the number of possible dependencies between time series variables becomes huge because one variable could affect another after a certain time lag. Therefore how to effectively utilise these dependencies becomes an important issue: to use all the possible dependencies in a variable grouping algorithm will be computationally infeasible for many, especially real-time, applications.

This paper is about a systematic study of the "variable groupings" problem in MTS. We investigate different heuristic methods for utilising the information regarding dependencies among MTS variables; this type of method does not appear to have been studied before. In all, 15 such methods are suggested and applied to six datasets where there are identifiable mixed groupings of MTS variables. Our methodology scores possible groupings based on a list of highly correlated pairings of variables. This list is not necessarily constructed from an exhaustive search and, therefore, makes the method applicable to massive data. The list size will strongly influence the final groupings and so a method for determining this parameter is sought for based on probabilistic simulation.

II. Grouping in Multivariate Time Series

MTS data are widely available in different fields including medicine, finance, science and engineering. Modelling MTS data effectively is important for many decision-making activities. A MTS is a series of observations, $x_i(t)$; [i=*1, ...,n; t=1,...,T*], made sequentially through time were i indexes the measurements made at each time point t.

Although much research has been carried out on modelling MTS for different purposes, little has been done on an important pre-processing issue: the grouping of MTS. When dealing with an n dimensional MTS, it is desirable to model the data as a group of smaller MTS models as opposed to a single one. Firstly, not all of the variables may be related, and secondly the number of parameters to be located in such a model would be very high. For example in forecasting, there are many statistical MTS modelling

This is the author affiliation (AuthorInfo style) area, The authors are adjuncts to IEEE Publishing Services, 445 Hose lane, Piscataway NJ.

methods such as the Vector Auto-Regressive (VAR), and other linear, non-linear and Bayesian systems [3,15,18]. Take the VAR(P) process as an example. There would be at least n^2P parameters to locate where P is the order of the VAR process and n is the number of variables in the data set. In explaining MTS, suppose we are trying to learn Dynamic Bayesian Network (DBN) models [5,8] from a MTS which has very high dimensionality, n, and large possible time lags, then the number of possible candidate networks will be $2^{MaxLag \cdot n^2}$ where *MaxLag* is the maximum time lag [23].

Decomposing the data into smaller dimensional time-series that are independent to some degree would narrow the search space a great deal allowing the speedier production of MTS models. Therefore we are interested in finding out how to decompose a high-dimensional MTS into groups of smaller MTSs, where the dependency between variables within the same group is high, but very low with variables in another group. Note that this is different from *dimensionality reduction* techniques such as principal component analysis or factor analysis which make some sort of multivariate transformation of the data [17].

III. Methodology

Given a MTS, we want to partition the variables into a number of smaller dimensional time series. The proposed methodology consists of two stages. Firstly a search over combinations of both variables and time lags (because time lag will affect the correlation between two MTS variables) is carried out in order to find a list of highly correlated variables. Let us call this list Q, which will be of length R. Q will consist of *triples* where a triple is made up of two variables and a time lag. For example, the triple $(x_1,\ x_2,\ 5)$ represents the correlation between x_1 and x_2 with a time lag of 5. Essentially all of the triples in Q represent the variable pairs that are deemed to be significantly correlated with the corresponding time lag. Therefore, it is important to estimate what R should be with a high degree of accuracy. We discuss this further in section IV. Stage two consists of an algorithm which is applied to Q where a specifically designed metric is used to group the variables in the original MTS based on the pairs of variables found in Q. Note that the lag portion of the triple is no longer used once the grouping algorithm is applied. This is because we are interested in grouping highly correlated variables irrespective of the time lag between them.

This section is arranged as follows. After, outlining the basic notation in section A, we introduce three methods for generating Q in section B. These methods are capable of generating a list of highly correlated variable pairs, which can then be used along with an appropriate metric by a grouping algorithm. In section C, a grouping metric is defined and its properties are studied. This is followed by the presentation of five different grouping search algorithms based on conventional clustering methods, hill climbing or evolutionary methods in section D.

A. Preliminaries

Given a MTS with n variables and of length T we want to partition each variable x_i into m groups where the size of each group will be denoted by k_i. This will be achieved by generating a list of "strong" correlations, Q, which will be of length R. Q will be calculated by using different searches through the number of possible correlations, s, where the number of calls to the correlation coefficient will be denoted by c. The aim of this search is to find the true underlying dependencies that generated the data. The number of "true" dependencies will be denoted by r.

B. The Correlation Search

The first stage of the methodology constructs Q which contains R pairs of highly correlated variables over all possible integer time lags from zero to some positive maximum, *MaxLag*. We want to find these correlations after exploring a fraction of the search space. Previously, we have compared different methods for performing this task [22] and have found that for operations where speed is essential, an evolutionary programming approach performs best. The correlation list generated using this method is then used in conjunction with the grouping strategy described below. Note that at time lag zero, the correlations represented by the triples $(x_i,x_j,0)$ and $(x_j,x_i,0)$ are effectively the same so duplicates are considered *invalid*. All triples of the form (x_i,x_i,lag) will also be considered invalid since these are auto-correlations and do not show relationships between different variables. All invalid triples are removed during the procedure.

1) The Exhaustive Search (EX)

The exhaustive search consisted of simply exploring all of the variables, at each time lag. The algorithm is detailed below.

```
Input: $X$ (a $T \times n$ MTS)
Set $Q$ = Empty List
For $i$ = 0 to $n$-1
    For $j$= 0 to n-1
        For $lag$ = 0 to $MaxLag$
            If the triple $(i, j, lag)$ is valid Then
                Insert the new triple, $(x_i,x_j,lag)$, into $Q$ and order
                (descending order of correlation magnitude)
                If size of $Q$ = $R$+1 Then remove the end triple of $Q$
            End If
        End For
    End For
End For
Output: $Q$ of length $R$
```

2) The Random Bag (RB)

This is a heuristic approach whereby a random selection of triples is placed in a "bag" containing R triples. With each iteration a new random triple is added to the bag. When the bag overflows, the worst correlation falls out. This is repeated for a predefined number of iterations. The algorithm is described below:

```
Input: $X$ (a $T\times n$ MTS)
Set $Q$ = Empty List
For $i = 0$ to $c$
        $i = U(0,n-1)$, $j = U(0,n-1)$, $lag = U(0,MaxLag)$ where $(i,j,lag)$ is valid
        If $a \notin Q$ then insert new triple, $(x_i,x_j,lag)$, into $Q$ and order
        (descending order of correlation magnitude)
        If size of $Q = R+1$ then remove the end triple from $Q$
End For
Output: $Q$ of length $R$
```

Note that c is the maximum number of allowed calls to the correlation function and U(min,max) returns a uniformly distributed random integer between min and max inclusive.

3) Evolutionary Programming (EP)

Evolutionary Programming is based on a similar paradigm to Genetic Algorithms. However, the emphasis is on mutation and the method does not use any recombination. The basic algorithm is outlined as follows [2,7]:

```
Input: $X$ (a $T\times n$ MTS)
Set $Q$ = Empty List
Generate $R$ random triples and insert into $Q$
Set $CallCount = R$
While $CallCount < c$
        Set Children to $Q$
        Apply Mutate operator to Children
        Insert valid Children into $Q$
        Update $CallCount$ by the number of valid Children
        Sort $Q$
        Apply Survival operator to $Q$
End While
Output: $Q$ of length $R$
```

A Child will be considered invalid if it is already in Q. Traditionally, EP algorithms use *Tournament Selection* [1] during the survival of the fittest stage and the best chromosome out of the final population will be the solution to the problem. However, it was decided that the entire population would be the solution for our EP method as in the RB 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 consisted of keeping the best R individuals. Although the entire population would represent the solution, it must be noted that the fitness of each individual would still be independent of the rest of the population. Each individual would try to maximise the magnitude of the correlation coefficient that it represents. This in turn would maximise the population's fitness by improving the correlations represented by the population.

The Mutate Operator

Within the EP a gene is either x_i ,x_j, or the *lag*. We have used the idea of *Self-Adapting Parameters* [2] in this context. Here each gene, $gene_i$, in each chromosome is given a parameter, σ_i. Each gene within a chromosome is mutated according to the Normal distribution with mean 0 and standard deviation equal to the gene's corresponding standard deviation, σ_i, in equation 1. Each σ_i is then mutated according to equation 2.

$$gene_i = gene_i + N(0,\sigma_i) \qquad (1)$$

$$\sigma_i = \sigma_i \cdot \exp(N(0,\tau) + N(0,\tau_i)) \qquad (2)$$

$$\tau = \frac{1}{\sqrt{2len}} \qquad (3) \qquad\qquad \tau_i = \frac{1}{\sqrt{2\sqrt{len}}} \qquad (4)$$

Note that τ is constant for each gene in each chromosome but different between chromosomes, and τ_i is different for all genes. Both parameters are generated each time mutation occurs. Each chromosome consisted of three parameters and their corresponding σ_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 fell into this category were repeatedly mutated until they became valid

The Survival Operator

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

C. The Partition Metric

The Partition metric, which we define below, is used to group variables together where they have strong mutual dependency and to separate them into different groups where the dependency is low. Let n be the number of variables, G be the list of groups and $m = |G|$ (the number of groups). Let g_i be the ith member of the list G where $1 \leq i \leq m$ and let $k_i = |g_i|$. The notation g_{ij} refers to the jth element of the ith list of G. G is restricted such that $\bigcup_{i=1}^{m} g_i = \{x_1..x_n\}$ and $g_u \cap g_v = \phi, \forall u \neq v$ where $k_i \geq 1$. Therefore $\sum_{i=1}^{m} k_i = n$. It is clear that in all cases $m \leq n$. The *partition metric* for any fixed list G, *f(G)*, is defined as follows, where *corr*(x_i, x_j) returns true if there exists in Q any triple of the form (x_i, x_j, lag) or (x_j, x_i, lag) for any valid *lag*.

$$f(G) = \sum_{i=1}^{m} h(g_i) \qquad (5)$$

$$L(g_{ia}, g_{ib}) = \begin{cases} 1 \text{ if corr}(g_{ia}, g_{ib}) \\ 0 \text{ if a = b} \\ \text{-1 otherwise} \end{cases} \tag{6}$$

$$h(g_i) = \begin{cases} \sum_{a=1}^{k_i} \sum_{b=1}^{k_i} L(g_{ia}, g_{ib}) \text{ if } k_i > 1 \\ 0 \qquad\qquad\qquad \text{otherwise} \end{cases} \tag{7}$$

The metric has the following characteristics (proofs for these can be found in the Appendix):

1. If there are no correlations, the maximum value is obtained when all variables are in separate groups.
2. If a correlation exists for each pairing of variables (the search space), then the maximum fitness is obtained when all of the variables are in one group.
3. If the data generating the correlations came from a mixed set of MTS observations, then the metric will be maximised when the variables within the same group have as many correlations within the list *Q* as possible and variables within differing groups contain as few correlations as possible.

In this paper we have chosen a *correl* that is a well established correlation coefficient - Spearman's Rank Correlation [20]. Spearman's Rank Correlation (SRC) measures linear and non-linear relationships between two variables, either discrete or continuous, by assigning a rank to each observation. We can calculate the SRC between two variables over differing time lags by shifting one variable in time. The equation incorporates the sums of the squares of the differences in paired ranks, according to the formula:

$$correl(x_i, x_j, lag) = 1 - \left(\frac{6 \sum_{t=1}^{T-lag} \left(rank(x_i(t)) - rank(x_j(t+lag)) \right)^2}{(T-lag)((T-lag)^2 - 1)} \right) \tag{8}$$

where T is the length of the MTS and $rank(x_i(t))$ is calculated from ordering and ranking every observation of the variable x_i on its value and recording the rank of the value at position t.

We chose Spearman's Rank as it is well recognised and not limited to finding linear dependencies although the methods are not restricted to using this particular coefficient and others such as Pearson's could have easily been used.

D. The Grouping Search

We have looked at various methods for maximising the metric outlined above in the context of grouping MTS. First of all we describe the general Genetic Algorithm approach that we have adopted before we explain three different forms of this algorithm. Next we describe a hill climb technique and finally a heuristic clustering method.

1) The Genetic Algorithms (GA)

The general Genetic Algorithm [12,10] described below uses the notion of a *Population* of *chromosomes* which represent a number of possible solutions to a particular problem. *Crossover* and *Mutation* operators are applied to these chromosomes according to *CrossoverRate* and *MutationRate*, respectively. A selection process is applied to *Population* in order to preserve "good" solutions and discard "poor" ones. The process is iterated over the *Population* chromosomes for a specified number of times, *Generations*. The general algorithm for generating a set of groups, *G*, from a set of correlations, *Q*, is given below:

```
Input: Q , Population, CrossoverRate, MutationRate, Generations
Fitness: The Partition Metric applied to a chromosome given Q
Generate Population chromosomes
Repeat Generation times
    Select CrossoverRate × Population chromosomes (with fitter
    chromosomes being chosen with higher probability and a
    chromosome can be chosen more than once)
    Randomly pair up selected chromosomes creating parent pairs
    Crossover parents to generate Offspring1 and Offspring2
    Mutate offspring based on MutationRate
    Insert offspring into the population
    Sort the population according to Fitness
    Retain the Population fittest chromosomes
End Repeat
Output: G (a set of groups, constructed from the final fittest individual)
```

The following describes three different representations, forms of crossover and mutation that were used with this general algorithm. For the scope of this paper, the fitness function for the methods will be the partition metric defined in equation 5.

<u>Gene Per Variable (GPV)</u>

This representation 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. For example, 10 variables to be placed into three groups:

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

This would be represented by the following chromosome: **0 1 1 0 1 1 2 1 0 2**. The Crossover operator we use for this representation is Holland's [12] standard one point crossover and the Mutation operator involves randomly mutating genes within the chromosome. Each gene has Mutation Rate probability of being mutated to a value from a uniform distribution U(0,*n*-1). For example:

Parent 1: **0 1 1 0 1 1 2 1 0 2**
Parent 2: **0 0 0 1 0 2 2 0 1 2**

1. Crossover (Crossing Point = 3):
   0 1 1 1 0 2 2 0 1 2 **0 0 0 0 1 1 2 1 0 2**
2. Mutate
   0 1 0 1 0 2 1 0 1 2 **0 2 0 0 1 1 2 1 0 2**

Goldberg's Partially Mapped Crossover (PMX)

This form of crossover applies to a new representation of the grouping problem where the chromosome consists of variables interspersed with group dividers. For example, let a group divider be represented by the symbol $\square_i$ where the subscript is unique and each of 10 variables within a domain be represented by a unique integer. Therefore the chromosome: **0 3 8 $\square_1$ 2 7 4 1 5 $\square_2$ 6 9** would represent the groupings in the previous example. In other words, variables within the same group dividers will be classed within the same group. This representation requires a new crossover operator in order to ensure that invalid offspring are not produced. It can be seen that standard crossover as used in the GPV representation would produce many invalid offspring as it would be highly likely to result in offspring with variables appearing in more than one group. Goldberg introduced the PMX operator [11] which prevented this and developed an o-schema theory (closely linked to Holland's original schema theory). It ensures all offspring are valid (i.e. it is a closed operator) and works as follows:

1. Select two crossing points for both parents
2. Swap all elements between the crossing points
3. For all repeating elements in the old part of the chromosome, replace with the value found on the corresponding position on the other chromosome.

Mutation involves randomly swapping two genes within the chromosome according to the Mutation Rate. Each gene has Mutation Rate probability of being swapped with another. For example:

Parent 1: **4 $\square_3$ 0 $\square_1$ 1 6 5 2 $\square_2$ 3**
Parent 2: **5 4 $\square_2$ 2 3 $\square_3$ 0 1 $\square_1$ 6**

1. Crossing points = 3 and 6
2. Swap elements "**$\square_1$ 1 6**" with "**2 3 $\square_3$**"
   4 $\square_3$ 0 2 3 $\square_3$ 5 2 $\square_2$ 3   **5 4 $\square_2$ $\square_1$ 1 6 0 1 $\square_1$ 6**
3. Replace repeated values
   4 6 0 2 3 $\square_3$ 5 $\square_1$ $\square_2$ 1   **5 4 $\square_2$ $\square_1$ 1 6 0 3 2 $\square_3$**
4. Mutate
   4 $\square_2$ 0 2 3 $\square_3$ 5 $\square_1$ 6 1   **3 4 $\square_2$ $\square_1$ 1 6 $\square_3$ 5 2 0**

Falkenauer's Grouping Genetic Algorithm (GGA)

This representation is similar to the GPV except that it also has an extra part on the chromosome which represents each group without any information about their contents. For example the same groupings as the previous examples 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 this 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 (i) to (iv) 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 [6]). 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). Falkenauer proves [6] that this method allows the schema theory to hold even for grouping problems. In contrast, PMX and standard crossover as used in GPV, with their schema and o-schema theories, appear to collapse when applied to these sort of problems.

2) *Hill Climbing (HC)*

A Hill Climbing Search [19] iteratively moves in the direction of increasing value for some metric. Our version of Hill Climb involves using the GPV representation and making simple changes to the current groupings with each iteration. Within each iteration one variable is moved into another existing group or placed into a newly formed group and if this change improves the score of the individual, it is retained. The algorithm is outlined below.

```
Input: $Q$
Generate a random selection of groupings using the GPV representation
Set $Score$ to the Partition Metric applied to $Q$ given the grouping
For $i = 1$ to $Iterations$ do
    Make a random change to one gene in the chromosome
    Set $New\_Score$ to the Partition Metric applied to $Q$
    If $New\_Score < Score$ Then undo changes
End For
Output: $G$ (a set of groups)
```

3) Mirkin's Separate and Conquer (S&C)

This method is based on the clustering technique of Separate and Conquer [16]. The algorithm is amended to allow it to cluster on the relationships between variables rather than on the value of variables. The algorithm is as follows and uses equation 7 to calculate $h(\mathrm{g}_i)$:

```
Input: $Q$
Let $G$ be a set of Groups (empty)
Let $X$ be a set of variables $\{1..n\}$
Create a group $g_1$ containing the best correlation pair in $Q$
Add $g_1$ to $G$ and set $m = 1$
For $i = 1$ to $n$
    Set $skip$ = false and $j = 1$
    While $j < m+1$ and $skip$ = false
        If $x_i \notin g_j$ Then
            Add $x_i$ to $g_j$ to create $g'_j$
            If $h(\mathrm{g}'_\mathrm{j}) > h(\mathrm{g}_\mathrm{j})$ Then
                Add $x_i$ to $g_j$ and set $skip$ = true
            End If
        End If
        $j = j + 1$
    End While
    If $skip$ =false Then
        Create a group $g*$ containing only $x_i$
        Add $g*$ to $G$ and set $m = m + 1$
    End If
End for
Output: $G$ (a set of groups)
```

To summarise, a new group is created containing the two variables that have the highest correlation between them. The next step is to take each variable in turn, and iterate through each group that exists, seeing if adding the variable to that group increases the groups' score. If this is the case, then the variable is added to that group. If there are no more groups to test a given variable with, then it is placed into a new group on its own.

IV. Parameter Estimation

In order to retrieve groupings that correspond closely to the correlations that represent actual dependencies, we will have to determine the ideal set of parameters for the correlation search, most importantly R, the size of the Q. As this will determine the cut off point for significant correlations, it will affect the overall algorithm a great deal. For example, a cut off point that is too high will mean there are too few significant correlations resulting in smaller groups; a cut off point that is too low will mean there are too many significant correlations and so groups will be combined into larger ones due to the inclusion of low correlated variables in the list. We have decided to try and determine the parameters through simulations of the random bag method described in section III.B. Random bag was chosen since it is the simplest to model. It should also be the weakest of the three methods for correlation search and so by coming up with confidence intervals for selecting all the true correlations for this method should mean we have a worst case scenario for the chosen parameters; namely 95% confidence on Random Bag should mean *at least* 95% confidence on EP. This has been shown to be true in our previous work in [21], and through the experiments within this paper. These simulations were used to generate probability distributions of selecting correlations that represent actual dependencies. These distributions could be used to determine confidence limits for the correlation list size and the number of calls to the correlation function.

A. Simulation of Random Bag

Simulations were carried out in order to mimic the way in which the random bag searches for good correlations. These consisted of setting the size R of Q, the size of the total search space (s) and the number of calls to the correlation function (c) to particular plausible instantiations and then simulating the act of randomly selecting a correlation from the search space and then recording whether it was a pre-defined "true" dependency. This process can be compared to repeatedly picking a selection of c random cards from a pack without replacement and recording the number of Aces found. Therefore for this case $R = 4$ (the number of Aces) and $s = 52$ (the number of cards in a pack). We were able, therefore, to generate approximations of the distributions associated with the probability of picking a "true" dependency. The number of these "true" dependencies will be referred to as r, where $r \leq R$. These distributions were then tested for normality using the Lilliefors' test (see section IV.B). The mean and standard deviation were then calculated for each distribution so that a method for symbolic regression could be used to learn a function to determine the mean and standard deviation given R, s and c (see section IV.C). The simulation algorithm is as follows and was repeated for N_{sims} different values of R, s and c.

```
Input: $R$, $r$, $s$, $c$ and $SimulationSize$
Set $dependencies = r$ randomly selected correlations
Set $Distribution$ to be a zero array of length $R$
For $i = 1$ to $SimulationSize$
    $count = 0$
    For $j = 1$ to $c$
        Randomly choose $R$ different correlations
        If $(x_i, x_j)$ is in $dependencies$ Then $count = count + 1$
    End For
    $Distribution_{count} = Distribution_{count} + 1$
```

```
End For
Output: Distribution
```

The probability distribution for selecting a true dependency is found by dividing each element in the distribution array by *SimulationSize*. *SimulationSize* is a variable that dictates the number of times the process is repeated to ensure that a good approximation to the random bag process is reached.

B. Lilliefors' Test

Lilliefors' test [14] is a simple test for normality that can be performed on a known distribution function. The simulations performed in section IV.A can easily be transformed into the required format for this method and the test can be performed to see if the random bag method can be approximated by a normal distribution. Given v observations, a metric D_{max} is computed as in equation 9.

$$D_{\max} = MAX\left|F^*(r) - S_v(r)\right| \qquad (9)$$

where $S_v(r)$ is the sample cumulative distribution function, $F^*(r)$ is the cumulative normal distribution with μ equal to the sample mean, σ^2 equal to the sample variance, and v is equal to R+1. Within the simulations, these two summary statistics can be computed directly from the data. If the value of $D_{\max}$ exceeds the critical value supplied by Lilliefors in his paper, one rejects the hypothesis that the observations closely follow the normal distribution. For the purpose of this paper we shall choose the 99% confidence limit, which requires D_{max} not to exceed $1.031/\sqrt{v}$. From the results of these tests we can assume that the random bag can be approximated by a normal distribution with a 99% certainty. In fact all of the 150 simulations passed the test for normality at this level.

C. Finding μ and σ

Once it has been ascertained that the distribution of the Random Bag process can be approximated as Normal, a value for the mean and standard deviation is needed in order to place confidence limits on the number of function calls needed to find the required R, the size of Q. Since many simulations have been performed, tabulating R, c, s and the associated μ and σ, these can be used to evaluate the relationship between μ and σ. We shall assume that μ is a function of R, c and s, and that σ is another function of R, s and c. The Genetic Programming technique of Symbolic Regression is used for this, [13], rather than applying a set of parameterised functions because there is no knowledge whatsoever of relationships between any of the variables.

The functions for μ and σ, which shall be denoted $\mu(R,c,s)$ and $\sigma(R,c,s)$, will be assumed to be functions in terms of the operators +,-,X,/, and the terminal symbols R, c, s along with the constant integers 0 to 9. The exact form is unknown. A binary tree will be used to represent a regular expression in terms of these symbols, with the terminal nodes being a variable or constant and the non-terminals being an operator. The worth of any given tree (its fitness) will be the difference between the observed value of μ and/or σ vs. the calculated value, using the equation formed from the tree, and all of the available data. This is defined in equations 10 and 11.

$$\text{Fitness for } \mu = -Nodes(\mu)\cdot\sum_{i=1}^{N_{sims}}\left|\mu(R_i,c_i,s_i)-\mu_i\right|^2 \qquad (10)$$

$$\text{Fitness for } \sigma = -Nodes(\sigma)\cdot\sum_{i=1}^{N_{sims}}\left|\sigma(R_i,c_i,s_i)-\sigma_i\right|^2 \qquad (11)$$

where *Nodes(●)* represents the number of nodes in the corresponding binary tree, and i indexes a variable from the table of simulated examples (where there are a total of N_{sims} examples). As with a Genetic Algorithm, the initial population will be a certain number of random binary trees as described above. This population will be improved (better fitness) over subsequent generations through the use of the standard genetic programming operators of Mutation and Crossover. Note that the negative fitness function ensures that the process tries to improve the population by minimising the fitness. Adjusting the fitness by penalising it on the tree's size will force the genetic program to look for a smaller tree. The resulting functions for μ and σ can be found in equations 12 and 13.

$$\mu = \frac{2cR}{2s+c} \qquad (12) \qquad\qquad \sigma = \frac{R}{63}+\frac{11c}{s} \qquad (13)$$

D. Confidence Limits on c

Once values for the mean and standard deviation have been found, one can place confidence limits on the probability of the random bag finding a number of correlations that lie between r and R, where R is the size of the random bag and r is the number of correlations being searched for. This is the cumulative normal distribution where the probability that the number of correlations found is greater than r. For the purpose of this paper, we have chosen the ratio of R to r as 5 and the confidence limit as 95%. The aim of this exercise is to recommend a value for c based on the known parameters R, r and s. Given that the pr(number of correlations ≥ r) = 0.95, we can use the standard normal distribution tables with $z = (r-\mu)/\sigma$ to find what the corresponding value of c should be. For the 95% level, the value of z should be –1.645. Since we know μ and σ, an equation can be formed in terms of z, r, R, c and s where only c is unknown. We start with equations 14 and 15.

$$z = \frac{r-\mu}{\sigma} \qquad (14) \qquad\qquad z = \frac{r - 2cR/(2s+c)}{R/63+11c/s} \qquad (15)$$

Unfortunately this requires a lot of algebra to solve the above equation for c. The final solution is a quadratic equation, and when some reasonable approximations are made, is as found in equation 16.

$$c \approx \frac{s}{22}\left((1.3r - 2R) + \sqrt{(2R - 1.3r)^2 + \frac{88}{63}(Rz - 63r)}\right) \qquad (16)$$

The parameter c is a guide towards how long the procedure is going to take, in terms of how many correlation function evaluations are made. For example if c is greater or equal to the number of calls made by the exhaustive search (s), then it is pointless to use the random bag to locate the required number of correlations. As a guideline, we aim for a 95% confidence at finding the required number of correlations.

V. Experimental Results

We describe the generated datasets in section V.A and the results of estimating the parameters for the algorithms in section V.B. We then describe a metric for evaluating the discovered groupings in section V.C which is followed, in section V.D, by the results of numerous experiments which compare different grouping strategies consisting of all combinations of the proposed methods for grouping search and for correlation search. These 15 strategies are applied to six datasets where there are identifiable mixed groupings of MTS variables. For each experiment we have recorded:

1. The Partition metric of the best solution after a varying number of calls to the fitness function for various different datasets. This is a measure of how well the groupings represent the correlations that were discovered during the correlation search.
2. The score as calculated by the Evaluation metric described in V.C, which is independent of the correlation search results. This can be considered as a measure of accuracy of the resulting groupings. It is essentially a measure of distance between the groups that were used to generate the data and the resultant groups found using our methods.
3. The number of function calls to find the solution with the highest Partition metric (a measure of efficiency).

All stochastic grouping algorithms (all methods except Separate and Conquer) were repeated 10 times and the average recorded in order to remove any sampling bias. We then calculated the marginal statistics over the correlation searches, the grouping strategies and the datasets.

A. *Multivariate Time-Series Datasets*

Based on the two problems being tackled by the grouping methods - the search for DBN structure and the generation of VAR models, two types of datasets have been produced. One set has been generated by hand-coded DBNs and the other by VAR models. We have generated five datasets of each type with varying dimensionality and order. These are described below. For the experiments we mixed various variables from these datasets to produce some which had only DBN generated data, some which had only VAR generated data and some with a mixture of the two. This was to see how the methods performed under different conditions and for different types of data.

1) *Dynamic Bayesian Networks*

Dynamic Bayesian Networks can be used to model MTS. A Dynamic Bayesian Network consists of a set of nodes, representing variables in the domain at different time lags and directed links between these nodes. To each node, with a set of parents, there is an associated probability table and these can be used to infer probabilities about certain events in the system [5]. Five different Dynamic Bayesian Network topologies were created with the conditional distributions hand-coded in order to generate 5 separate MTS. The number of variables within each network was 3, 5, 5, 10 and 10. The size of the largest time lag for each network varied between 5 and 60.

2) *VAR Processes*

Just under half of the test data was generated from a selection of Vector Auto-Regressive MTS models. These types of models have various applications from medical domains [21] to economic domains [4]. A VAR process of order P, written VAR(P), is defined in equation 17.

$$\underline{x}(t) = \sum_{i=1}^{P} \mathrm{A}_i \cdot \underline{x}(t-i) + \underline{\varepsilon}(t) \qquad (17)$$

where $\underline{x}(t)$ is the next data vector of size n (the number of variables in the model) at time t, A_i is a $n{\times}n$ coefficient matrix at time lag i, and $\underline{\varepsilon}(t)$ is a n length noise vector at time t (usually Gaussian) with zero mean. The value of each element in A_i is usually a real number in the range ±1. In order for this process not to rapidly tend towards infinity or zero over time, certain conditions must be placed on the parameter matrices, referred to as stability [15]. This condition can be imposed through the use of a genetic algorithm to generate a random VAR(P) process and then the use of Crossover and Mutation to improve its fitness (which is a measure of its stability) through subsequent generations.

3) *Dataset Organisation*

Table 1 describes the datasets that were generated using the two methods described above:

MTS	Order	Dimensionality
a	10	3
b	20	5
c	5	5
d	30	10
e	60	10
f	2	10
g	3	7
h	4	6
i	5	3
j	2	2

Table 1 - The different MTS descriptions

These 10 MTS were grouped into various different combinations to produce 6 datasets. The first consisted of all 61 variables, the second consisted of only DBN generated data, the third only VAR generated data and the remaining three consisted of various mixtures. All datasets except the first consisted of 28 variables so as to keep the search space identical. Table 2 shows the breakdown of each dataset.

Dataset	MTS
1	a b c d e f g h i j
2	a b d e
3	f g h i j
4	a e f i j
5	a b c g h j
6	d g h i j

Table 2 - The Breakdown of each Dataset

B. Parameter Estimation Results

If we apply the parameter estimation analysis, from section IV, to Datasets 1-6, we obtain the results as listed in table 3. The equation for s represents the total possible number of correlations at varying time lags, once invalid correlations are removed (see section III.B). μ and σ are defined in equations 13 and 14 respectively, z is the standard normal variable, and c is defined by equation 17. Two new parameters are introduced: γ and β. γ is the ratio of c to s and gives an indication of how efficient the procedure is going to be. As a guideline, we would suggest that for the random bag to be effective, this value should be less than 1/3. The parameter β needs defining. This represents the ratio r/R. We suggest a value of 0.2, this being found by experimentation, and provides a good trade off between the number of calls to the correlation function, c, and how many correlations needed to be stored in memory. As can be seen, the use of the approximation in equation 17 has resulted in the confidence limit not being exactly 95%, but rather 94.4% (on average).

Parameter	Dataset 1	Dataset 2-6
MaxLag	75	75
n (number of variables)	61	28
r	150	64
R	750	320
c	72201	15585
$s = n(n-1)(MaxLag + 0.5)$	276330	57078
$\mu = \dfrac{2cR}{2s + c}$	173.321	76.879
$\sigma = \dfrac{R}{63} + \dfrac{11c}{s}$	14.779	8.083
$z = \dfrac{r - \mu}{\sigma}$	-1.578	-1.593
$\gamma = c/s$	0.273	0.261
$\beta = r/R$	0.200	0.200
Confidence	0.943	0.945

Table 3 - Parameters for Datasets 1-6

Figure 1 shows an example where s = 1,000,000, β = 0.2, γ is allowed to vary between 0.22 and 0.32, and three values of R are displayed. The values of R corresponds to 2.5%, 0.25% and 0.025% respectively of s. It can be immediately seen from the graph that R=250 requires more correlation function calls for any level of confidence than for the other two values of R. However there is not much between R=25,000 and R=2,500. Other similar experiments have shown that the optimal value for R/s is near the 0.25% mark.

To conclude, we have shown that c should be calculated from equation 17 once a confidence limit has been assigned (e.g. 95%, giving a value for z: -1.645); here we recommend the value for R/s to be about 0.25%. And finally, we have found that having the ratio of r to R being 0.2 proves to be efficient. However a more systematic study should be conducted on how these parameters relate to each other.

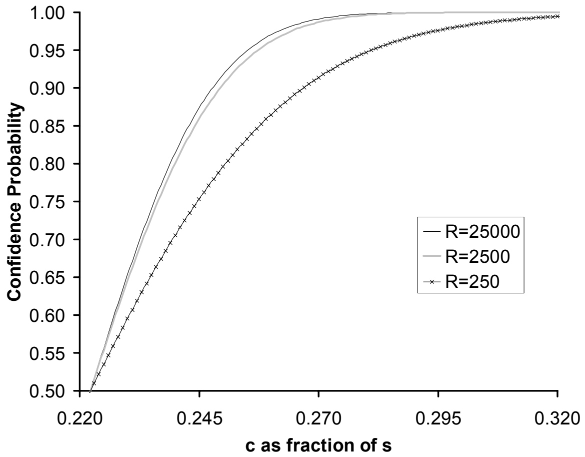

Figure 1 – Confidence against γ with varying R

C. Evaluation Metric

A metric is needed to show how similar or dissimilar two groups are. We define this by pairing all of the variables up and incrementing the score each time that the pair appears in the correct group within the two groups or when the pair appears in different groups. The metric is scaled so that it returns a value between 0 and 1 inclusive, where 0 represents very dissimilar groups and 1 represents very similar groups. This metric is defined as follows:

<u>Definition of the evaluation metric function EVM(G_1,G_2)</u>

```
Let $G_1$ and $G_2$ be two groupings
Let $n$ be the number of variables
Let $EVM = 0$
For $i = 1$ to $n - 1$
    For $j = i+1$ to $n$
        Let $g_1$ be the group within $G_1$ containing $i$
        Let $g_2$ be the group within $G_2$ containing $j$
        If $j$ in $g_1$ and $i$ in $g_2$ Then $EVM = EVM + 1$
        If $j$ not in $g_1$ and $i$ not in $g_2$ Then $EVM = EVM + 1$
    End For
End For
Update $EVM$ to $2EVM/n(n-1)$
```

D. *Results*

In this section we first look at the results from the 15 different combinations of correlation search and grouping strategy to see how they performed when averaged over the 6 datasets. We then look at some of the marginal statistics to see how the correlation searches and the grouping strategies performed irrespective of each other. We also see how the different datasets affected the outcome by looking at their marginal statistics. Finally we discuss the grouping results using three examples.

The parameters for all the grouping genetic algorithms were identical and are found in table 4. The exception to this was GPV which was allowed to run for 1000 generations due to its slow convergence.

Parameter	Dataset 1	Dataset 2-6
Population	150	100
CrossoverRate	0.8	0.8
MutationRate	0.1	0.1
Generations (GPV)	150	100 (1000)

Table 4 - The Parameters for the GGAs

1) *The 15 Methods*

	Partition Metric	Evaluation Metric	Function Calls
RB / GPV	110.60	0.91	232292.5
RB / PMX	114.90	0.92	12974.00
RB / GGA	121.10	0.93	8697.667
RB / HC	125.00	0.92	4881.200
RB / S&C	113.67	0.90	400.6667
EP / GPV	105.43	0.90	232327.7
EP / PMX	109.87	0.90	10545.67
EP / GGA	113.67	0.92	8152.333
EP / HC	118.80	0.91	5331.520
EP / S&C	109.33	0.89	427.6667
EX / GPV	117.80	0.91	232237.4
EX / PMX	122.50	0.92	11325.67
EX / GGA	128.87	0.93	8693.667
EX / HC	130.03	0.93	4778.470
EX / S&C	122.67	0.92	411.6667

Table 5 - The five Grouping Strategies applied to the Three Methods for Generating *Q*

We can see from the results of the 15 different methods in Table 5 that whilst there is a lot of variation in the number of calls to the Partitioning Function (FC), the metrics, in particular the evaluation metric does not vary a great deal at all. This implies that the initial process of searching for *Q* does not have to be exhaustive to get good results. This property would be very useful for those applications where the partitioning of a MTS must occur on a real time basis. By far the fastest to converge is the Separate and Conquer Method taking little more than 400 function calls. However, it must be noted that this method is deterministic and is not guaranteed to find the best groupings.

The most important statistic is the evaluation metric and the method that seems to perform best over all the datasets is the Exhaustive Search / Hill Climb. Although the Falkenauer finds just as good a solution, it takes almost twice as many function calls. However, as the marginal statistics will show, the Falkenauer method performs better when averaged over all the correlation search strategies. Therefore, it appears that if the exhaustive search cannot be carried out then a combination of Random Bag or Evolutionary Program with Falkenauer is the best option.

2) *A Note Regarding RB and EP*

Table 6 displays the average of the top *r* correlations for each method of correlation mining, and displays the average over all of the datasets.

Dataset (*r*)	EX	RB	EP
Dataset 1 (150)	0.592	0.527	0.547
Dataset 2 (64)	0.536	0.488	0.402
Dataset 3 (64)	0.694	0.629	0.659
Dataset 4 (64)	0.641	0.575	0.569
Dataset 5 (64)	0.548	0.509	0.497
Dataset 6 (64)	0.625	0.568	0.558
Dataset's Avg.	0.606	0.549	0.539

Table 6 - The Average Top *r* Correlations for the Three Methods for Generating *Q*

As can be seen in the table, the exhaustive search performs the best, followed by the random bag, and then evolutionary programming. It should be noted that this result only applies to the situation where there are a large number of correlation calls made (*c* is large). It has been shown in [22] that the EP method outperforms the RB method for smaller values of *c*. Based on the extensive analysis and experiments performed so far we can recommend that if *c* is more than 30% of *s* then the exhaustive search method should be used. If this is not the case and if *z* corresponds to less than 50%, use the EP method, otherwise use the RB method.

3) *Marginal Statistics*

In order to explore more fully the effect of the different correlation searches, grouping strategies and datasets, we calculated various marginal statistics. Essentially this involved averaging over the correlation searches, the grouping strategies and the datasets to see how each of these methods compared. These results can be found in tables 7 to 9 below and each table is discussed in the next section.

	Partition Metric	Evaluation Metric	Function Calls
Average EX	124.373	0.923	51489.37

Average EP	111.420	0.905	51356.97
Average RB	117.053	0.915	51849.21

Table 7 - Averaging over Correlation Search

The correlation summary statistics (Table 7) support the conclusion that the method used for generating a good set of correlations does not have a very significant effect on the final groupings. In other words, the evaluation metric which measures the distance between the original groupings and the discovered groupings are very similar for all correlation search methods (approximately 0.9). Therefore, it would make more sense to perform a fast approximate correlation search on datasets where the search space is so large that the exhaustive search is infeasible.

	Partition Metric	Evaluation Metric	Function Calls
Average GPV	111.278	0.907	232285.8
Average PMX	115.756	0.916	11615.11
Average GGA	121.211	0.926	8514.556
Average HC	124.611	0.922	4997.061
Average S&C	115.222	0.903	413.3333

Table 8 - Averaging over Grouping Strategy

The best grouping strategies, as shown by the grouping summary statistics (Table 8), are the Hill Climb method and Falkenauer's GGA. This is probably due to the economical use of function calls made by Hill Climb (unlike the GA methods which require evaluating populations) and the efficient crossover developed by Falkenauer. The other GA methods used less efficient crossovers and the Separate and Conquer method is deterministic and therefore can never be guaranteed to find the global solution. It is, however, very fast at finding a good set of groupings after a very small number of function calls.

	Partition Metric	Evaluation Metric	Function Calls
Avg. Dataset1	222.120	0.958	99294.81
Avg. Dataset2	69.227	0.830	42105.42
Avg. Dataset3	105.667	0.934	41916.15
Avg. Dataset4	101.200	0.896	42154.41
Avg. Dataset5	99.493	0.948	42003.80
Avg. Dataset6	107.987	0.922	41916.49

Table 9 - Averaging over Dataset

Looking at the dataset statistics (Table 9), it appears that Dataset 1 (the mixture of both types of data) produced a higher fitness and independent metric score than the rest and Dataset 3 (the purely VAR data) produced better results than Dataset 2 (the purely DBN data). A reason for this could be that the VAR data generator produced variables with higher correlations between true dependencies. These may then have outflanked any spurious correlations. It is encouraging to note that the largest dataset, Dataset 1, with a mixture of DBN and VAR data produced such good results. Datasets 4 to 6 which contain a mixture of VAR and DBN exhibit the most variations in the evaluation metric. This is most likely down to the strength of correlations that were reflected in the generated data as well as the existence of spurious correlations.

Table 10 shows a selection of groupings learnt from the 3 datasets using the Falkenauer algorithm with differing correlation searches. It can be seen that the majority of variables have been grouped correctly in the all three experiments. In fact, 15 out of the 21 groups have been perfectly recreated. Some of the variables have been placed in a group on their own implying that they are independent when in actual fact there should be some correlation between them and other variables. This could be due to spurious correlations which have prevented the true correlations from being included on the correlation list.

This effect is also evident in the summary tables where the independent metric (which simply measures the distance between the discovered groupings and the original) is higher for some experiments than others but the fitness (which relies on the correlations between variables) is lower. The opposite is also evident in the results. Once again, this is most likely due to spurious correlations between variables in different groups.

4) *Sample of Groupings*

Grouping Method	Original MTS Groupings	Discovered Groupings
EP / GGA	0 1 2	0 6
Dataset 1		1
		2
	3 4 5 6 7	3 4 5 7
		8
	8 9 10 11 12	9 10
		11 12
	13 14 15 16 17 18 19 20	13
	21 22	14 15 20 21 22
		16 17 18 19
	23 24 25 26 27 28 29 30	23 24 25 26 27 28 29 30
	31 32	31 32
	33 34 35 36 37 38 39 40	33 34 35 36 37 38 39 40
	41 42	41 42
	43 44 45 46 47 48 49	43 44 45 46 47 48 49
	50 51 52 53 54 55	50 51 52 53 54 55
	56 57 58	56 57 58
	59 60	59 60
EX / GGA	0 1 2	0 1 2
Dataset 5	3 4 5 6 7	3 4 5 6 7
		8
	8 9 10 11 12	9 10
		11 12
	13 14 15 16 17 18 19	13 14 15 16 17 18 19
	20 21 22 23 24 25	20 21 22 23 24 25
	26 27	26 27
RB / GGA	0 1 2 3 4 5 6 7 8 9	0 1 2 3 5 7 8 9
Dataset 3		4
		6
	10 11 12 13 14 15 16	10 11 12 13 14 15 16
		17
	17 18 19 20 21 22	18 19 20 21 22
	23 24 25	23 24 25
	26 27	26 27

Table 10 - A sample of grouping results from the Falkenauer method along with the original groupings that were used to generate the MTS

An interesting result that was found in the DBN data groups was that if a group of variables was incorrectly split into 2 or more groups, then the divide(s) made topological sense when compared to the structure of the DBNs that generated the data. In some networks the algorithm split them into relatively independent structures.

VI. CONCLUDING REMARKS

In this paper we have outlined a framework with which we can decompose high dimension MTS into smaller dimension MTS which are relatively independent of one another based on the correlation between the variables. This can be very useful in problems where the high dimensionality of a MTS prevents certain algorithms from being applied, for example the generation of Vector AutoRegressive (VAR) models or Dynamic Bayesian Networks (DBNs). Our results have shown that whilst the initial search for good correlations to generate the groupings does not have to be exhaustive to produce equally good results, the best method of grouping search appears to be either a Hill Climb strategy or Falkenauer's Grouping Genetic Algorithm. The results have been very promising on both VAR data and DBN data and, in most cases, the metric used to find the groupings proved robust enough to avoid mistaken groups due to spurious correlations. We have also provided some concrete practical recommendations on the correlation search step of our methodology.

Future work will involve looking at how the grouping methodology performs on some real world datasets with the aim of building DBN and VAR models. We will also be looking at ways of tackling the spurious correlations problem using a combination of standard statistical techniques and heuristics. Relationships between the parameters as described in section IV also need a more rigorous and in-depth study.

ACKNOWLEDGEMENTS

The authors wish to thank the project sponsors: Moorfields Eye Hospital, UK; Honeywell Technology Centre, USA; Honeywell Hi-Spec Solutions, UK and the Engineering and Physical Sciences Research Council, UK. We would also like to thank Jason Crampton for his mathematical assistance.

BIBLIOGRAPHY

[1] T. Baeck, G. Rudolph and H.-P. Schwefel, “Evolutionary Programming and Evolution Strategies: Similarities and Differences”, D.B. Fogel and W. Atmar, editor: Proceedings of the Second Annual Conference on Evolutionary Programming, 11-22, 1993.

[2] T. Baeck, “Evolutionary Algorithms: Theory and Practice”, Oxford University Press, 1996.

[3] M. Casdagli, and S. Eubank, “Non-linear Modelling and Forecasting”, Addison Wesley, 1992.

[4] C. Chatfield, “The Analysis of Time Series - An Introduction”, Chapman and Hall, 4th edition, 1989.

[5] P. Dagum, A. Galper, E. Horvitz and A. Seiver, “Uncertain Reasoning and Forecasting”, International Journal of Forecasting 11, pp 73-87, 1995.

[6] E. Falkenauer, “Genetic Algorithms and Grouping Problems”, Wiley, 1998.

[7] D.B. Fogel, “Evolutionary Computation - Toward a New Philosophy of Machine Intelligence”, IEEE Press, 1995.

[8] N. Friedman, K. Murphy and S. Russell, “Learning the Structure of Dynamic Probabilistic Networks”, Proceedings of the 14th Conference on Uncertainty in AI, pp 139-147, 1998.

[9] M. Garey and D. Johnson, “Computers and Intractability - A Guide to the Theory of NP-Completeness”, W.H. Freeman, San Francisco, 1979.

[10] D. E. Goldberg, “Genetic Algorithms in Search, Optimisation, and Machine Learning”, Addison Wesley, 1989.

[11] D.E. Goldberg and R. Lingle, “Alleles, Loci, and The Travelling Salesman Problem”, Proceedings of an International Conference on Genetic Algorithms and Their Applications, pp 154-159, 1985.

[12] J.H. Holland, “Adaptation in Natural and Artificial Systems”, University of Michigan Press, 1995.

[13] J. Koza, “Genetic Programming: On the Programming of Computers by Natural Selection”, MIT Press, 1992.

[14] H. W. Lilliefors, “On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown”, Journal of the American Statistical Association, 62, 399-402, 1967.

[15] H. Lutkepohl, “Introduction to Multivariate Time Series Analysis”, Springer-Verlag, 1993.

[16] B. Mirkin, “Concept Learning and Feature Selection Based on Square-Error Clustering”, Machine Learning 35, pp 25-39, 1999.

[17] D. Pena and G. Box, “Identifying a Simplifying Structure in Time Series”, Journal of American Statistical Association 82, pp 836-843, 1987.

[18] A. Pole, M. West, and P.J. Harrison, “Applied Bayesian Forecasting and Time Series Analysis”, Chapman-Hall, 1994.

[19] S. Russell and P. Norvig, “Artificial Intelligence, A Modern Approach”, Prentice Hall, 1995, Ch.4, pp 111-112.

[20] G. Snedecor and W. Cochran, “Statistical Methods”, Iowa State University Press, 6th edition, 1967.

[21] S. Swift and X. Liu, “Modelling and Forecasting of Glaucomatous Visual Fields using Genetic Algorithms”, Proceedings of the Genetic and Evolutionary Computation Conference, pp 1731-1737, Morgan Kaufmann, 1999.

[22] S. Swift, A. Tucker and X. Liu, “Evolutionary Computation to Search for Strongly Correlated Variables in High-Dimensional Time-Series”, Proceedings of Intelligent Data Analysis 99, LNCS 1642, Springer-Verlag, pp 51-62, 1999.

[23] A. Tucker and X. Liu, “Extending Evolutionary Programming to the Learning of Dynamic Bayesian Networks”, Proceedings of the Genetic and Evolutionary Computation Conference, pp 923-929, Morgan Kaufmann, (1999).

Appendix - Proofs for Grouping metric

Proof 1. When there are no correlations, then $Q = \phi$. Therefore *max(f(G))* is 0, because there will never be any cases where *L* is 1. This therefore requires that the size of any of the groups in *G* will be 1. This is by definition of the functions *L* and *h*.

Proof 2. If a correlation exists for each pairing of variables, then the maximum size for *Q* will be $\frac{n(n-1)}{2}$, because of the duplicate restriction. It therefore follows that the value for $h(g_i)$ will be $\frac{k_i(k_i-1)}{2}$ using the same logic. Using equation 5, we have

$$\max(f(G)) = \max(\sum_{i=1}^{m} h(g_i))$$

therefore

$$\max(f(G)) = \max\left(\sum_{i=1}^{m} \frac{k_i(k_i-1)}{2}\right)$$

since

$$\max\left(\sum_{i=1}^{m} \frac{k_i(k_i-1)}{2}\right) = \frac{1}{2}\max\left(\left(\sum_{i=1}^{m} k_i^2\right) - n\right)$$

then *f(G)* will be a maximum when $\sum_{i=1}^{m} k_i^2$ is a maximum.

We shall assume that $1 \le k_1 \le k_2 \le \ldots k_m$. If we write $k_1' = k_1 + k_2$ then

$$(k_1 + k_2)^2 = k_1^2 + k_2^2 + 2k_1k_2$$
$$\therefore (k_1')^2 > k_1^2 + k_2^2$$

This process can be repeated until there is only one value k_1 remaining where $k_1 = n$, and *f* attains its maximum value. Hence when *Q* is at a maximum size (as above), the arrangement with the maximum fitness will be all variables in a single group.

Proof 3. If the data generating the correlations came from a mixed set of multivariate time series observations, then for a given grouping arrangement *G* and correlation set *Q*

$$\max(f(G)) = \sum_{i=1}^{m} \max(h(g_i))$$

$$\max(h(g_i)) = \max\left(\sum_{a=1}^{k_i}\sum_{b=1}^{k_i} L(g_{ia}, g_{ib})\right)$$

This will be a maximum when all instances of the function *L* are 1. If *Q* contains an additional spurious correlation or is missing a correlation, then this value will be reduced by 1, by definition of *L* and proof 2. Hence the maximum value of the fitness for a given *G* will be when *Q* contains the all of the correlations that can exist for each grouping.

Glossary

X	A Multivariate Time Series
n	Number of variables in the MTS
T	Number of cases / observations
$x_i(t)$	An observation of the MTS variable, *i* at time, *t*
lag	Time lag of a correlation
MaxLag	Maximum limit for a time lag
P	The order of a VAR process
Q	List of discovered high correlations
R	Length of *Q*
G	Set of groups
m	Number of groups
g_i	The *i*th group
k_i	Size of the *i*th group
s	Size of the search space of all possible correlations with all lags up to *Maxlag*
r	Number of true underlying dependencies (i.e. excluding spurious correlations)
c	Number of calls to the correlation coefficient
correl	Correlation in the form of a triple, (x_i, x_j, lag)
corr	Boolean Function that returns true if a correlation pair exists in Q irrespective of direction
z	z statistic for the Normal Distribution
β	The ratio *c/s*
γ	The ratio *r/R*