The Modelling of Short High-Dimensional Multivariate Time Series

Abstract

In bio-medical domains there are many applications involving the modelling of multivariate time series (MTS) data. One area that has been largely overlooked so far is the particular type of time series where the dataset consists of a large number of variables but with a small number of observations. This thesis presents a methodology for the modelling of this type of data and introduces a novel computational method, which combines evolutionary algorithms and traditional statistical methods.

Two key issues in modelling short MTS are addressed in this thesis. Firstly, the curse of many time series modelling methods, particularly those from the traditional statistical approaches, is the number of parameters that must be located in order to apply the models. The method proposed in this thesis bypasses this parameterisation problem by locating the key relationships between the multivariate time series variables using a cross-correlation search and decomposing these variables into mutually exclusive but highly interrelated subsets using evolutionary algorithms. Secondly, the short length of these time series pose significant challenges since traditional statistical methods often place constraints on the minimum number of observations in the dataset. Towards this end, an effective way of modelling this type of data has been developed based on evolutionary algorithms and the Vector Autoregressive Process, which avoids these constraints.

The work in this thesis has been extensively evaluated against both simulated and real world biomedical time series. Evaluation is performed from both a theoretical and empirical angle and the results obtained suggest the proposed methodology is highly effective. This thesis makes the following key contributions: the introduction of a rapid correlation mining method, the effective decomposition of high dimensional MTS into subgroups, and the novel modelling of short MTS datasets.

Acknowledgements

The author would also like to thank the following people (listed in no particular order): *Janet McFall* for her understanding, patience and advice which has all been readily available throughout the whole project. *Jason Crampton* for his mathematical assistance. *Steve Counsell* for his advice throughout the write-up of this thesis. *Dr Allan Tucker* for being there to “bounce” ideas off and his collaboration on some research papers, particularly the work reported in chapter 4. *Professor Xiaohui Liu* for providing an excellent level of supervision throughout the whole PhD. *Mr Richard Wormald* for providing help on all of the medical aspects of this thesis and being the project sponsor tutor. *Professor George Loizou* for his advice on structuring the thesis. *Katherine Henshaw* for providing employment in the Cochrane Eyes and Vision group at Moorfields eye hospital between the PhD funding running out and a Research position becoming available. *Shona Burman-Roy* for putting up with the author as the manager during this latter period. *James Cain* (*Blunder*) and *Pete Newson* for their very valuable C++ and STL programming advice. *Catey Bunce* and *Dr Suzanne Evans* for their very helpful, well explained and useful statistical help. And finally, *Dr Phil Docking*, *Philip Gregg*, *Graham Sadler* and *Andrew Watkins* from Birkbeck College’s systems support group for the superb level of computer assistance and support they provided. I would also like to thank the rest of my friends and family for putting up with me during the whole PhD studentship.

The author also wishes to thank the project sponsors: Moorfields Eye Hospital, UK and the Engineering and Physical Sciences Research Council, UK.

“All models are wrong, but some are useful”

– George Box

Table of Contents

Table of Algorithms

Table of Equations

Table of Figures

Table of Tables

Notation

The following notations are used throughout this thesis.

Mathematical and Statistical

$\underline{x}$, $\underline{y}$	A vector
$\hat{A}, \hat{x}$	Estimator for a matrix or variable
$\underline{X}, \underline{\underline{X}}$	Transformation of a matrix or a change of matrix notation
$\underline{0}$	A matrix of zeros
A, B	A matrix
A^{T}	Transpose of a matrix A
A^{-1}	Inverse of a matrix A
a_{ij}	The element in the ith row and jth column of a matrix A
C_i	A Gershgorin circle
D	The union of some Gershgorin circles
$\mathrm{E}(x)$	The expected value of a variable x
I, I_a	The identity matrix (of size a if specified)
i, j	Typical index variables
MAXLAG POPULATION etc…	A parameter constant
$\mathrm{N}(\mu,\sigma)$	Normal (Gaussian) distribution with mean μ and standard deviation σ (the density function)
N, M	Length or size of an object
$\Pr(x>a)$	The probability that a random variable x is greater than a
R_s	Spearman's Rank Correlation Coefficient
$x \sim \mathrm{N}(\mu,\sigma)$	Random variable x is distributed by the specified Normal distribution
z	The value of a variable distributed under the standard Normal distribution, i.e. $\mathrm{N}(0,1)$, $\Pr(x>z) = \varphi(z)$

x_i, y_i	The *i*th element of a vector
λ, λ_i	An eigenvalue
μ, μ_x	Mean
ρ, ρ_{xy}	Pearson’s Correlation Coefficient
σ, σ_x	Standard deviation
Σ, Σ_ε	Covariance matrix
$\varphi(z)$	The cumulative standard Normal distribution, the density function is N(0,1)

Functional

$[A\ B]$	Concatenation of matrices and/or vectors where the number of rows of each object being operated on is equal. The operator produces a single matrix formed from each column of each object as they appear, traversing from left to right
$[a, b]$	A set of numbers between *a* and *b* inclusive
(a, b)	A set of numbers between *a* and *b* exclusive, equal to $[a+1, b-1]$ for integers
$(a, b]$	A set of numbers between *a* and *b* exclusive of *a*, equal to $[a+1, b]$ for integers
$[a, b)$	A set of integers between *a* and *b* exclusive of *b*, equal to $[a, b-1]$ for integers
$a_1,...,a_N$	An alternative notation to the interval and set notation used above, using variables as opposed to numeric constants
$\mathrm{CP}(z,y)$ $\mathrm{F}^*(a)$ $\mathrm{S}_v(a)$ etc…	Functions with parameters
$\mathrm{UI}(a,b)$	A random uniformly distributed uniform integer between *a* and *b* inclusive
$\mathrm{UR}(a,b)$	A random uniformly distributed real number between *a* and *b* inclusive

Δ	Determinant of a given matrix
$\lvert A \rvert$	Alternative notation for the determinant of a matrix
$\lvert \underline{x} \rvert$	Vector magnitude = $\sqrt{\sum_{i=1}^{N} x_i^2}$
$\lvert a \rvert$	Absolute value of a scalar
$\lvert g_i \rvert$	The number of elements in a set or group g_i

General Time Series

t	A time point of a time series
T	The length of a time series

Univariate Time Series

L_t	The local level of a Holt-Winters univariate forecast at time t
T_t	The local trend of a Holt-Winters univariate forecast at time t
X	A univariate time series
X_t	An observation of a univariate time series variable at time t
α	The Holt-Winters smoothing parameter for the local level
$\hat{\gamma}$	The Holt-Winters smoothing parameter for the local trend

Multivariate Time Series and Space-Time Series

$\underline{x}(t), \underline{y}(t)$	A vector observation of a multivariate time series at time t
$\underline{x}(t+h)$	A vector forecast of a time series h steps ahead of time t
$\underline{\varepsilon}(t)$	Vector noise observation in a time series at time t
$\hat{\Sigma}_{\hat{p}}$	Estimated covariance matrix for a fitted model of order $\hat{p}$
A_i	The ith parameter matrix of a VAR process
M_i	The ith parameter matrix of a VMA process
n	The number of variables in a multivariate time series

nVAR(p)	An n variable VAR process of order p
nVARMA(p,q)	An n variable VARMA process of order p and q
nVMA(q)	An n variable VMA process of order q
p	The order of a VAR or STAR process
q	The order of a VMA process
$x_i(t)$	The ith variable of a multivariate time series at time t
$\Gamma(h)$	Auto-covariance function at time lag h
Ω	Margin of error for recording no change for a visual field variable forecast
Θ_i	Autoregressive spatial lag at time lag i
A_{ij}	Autoregressive parameters
$W^{(j)}$	Weighting Matrix for lag j

Grouping and Correlation

$\square_i$	The ith group partition in the PMX crossover operator
G, G_i	A grouping, consisting of a non-overlapping set of sets
g_i	A group, a set of variables
g_{ij}	The jth member of group g_i
k_i	The size of a group g_i
m	The number of groups
Q	The list of correlations that have been mined, of length R
R	The number of correlations to be mined (the size of Q)
r	The true number of correlations to be mined (the best r from Q)
c	The number of correlations calls an algorithm is allowed to make
s	The number of possible correlations in a multivariate time series
β	The ratio r/R
γ	The ratio c/s

1. Introduction

In bio-medical domains there are many applications involving the modelling of *Multivariate Time Series* (MTS) data [Hannan1970]. One area, which has been largely overlooked, is the particular type of time series where the dataset consists of a large number of variables but with a small number of observations. When modelling this type of dataset using traditional statistical methods there are two key issues that need to be addressed. The first is the number of parameters which need to be located; this increases in relation to the dimensionality of the dataset. Secondly the small length of such time series can prove to be problematic. Being both high in dimensionality and short in length complicates these problems further. Many of these traditional statistical methods simply impose a restriction on the minimum length, thus effectively preventing their application to the problem.

This work applies *Evolutionary Computation* [Bäck1997] techniques to the modelling of short MTS. Such a time series is decomposed into groups of lower dimensionality series, and then modelled for short term forecasting. This work shows where the classical techniques fail in this task, and demonstrates that *Evolutionary* techniques can be applied successfully.

1.1 Motivation

This research project was motivated by a common problem within bio-medical domains. This problem is that an extraordinary amount of data is collected on bio-medical experiments, medical conditions and patients, and is very rarely analysed to a significantly satisfactory level [Wyatt1995]. The domain that this thesis concentrates on is Ophthalmology, although techniques developed are applicable to a range of applications including the analysis of short MTS such as gene expression data in functional genomics [Lockhart2000].

The application domain involves an eye condition called *Normal Tension Glaucoma*. This condition, if left untreated, can lead to a severe reduction in a person's capacity to see and, if left untreated long enough, perhaps to blindness. The condition has no known cure, however drugs can be applied to slow down the progression. Unfortunately these drugs can have some severe side effects. Throughout the treatment process of those who are diagnosed with normal tension glaucoma a set of visual tests are performed at approximately six months intervals to record progression of the condition at key points on the eye. A clinician then examines the whole set of tests and decides on whether the rate of deterioration has changed. If rate of deterioration has increased then it might warrant an increase in the dosage of medication prescribed or perhaps even surgery. If the rate of deterioration has decreased, the clinician might prescribe a lower dosage of the medication because of the side effects. Automatically forecasting this deterioration for a given patient will help the clinician in this task, and hence improve the quality of treatment the patient receives. This dataset can be modelled as a *Short Multivariate Time Series* dataset. With the visual field dataset each of the points measured on the eye can be seen as a variable and each test as a set of observations at a time point.

There are many existing techniques that are well proven with MTS data, but when the time series becomes short such methods start to encounter problems, e.g. bias or complete failure. Evolutionary computation techniques can be used to avoid these problems. This research project shows that a high dimensionality short MTS can be modelled better when it is decomposed into a set of lower dimensional series. The project then shows how improvements can be made in the short term forecasting of short multivariate time series by computing the parameters of such models using evolutionary computation techniques.

The glaucoma data is described in detail in chapter 2, expanding on how visual field data is currently forecast and modelled. Essentially the data is modelled using linear regression [Mosteller1977] and not as an MTS. This, combined with a graphical user interface indicating the "goodness of fit" of the regression line has been made into a tool called Progressor (see chapter 2).

The time series model that is deemed suitable for forecasting the visual field data is the *Vector Autoregressive* (VAR) process [Lütkepohl1993]. This process has been used in a variety of applications, e.g. [Webb1995] and there are various different types of this model, for example the *Bayesian Vector Autoregressive* process [Dua1995]. Although the basic VAR process has been superseded by more advanced models [Holden1995], the work in this thesis initially uses this basic model to demonstrate the problems inherent in short multivariate time series and to show the validity of the ideas presented, and then details the implementation for one of these advanced models. The modelling of short high dimensional multivariate time series has not been addressed with the VAR process as well as many other statistical methods.

Finally it is worth noting that many forecasting problems are now performed by neural networks [Ungar1995, Zirilli1997] however unlike the VAR process these models do not provide any readily comprehensible information about the relationships between variables over time [Faraway1998].

1.2 Methodology

Figure 1.1 outlines the process diagram for the methodology for this research project, which consists of four steps. Three of the steps involve the development of the required MTS models, and the final step is an appraisal of the resulting models.

The first step is the mining of correlations from the dataset. This stage concerns the temporal mining of relationships in a time series.

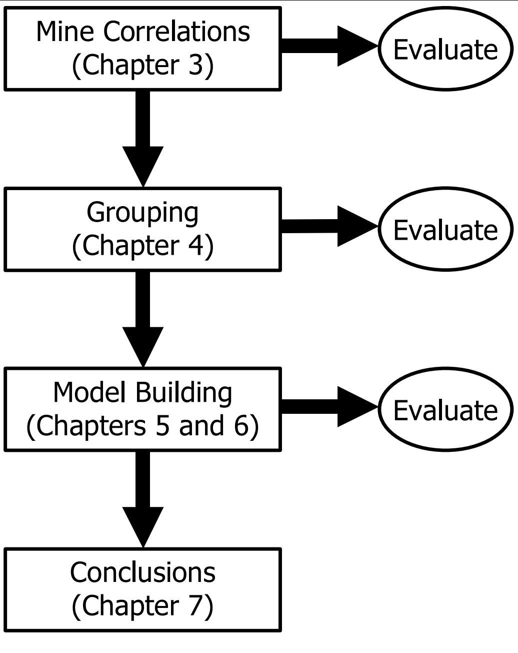


Figure 1.1: Process Diagram for Methodology

The goal is to develop an algorithm that finds a *good-but-not-optimal* selection of *interesting* highly related time series variables in as short amount of time as possible. Three methods for finding the approximate correlation structure are presented and compared to the exhaustive search method for verification. All the methods utilise two procedures for calculating correlations.

Very little work has been done on the temporal correlation mining problem, but some work has been done on grouping variables [Falkenauer1998, Venugopal1992] which is the second stage of the methodology. Once the correlations have been located, the next step is to use this information to construct a set of groups. Here the optimal arrangement is where there are strong relationships between variables in the same group, and weak relationships between variables in different groups. The decomposition of high dimensional 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 part of the methodology contains a systematic study and application of the *variable grouping* problem in MTS. In particular, different methods of utilising the information regarding correlations among MTS variables are investigated.

Having decomposed a high dimensional MTS into a number of lower dimensional MTS, the third step of the methodology presents an evolutionary computation based method specially tailored to modelling short MTS using the VAR process. Much research has been done on the modelling of MTS data in both the statistical and artificial intelligence communities. However, one area that has been largely overlooked is MTS in which the data set consists of a large number of variables but with a small number of observations. There are inherent difficulties in using traditional statistical techniques to model this type of MTS. By combining the decomposition of an MTS into a collection of smaller dimensionality series and the specially adapted VAR process, it is demonstrated that an effective and efficient methodology for the modelling of short and high dimensional MTS is created.

1.3 Contributions

This thesis contributes to five key areas in data analysis, data modelling and visual field analysis.

1. Correlation Mining. The temporal correlation mining method that proves to be the best in performance is based on an evolutionary programming technique, and produces a pre-specified number of the highest correlated variables in a MTS. This list is mined without having to explore the whole correlation space (which would be highly time consuming), and in test and real world datasets has managed to provide results that are ~95% accurate while only exploring ~27% of the possible correlations. This has a wide variety of applications, particularly if such a correlation list is needed quickly or within a real time application. Very little work seems to have been done on this important data pre-processing technique.

2. Grouping. The decomposition of a set of MTS variables into exclusive related groups with high inter-dependencies and low dependencies with variables in other groups is another useful data pre-processing procedure, which naturally leads on from the correlation mining results. The proposed method is a *Genetic Algorithm* based technique, which includes the definition of a fitness function to rate candidate solutions based on the results of the correlation mining stage of the methodology. Furthermore, this fitness function is demonstrated to have the required mathematical properties desirable in such a function. The proposed method is highly accurate and performs better than the other standard techniques it is compared with. The subsets of variables can be used for a variety of applications, such as model building. Because the groups are smaller than when they are treated as a single group, any models will be easier to apply and locate because of the fewer number of parameters needed to make up the models. Most work in this area involves data clustering methods; however these methods do not make use of time dependencies between variables.

3. VARGA. This method incorporates the use of genetic algorithms to find the order and parameters of a VAR process for modelling short MTS. The method avoids the length restriction imposed by some statistical methods, locates the parameters for the model and creates subset models in one step to a higher degree of accuracy than the standard techniques. An initial version of the VARGA paradigm is developed and demonstrated to be superior when tested against a univariate time series forecasting method and a standard statistical package. Two further improvements of the VARGA method are made, and each variant is extensively tested on a number of criteria. The final version of VARGA is shown to perform exceptionally well against standard statistical methods, a heuristic search method, and the other VARGA variants.

4. The Process. Points 1 to 3 above, when combined, have been shown to be an effective procedure for modelling short, high-dimensional MTS datasets that would otherwise prove very difficult or impossible to model using more conventional methods. This procedure has a wide range of applications to many other important datasets of this nature.

5. Visual Fields. The process described in point four above has been successfully applied to the modelling of normal tension glaucoma patient's visual field tests. The results demonstrate that the visual field dataset can be successfully and efficiently modelled as an MTS with a high degree of accuracy. This is an important finding since many of the current methods for predicting visual field loss are based on univariate linear regression. Additionally the correlation mining results graphically depicts the spread of the condition, which corresponds to the medical literature; and the grouping results indicate that size of the subset MTS corresponds to how advanced the condition is.

1.4 Summary of Thesis

This thesis is organised as follows:

Chapter 2 provides a literature review of the concepts, techniques and methods that are used within this research project. Firstly, time series analysis, both univariate and multivariate is described, particularly the vector autoregressive process. Next, a description of the MTS datasets is outlined. Further details of the glaucoma condition are given, and then of the visual field data, along with associated current methods for predicting deterioration. Finally, the elements of evolutionary computation which have been utilised for this research are described.

Chapter 3 examines correlation mining as a method for determining relationships within MTS datasets. Two correlation metrics are used and their time series extensions are introduced. The number of correlations that are needed for the grouping problem is considered and an approximate method is introduced based upon evolutionary programming. The results of this method are then compared with several conventional methods, and it is demonstrated that the evolutionary approach is better under certain conditions.

Chapter 4 looks at decomposing an MTS into a number of smaller series. The goal is that such groups should have strong relationships between members of the same group but weak relationships between members of other groups. A number of methods are presented, including methods based upon genetic algorithms, a well established clustering method and a hill-climbing based method. An evaluation of these methods is then performed based on their accuracy.

Chapter 5 introduces VARGA, a modified genetic algorithm which is specifically designed for identifying the parameters of a vector autoregressive process. This method is described in detail, along with several variants that offer improvements under certain circumstances. Each variant is tested on a real world dataset, and compared with a well-established conventional technique.

Chapter 6 continues the work introduced in chapter 5, and produces a method which is tailored towards finding a vector autoregressive process where the parameter matrices are sparse [Zlatev1991], i.e. where many of the elements are zero. These models are also known as VAR subset models [Lütkepohl1993]. Advantage is taken of this new representation to develop fast operators and a more efficient fitness function evaluation. Furthermore, the method is tested against a number of traditional statistical and artificial intelligence methods using additional evaluation criteria. The results are then combined with the work of chapters 3 and 4, demonstrating how the entire methodology can be applied to a real world dataset.

Chapter 7 summarises the whole thesis. This chapter examines what has been developed within this research project and how this can be extended as further work.

2. Background

This chapter consists of three parts. The first is an introduction to the statistical modelling of time series datasets, followed by the description of the datasets that these techniques are applied to. The final section described some artificial intelligence techniques that have been applied to improve the performance of the time series models.

2.1 Time Series Analysis

Time series data are widely available in different fields including medicine, finance, science and engineering. Modelling time series data effectively is important for many decision-making activities. Time series models can be used to forecast future values and to help understand the underlying relationships within the time series.

2.1.1 General Time Series

A time series is a series of observations, $x_i(t)$; $[i=1,...,n;\ t=1,...,T]$, made sequentially through time. Here i indexes the different measurements made at each time point t; n is the number of variables being observed and T is the number of observations made. If n is equal to one then the time series is referred to as *univariate* [Chatfield1989], and if it is greater than one the time series is referred to as *multivariate* [Hannan1970]. Each observation, $x_j(t)$, has the same meaning as time increases, for example if $x_1(1)$ (variable one at time point one) is a measurement of weight then $x_1(2)$, $x_1(3)$,…, $x_1(T)$ are also measurements of weight. The vector notation $\underline{x}(t)$ is a shorthand way of referring to the whole set of observations made at time t, i.e. $\underline{x}(t)$ stands for the observations $x_i(t)$ where $1 \le i \le n$. Each time point t has a distinct order, and dictates the time at which measurements were recorded, i.e. if $\underline{x}(t_1)$ and $\underline{x}(t_2)$ are two observations, and t_1 is less than t_2, then $\underline{x}(t_1)$ was observed before $\underline{x}(t_2)$.

Time series forecasting [Weigend1994] is the process of trying to estimate future values of $\underline{x}(t)$ based upon previous observations; for example, deriving $\underline{x}(T+h)$ if $\underline{x}(1),\ldots,\underline{x}(T)$ are known, where values of h are positive whole number representing the number of *time points* in the future that the forecast is being made.

Much work has been done towards determining the best method of forecasting [Chatfield1988a, Mahmoud1984], and work has been done in the use of artificial intelligence methods for forecasting [Faraway1998, Numata1998].

2.1.2 Univariate and Holt-Winters

The majority of the work on time series analysis has been highly concentrated on univariate models, which are the simplest types to work with. Within this area, most work has been concentrated on the family of linear models, which includes the autoregressive (AR), moving average (MA) and autoregressive moving average models (ARMA) [Box1970].

Since much of this research is concerned with the short term forecasting of glaucomatous visual field deterioration, and knowing that a univariate model can provide a reliable and accurate short term forecast [Fildes1998], a univariate forecasting method will be used to evaluate the methods presented in this thesis.

The *Holt-Winters* (HW) forecasting method [Chatfield1988b] is a simple way of predicting the next value in a univariate time series. The non-seasonal HW method is defined as follows:

$$L_t = \alpha X_t + (1-\alpha)(L_{t-1} + T_{t-1}) \tag{2.1}$$

$$T_t = \hat{\gamma}(L_t - L_{t-1}) + (1-\hat{\gamma})T_{t-1} \tag{2.2}$$

$$\hat{X}_{t+1} = L_t + T_t \tag{2.3}$$

Equation 2.1 defines the series mean level L_t at time t; equation 2.2 defines the series trend T_t at time t, and equation 2.3 defines the forecast of the dependant variable at time t, given values of X from $1,..,t$-1. All the variables are scalars. The method needs the smoothing parameters α and $\hat{\gamma}$, as well as the mean level and trend starting values L_0 and T_0 to be defined, so that the subsequent values of L_t, T_t and $\hat{X}_t$ can be calculated. According to [Chatfield1988b] α and $\hat{\gamma}$ lie between zero and one. However L_0 and T_0 are usually determined as a function of the observed values of X, and are often limited by the maximum and minimum values. To allow for some variation from this rule, they are assumed to lie within ±100 for the visual field application, which exceeds the limits of the corresponding time series. There are various ways of finding the values of these parameters as suggested in [Chatfield1988b]. In [Swift1999a] it has been suggested that a genetic algorithm (see section 2.3.1) can be used to find a near optimum set of parameters for a one-step ahead forecast.

2.1.3 Multivariate and the Vector Autoregressive Process

There are many types of MTS models to choose from. First of all there is a decision about linear and non-linear models [Casdagli1992], and then the specific type of model within these two categories. Many non-linear models have been specially designed and tailored for the problem area they are applied to, especially within the financial domain, the choice of which model to use can be very difficult. The dataset under consideration within this thesis will be modelled by one of the multivariate linear time series models, thus avoiding the difficult decisions non-linear model selection can entail; further justification of the choice of a linear model is detailed in section 2.2.2. The main linear models are the *Vector AutoRegressive* (VAR) process, the *Vector Moving Average* (VMA) process and the *Vector AutoRegressive Moving Average* (VARMA) process [Lütkepohl1993].

A VARMA process of order p, q written VARMA(p,q), is defined in equation 2.4.

$$\underline{x}(t)=\sum_{i=1}^{p}A_i\underline{x}(t-i)+\underline{\varepsilon}(t)+\sum_{i=1}^{q}M_i\underline{\varepsilon}(t-i) \tag{2.4}$$

where $\underline{x}(t)$ is the next data vector of size n (the number of variables in the model), A_i is a $n\times n$ autoregressive coefficient matrix at time lag i, M_i is an $n\times n$ moving average coefficient matrix at time lag i, and $\underline{\varepsilon}(t)$ is a n dimensional noise vector at time t (usually of Gaussian distribution) with zero mean. If p=0 then the model represents a VMA process, and if q=0 then the model represents a VAR process. The VAR process follows in equation 2.5, with notation as above:

$$\underline{x}(t)=\sum_{i=1}^{p}A_i\underline{x}(t-i)+\underline{\varepsilon}(t) \tag{2.5}$$

$\underline{x}(t)$ is determined by the sum of some matrix transformations applied to previous observations plus some random noise.

Definition 2.1. $\Sigma_\varepsilon = E(\underline{\varepsilon}(t)\underline{\varepsilon}^T(t))$ defines the covariance matrix for the noise vector and it is assumed that $E(\underline{\varepsilon}(t)\underline{\varepsilon}^T(s))=0,$ where $s \neq t$.

Definition 2.2. The notation nVAR(p) will be used to refer to a n-variable VAR process of order p.

Definition 2.3. A nVAR(p) process has an equivalent (np)VAR(1) representation. See appendix B.

Proposition 2.1. A nVAR(p) process has an equivalent nVMA(∞) representation, that is, an infinite order MA process of the same dimensionality.

Proof. If $\underline{y}(t)$ is the time series generated by the (np)VAR(1) representation of a nVAR(p) process then

$$\underline{y}(t) = A\underline{y}(t-1) + \underline{\varepsilon}(t)$$
$$\underline{y}(t-1) = A\underline{y}(t-2) + \underline{\varepsilon}(t-1)$$
$$\underline{y}(t-2) = A\underline{y}(t-3) + \underline{\varepsilon}(t-2)$$
$$\ldots$$
$$\underline{y}(t) = AA\underline{y}(t-2) + A\underline{\varepsilon}(t-1) + \underline{\varepsilon}(t)$$
$$\underline{y}(t) = AAA\underline{y}(t-3) + AA\underline{\varepsilon}(t-2) + A\underline{\varepsilon}(t-1) + \underline{\varepsilon}(t)$$
$$\ldots$$
$$\therefore \underline{y}(t) = \sum_{i=1}^{\infty} A^i \underline{\varepsilon}(t-i) + \underline{\varepsilon}(t)$$

which is an infinite order VMA process where the coefficients $M_i = A^i$. ■

Forecasting

To use equation 2.5 for prediction purposes, the parameter matrices A_i and the order p must be estimated from the data.

Since the noise, $\underline{\varepsilon}(t)$, within the model is assumed to have zero mean, then a *forecast* of a VAR(p) process is defined in equation 2.6, where the forecast will be equivalent to the conditional expectation [Lütkepohl1993], therefore if $\underline{x}(t)$ is known then $E(\underline{x}(t)) = \underline{x}(t)$.

$$E(\underline{x}(t+h)) = \sum_{i=1}^{p} A_i E(\underline{x}(t+h-i)) \tag{2.6}$$

VAR Parameter Estimation

As previously mentioned, in order to use a VAR process, the order and corresponding parameter matrices must be known. In many practical applications neither of these two sets of parameters are known.

However if just the order is known then there are methods available to help estimate the parameters for a set of data. Three commonly used techniques for estimating the parameter matrices (given the order) are the *Yule-Walker Equations*, *Least Squares* and

Maximum Likelihood methods. With the *Least Squares* method, there can be a restriction on the minimum length of the time series. With the *Maximum Likelihood* method, the distribution of the $\underline{\varepsilon}(t)$ must be known. Unfortunately in some applications this is not the case, such as the visual field data used in this thesis. This dataset's noise term probably does not fall into any standard distribution. Even if it did, since the visual field values always lie between zero and 60 (see appendix A) a distribution would be difficult to identify. So for comparison purposes the Yule-Walker method will be used.

Yule-Walker Equations

The method described by Lütkepohl in [Lütkepohl1993] will be used.

Definition 2.4. The auto-covariance function of $\underline{x}(t)$ is defined as $\varGamma(h) = E(\underline{x}(t)\underline{x}^T(t-h))$ for $h \geq 1$, further if $E(\underline{x}(t)) = 0$ then an estimator for $\varGamma(h)$ can be computed as follows: $\hat{\varGamma}(h) = \frac{1}{T-|h|}\sum_{t=h+1}^{T}\underline{x}(t)\underline{x}^T(t-h).$

Note it can be shown that $\varGamma(h) = \varGamma^T(\text{-}h)$.

Given the definition of a VAR(p) process and post multiplying by $\underline{x}^T(t-h)$ results in:

$$\underline{x}(t)\underline{x}^T(t-h) = \sum_{i=1}^{p} A_i\underline{x}(t-i)\underline{x}^T(t-h) + \underline{\varepsilon}(t)\underline{x}^T(t-h)$$

Taking expectations results in

$$E(\underline{x}(t)\underline{x}^T(t-h)) = \sum_{i=1}^{p} A_i E(\underline{x}(t-i)\underline{x}^T(t-h)) + E(\underline{\varepsilon}(t)\underline{x}^T(t-h))$$

Applying definition 2.4 gives

$$\varGamma(h) = \sum_{i=1}^{p} A_i\varGamma(h\text{-}i) + E(\underline{\varepsilon}(t)\underline{x}^T(t-h))$$

From proposition 2.1 and definition 2.1 it follows that

$$E(\underline{\varepsilon}(t)\underline{x}^T(t-h)) = \begin{cases} \varSigma_\varepsilon & ,h = 0 \\ 0 & ,\text{otherwise} \end{cases}$$

Therefore

$$\Gamma(h) = \sum_{i=1}^{p} A_i \Gamma(h\text{-}i), \text{ where } h \neq 0$$

$$\Gamma(0) = \sum_{i=1}^{p} A_i \Gamma^T(i) + \Sigma_\varepsilon$$

When $\Gamma(h)$ is replaced by estimator $\hat{\Gamma}(h)$ the Yule-Walker equations are obtained. These equations can be rearranged as follows:

$$\hat{\Gamma}(h) = [\hat{A}_1,\ldots,\hat{A}_p]\begin{bmatrix}\hat{\Gamma}(h\text{-}1)\\ \vdots \\ \hat{\Gamma}(h\text{-}p)\end{bmatrix} \text{where } h > 0, \text{and } \hat{A}_i \text{ is an estimator for } A_i$$

$$\underbrace{[\hat{\Gamma}(1),\ldots,\hat{\Gamma}(p)]}_{\hat{\Gamma}} = \underbrace{[\hat{A}_1,\ldots,\hat{A}_p]}_{\hat{A}}\underbrace{\begin{bmatrix}\hat{\Gamma}(0) & \cdots & \hat{\Gamma}(p\text{-}1)\\ \vdots & \ddots & \vdots\\ \hat{\Gamma}(\text{-}p+1) & \cdots & \hat{\Gamma}(0)\end{bmatrix}}_{\hat{B}} \tag{2.7}$$

Which can be written as

$$\hat{\Gamma} = \hat{A}\hat{B}$$

$$\therefore \hat{A} = \hat{\Gamma}\hat{B}^{-1}$$

Hence by using the approximation from definition 2.4, an estimate of the parameter matrices can be obtained. Because the set of equations in equation 2.7 are in fact a *Toelplitz* matrix, *Whittles* extension to the *Levinson-Durbin* recursion can be used to solve them [Whittle1984]; this is the method used by the statistical package S-Plus – see chapter 5. The estimator detailed by Lütkepohl (which is almost identical to the corresponding least-squares estimator and the maximum likelihood estimator under certain circumstances), is defined in equation 2.8.

$$\underline{X} = [\underline{x}(1),\ldots,\underline{x}(T)] \tag{2.8}$$

$$\underline{X(t)} = \begin{bmatrix}\underline{x}(t)\\ \cdots \\ \underline{x}(t-p+1)\end{bmatrix}$$

$$\underline{\underline{X}} = [\underline{X}(0),\ldots,\underline{X}(T-1)]$$

$$\hat{A} = \underline{X}\,\underline{\underline{X}}^T\left(\underline{\underline{X}}\,\underline{\underline{X}}^T\right)^{-1}$$

VAR Order Estimation

The previous section has shown that under certain conditions, and assuming that the order of a nVAR(p) process is known, then an estimation of the underlying parameter matrices that derived the time series can be computed. However in practice, this still leaves an estimation of the order being required.

One way to estimate the order of a nVAR(p) process is to use a metric. Most of these metrics are based upon information theory and some examples listed below. An estimator of p, denoted $\hat{p}$, is chosen that minimises the metric being used. Table 2.1 details some of these metrics. The notation $\hat{\Sigma}_{\hat{p}}$ will refer to an estimation of Σ_{ε} when a VAR($\hat{p}$) has been fitted to a set of data. The notation Δ will be used to represent $\left|\hat{\Sigma}_{\hat{p}}\right|$. Within table 2.1 T and n are the length and dimensionality of the MTS respectively.

Metric	Equation
AIC [Akaike1974] *Akaike's Information Criterion*	$\ln(\Delta)+\frac{2\hat{p}n^2}{T}$
FPE [Akaike1971] *Final Prediction Error*	$\left(\frac{T+n\hat{p}+1}{T-n\hat{p}-1}\right)^n\cdot\Delta$
HQ [Quinn1980] *Hannan and Quinn*	$\ln(\Delta)+\left(\frac{2\ln(\ln(T))}{T}\right)\cdot\hat{p}n^2$
SBC [Schwarz1978] *Schwarz's Bayesian Criterion*	$\ln(\Delta)+\left(\frac{\ln(T)}{T}\right)\cdot\hat{p}n^2$
MSC [Neumaier] *Modified Schwarz Criterion*	$\frac{\ln(\Delta)}{n}-\left(1-\frac{2.5(n\hat{p}+1)}{T-(n\hat{p}+1)}\right)\cdot\ln(T-2.5)$

Table 2.1: Some Order Selection Metrics

AIC is known to have a significant bias when T is small (there has been work down to correct this bias in the univariate case [Hurvich1989]).

All of the metrics have differing behaviours depending upon the circumstances they are used in. For example, FPE and MSC require $T > n\hat{p} + 1$. With a MTS involving 10 variables, to find the most appropriate order of a VAR process with a maximum order of five under consideration, T must be at least 52. This restriction is unacceptable for modelling many short time series. To demonstrate this, the five metrics listed in table 2.1 will be applied to a 7VAR(3) process. The results are given in table 2.2.

Metric	**Multivariate Time Series Length Order**				
	T=1000	T=500	T=100	T=50	T=25
AIC	3	3	3	8	4
FPE	3	3	3	(1-6):3	(1-3):3
HQ	3	3	3	8	4
SBC	3	3	3	8	4
MSC	1	1	1	(1-6):1	(1-3):1

Table 2.2: Order Selection Example

Table 2.2 shows the length of several MTS and the corresponding results from applying the order selection metrics described in table 2.1. The time series vary in length and the metrics are evaluated for candidate orders, $(\hat{p})$, from one to ten. T = 1000 for the 7VAR(3) process and shorter MTS have been created by truncating at the required length; for example in the case of T=25, the shorter time series is created by taking the first 25 observations of the 7VAR(3) MTS. The notation *(x-y):z* is used to indicate that the metric could only consider candidate values $\hat{p}$ from *x* to *y* inclusive, and the best order for that subset is *z*.

Table 2.2 clearly shows all metrics except MSC compute the correct order when $T \geq 100$. However when $T \leq 50$, all of the methods encounter problems. AIC, HQ and SBC choose an incorrect order, and FPE and MSC can only consider a reduced set of orders based on the length restriction outlined above.

It should be noted that FPE does choose the correct order in all cases, but this is because the required order lies within the restricted order range. FPE would not have indicated the correct order if the data would have been from a 7VAR(7) process (or higher). Note also that MSC consistently computes the wrong order for all time series; it is suggested that even a length of 1000 is too short for this metric.

A conclusion can be drawn that the metric based order selection methods could be unreliable, especially when T is small.

VAR Stationality and Stability

Much of the theory for both the linear univariate and multivariate time series analysis has been concentrated on a particular family of times series referred to as *stationary* time series. In short, a time series is stationary if its first and second moments (mean and variance) are time invariant. One interpretation of the Wald decomposition theorem [Lütkepohl1993] is that a VAR process can be used to represent a stationary MTS. With MTS, there is another family called *stable* multivariate time series, which are all stationary. Appendix B defines what a stable time series is. Stationary time series have properties that make them particularly suitable for analysis, for example in solving the Yule-Walker equations.

Unfortunately in many applications stationality cannot be assumed. For example when examining some of the patient's visual field tests, it can be seen that many of the measurements eventually die off to zero. Thus this dataset falls into the category of non-stationality. One commonly used technique to enforce stationality is a difference transformation to create a stationary time series [Box1970]. The data is repeatedly differenced until it becomes stationary, and then this resultant dataset is modelled; which has worked well on many univariate datasets. But if the time series is short, then differencing reduces the size being modelled, losing valuable data points which could be vital to build an accurate model. Additionally with the visual field data not all of the points have the same trend, so differencing might not guarantee stationality at all.

However, consider figures 2.1 and 2.2 which show the two variables of a 1000 length stable 2VAR(2) process.

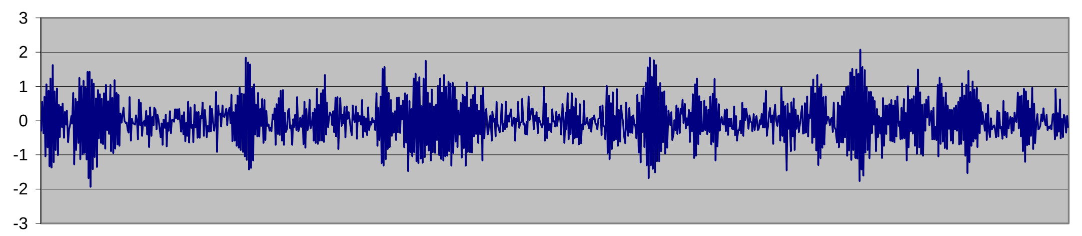


Figure 2.1: Variable 1 of a 2VAR(2), 1000 Time Points

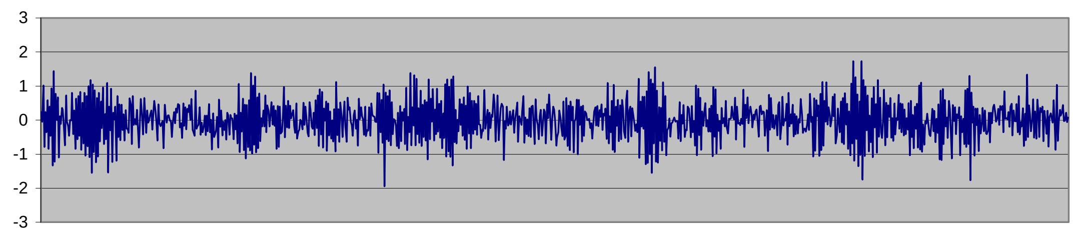


Figure 2.2: Variable 2 of a 2VAR(2), 1000 Time Points

This two variable time series can be inspected visually and can be seen to be stationary. However if the section of the time series corresponding to time points 770-800 is examined as in figure 2.3:

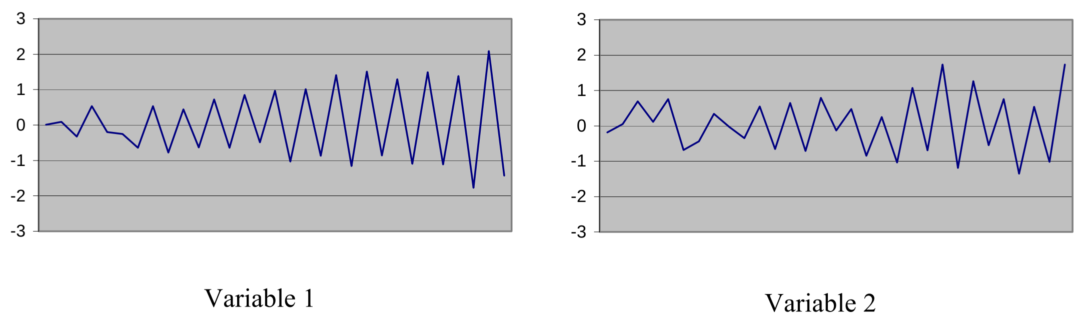


Figure 2.3: Variables 1 and 2 of a 2VAR(2), 1000 Time Points, Points 770-800

It can then be seen that the time series seems to be non-stationary, in this case the variance of variable 1 looks like it is increasing with time. It is only when considered as part of the 1000 time point time series that these two sections can be identified as local fluctuations. This is a problem with trying to model short time series; what seems to be convergence to infinity or zero might be some behaviour that is part of a longer cycle; so a VAR process might be suitable to model such time series, even if they do not appear to be stationary.

Other Details About VAR Processes

There are many variations on a VAR process, each one designed to cater for data of a particular type. The following briefly describes some of the more common variants.

VAR Subset Models [Lütkepohl1993] are VAR processes where one or more of the parameter matrices elements are set to zero. It is desirable to create such a model from a fully dense VAR process since relationships between variables can be visually inspected, a zero parameter indicating no relationship.

Often in real world datasets there are *missing data* points or the time series is *unequally* spaced. Missing data can be imputed by using the EM Algorithm [Dempster1977] or by using Gibbs Sampling [Gilks1996]; note that these two methods are not specifically designed for dealing with VAR processes but have had a lot of success in various applications. In [Jones1984] a method for dealing with unequally spaced observations specifically for a VAR process is suggested. The VAR process has had a high degree of accuracy in forecasting, for example [Webb1995], but recent research has improved upon this basic process [Holden1995]. The Bayesian VAR process (BVAR, e.g. [Dua1995]) is one example. Finally, there are a variety of models for dealing with non-stationary VAR processes, for example the *Time Varying VAR Process* has the parameters matrices being time dependent [Lütkepohl1993].

2.2 The Datasets

Two types of dataset are used to validate the work in the thesis: simulated data and the normal tension glaucoma visual field dataset.

2.2.1 Simulated Data

Simulated data consists of two types of MTS models that are concatenated together to form a higher dimensional MTS. The first types are simulated VAR processes and the second are simulated *Dynamic Bayesian Networks*. Simulated VAR processes are described fully in chapter 4 where they are first used and dynamic Bayesian networks are described below, and how they are simulated is described again in chapter 4.

A *Bayesian Network* (BN) [Heckerman1996] is a *Graphical Model* for representing the relationships and influences between a set of variables. Figure 2.4 gives a simple example of such a network, and is taken from an example in [Pearl1988]. The graphical part of a BN is a directed acyclic graph (DAG), a description of which can be found in [Whittaker1990]. Most BNs work with variables that are discrete or categorical in nature.

Within a BN a variable is influenced by a certain number of other variables. The term *Parents* is used to refer to the set of variables that influences a given variable, and the term *Children* is used to refer to the set of variables that is influenced by a given variable. Within figure 2.4, *Alarm Sound* has *Burglary* as a parent and *Gibbon's Testimony* and *Watson's Testimony* as children. Parents have a probabilistic effect upon their children, indicated by the direction of the arrows.

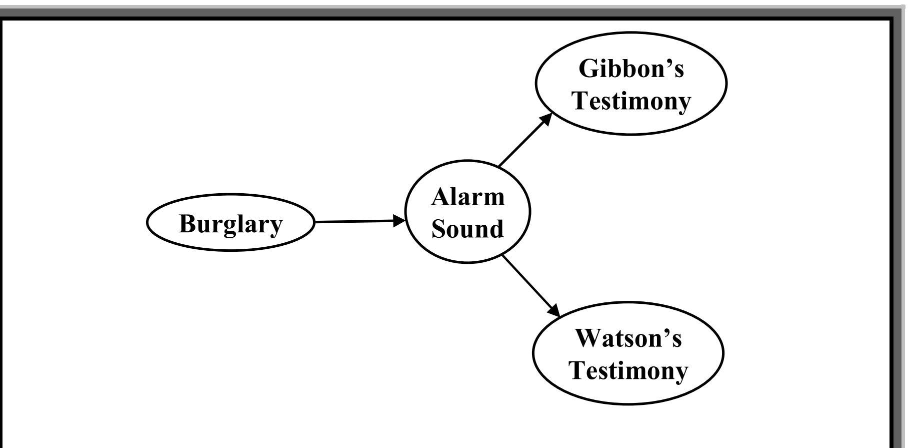


Figure 2.4: A Simple Bayesian Network

Equation 2.9 is the joint probability distribution for the set of variables *X* under the conditional independence assumption. $p(x_i|\pi_i)$ is the local conditional probability for variable x_i. In this equation, *X* is a set of variables $\{x_1,...,x_n\}$, which represents the fields of a database or the nodes in a DAG and π_i is the set of parents of variable *i*.

$$p(X) = \prod_{i=1}^{n} p(x_i \mid \pi_i) \tag{2.9}$$

The BN works as follows: given instantiations of some of the variables in *X*, and the relationships in equation 2.9, derivation of values for the remaining variables in *X* is possible. Algorithms for this procedure are given in [Buntine1996]. A BN can be used for predicting values for a partial record, and can offer explanation for why various instantiations have occurred.

Dynamic Bayesian Networks [Friedman1998] can be used to model a MTS, and are very similar to the standard BN. A dynamic Bayesian network (DBN) 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 [Dagum1995]. Much of the current developments in DBNs can model MTS provided that the data are in discrete states.

2.2.2 Glaucoma Visual Field Data

As detailed in the introduction of this thesis, the dataset that is the main subject for analysis is a set of *Normal Tension Glaucoma Visual Field* data. Glaucoma [Hitchings2000] is a condition that affects the human eye. The visual field data consists of 76 points recorded approximately every six months for each patient. This falls into the category of MTS data (the VF dataset can also be considered to be a *Spatial Time Series*, for example [Pfeifer1980a, Pfeifer1980b], however this will be addressed later).

Anatomy of the Eye

Figure 2.5 shows a schematic of the human eye, which is essentially the same for both males and females.

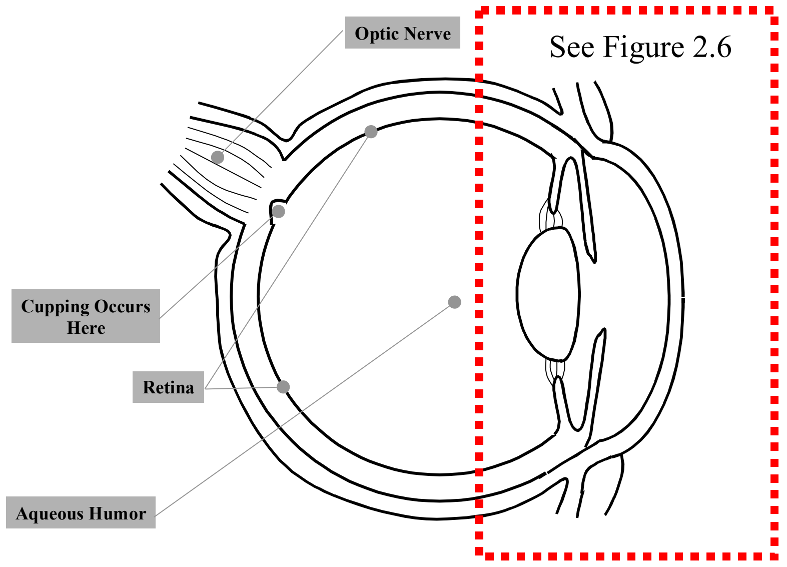


Figure 2.5: Schematic of the Human Eye

The eye acts rather like a video camera. The front part of the eye, the dotted section in figure 2.5 and expanded in figure 2.6, focuses light onto a set of light sensitive cells which is converted into an electrical signal and sent to brain; the *Retina* is this light sensitive section. The label "*cupping occurs here*" is pertinent to the glaucoma condition, which will be explained in the next section. *Aqueous Humor* is a liquid that helps to maintain the eyes shape and rigidity, which is essential for the eye to focus light correctly (a camera that changes shape continuously would produce photographs that varied in focus). The *Optic Nerve* is a bundle of nerves that transmits what is focused by the front of the retina to the brain. Like a camera this image is upside down when focussed and transmitted; the brain performs a pre-processing action and corrects this.

Glaucoma

Glaucoma is the name given to a family of eye conditions. The common trait of these conditions is a functional abnormality in the optic nerve, leading to loss of visual field. This vision loss is usually only part of the visual field, however untreated glaucoma can lead to blindness.

Even though this condition was identified many years ago, for example [Graefe1857], little is known about its cause. It is thought that a build up of eye pressure in the eye, (this pressure is known as *Intra-Ocular Pressure* – IOP), causes damage to the retina, by pushing the optic nerve head inwards causing *cupping*. Figure 2.6 shows the front part of the eye, which is relevant to understanding the cause of this damage. For a more detailed description of the eye, see [Hollwich1985]. The *Cornea* is the optical window to the eye; the *Iris* is a muscle that expands or contracts, thus altering the perceived size of the *pupil* (the gap), thus focusing light correctly through the *lens* onto the retina. The *Anterior Chamber* contains fluid to keep the front of the eye rigid, which is called *Vitreous Humor*. The *Conjunctiva* is the transparent mucous membrane lining the inside of the eyelids and the white of the eyeball (which is called the *Sclera* and is very tough). The fluid in this part of the eye is continually refreshed, flowing out of the *Cilary Epithelium* around the iris and drained off through the *Trabeculum* (a network of meshing).

The pressure within the eye can be increased in a variety of ways. In particular the trabeculum may become blocked in some manner, thus the influx of vitreous humor is not drained away at the rate it is increasing. If flakes of pigment from the iris block the trabeculum, the condition is called *Pigmentry Glaucoma*. If the *Angle* within the diagram, (measured between the iris and the cornea wall), becomes small for some reason, thus again reducing the rate of drainage, it is referred to as *Closed Angle Glaucoma*. *Primary Open Angle Glaucoma* refers to similar conditions where the angle has not been closed, the trabeculum appears unblocked, and IOP is above normal.

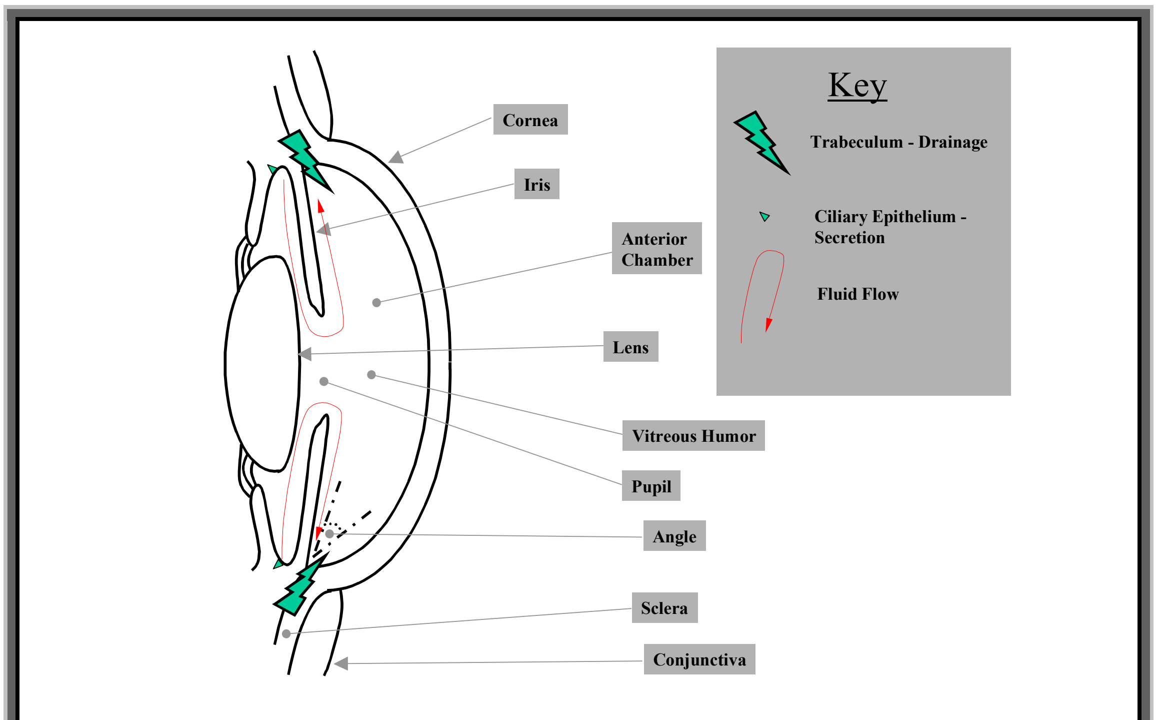


Figure 2.6: Schematic of the Front of a Human Eye

IOP has not been proven to be a cause of glaucoma, but it has been shown to be a factor. Figure 2.7 shows a rough relationship. The normal IOP for a person is between 12 and 20 mmHg. If a person’s IOP is above 20 mmHg then they are referred to as suffering from *ocular hypertension*.

In order to reduce IOP, drugs can be issued to reduce the flow of fluid and hence, in theory, the IOP. However these drugs can have some serious side effects. Surgery can sometimes help in clearing the trabeculum if it is blocked, or widening it in order to improve drainage. However eye surgery is seldom desirable.

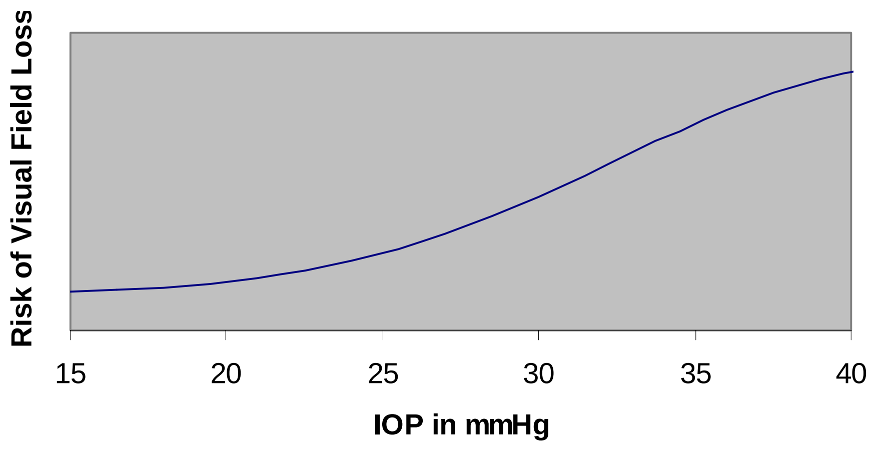


Figure 2.7: IOP vs. Risk Factor

The dataset for this thesis is a section of normal tension glaucoma visual field data [Haley1987]. Normal tension glaucoma is different to types mentioned above, and occurs when the IOP is within the expected bounds, i.e. 12 to 20 mmHg. This is one of the more difficult types of glaucoma to deal with since pressure seems to be less of a cause.

Visual Field Analysis

Visual fields tests are crucial to the diagnosis and management of all types of glaucoma. To conduct visual field analysis the retina is divided into a set of points, the level of sensitivity of the eyesight of a patient is tested at each point and is assigned a value between 0 (no perception) and 60 (perfect perception). A specialised machine is used to conduct these tests, which can take several hours for all the points of both eyes. A patient once diagnosed with glaucoma or is a glaucoma suspect, for example ocular hypertension, will undergo these tests approximately every six months.

The rate of change of the patient's sensitivity in parts of the eye, and as a whole, can be a useful guide for clinicians who are treating these patients. For normal tension glaucoma the *Central Threshold 30-2* test is usually used which tests 76 points. This can be seen in figure 2.8.

Figure 2.8 shows these points as co-ordinates measured from the centre of the retina for the right eye. The squares in black designate the position of the blind spot. The retina is connected to bundles of nerves that transmit the image to the brain. The mapping of these nerve fibre bundles to the 76 points of the 30-2 test are shown in figure 2.9. The diagrams for the left eye are a mirror image of figures 2.8 and 2.9

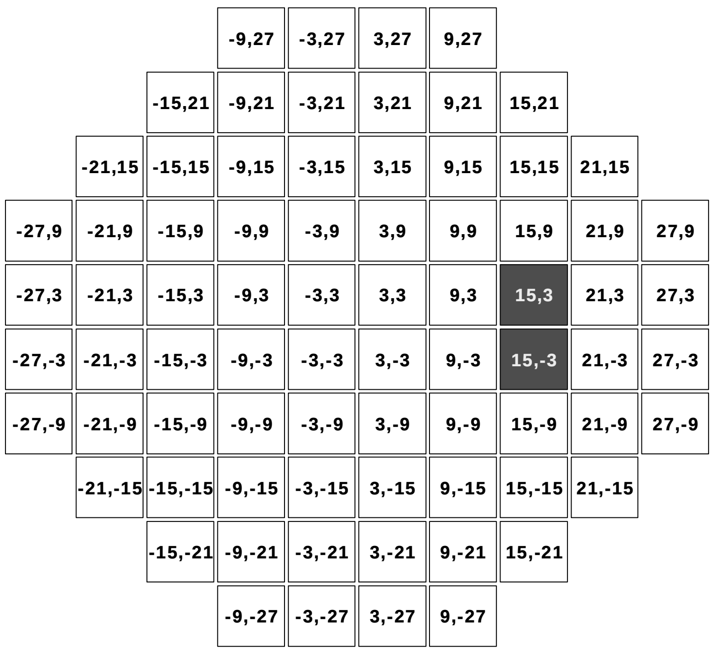


Figure 2.8: The 76 Points of the Central Threshold 30-2 Test

In figure 2.9 the blind spot is marked with a *B* and each nerve fibre bundle is assigned a different number between 1 and 15. The different shading is simply to make each group more distinct. Current theory [Crabb1996/97, Heijl1988/89a] states that deterioration of the visual field can be highly correlated if two points lie on the same nerve fibre bundle.

The prediction of visual field deterioration in patients who are suffering from glaucoma plays an important role in the management, treatment and control of the diseases progress. For example, if the rate of deterioration is decreasing it might be appropriate to reduce the medication; or if the rate of deterioration is increasing, more medication might be needed or even surgery might be necessary.

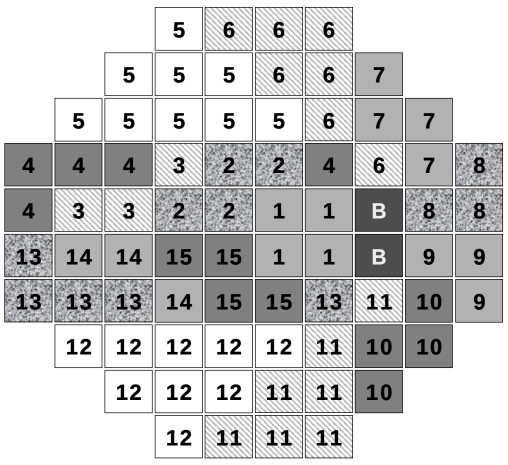


Figure 2.9: The 76 Points of the Central Threshold 30-2 Test By Nerve Fibre Bundle

The Dataset

Appendix A details the database used to store the data for the patient's visual fields that were the subjects of this research. For many of the methods applied to this data it is convenient to map the coordinates of each point to a unique natural number between 1 and 76. For example when storing the visual field tests as vectors of time series observations, each point needs to correspond the same row in each vector. This mapping is shown in figure 2.10.

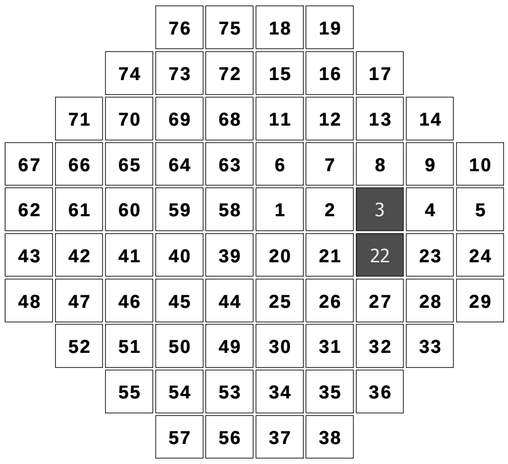


Figure 2.10: The 76 Points of the Central Threshold 30-2 Test By Mapping

For the purpose of this research the dataset will be considered to be a MTS and will be modelled using the VAR process. The justification for this and any assumptions that have been made to use such a model are as follows:

i) The medical literature shows that there are relationships between points, and current theory states that deterioration of the visual field can be highly correlated if two points lie on the same nerve fibre bundle [Crabb1996/97, Heijl1988/89a].

ii) Any noise within the model can be explained by experimental noise, patient fatigue etc.

iii) It is also logical to assume that an effect from one point to another must take some time, i.e. a cause cannot be instantaneous.

The visual field data appears to be non-stationary. This is compatible with the nature of the condition, i.e. a gradual degradation to blindness in part or all of the field of vision. However this short MTS could be modelled by considering a much longer stationary series that does not exhibit this behaviour, rather like with variable one in figure 2.3. The VAR process therefore is suitable for modelling the visual field data, however it will be a challenging task to identify the parameters and order of such a model.

Existing Visual Field Modelling Methods

Much work has been done to try and predict, interpret and analyse visual field test data. The literature for visual field analysis can be divided into three groups: prediction and classification detailed in table 2.3; clustering detailed in table 2.4; and the detection and management of outliers detailed in table 2.5. Additionally a useful survey into some of the applications of computer technology within the domain of visual field analysis can be found in [Åsman1992].

Method	Reference(s)	Description
Progressor	[Fitzke1996] [Mcnaught1995] [Mcnaught1996] [Viswanathan1997]	The use of linear regression and the corresponding significance level, combined with a user interface, to model progression. Aimed as a computerised tool for clinicians
Linear Regression	[Wild1997] [Birch1995]	The use of univariate pointwise linear regression to classify and forecast progression
Topographical and Longitudinal Modelling	[Wild1993]	The use of a mixed topographical and longitudinal model to forecast progression
Multivariate Regression	[Nouri-Mahdavi1997]	Multivariate Regression is applied to the visual field data and then compared with clinicians, and linear methods in order to examine the level of agreement
Neural Networks	[Brigatti1996] [Brigatti1997] [Goldbaum1994] [Spenceley1994]	A variety of *Neural Network* configurations have been used to classify patients visual fields into a variety of categories and groups, for example distinguishing between normal and glaucomatous visual fields

Table 2.3: Visual Field Prediction and Classification

Linear Regression based methods [Mosteller1977] assume that the data is univariate, and fit the best straight line through the data. Progressor (see table 2.3 for references) is a system based on linear regression that gives the user a rate of change parameter and associated significance level for each point in a set of visual field tests. Linear regression will not be evaluated against the methods presented in this dissertation. The justification for this is that the visual field data has been found to be inherently multivariant (see chapter 5), and that Progressor does not provide forecasts for all of the visual field points, only those that are found to have a statistically significant fit after linear regression has been modelled. *Topographical Modelling* tries to handle the spatial dependencies between test points, and *Longitudinal Modelling* looks at the previous tests from a longitudinal data analysis point of view, for example [Diggle1994]. The combination of these two techniques is considered in [Wild1993]. Time series analysis is a more suitable way of treating the visual field data (see section 2.1). Spatial modelling of the visual fields becomes complex because the inter-point dependencies change over time due to the progression of the condition. However these changing spatial dependencies should not be disregarded. Consideration of this is made in section 7.3.3. *Multivariate Regression* is more suited to non-time series data, and a VAR is effectively a multivariate regression model over time. A variety of classification methods have been used to distinguish glaucomatous fields from other conditions, a review of which can be found in [Hand2001].

Method	**Reference(s)**	**Description**
Hierarchical Clustering	[Mandava1993]	The use of a *Hierarchical Clustering* technique on the visual field points to see if they can be grouped together
K-SOM	[Spenceley1996]	*Kohonen Self-Organising Maps* have been used to spatially cluster visual field loss
Non-Hierarchical Clustering	[Åsman1993] [Chauhan1988] [Heijl1988/89b]	The use of a clustering techniques other than *Hierarchical Clustering*

Table 2.4: Visual Field Clustering

Various clustering techniques have been applied to visual fields data to try and group together points that have similar behaviour or features (see table 2.4 for details). *Hierarchical Clustering* yields a dendrogram (binary tree) representing the nested clusters of patterns and similarity levels at which clusters change. The dendrograms can be broken at different levels to yield different clusterings of the data. *Kohonen's Self Organising Map* (K-SOM) is a form of neural network for transforming an incoming signal pattern of high dimension into a one or two dimensional discrete map [Kohonen1989]. This mapping can then be examined visually to see if any patterns between the data emerge. Many other clustering techniques have been applied to the visual field data; a good review of *clustering* can be found in [Jain1999].

Method	Reference(s)	Description
Filters	[Fitzke1995]	*Spatial Filters* have been applied to try and improve the accuracy of Progressor
K-SOM	[Cheng1996] [Henson1997] [Liu1994]	*Kohonen Self-Organising Maps* have been used to classify noise in visual field tests

Table 2.5: Visual Field Outlier Management

Spatial Filters have been applied to the visual field data to try and remove the influence of any outliers [Barnett1994]. The process involves modifying each point by a linear combination of its spatial neighbours, which has the effect of "blurring" the data. A small increase in forecast accuracy has been noted if this pre-processing step is applied before the application of Progressor. *Kohonen's Self Organising Map* has been used in various forms to classify and explain potential outliers within the visual field data. For example in [Liu1994] binary visual field data is collected using a portable visual field test and then K-SOM is then applied to compress the data onto a two dimensional map that allows the identification of outliers and the understanding of test behaviour.

2.3 Evolutionary Computation

Evolutionary Computation [Bäck1997] is the name given to the procedure of simulating evolution [Darwin1998] on a computer. An *Evolutionary Algorithm* is an algorithm that performs evolutionary computation. The field of evolutionary computation is relatively new and is considered to contain the *sub-fields* of *Genetic Algorithms*, *Evolutionary Programming* and *Genetic Programming*. These three techniques are described in the following sections.

2.3.1 Genetic Algorithms

A *Genetic Algorithm* (GA) [Holland1975] represents a solution to a problem as a binary string, called a *Chromosome*. Each bit of a chromosome is called a *gene*. A population of *chromosomes* is maintained, which represents a subset of the space of all possible solutions. Through subsequent generations (iterations) the suitability of the population is improved through a process of breeding, mutation, and survival of the fittest (to maintain a constant size population), analogous to the real-world equivalents. The problem solution is the fittest individual of the last population.

Breeding within a genetic algorithm is referred to as *crossover* or *recombination*. This procedure is used to create children by recombining sections (portions) from one or more parents. *Mutation* changes a number of the population, and is usually applied to children resulting from the crossover stage. Survival of the fittest selects a number of the parents and children to be carried over to the next generation. The suitability of a chromosome to solve a particular problem is usually referred to as the chromosome's *fitness*.

Genetic algorithms are suitable for solving optimisation type problems [Michalewicz1995]; search type problems [Hackworth1999] and permutation and ordering type problems [Blanton1993]. Any genetic algorithm must define the concepts of *representation*, *fitness*, *crossover*, *mutation* and *survival*. The genetic algorithm follows algorithm 2.1.

The parameters POPULATIONSIZE, GENERATIONS and NBITS are implementation and application dependent. *Validity* will be described in the next section. Crossover, mutation and survival are referred to as genetic operators. In Holland's original genetic algorithm, survival is performed before crossover. However the difference between this and algorithm 2.1 is whether a single survival is applied at the start (Holland) or at the end (algorithm 2.1) after the sequence of operators has been applied for all generations.

Algorithm 2.1: The Genetic Algorithm

```
1)    Input:  Fitness function for a Chromosome
              POPULATIONSIZE – the number of chromosomes in each population
              NBITS – the number of bits making up each chromosome
              GENERATIONS – How many iterations to run the genetic algorithm for
              CROSSOVERRATE – The chance of a chromosome becoming a Parent
              MUTATIONRATE – the chance each chromosomes gene has of Mutating
2)    Generate POPULATIONSIZE Valid chromosomes of size NBITS bits
3)    For loop = 1 to GENERATIONS
4)            Crossover the Population to Create Children
5)            Mutate the Population
6)            Remove Invalid chromosomes
7)            Apply Survival of the Fittest to the Population
8)    End For
9)    Output: The fittest individual of the last population
```

Representation

Genetic algorithms were initially designed to work on a problem whose solution could be represented by a binary string. The particular solution a chromosome represents is called the *phenotype*, and the corresponding binary representation is called the *genotype*. There is no hard or fast rule in determining the representation of a solution as a binary string, but the general rule is to keep the representation as close to a one to one mapping as possible, i.e. each phenotype has only one genotype and vice-versa. A chromosome is considered valid if there is a mapping from its genotype to a phenotype. The conversion from the genotype to the phenotype is usually through a mathematical formulae, algorithm or look-up table. Each position of each bit therefore should have some function or part in building up the phenotype. If part of chromosome does not play any part of constructing the phenotype then it is referred to as *redundant*. A gene's (bit) position is referred to as its *locus*. In order for a genetic algorithm to function correctly then the interpretation of a locus should not change, i.e. if it corresponds to red when set to one in the phenotype in a chromosome then it should correspond to red when set to one in any other chromosome. The values a particular locus can take are called *alleles*; in the case of a binary chromosome, all loci have alleles of zero or one. The genetic algorithm proposed by Holland works on fixed length chromosomes. The choice of a suitable representation is one of the most important stages in the development of a genetic algorithm.

For example, consider the use of a genetic algorithm in a two variable optimisation problem. This example problem could consist of the variables *X* and *Y*, which lie in the integer range [0,100] and [0,20] respectively. To represent this as a binary string, 0 to 100 can be represented in seven bits and 0 to 20 can be represented in five bits. This gives a total of twelve bits to represent every possible solution in the problem space, see figure 2.11. Note that bits 0 to 6 (counting left to right) construct *X* and bits 7 to 11 construct *Y*, using a base 2 encoding. Seven bits can represent numbers in the range [0,127] and five bits can represent numbers in the range [0,31], which means that approximately 48.2% of random chromosomes will be *invalid*. This means that the representation is fairly poor since almost half of any children created will be thrown away.

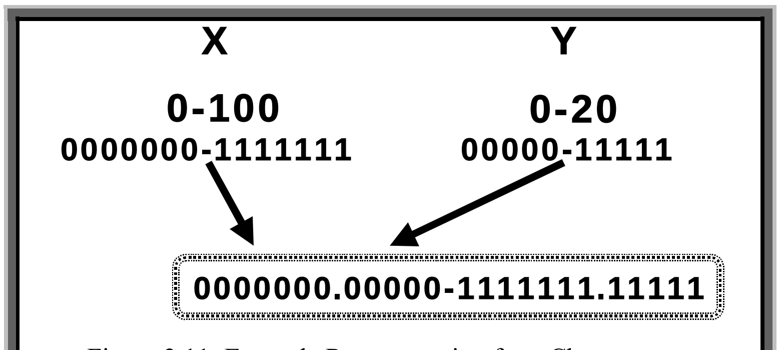


Figure 2.11: Example Representation for a Chromosome

If the chromosome contained ten bits so that *X* could take values between [0,63] and *Y* values between [0,15] then there would be no invalid chromosomes. However not all of the phenotypes would have a corresponding genotype thus the genetic algorithm will never reach the best solution to the problem if it lies outside the values the genotype can represent.

Note that many implementations use real or integer number for genes [Goldberg1990] instead of binary digits. A genetic algorithm has been shown to converge to good solutions in the binary case (see the *Schema Theorem* below), but proofs are more difficult for the real or integer cases. With non-binary genes crossover essentially remains the same, but mutation must be modified.

Fitness

The *fitness function* for a genetic algorithm is usually a mapping from a chromosome's phenotype to a real number. The fitness function is used to rate how well a chromosome solves the problem in question. This fitness is traditionally maximised, but a genetic algorithm can function equally well on fitness minimisation problems. The mapping between phenotype and fitness should be one to one, but a mapping of many to one will still work. However if a phenotype has many fitness's then the genetic algorithm will function poorly if not at all. The choice of a suitable fitness function is as important as choosing a suitable representation. The term *population fitness* is usually defined as either the sum or the average of the fitness of all of the chromosomes in a given population.

Crossover

Crossover is the procedure of recombining parts of chromosomes to create new individuals with the aim of improving the population's fitness. The two most common types of crossover are *one-point crossover* [Holland1975] and *uniform crossover* [Syswerda1989]. Both types of crossover require two parents. The method of choice of the parents can vary, but usually each member of the current population is given a chance of breeding (*CrossoverRate*). If the number of chromosomes in the breeding subset is odd, then there is a 50% chance that a random one is either added from the remaining population or is removed. Each chromosome in the breeding subset is randomly allocated a partner giving a number of parent pairs. The resultant chromosomes of each parent pair are referred to as the parent's *children*, and both methods of crossover described in this section produce *two children*.

One-point Crossover. A uniform random whole number, *i*, is generated in the interval [2,NBITS). *Child one* is created by copying genes *1..i* of parent one and then appending genes (*i+1*)..NBITS of parent two to the end. Similarly, *Child two* is created by copying genes *1..i* of parent two and then appending genes (*i+1*)..NBITS of parent one.

Uniform Crossover. Child one is created by copying parent one, and child two is created by copying parent two. For each gene indexed 1…NBITS, left from right within the chromosome, a uniformly distributed random whole number is generated in the interval [1,2]. For each case where this number is one, the two corresponding genes of child one and child two are swapped.

Figure 2.12 graphically depicts these two forms of crossover. Within this figure, the cells labelled *P1* and *P2* correspond to genes from parent one and parent two respectively. In the one-point crossover example, the *crossover point* is *i*=6. For the *uniform crossover* example, swaps are made for genes 1, 2, 4 and 9.

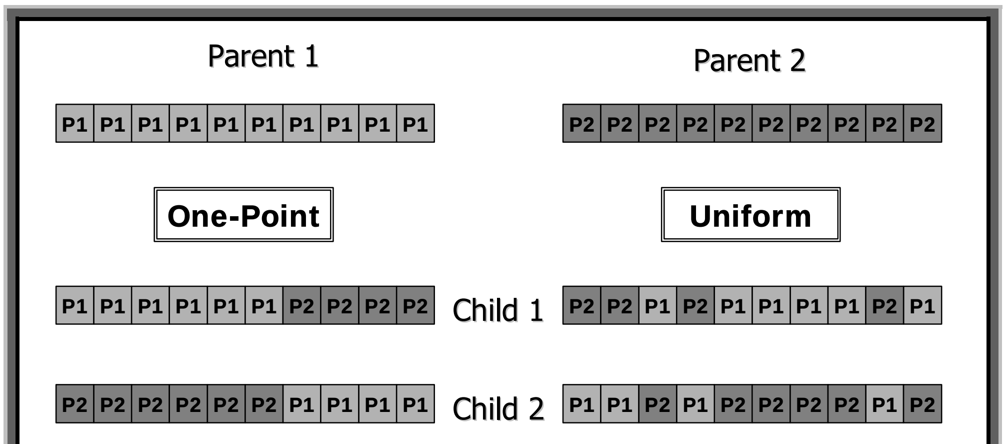

Parent 1
Parent 2
P1 P1 P1 P1 P1 P1 P1 P1 P1 P1
P2 P2 P2 P2 P2 P2 P2 P2 P2 P2
One-Point
Uniform
P1 P1 P1 P1 P1 P1 P2 P2 P2 P2
Child 1
P2 P2 P1 P2 P1 P1 P1 P1 P2 P1
P2 P2 P2 P2 P2 P2 P1 P1 P1 P1
Child 2
P1 P1 P2 P1 P2 P2 P2 P2 P1 P2

Figure 2.12: One-Point and Uniform Crossover Operators

Mutation

Mutation is a technique that randomly changes the values of zero or more genes of a chromosome. Algorithm 2.2 describes mutation in the binary case.

Definition 2.5. The function *UR*(MIN,MAX) is a uniformly distributed random number generator that returns a *real* number in the interval [MIN,MAX].

Definition 2.6. The binary operator *mod* is the modulo arithmetic operator. That is, it returns the remainder from integer division (*div*), e.g. 17 *mod* 5 = 2 since 17 *div* 5 = 3 remainder 2.

Algorithm 2.2: The Mutation Operator

```
1)     Input:  The Population of Chromosome to under go mutation
               MUTATIONRATE – the chance each Chromosomes gene has of Mutating
               NBITS – the number of bits making up each Chromosome
2)     For Each Chromosome CHROME that is to Mutate
3)             For i = 1 to NBITS
4)                     If UR(0.0,1.0) < MutationRate then
5)                             Set $g_i$ = ith gene of CHROME
6)                             Set ith gene of CHROME = ($g_i$ + 1) mod 2
7)                     End If
8)             End For
9)     End For
10)    Output: The mutated population
```

MutationRate is a user defined parameter. The expected number of mutations is defined in equation 2.10.

$$\mathrm{E}\left(\textit{Number of Mutations}\right) = \textit{MutationRate} \times \mathrm{NBITS} \tag{2.10}$$

The general rule of thumb for the mutation rate is to set the expected value equal to one; hence *MutationRate* is defined as the reciprocal of NBITS. For a real valued genetic algorithm, mutation must be amended. Typically the gene is replaced with a uniformly distributed random number between the limits the gene can take. Other forms of mutation involve perturbing the genes value according to a random Gaussian distribution (this is sometimes called *Creep Mutation* [Goldberg1990]).

Survival

Survival of the Fittest is defined by Charles Darwin [Darwin1998] as follows:

> *"This preservation of favourable individual differences and variations, and the destruction of those which are injurious, I have called Natural Selection,* [*in later editions:*] *or the Survival of the Fittest."*

In short, this is the process where creatures well adapted to the environment have more chance of living, and hence breeding, than those less suitably adapted. In genetic algorithms (and most other evolutionary algorithms) the process is applied to a population of chromosomes (parents and children) in order to reduce the population size to that of the starting population; making sure those chromosomes with higher fitness are more likely to be retained than those with lower fitness. Without the survival operator, the population size would increase exponentially each generation. An initial population of ten would increase to approximately 10,000 after 10 generations, and to approximately 10^7 after 20 generations; thus the a genetic algorithm would use all of the resources of a computer long before it found a good solution. The survival of the fittest stage is often referred to as *selection*.

The aim of the survival stage is to allow individuals to pass on to the next generation with a probability that is a function of its fitness. The most common method is called the *roulette wheel* (or *proportional selection*), which was the original method utilised by Holland. When survival is applied to a population it is assumed that the current population size is larger than the target population size (which is usually equal to the initial population size). The roulette wheel selection method works by giving each chromosome a chance of surviving until the next generation based upon its fitness as a proportion to the fitness of the whole population. Formally, given a population (numbering P_{size}) of chromosomes $C_1,\ldots,C_{P_{size}}$, the probability that a chromosome survives $Pr(C_i)$ is defined in equation 2.11.

$$\Pr(C_i) = \frac{Fitness(C_i)}{\sum_{j=1}^{P_{size}} Fitness(C_j)} \tag{2.11}$$

where *Fitness* is the function that returns the fitness of a given chromosome. Note that a chromosome can be chosen zero or more times. The method is called the roulette wheel since the required number of chromosomes are chosen in a similar manner to having a biased roulette wheel, where the size of a pocket is proportional to the chromosomes fitness.

To implement this, the current population is sorted according to fitness, and then the probabilities computed for each individual according to equation 2.12. A random uniformly distributed real number between zero and one is generated to select each chromosome to go forward to the new population. The individual (*C_i*) is selected that satisfies equations 2.13 to 2.14.

$$CP(i) = \sum_{j=1}^{i} \Pr(C_i),\ where\ 1 \le i \le P_{size} \tag{2.12}$$

$$CP(i) < UR(0.0,1.0) \le CP(i+1) \tag{2.13}$$

$$Fitness(C_i) \ge Fitness(C_{i+1}) \tag{2.14}$$

CP(i) is the cumulative probabilities from chromosome number one to *i*. Equation 2.14 simply reasserts that the chromosomes are in order of fitness with chromosome number one having the highest fitness.

Other survival methods have been developed, such as *Tournament Selection* and *Elitism*. *Tournament selection* [Bäck1993a] is where each individual is compared with a number of random other individuals in the population. Each time the individual betters its random competitor it gets a point. After every individual in the current population has been scored, the required number of individual to survive is chosen by selecting those with the highest score. *Elitism* [Dejong1975] is where a number of the best (fittest) individuals of the current population is allowed to survive (unchanged) automatically, thus ensuring the best fitness never decreases over subsequent generations. Elitism is often combined with other survival operators. A description and comparison of several survival methods can be found in [Goldberg1991].

The Schema Theorem

The fact that a genetic algorithm converges to a solution is very complex, and hence only a brief outline will be explained. This result is called the *Schema Theorem* [Holland1975, Goldberg1989], and the form of this theorem outlined below only applies to binary encoded genetic algorithms. First of all some necessary definitions will be outlined, and then the theorem will be described. This section is taken from [Michalewicz1996].

A *Schema* is a template that describes similarities among chromosomes. A schema incorporates the idea of a wild card character '*', this is treated as a "don't care" symbol, for example the 4 bit schema 1*00 matches the 4 bit chromosomes 1100 and 1000. The value of a schema is the average of all the chromosomes in a population containing that schema (or matched by a schema). The order of a schema is its length, which is the number of bits between the first and last non-wild card characters. For example the 4 bit schema above is of length four; the 6 bits schema *11*0* is also of length four. It can be shown that the Holland genetic algorithm outlined above will result in populations favouring above average schemas as time (generation number) increases.

The Schema Theorem:

> *"Short, low order, above average Schemata receive increasing occurrences in subsequent generations of a Genetic Algorithm."*

Definition 2.7. *Epistatsis* is the measure of how important two loci are for determining the fitness of a chromosome, relative to their physical distance apart in the representation. For example the requirement that all good solutions must have a '1' at both the end and rear of a chromosome. In biological systems, a gene is epistatic if its presence suppresses the effect of another gene in another locus.

These short low order schemas will have “medium” epistasis. The theorem states that such schema will occur more and more frequently as the number of generations increase. These schemas will be unknown, but will start to emerge (if such an analysis was undertaken) dependent on the representation used (and the problem).

The *Building Block Hypothesis* states how these schemas work together in a genetic algorithm:

> *“A Genetic Algorithm seeks near optimal performance through the juxtaposition of short, low order, above average Schema. These are called building blocks.”*

To summarise the schema theorem:

i) The random initial population creates some random schema
ii) Crossover and mutation aid the creation of new schema
iii) Survival of the fittest gets rid of low scoring schema
iv) Hence the populations average score tends to increase as the generation number increases

2.3.2 Evolutionary Programming

Evolutionary Programming (EP) is based on a similar paradigm to *genetic algorithms*. However, the emphasis is on mutation and it does not use any recombination. The basic algorithm is outlined in algorithm 2.3 [Bäck1993b, Fogel1995].

The best out of the final population will be the best solution to the problem. Traditionally, EP algorithms use tournament selection during the survival of the fittest stage. The representation is usually *not* binary, and the mutation operator can be quite complex. All the other parts of an evolutionary program are the same as with a genetic algorithm. Evolutionary programming will be further covered in chapter 3.

Algorithm 2.3: Evolutionary Programming

```
1)    Input:  Fitness function for a Chromosome
              POPULATIONSIZE – the number of Chromosomes in each population
              NGENES – the number of genes making up each Chromosome
              GENERATIONS – How many iterations to run the Evolutionary Program for
2)    Generate POPULATIONSIZE Chromosomes of size NGENES
3)    For i = 1 to GENERATIONS
4)            Duplicate the Population to Children
5)            Mutate all of the Children
6)            Add the Children back to the Population
7)            Apply Survival of the Fittest to the Population
8)    End For
9)    Output: The fittest individual of the last population
```

2.3.3 Genetic Programming

Genetic Programs [Koza1992] are used to solve a problem that can be modelled as a grammar, for example BNF, see [Aho1986]. In this thesis, the technique that is of interest is *Symbolic Regression*, which is a type of *Genetic Programming*. With symbolic regression, a mathematical expression is represented as a tree structure. Terminal nodes within this tree are usually variables, and non-terminals are operators (for example +,-,/,×) or functions (for example – logarithm). The fitness of such a tree is a function of the observed data versus the calculated data resulting from evaluating the expression the tree represents. A commonly used example is the least squares error as in linear regression [Mosteller1977]. Crossover and mutation are redesigned to handle tree structures. The genetic programming algorithm is the same as the genetic algorithm in all other respects and is detailed in algorithm 2.4.

Algorithm 2.4: Genetic Programming

```
1)  Input:  Fitness function for a Chromosome
            POPULATIONSIZE – the number of Chromosomes in each population
            GENERATIONS – How many iterations to run the Genetic Program for
            Implementation dependent parameters
2)  Generate POPULATIONSIZE Chromosomes of size NGENES
3)  For i = 1 to GENERATIONS
4)          CROSSOVER the Population to Children
5)          MUTATE all of the Children
6)          Add the Children back to the Population
7)          Apply Survival of the Fittest to the Population
8)  End For
9)  Output: The fittest individual of the last population
```

Typical operators [Banzhaf1998] are described in table 2.6 below.

Operator	Description
Sub-tree Exchange Crossover	Two sub-trees are swapped between parents
Self Crossover	Sub-trees are exchanged within an individual parent
Point Mutation	A node in the tree is changed to a different symbol
Permutation Mutation	Two terminal symbols from the same sub-tree are swapped
Hoist Mutation	A sub-tree creates a new individual
Expansion Mutation	A sub-tree is added to the base of a tree
Prune Mutation	A sub-tree is removed
Sub-tree Mutation	A sub-tree is replaced for a random sub-tree

Table 2.6: Some Genetic Programming Operators

Genetic Programming will be further covered in chapter 4.

3. Correlation Analysis

The work presented in this chapter presents the first stage of the three-stage procedure for the modelling of high dimensional short multivariate time series. This stage concerns the mining of relationships in a time series. The goal is to develop an algorithm that finds a *good-but-not-optimal* selection of *interesting* highly related variables in as short amount of time as possible.

One way to find the structure in a dataset is correlation analysis. In this chapter three methods for finding the approximate correlation structure are presented and compared to the exhaustive search method for verification. Two methods for calculating correlations are utilised which are described in the next section. The methods for performing fast, approximate search include a version of an evolutionary programming algorithm and a genetic algorithm. The results are presented on the visual field dataset using the different algorithms and correlation coefficients. Finally the results are discussed and future work considered.

Once it has been demonstrated that the methods for discovering an approximate set of relationships are effective and efficient, this information is then used in the grouping problem described in chapter 4. This chapter is an extended version of work presented in [Swift1999b].

3.1 Correlations in Time Series

Correlation [Snedicor1967] is a measure of the relationship between two sets of observations or variables. Correlation between points at different times lags can play a useful role in the monitoring of the disease progression; since many mathematical methods for time-series forecasting need the correlations between variables to develop the models. It would be useful to be able to do this during a patient's regular consultation so that any decisions could be made while they wait, hence in as short a time as possible. Correlation analysis is a way to measure how 'coupled' two or more variables are. Although this is not a reliable method with which to infer causality amongst variables, it can be useful in determining the underlying structure of a dataset or set of observations. Most correlation coefficients take values between –1 and +1 where –1 shows a strong negative correlation and +1 shows a strong positive correlation.

3.1.1 Correlation Metrics

There exist various methods for calculating the correlation between two variables. The most common is *Pearson's Correlation Coefficient*.

Pearson's Correlation Coefficient (PCC) [Pearson1896] measures the linear relationship between two numerical variables x and y as in equations 3.1 and 3.2.

$$\rho_{xy} = \frac{Cov(x,y)}{\sigma_x \sigma_y} \tag{3.1}$$

$$Cov(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y) \tag{3.2}$$

where ρ_{xy} is the value of PCC between x and y, N is the number of x,y pairs, σ_x σ_y are the standard deviations for x and y respectively, x_i and y_i are the *ith* instances of the variables x and y, and μ_x and μ_y are the expectations of the variables x and y. The computation time of PCC is proportional to N. Note that $-1 \le \rho_{xy} \le 1$.

Spearman's Rank Correlation (SRC) [Spearman1904] measures linear and/or non-linear relationships between two variables, either discrete or continuous, by assigning a rank to each observation. It then calculates the sums of the squares of the differences in paired ranks d_i^2 according to equation 3.3.

$$R_s = 1 - \frac{6\sum_{i=1}^{N} d_i^2}{N(N^2-1)} \tag{3.3}$$

where R_s is the value of SRC between the two variables, N is the number of pairs and each d_i is calculated by taking the difference between the ranks of each variable pair x_i and y_i. This means the computation time of SRC is proportional to $N\log_2(N)$ since sorting must be used on data that is not already ranked.

If the data to be analysed is a time series then two types of correlation functions can be defined.

Definition 3.1. The *Cross-Correlation Function* (CCF) is a standard correlation function applied to two time series variables where the data pairs are constructed by time shifting one of the variables. For example if the two series are $x(1),...,x(10)$ and $y(1),...,y(10)$ then the CCF between series x and y with a time lag of one is the correlation between $x(1),...,x(9)$ and $y(2),...,y(10)$. The general form is written as CCF(x,y,lag) which is calculated between pairs of variable, $x(t)$ and $y(t+lag)$.

Definition 3.2. The *Auto-Correlation Function* (ACF) measures how closely correlated a variable is with itself over varying time lags. Therefore ACF(x,lag) = CCF(x,x,lag).

These functions will give an indication of the variation of correlation over different time lags between time series variables. The CCF also indicates the direction of influence since CCF(x,y,lag) is different to CCF(y,x,lag). For time-series in which the lags are large, many different coefficients must be calculated. If n is the number of variables and lag is the time lag that is under consideration, then the number of possible correlations is $lag \times n^2$. There may be various real world applications where the number of possible correlations may pose a problem, for example, where the structure of a dataset would be required in real time as the data is produced, or where a dataset is so huge that it would take an unreasonable amount of time to process.

The CCF for SRC between two variables over differing time lags is calculated by shifting one variable in time is given in equation 3.4.

$$R_s(x_i,x_j,lag) = 1 - \left(\frac{6\sum_{t=1}^{T-lag} \left(rank(x_i(t)) - rank(x_j(t+lag))\right)^2}{(T-lag)((T-lag)^2-1)} \right) \tag{3.4}$$

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. A similar form exists for PCC.

3.1.2 Combining Correlations

Given a set of correlations from a number of samples from two variables, it is useful to combine these correlations into a single correlation. The procedure to do this is more complex than simply taking the average since the correlation coefficients may have come from differing sized samples. A correlation combining procedure is outlined as follows:

Fisher's z-Transformation. Fisher [Fisher1921] suggested that in order to place confidence intervals on Pearson's correlation coefficient (see section 3.1.1), a transformation is needed. This transformation is called the *Fisher's z-transformation*, but a slightly different notation will be used to avoid confusion with the standard normal distribution variable notation. The transformation is defined in equation 3.5.

$$F_z(\rho) = \frac{1}{2}\ln\left(\frac{1+\rho}{1-\rho}\right) = \mathrm{Tanh}^{-1}(\rho) \tag{3.5}$$

$$\sigma_{F_z} = \frac{1}{\sqrt{n-3}} \tag{3.6}$$

$$\rho = \frac{e^{2F_z}-1}{e^{2F_z}+1} = \mathrm{Tanh}(F_z) \tag{3.7}$$

The variable $F_z(\rho)$ is approximately normally distributed with standard error of equation 3.6. Here n is the number of pairs of observations of the variables. Confidence intervals can be computed for this transformed variable in the normal way. The inverse transformation is shown in equation 3.7. Fisher also suggested a way of combining correlations based on Fisher's z-transformation [Fisher1921]; a successful implementation was made by Lush [Lush1931]. The procedure is outlined below:

1) $\{\rho_1,...,\rho_{N_\rho}\}$ is a list of correlations to be combined.
2) For each ρ_i a corresponding $F_{(z)i}$ is computed according to equation 3.5.
3) A weighted transformed variable $W_{(z)i}$ is then computed for each $F_{(z)i}$. This is computed as follows: $W_{(z)i} = (n_i\text{-}3)F_{(z)i}$ where n_i is the number of pairs of observations of the variables used to calculate ρ_i. N_{Wz} is the sum of the weights, $N_{Wz} = \sum_{i=1}^{N_\rho}(n_i - 3)$ and $\overline{W_z}$ is the sum of weighted transformations, $\overline{W_z} = \sum_{i=1}^{N_\rho} W_{(z)i}$.
4) The combined correlation value $\hat{\rho}$ is calculated as follows: $\hat{F}_z = {\overline{W_z}}\big/{N_{Wz}}, \hat{\rho} = \frac{e^{2\hat{F}_z} - 1}{e^{2\hat{F}_z} + 1}$ (as in equation 3.7).

Theory, e.g. [Snedecor1967], states that the same transformation can be used for Spearman's rank correlation coefficient.

3.2 Methods

Given an MTS the objective of correlation mining is to search for a list of correlations Q which contains the top R (referred to as the *RankSize*) correlated variables over all possible positive integer time lags up to some maximum, *MaxLag*. Q will consist of *triples* made up of two variables and a time lag. For example, the triple $(x_1,x_2,5)$ represents a correlation between x_1 and x_2 with a time lag of 5. Essentially the triples in Q represent the pairs of variables that are significantly correlated for some time lag. Therefore, it is important to estimate what R should be with a high degree of accuracy. This is discussed further in chapter 4.

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*. The triples (x_i,x_i,*lag*) are all autocorrelations and hence these are considered invalid too, since they give no information about relationships between variables.

Proposition 3.1. The maximum number of valid correlations for an n dimensional MTS considering a time lag up to and including *MaxLag* is $n(n-1)(MaxLag + \frac{1}{2})$.

Proof. Ignoring validity, the maximum number of correlations would be one for each variable pairing for lags between [0,*MaxLag*] which is n^2(*MaxLag*+1). The number of duplicate correlations at time lag zero is equal half of the possible correlations which equals $\frac{1}{2}n(n-1)$. The number of autocorrelations is equal to the number of variables for all of the time lags which is n(*MaxLag*+1). Therefore the total number of valid correlations, denoted s, is as follows:

$$
\begin{aligned}
s &= n^2(MaxLag+1) - \frac{1}{2}n(n-1) - n(MaxLag+1) \\
&= n^2 MaxLag + n^2 - \frac{n^2}{2} + \frac{n}{2} - nMaxLag - n \\
&= n(n-1)MaxLag + \frac{1}{2}n(n-1) \\
&= n(n-1)(MaxLag + \tfrac{1}{2})
\end{aligned}
$$
■

Note that the triple (x_i,x_j,*lag*) and (x_j,x_i,*lag*) where *lag* is greater than zero are not equivalent. A correlation is considered valid if it represents a pair of variables that is not already in Q, i.e. there is no other triple of the form (x_i,x_j,*lag*) and (x_j,x_i,*lag*). This ensures that a pair of variables that are highly correlated do not dominate Q for a variety of lags, e.g. (x_1,x_2,5), (x_1,x_2,4), (x_2,x_1,3) etc. and hence only appear once. To make sure that the best lag and variable order is maintained, a side effect of validity is that the stronger correlated triple stays in Q, hence there is a possibility that an existing triple would have to be removed. The method of validity checking and removal is implemented in all of the methods.

The following sections describe the methods employed to find subsets of highly correlated points through time. All of these methods use the absolute value of the correlation coefficients, in order to rank a relationship between zero and one inclusive, (the objective was to locate dependencies but not their nature).

Four methods are implemented: these are *Random Bag* (RB), *Genetic Algorithm* (GA) (see section 2.3.1), *Evolutionary Programming* (EP) (see section 2.3.2) and an *EXhaustive search* (EX). It has proved hard to find any existing methods that do not rely on the data being categorical, for example [Steeg1998]. Note that the standard statistical solution would be to explore the whole search space, sample the time-series, or restrict the search space through the use of expert knowledge; all of which are inappropriate for the application which the methods are evaluated on.

3.2.1 The Exhaustive Search

Algorithm 3.1: The Exhaustive Search

```
1)   Input:  X – a T×n MTS
             MAXLAG – the maximum lag with which to time shift the MTS by
             R – the required number of correlations
2)   Set Q = Empty List
3)   For i = 0 to n-1
4)        For j= 0 to n-1
5)             For lag = 0 to MAXLAG
6)             If the triple ($x_i$, $x_j$,lag) is valid then
7)                  Insert a new triple ($x_i$,$x_j$,lag) into Q
8)                  Sort Q in descending order of correlation calculated from X
9)                  If size of Q = R+1 then remove a triple from the tail of Q
10)                 End If
11)            End For
12)       End For
13)  End For
14)  Output: Q of length R
```

This method was performed on the visual field dataset using Pearson's and Spearman's rank coefficients, detailed in section 3.1.1. Although in practice datasets could be of sufficiently large dimensionality and time-series length to preclude such a search, this method was implemented regardless of overheads so that the results could be used as a benchmark for the other methods. The exhaustive search consisted of simply exploring all of the variables, at each time lag and is described formally in algorithm 3.1.

3.2.2 The Random Bag

This is a heuristic approach whereby a random selection of triples is placed in a *bag* containing *RankSize* triples. With each iteration a new random triple is added to the bag. When the bag overflows, the worst correlations fall out. This is repeated for a predefined number of iterations. The procedure is described in algorithm 3.2.

Definition 3.3. The function *UI*(MIN,MAX) is a uniformly distributed random number generator that returns an *integer* number in the interval [MIN,MAX].

Algorithm 3.2: The Random Bag

```
1)      Input:  X – a T×n MTS
                MAXLAG – the maximum lag with which to time shift the MTS by
                R – the required number of correlations
                c – the number of calls to the correlation function
2)      Set Q = Empty List
3)      Repeat c times Do
4)              Set i = UI(0,n-1), Set j = UI(0,n-1), Set lag = UI(0,MaxLag) where ($x_i$,$x_j$,lag)
                is valid
5)              Set a = new triple ($x_i$,$x_j$,lag)
6)              If a$\notin$ Q then insert a into Q
7)              Sort Q in descending order of correlation calculated from X
8)              If size of Q = R+1 then remove a triple from the tail of Q
9)      End Loop
10)     Output: Q of length R
```

The RB method is essentially the selection of the best *R* triples from a non-overlapping random sample of size *c* from the *s* possible correlations. Note that *c* is the maximum number of allowed calls to the correlation function. Note also that a triple is valid if it does not warrant removal. The final contents of the *Bag* represent the solution, i.e. the required *RankSize* correlations.

3.2.3 Genetic Algorithm

Recall that a genetic algorithm is a method for search based on the mechanics of natural selection and genetics. A population of chromosomes that represent possible solutions is used to explore the search space. This is achieved by updating the population with the creation of new chromosomes, formed through the recombination (using an operator called crossover) of others and a small perturbation analogous to mutation and the destruction of less fit chromosomes.

For the correlation problem, a chromosome was considered to consist of a number of genes corresponding to the required correlation *RankSize*. Each gene consisted of a correlation triple. Uniform crossover was used and was not allowed to split any gene, whereas mutation could affect part of a triple. The GA specific parameters were selected through experimentation for optimal performance. These are listed in table 3.1. Within this table *#Genes* is the chromosome length, i.e. the number of genes.

Parameter	**Value**
MutationRate (%)	0.5
CrossoverRate (%)	100
Population	10
Generations	~21
#Genes	100 (=*Ranksize* =*R*)
Survival	The roulette wheel

Table 3.1 GA Parameters

Algorithm 3.3: The Correlation Genetic Algorithm

```
1)  Input:  Fitness function for a chromosome
            POPULATIONSIZE – the number of chromosomes in each population
            #GENES – the number of genes making up each chromosome
            GENERATIONS – How many iterations to run the Genetic Algorithm for
            CROSSOVERRATE – The chance of a chromosome becoming a Parent
            MUTATIONRATE – the chance each chromosomes gene has of Mutating
2)  Generate POPULATIONSIZE Valid chromosomes of size #GENES bits
3)  For loop = 1 to GENERATIONS
4)          Crossover the Population to Create Children
5)          Mutate the Population
6)          Remove Invalid chromosomes
7)          Apply Survival to the Population
8)  End For
9)  Output: The fittest individual of the last population
```

Algorithm 3.3 describes the correlation genetic algorithm. Each chromosome is represented as a string of integer numbers, the length being equal to three times the *RankSize*. This corresponds to a correlation being represented as a triple, i.e. *(x,y,lag)*. Since the size (range) of the lag is usually different to the size of the variables *x* and *y* (size is the number of variables *n*), uniform crossover was restricted to be in multiples of three.

Mutation is defined according to equation 3.8.

$$Gene_i = \mathrm{UI}(1, \mathrm{Limit}(i)), \text{if } Gene_i \text{ is to mutate}$$
$$\mathrm{Limit}(i) = \begin{cases} MaxLag & , i \bmod 3 = 0 \\ n & , \text{otherwise} \end{cases} \tag{3.8}$$

where $Gene_i$ is the *i*th Gene (where $1 \leq i \leq 3(\#Genes)$) of the chromosome.

3.2.4 Evolutionary Programming

Evolutionary Programming is based on a similar paradigm to genetic algorithms. However, the emphasis is on mutation and it does not use any recombination. The basic procedure is outlined in algorithm 3.4, and is discussed in section 2.3.2.

Algorithm 3.4: The Correlation Evolutionary Program

```
1)      Input:  X – a T×n MTS
                R – the required number of correlations
                MAXLAG – the maximum lag with which to time shift the MTS by
                c – the number of calls to the correlation function
2)      Set Q = Empty List
3)      Generate R random triples and insert into Q
4)      Set CallCount = R
5)      While CallCount < c
6)              Set Children to Q
7)              Apply Mutate operator to Children
8)              Insert valid Children into Q
9)              Update CallCount by the number of valid Children
10)             Apply Survival operator to Q
11)     End While
12)     Output: Q of length R
```

A child will be considered invalid if it is already in *Q* or fails the validity rules described earlier in this section. The best out of the final population will be the best solution to the problem. Traditionally, EP algorithms use tournament selection (see section 2.3.1) during the survival of the fittest stage. However, it was decided that the entire population would be the solution for the 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 (*PopulationSize*=*R*). Hence the survival operator consisted of keeping the best *R* individuals. This therefore required a check for any duplicates after mutation, and for any invalid chromosomes. Any children that fell into this category were repeatedly mutated until they became valid. 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 correlation coefficient that it represents. This in turn would maximise the population's fitness by improving the set of correlations represented by the population.

Algorithm 3.4 used uses the idea of *Self-Adapting Parameters* [Bäck1996] as part of the mutation phase of the method**.** Here each gene (x_i) in each chromosome is given a parameter, σ_i. Mutation is defined as follows (equations 3.9 to 3.12):

$$x_i' = x_i + N(0,\sigma_i) \qquad (3.9)$$

$$\sigma_i' = \sigma_i e^{\psi+\psi_i} \qquad (3.10)$$

$$\psi \sim N(0,\frac{1}{\sqrt{2N_c}}) \qquad (3.11)$$

$$\psi_i \sim N(0,\frac{1}{\sqrt{2\sqrt{N_c}}}) \qquad (3.12)$$

Note that ψ is constant for each gene in each chromosome but different between chromosomes, and ψ_i is different for all genes. Both parameters are generated each time mutation occurs. Initial examination of the performance of the RB method found that the performance was better than the GA. Similarities were drawn between this basic method and the RB algorithm. The major difference was, rather than adding a new random chromosome to the population, an existing member of the population is copied and mutated in a controlled manner. Each chromosome consisted of three genes and their corresponding σ values. The value of N_c is the size (number of genes) of each chromosome, i.e. three.

Each gene within a chromosome is mutated according to the normal distribution with mean zero and standard deviation equal to the gene's corresponding σ value (equation 3.9). The σ values, themselves, are mutated according to equation 3.10. Initials values for these standard deviations are shown in table 3.2.

σ_i	**Starting Value**
Variable 1	3.5
Variable 2	3.5
Lag	1.0

Table 3.2: σ_i Starting Value

3.3 Results

The dataset used for evaluation is the visual field dataset as described in section 2.2.2. Within this dataset, for each patient's visual field data, correlations will be calculated using the 76 visual field points at a time lag of up to five, approximately 30 months. This would result in a total of 31,350 correlations. All of the 82 available patients are used in this chapter to create a dataset that is of significant complexity (a total of 2,570,700 possible correlations). The combined correlations for all of the patients were "averaged" (this averaging is described in section 3.1.2) to get a single value for two visual field points at a given time lag. This is so that the general dependencies over a representative population can be compared with the medical literature [Crabb1996/97] and [Heijl1988/89a] for verification.

For the visual field dataset, each of the different methods was run until the number of calls to the correlation function equalled that of the exhaustive search. With the GA, RB and EP methods, this meant setting an artificially high number of generations. In practice this would be pointless. A *RankSize* of 100 was chosen for each of these experiments since it is large enough to show if the relationships between the 76 points correspond to the same *nerve fibre bundles* (see chapter 2.2.2).

Figures 3.1 to 3.4 show the results from these experiments. Figures 3.1 is for Pearson's correlation coefficient, and figure 3.2 displays the first third of the search space (a third of the number of calls) to demonstrate the area of interest. Figures 3.3 and 3.4 show the same information but for Spearman's rank correlation coefficient.

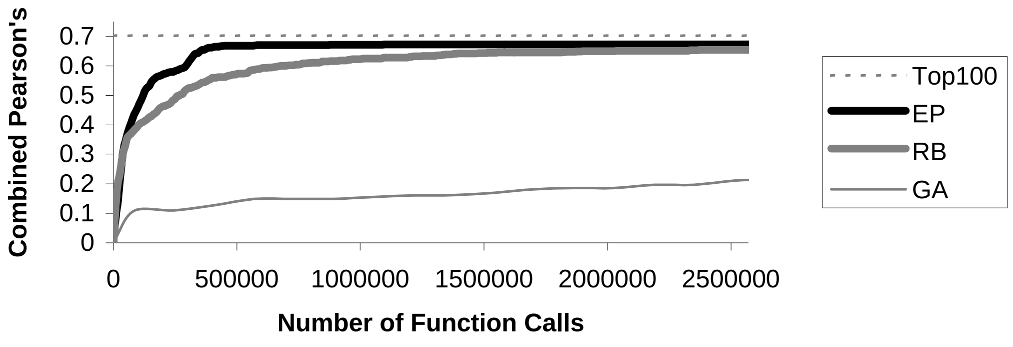


Figure 3.1: Pearson's Correlation Coefficient for the VF Data (100%)

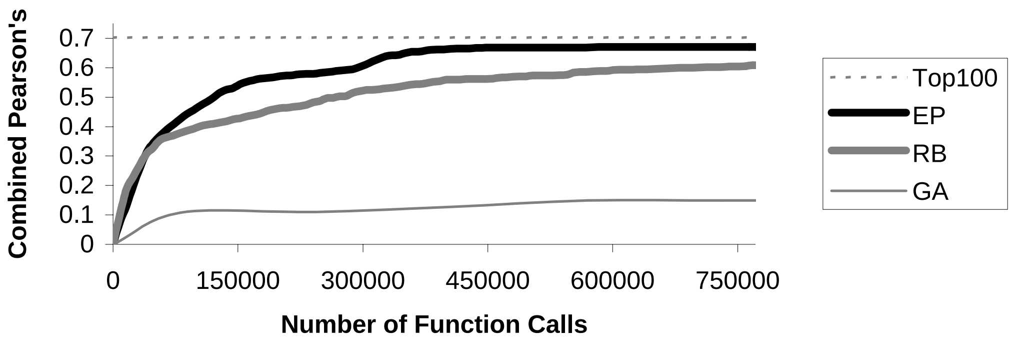


Figure 3.2: Pearson's Correlation Coefficient for the VF Data (30%)

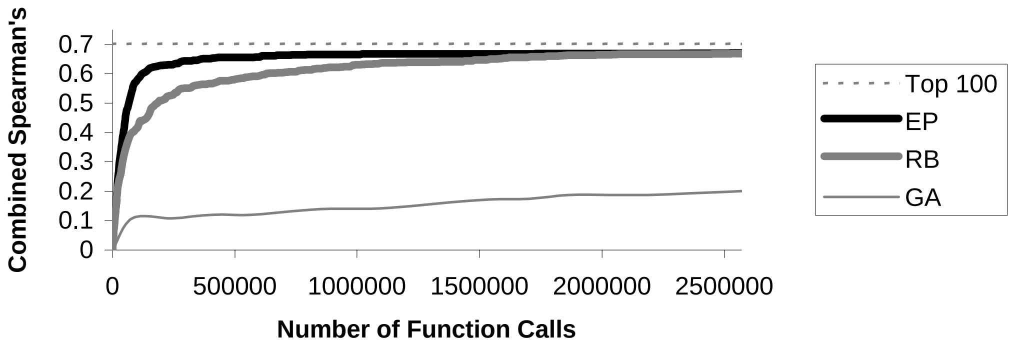


Figure 3.3: Spearman’s Correlation Coefficient for the VF Data (100%)

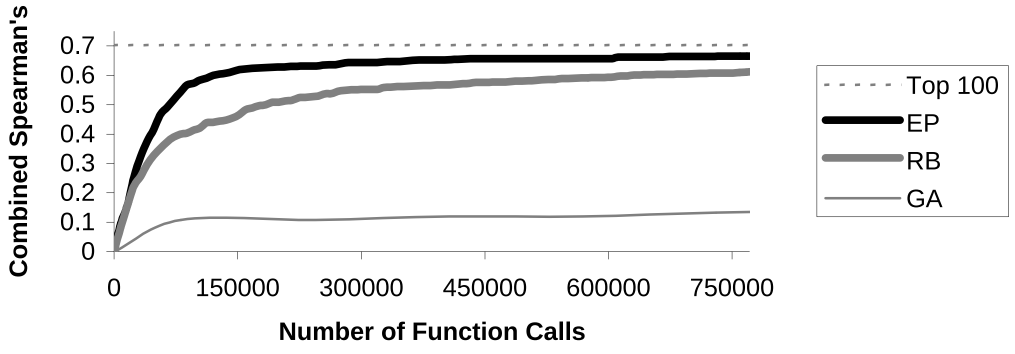


Figure 3.4: Spearman’s Correlation Coefficient for the VF Data (30%)

Within figure 3.1 and figure 3.3 it can be seen that the EP method converges nearer to this maximum during the first 500,000 function calls before slowing. However, it can be seen, in both sets of results, that the EP method performs consistently better than the other methods. The RB method does the next best, converging at a slower rate than the EP method; this is highlighted in figures 3.2 and 3.4. However the GA method performs very poorly and still seems to be slowly converging. Again, the result for the GA was the best produced from a number of experiments using differing parameter values. The Pearson and Spearman results are quite similar showing the performance of the method is independent of the correlation metrics.

Next, an analysis was made of each method after approximately five percent of the total number of correlations was called. Table 3.3 shows the analysis of the best correlations found. Within this table, *PCC* refers to results generated by using Pearson’s correlation coefficient; *SRC* to results generated using Spearman’s rank correlation coefficient; *EX* is the exhaustive search results; *TOP100* is the best 100 correlations from the exhaustive search results; *RB* is the Random Bag results; *EP* is the evolutionary programming results; *GA* is the genetic algorithm results; *#Calls* is the number of calls to the relevant correlation coefficient; *%Calls* is the number of calls as a percentage of the total number of possible correlations (i.e. the whole search space); *Max.* is the maximum correlation; *Min.* is the minimum correlation; *Mean* is the arithmetic average; *Median* is the median value and *StDev.* is the standard deviation for the correlations.

The exhaustive method shows the overall best and worst correlation in the entire possible 2,570,700 as would be expected. The number of calls for methods not based around the exhaustive method is approximately five percent of the whole search space. The EP method has the best average for both Pearson's and Spearman's correlation coefficient, and also a better median. The standard deviations are all of a similar magnitude, indicating that the distributions of the found correlations are similar in variability.

Method	**# Calls**	**% Calls**	**Min.**	**Max.**	**Mean**	**Median**	**StDev.**
PCC							
EX	2,570,700	100.000	0.000	0.873	0.069	0.039	0.000
TOP100	2,570,700	100.000	0.636	0.873	0.702	0.682	0.064
EP	136,566	5.312	0.412	0.810	0.516	0.505	0.090
RB	136,242	5.300	0.296	0.731	0.418	0.374	0.114
GA	244,215	4.750	0.000	0.535	0.110	0.052	0.135
SRC							
EX	2,570,700	100.000	0.000	0.883	0.071	0.043	0.000
TOP100	2,570,700	100.000	0.625	0.883	0.702	0.685	0.069
EP	136,161	5.297	0.505	0.869	0.607	0.580	0.090
RB	135,837	5.284	0.315	0.799	0.447	0.425	0.119
GA	244,782	4.761	0.001	0.517	0.100	0.047	0.131

Table 3.3: Summary Statistics for All Methods for Both Correlation Coefficients

Figures 3.5 and 3.6 show two histograms, one for each correlation coefficient for the whole exhaustive search. In these histograms, the number of correlations that falls into a given interval is given. In both cases it can be seen that there are many low value correlations, and as the value of the correlation increases, the corresponding frequency decreases, as would be expected.

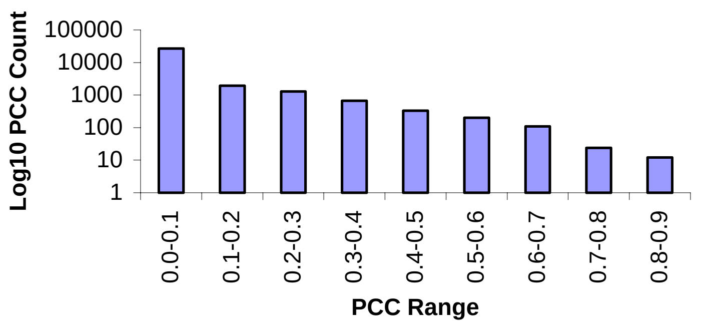


Figure 3.5: PCC Correlation Range Histogram

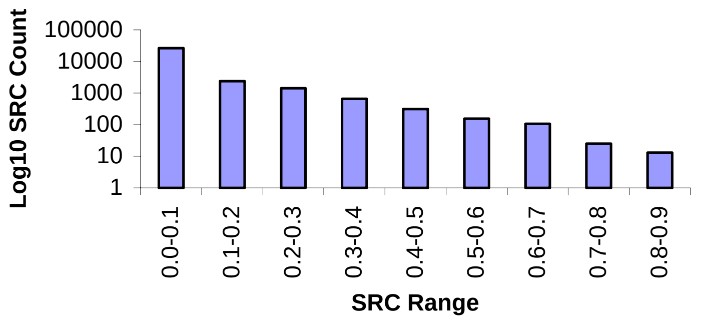


Figure 3.6: SRC Correlation Range Histogram

To summarise the results, the EP method behaved better than the RB method and the GA in both datasets, using both correlation coefficients; but only for cases where the number of calls is less than half the search space. It seemed that there was a larger difference in performance between EP and the others in the VF data. This is probably due to the fact that the numerical encoding of the variables map mathematically to spatial points upon the eye's retina, as described in section 2.2.2. A mutation would result in a form of rotation or translation of co-ordinates. The self-adapting standard deviations would adjust to a level where certain transformations would result in many useful and high correlations, perhaps within the same nerve fibre bundle.

The GA performed the worst, particularly when the number of generations was small, because the crossover operator merely mixes correlation genes around. The crossover operator is additionally designed to carry forward good schema, which cannot exist within this context (i.e. all genes are independent and hence exhibit low epistasis).

The GA could have been designed as in the EP method, with an individual representing a single correlation and the population representing the solution. However, unless a binary representation was used, the chromosome size would have been too small to fully exploit crossover. The binary representation would have been approximately 15 bits in the case of the VF data. This is still very small, and some elementary experimentation has verified this.

3.5 Discussion

Figure 3.7 displays three graphs of the 76 points of the visual field data. Each graph includes the correlation results from running the EP method for 5% of the whole search space, corresponding to the experiments displayed in table 3.3.

The first diagram in figure 3.7 displays the best 34 correlations found, the second graph, the next 33, and the last graph: the remaining 33.

A line is used to represent a correlation between two variables, and its colour is an indication of strength: the darker the line is, the higher the correlation.

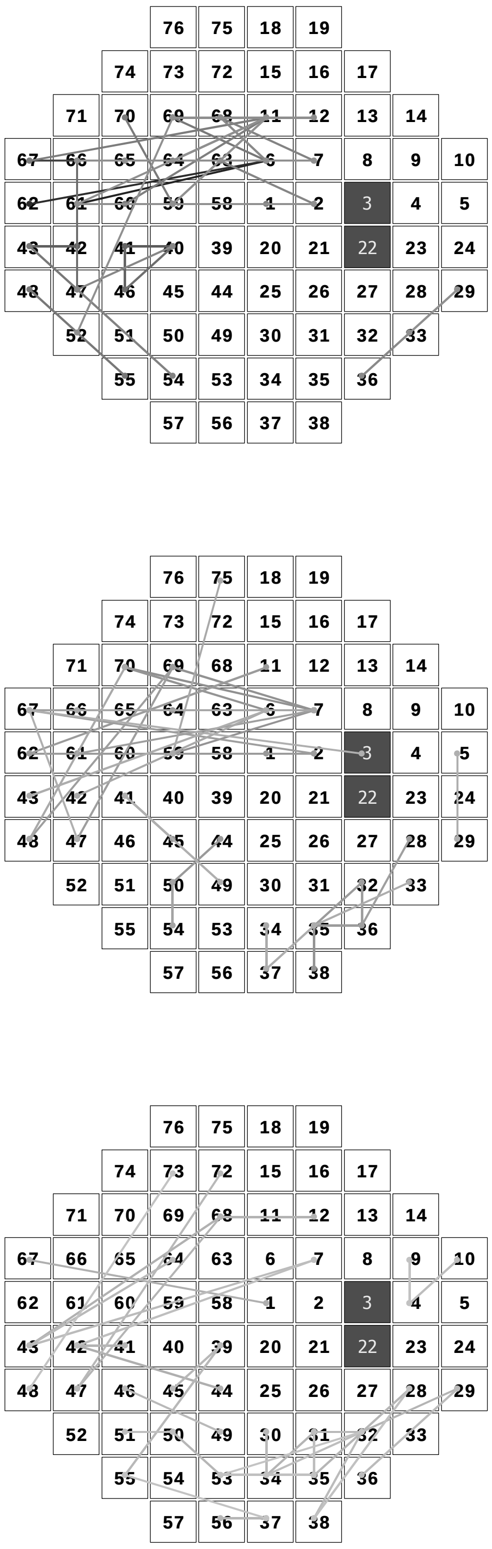


Figure 3.7: Correlation Results

It can be seen that the majority of the correlations correspond well with the medical literature, which indicates that deterioration often originates "*out of the blind spot*" [Hollwich1985] and then affects the periphery of the eye, moving counter clockwise. Many of the relationships are between points that lie on the same nerve fibre bundle, or with points that lie on adjacent bundles, again corresponding with what an ophthalmic clinician would expect.

One correlation that is quite strong suggests a relationship between point 3 (within the blind spot) and point 67. This can be explained since not all of the retina covered by points 3 and 22 correspond to the blind spot, and any visual field reading involving these points are likely to be noisy, thus this relationship could be pure coincidence.

3.6 Concluding Remarks

Within this chapter several methods for quickly learning the correlation structure of a large dataset have been explored. These methods have been applied to an important real world dataset that exhibit properties where this type of analysis is useful. That is, it is a large multivariate time-series where the fast identification of the approximate correlation structure is needed. The results show that the EP method is by far the quickest to converge to a high average correlation. The self-adapting parameters appear ideally suited to finding meaningful clusters of correlations; however it still remains to be investigated where this method is appropriate, and where it falls down. It is suggested that the EP method will perform no better than that of the RB method if there are no *patterns* within the underlying correlations of the dataset being explored, e.g. the spatial arrangement of the nerve fibre bundles within the visual field data.

These correlations will be shown to be highly useful in chapter 4 where they will be used as the basis for a variable grouping strategy that decomposes a high dimensional time series into a number of smaller dimensional series. Finally, consideration could be made of using them as seeds to a genetic algorithm based method that produces statistical time series models suited for short term forecasting.

4. Decomposing High Dimensional MTS

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 partitions. The term *grouping* will be used to refer to this type of problem [Falkenauer1998].

Examples of grouping applications include bin packing, workshop layout design, and graph colouring. Much research has been done on the grouping problem in different fields. It has been established that many, if not all grouping problems, are NP-hard [Garey1979]. Therefore 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.

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 very large because one variable could affect another after a certain time lag. Therefore effectively utilising these dependencies becomes an important issue: using all the possible dependencies in a variable grouping algorithm will be computationally impractical for many, especially real-time, applications.

This chapter is about a systematic study and application of the *variable grouping* problem in MTS. In particular, different methods of utilising the information regarding correlations among MTS variables are investigated, this information being provided by the methods presented in chapter 3. Five grouping methods to be introduced in this chapter and three correlations methods will be applied to six datasets where there are identifiable mixed groupings of MTS variables. This chapter describes the general methodology, reports extensive experimental results and concludes with useful insights on the strengths and weaknesses of this type of grouping method. This chapter is based upon work outlined in [Swift2001] and [Tucker2001b].

4.1 Grouping Problems and Multivariate Time Series

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 methods such as the Vector AutoRegressive (VAR), Vector AutoRegressive Moving Average, and other non-linear and Bayesian systems [Lütkepohl1993]. Take the VAR(p) process as an example. There would be at least $n \times n \times p$ parameters to locate where p is the order of the VAR process and n is the number of variables in the dataset. In explaining MTS, suppose the aim is to learn Dynamic Bayesian Network (DBN) models [Friedman1998] 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 [Tucker2001a].

The decomposition of the MTS into smaller dimensional MTS that are independent to some degree significantly reduces the search space, thereby allowing the speedier production of MTS models. Therefore the aim is to decompose a high-dimensional MTS into groups of smaller MTS, where the dependency between variables within the same group is high, but very low with variables in another group. This type of method does not appear to have been studied before. In [Mandava1993] visual field variables are clustered but time delays are ignored. In [Oates1996] categorical data is segmented along the time axis and these segments are clustered. In [Goutte1999] real valued MTS are clustered but expert knowledge is used to reduce the search space. Note that this is different from what the *dimensionality reduction* techniques such as principal component analysis or factor analysis try to achieve by making some sort of multivariate transformation of the data [Pena1987].

In the previous chapter, the result of the correlation pre-processing resulted in a list of correlations, denoted *Q*. Here each element of *Q* consists of a correlation triple (x_i,x_j,*lag*). In this chapter the grouping algorithms will be 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, but is maintained for use later on. This is because the purpose at this stage is to group highly correlated variables together, irrespective of the time lag between them; the lag helps in determining the strength of the relationship, which is no longer needed once identified.

This chapter is arranged as follows. After outlining the basic notation a grouping metric is then 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. Finally the results of applying correlation search and grouping are presented and considered.

4.1.1 Preliminaries

Given a multivariate time series with *n* variables of length *T*, the objective is to partition the list of variables $\{x_1,..,x_n\}$ into *m* groups. The size of the *i*th 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*.

4.1.2 The Grouping Metric

The *Grouping Metric* is used to score a group. It assigns higher scores to groups that contain variables which appear in a triple in *Q*. Let *G* be a list of groups $\{g_1,..,g_m\}$ that partitions the variables $\{x_1,..,x_n\}$, where $|g_i| = k_i$. That is $\sum_{i=1}^{m} k_i = n$,

$g_i \cap g_j = \phi, \forall i \neq j$ and $\bigcup_{i=1}^{k} g_i = \{1,..,n\}$ where $k_i \geq 1$. The notation g_{ij} refers to the jth element of the ith list of G. It is clear to see that in all cases $m \leq n$.

The *grouping metric* $f(G)$ for any fixed list of groups is defined as in equations 4.1 to 4.3, where $corr(x_i, x_j)$ returns true if there exists in Q a triple of the form (x_i, x_j, lag) or (x_j, x_i, lag) for any valid lag. Hence $corr(x_i,x_j)=corr(x_j,x_i)$. The set Q contains only unique correlations.

$$f(G) = \sum_{i=1}^{m} h(g_i) \tag{4.1}$$

$$h(g_i) = \begin{cases} \sum_{a=1}^{k_i-1} \sum_{b=a+1}^{k_i} L(g_{ia}, g_{ib}) & ,\text{if } k_i > 1 \\ 0 & ,\text{otherwise} \end{cases} \tag{4.2}$$

$$L(g_{ia}, g_{ib}) = \begin{cases} 1 & ,\text{if corr}(g_{ia}, g_{ib}) \\ -1 & ,\text{otherwise} \end{cases} \tag{4.3}$$

The metric has the following characteristics, the proofs of which can be found in appendix F:

i) If there are no correlations ($Q=\varphi$), the maximum value is obtained when all variables are in separate groups.

ii) If a correlation exists for each pairing of variables (the whole search space, i.e. $|Q| = \frac{1}{2}n(n-1)$), then the maximum fitness is obtained when all of the variables are in one group.

iii) If the data generating the correlations came from a mixed set of multivariate time series observations, then the metric will be maximised when the variables within the same group have as many correlations within the set Q as possible and variables within differing groups contain as few correlations as possible.

A suitable correlation function has been chosen that is a well-established correlation coefficient: *Spearman's Rank Correlation*, see section 3.1.1. It should be noted that the methods are in no way restricted to using this particular coefficient and others such as Pearson's could have easily been used. Spearman's Rank was chosen, as it is well recognised and not restricted to finding linear dependencies.

4.2 The Grouping Search

This section describes the methods employed to group the results of the correlation searching routines according to the metric outlined above. Three genetic algorithm based methods are presented, along with a *Hill-Climbing* and heuristic clustering method.

4.2.1 The Genetic Algorithms

The general algorithm for generating a list of groups, *G*, from a set of correlations, *Q*, is given in algorithm 4.1.

Algorithm 4.1: The Genetic Algorithm

```
1)      Input:  Q – The Correlation list
                Fitness function for a chromosome – the Grouping Metric applied to Q
                POPULATION – the number of chromosomes per population
                GENERATIONS – the number of genetic iterations
                CROSSOVERRATE – the percentage of chromosomes allowed to breed
                MUTATIONRATE – the chance a gene of a chromosome has of mutating
2)      Generate POPULATIONSIZE chromosomes
3)      For loop = 1 to GENERATIONS
4)              Crossover the Population to Create Children
5)              Mutate the Population
6)              Apply Survival of the Fittest to the Population
7)      End For
8)      Output: G, groups constructed from the fittest individual from the final population
```

Three different representations of a partition are used, each having their own specialised crossover and mutation operator. For the scope of this thesis, the fitness function for the methods will be the grouping metric defined in equation 4.1.

Gene Per Variable (GPV). This representation consists of a chromosome with each gene representing the index of a variable in the domain. The value of the gene determines which group the variable is a member of. For example the partition $\{\{x_0, x_3, x_8\}, \{x_2, x_7, x_4, x_1, x_5\}, \{x_6, x_9\}\}$ would be represented as follows:

$$\left.\begin{array}{l}\text{Group } 0: 0\,3\,8 \\ \text{Group } 1: 2\,7\,4\,1\,5 \\ \text{Group } 2: 6\,9\end{array}\right\} \Rightarrow 0\,1\,1\,0\,1\,1\,2\,1\,0\,2$$

The crossover operator used for this representation is Holland's standard one point crossover and the mutation operator involves randomly mutating genes within the chromosome according to the mutation rate. Each gene has mutation rate probability of being mutated to a value from a uniform distribution UI(0,*n*-1).

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, variable indices between two □s or a □ and the beginning or end of a chromosome belong to the same group.

This representation requires a new crossover operator in order to ensure that invalid children are not produced. It can be seen that standard crossover as used in the GPV representation would produce many invalid children, as it would be highly likely to result in children with variables appearing in more than one group. Goldberg introduced the PMX operator [Goldberg1985], which prevented this and developed an o-schema theory (closely linked to Holland's original schema theory). It ensures all children are valid (i.e. it is a closed operator) and works as follows:

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

For example, Parent 1: 4 $\square_3$ 0 $\square_1$ 1 6 5 2 $\square_2$ 3 and Parent 2: 5 4 $\square_2$ 2 3 $\square_3$ 0 1 $\square_1$ 6

i) Crossing points = 3 and 5
ii) Swap elements
iii) "$\square_1$ 1 6" with "2 3 $\square_3$"
iv) Giving 4 $\square_3$ 0 2 3 $\square_3$ 5 2 $\square_2$ 3 and 5 4 $\square_2$ $\square_1$ 1 6 0 1 $\square_1$ 6
v) Replace repeated values; 2 with $\square_1$, 3 with 1 and $\square_3$ with 6
vi) Giving 4 6 0 2 3 $\square_3$ 5 $\square_1$ $\square_2$ 1 and 5 4 $\square_2$ $\square_1$ 1 6 0 3 2 $\square_3$

Mutation involves randomly mutating genes within the chromosome according to the mutation rate. Each gene has mutation rate probability of being swapped with another random gene.

Falkenauer's Grouping Genetic Algorithm (GGA). This representation is similar to the GPV except that it also has an extra part on the chromosome. For the GGA, the representation of a chromosome consists of two parts. In the first part, each gene represents a variable in the domain. The value of the gene determines which group the variable is a member of. In the second part, each gene represents the actual groups without any information about their contents. Hence, the chromosome is of variable length since the number of groups can vary. For example, 8 variables to be placed into the following 3 groups:

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

This would be represented by the following chromosome: 01100212:012. It is the second part of the chromosome (after the colon) that crossover is applied to. Crossover works as follows:

i) Select two random crossing sites, delimiting the crossing sites in each of the two parents, denoted as: [Parent start position, Parent end position].
ii) Inject the contents of the crossing section of the first parent at the first crossing site of the second parent.
iii) Remove any elements that conflict with the groups that were members of the first parent.
iv) Remove any empty groups and reinsert any unassigned variables to existing groups.
v) Repeat (i) to (iv) to produce the second child by reversing the roles of the first and second parent.

Example for first child:	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

i) The crossing sites for Parent 1 = [0,1] and for Parent 2 = [1,3]

ii)	Inject group 0 into position 1:	0 ? ? 0 0 ? ? ? : 3 0 4 5 6
iii)	Remove group 4 and 5 due to conflicts:	0 ? 3 0 0 6 3 6 : 3 0 6
iv)	Reinsert variable 1 into random group (6):	0 6 3 0 0 6 3 6 : 3 0 6

A '?' denotes an unallocated variable.

4.2.2 Hill-Climbing

Hill-climbing [Michalewicz1998, Russell1995] is essentially an iterative search where the value of the solution can only increase or stay the same at each step. The version of hill-climb within this chapter involves using the GPV representation, making simple changes to the current groupings at each iteration; a random variable index is either moved into another existing group or into a new group. If this change improves the score of the partition, it is retained. The hill-climbing algorithm is described in algorithm 4.2.

Algorithm 4.2: The Hill-Climb

```
1)  Input:  Q – the Correlation list
            Score – the Grouping Metric applied to Q given a chromosome
            Iterations – the number of times the Hill-Climb is iterated
2)  Generate a random selection of groupings G (i.e. a single chromosome using the GPV
    representation)
3)  Set Score according to the Grouping Metric applied to Q given the grouping
4)  For i = 1 to Iterations do
5)          Make a random change to G (move a variable to an existing or new group)
6)          Set New_Score according to the Grouping Metric applied to Q given G
7)          If New_Score < Score Then undo changes
8)  Next i
9)  Output: G (a list of groups)
```

4.2.3 Separate and Conquer

This method is based on the clustering technique of Separate and Conquer [Mirkin1999]. The algorithm has been amended so that it clusters on the relationships between variables rather than on the value of variables. The procedure is described in algorithm 4.3 and uses equation 4.2 to calculate $h(\mathrm{g}_i)$. 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.

Algorithm 4.3: Separate and Conquer

```
1)  Input:   Q – the Correlation list
2)  Let G be a list of Groups (empty), Let X be a list of variables {1,..,n}, Let m=1
3)  Create a group $g_1$ containing the best correlation pair in Q and add $g_1$ to G
4)  For i = 1 to n
5)      Set skip=false and Set j = 1
6)      While j < m+1 and skip=false
7)          If $x_i \notin g_j$ then
8)              Set $g_j' = g_j \cup \{x_i\}$
9)              If $h(g_j') > h(g_j)$ then Set $g_j = g_j \cup \{x_i\}$ and Set skip=true
10)             End If
11)         End if
12)         j= j+1
13)     End While
14)     If skip =false then
15)         Set $g^* = \{x_i\}$ and Set $G = G \cup g^*$
16)         Set m=m+1
17)     End If
18) Next i
19) Output: G (a list of groups)
```

4.3 Parameter Estimation

In order to retrieve groupings that correspond closely to the correlations that represent actual dependencies, the ideal set of parameters for the correlation mining algorithm will have to be determined. The most important of these is *R*, the size of *Q*. This will determine the cut off point for significant correlations, it will affect the overall algorithm a great deal. For example, if *R* is too large there will be too few significant correlations resulting in smaller groups; if *R* is too small there will be too many significant correlations and so groups will be combined into larger ones due to the inclusion of low correlated variables in the list. It was decided to try and determine the parameters through simulations of the Random Bag method described in chapter 3. Random Bag was chosen since it is the simplest to model. It is the weakest of the three methods for correlation mining and so by coming up with confidence intervals for selecting all the true correlations for this method should mean 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 the previous work that this chapter builds on [Swift1999b], and through the experiments within this chapter. Simulations were used to generate probability distributions of selecting correlations that represent actual dependencies. These distributions could in turn be used to determine confidence limits for the correlation list size and the number of calls to the correlation function.

4.3.1 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 of Q (R), the size of the total search space (s) and the number of calls to the correlation function (c) to particular plausible instantiations, then simulating the act of randomly selecting a correlation from the search space, and finally 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). Therefore, approximations of the distributions associated with the probability of picking a *true* dependency can be generated. The number of these *true* dependencies will be referred to as r, which is always less than or equal to R. These distributions were then tested for normality using the Lilliefors' test (see section 4.3.2). 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.

Algorithm 4.4 describes this procedure. 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. This was repeated for N_{sims} different values of R, s and c.

Algorithm 4.4: Simulation of the Random Bag

```
1)  Input:  R, r, s, c and SimulationSize
2)  Set dependencies = r randomly selected correlations, Let Distribution be a zero array of
    length R
3)  For i = 1 to SimulationSize
4)      Set count = 0
5)      For j = 1 to c
6)          For k = 1 to R
7)              Set q to a random correlation
8)              If q is in dependencies Then count = count + 1
10)         Next k
11)     Next j
12)     Set $Distribution_{count}$ = $Distribution_{count}$ + 1
13) Next i
14) Output: Distribution
```

4.3.2 Lilliefors' Test

Lilliefors' test [Lilliefors1967] is a simple test for normality that can be performed on a known distribution function. The simulations performed in section 4.3.1 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. The test is as follows: given v observations, a metric D_{max} is computed by equation 4.4.

$$D_{\max} = MAX\left|F^*(r) - S_{R+1}(r)\right| \tag{4.4}$$

where $S_{R+1}(r)$ 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. Within the simulations, these two summary statistics can be computed directly from the data. If the value of D_{max} exceeds a critical value supplied by Lilliefors in his paper, one rejects the hypothesis that the observations closely follow the normal distribution.

For the purposes within this chapter the 99% confidence limit is selected, which requires D_{max} is less than or equal to $\frac{1.031}{\sqrt{R+1}}$.

A sample of tests on the 150 simulations can be found in appendix G. From the results of these tests it can be assumed 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.

4.3.3 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 means 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 the process itself does not have a very easy representation for the probability distribution, the algebraic representation for the mean and standard deviation is likely to be difficult to derive. 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. It will be assumed 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 will be used for this [Koza1992].

The functions which determine μ and σ, are denoted $\mu(R,c,s)$ and $\sigma(R,c,s)$ respectively. These functions are assumed to be constructed in terms of the operators +,-,×,/, the terminal symbols R, c, s, along with the integers zero to nine. The exact form of $\mu(R,c,s)$ and $\sigma(R,c,s)$ 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 fitness of a given tree will be the difference between the observed values of μ and/or σ and the value calculated from the simulation data; (using the equation formed from the tree and all of the available data). The fitness function is defined in equations 4.5 and 4.6.

$$\text{Fitness for } \mu(R,c,s) = -Nodes(\mu(R,c,s)) \cdot \sum_{i=1}^{N_{sims}} \left(\mu(R_i,c_i,s_i) - \mu_i\right)^2 \tag{4.5}$$

$$\text{Fitness for } \sigma(R,c,s) = -Nodes(\sigma(R,c,s)) \cdot \sum_{i=1}^{N_{sims}} \left(\sigma(R_i,c_i,s_i) - \sigma_i\right)^2 \tag{4.6}$$

where $Nodes(\bullet)$ represents the number of nodes in the binary tree corresponding to the function in question, and i indexes a variable from the table of simulated examples (where there are a total of N_{sims} examples). As with a GA, the initial population will be a certain number of random binary trees of the form 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. By maximising the fitness functions, the genetic program is encouraged to look for smaller trees.

4.4 Experimental Methods

Section 4.4.1 describes the synthetic dataset. The results of parameter estimation are described in section 4.4.2. A Weighted-Kappa metric for evaluating the discovered groupings is introduced in section 4.4.3, followed in section 4.4.4 by the parameters used in the GA based experiments. There are three correlation search methods and five grouping strategies to test making a total of 15 different combinations of methods. These 15 strategies are applied to six synthetic datasets in which there are identifiable mixed groupings of MTS variables. For each experiment the following is recorded:

i) The grouping metric of the best solution after a varying number of calls to the fitness function for various datasets. This is a measure of how well the groupings represent the correlations that were discovered during the correlation search.

ii) The score as calculated by the Weighted-Kappa metric described in appendix J, 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 each method.

iii) The average number of function calls to find the solution; this can be thought of as a measure of efficiency.

All stochastic grouping algorithms (all methods except Separate and Conquer) were repeated ten times and the average recorded in order to remove any sampling bias. The correlation mining methods produce *R* correlations whilst *r* are needed. *Q* is therefore truncated from size *R* to size *r* so that the best *r* out of *Q* are retained. Note that the *invalidity rule* described in chapter 3 ensures there are no duplicates, i.e. correlations containing the same variables but at differing or same time lag.

4.4.1 Dataset Organisation

Simulated datasets are used to verify the validity of the grouping procedure at this stage. Two sets of simulated data were generated. The first was a selection of VAR processes as described in section 2.1.3, and the second was a selection of hand coded DBN networks as described in section 2.2.1. Five datasets from both sets of simulated data with varying dimensionality and order were generated. These are described below and in appendix H. VAR process creation is described in appendix E, and the genetic algorithm generation method was chosen so that the choice of process would not be restricted. DBN generation has been omitted but can be found in [Tucker2001b]. Table 4.1 describes the MTS datasets that were generated.

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

Table 4.1: The different MTS descriptions

These 10 multivariate time-series were grouped into different combinations to produce 6 datasets. The first consisted of all 61 variables, the second consisted of only DBN generated data, and 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 4.2 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 4.2: The Breakdown of Each Dataset

4.4.2 Parameter Estimation Results

Parameters and result statistics of the parameter estimation experiments can be found in appendix I. The resulting functions for μ and σ are according to equations 4.7 and 4.8:

$$\mu = \frac{2cR}{2s + c} \tag{4.7}$$

$$\sigma = \frac{R}{63} + \frac{11c}{s} \tag{4.8}$$

Once values for the mean and standard deviation have been found, confidence limits on the probability of the Random Bag finding a number of correlations that lie between *r* and *R* can be placed, 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 chapter, the ratio of *R* to *r* has been chosen 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 $\geq r$) = 0.95, the standard normal distribution tables with $z = (r\text{-}\mu)\,/\sigma$ can be used to find what the corresponding value of *c* should be. For the 95% level, the value of *z* should be –1.645. Since μ and σ are known, an equation can be formed in terms of *z, r, R, c* and *s* where only *c* is unknown. The starting point is given in equations 4.9 and 4.10.

$$z = \frac{r - \mu}{\sigma} \tag{4.9}$$

$$z = \frac{r - \left(\dfrac{2cR}{2s + c}\right)}{\left(\dfrac{R}{63} + \dfrac{11c}{s}\right)} \tag{4.10}$$

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 according to equation 4.11.

$$c \approx \frac{s}{22}\left((1.3r - 2R) + \sqrt{(2R - 1.3r)^2 + \frac{88}{63}(Rz - 63r)}\right) \tag{4.11}$$

The parameter c determines how long the procedure is going to execute for, 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 reasonable guideline a 95% confidence of finding the required number of correlations is aimed for. If the parameter estimation analysis is applied to datasets 1-6, the results listed in table 4.3 are obtained.

The equation for s represents the total possible number of correlations at varying time lags, once invalid correlations are removed (see chapter 3). μ and σ are defined in equations 4.7 and 4.8 respectively, z is the standard normal variable, and c is defined by equation 4.11. 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, it is suggested that for the Random Bag to be effective, this value should be less than 1/3. However, the parameter β needs defining. This represents the ratio r/R. A value of 0.2 was found through experimentation to provide 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 4.11 has resulted in the confidence limit not being exactly 95%, but rather 94.4% (on average).

Figure 4.1 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 difference between R=25,000 and R=2,500. Other similar experiments have shown that the optimal value for R/s is approximately 0.25%.

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
$\mu = \frac{2cR}{2s+c}$	173.321	76.879
$\sigma = \frac{R}{63} + \frac{11c}{s}$	14.779	8.083
$z = \frac{r-\mu}{\sigma}$	-1.578	-1.593
$\gamma = {}^{c}\!/_{s}$	0.273	0.261
$\beta = {}^{r}\!/_{R}$	0.200	0.200
Confidence	0.943	0.945

Table 4.3: Parameters for Datasets 1-6

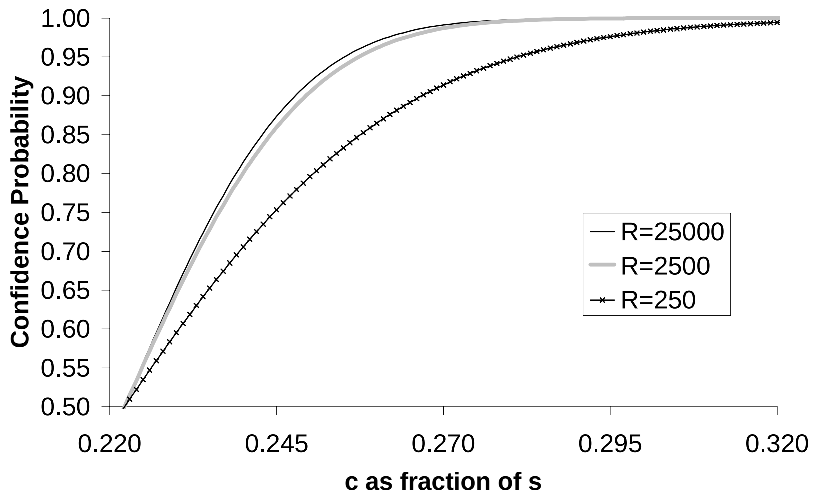


Figure 4.1: Confidence against γ with varying *R*

To conclude this section, it has been shown that *c* should be calculated from equation 4.11 once a confidence limit has been assigned (e.g. 95%, giving a value for *z*: -1.645); here it is recommended that the value for *R/s* to be about 0.25%. Finally, 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.

4.4.3 Genetic Algorithm Parameters

Table 4.4 displays the genetic algorithm parameters for the relevant grouping methods, i.e. those that are genetic algorithm based.

GA Parameter	**GPV**		**PMX**		**Falkenauer**	
	Dataset 1	**Dataset 2-6**	**Dataset 1**	**Dataset 2-6**	**Dataset 1**	**Dataset 2-6**
Population	150	100	150	100	150	100
Generations	1500	1000	150	100	150	100
Crossover Rate	0.80	0.80	0.80	0.80	0.90	0.90
Mutation Rate	0.05	0.10	0.008	0.018	0.90	0.90

Table 4.4: Genetic Algorithm Parameters

4.4.4 The Weighted-Kappa Metric

A metric is needed to show how similar or dissimilar two groups are. This is based on the *Weighted-Kappa* metric (see appendix J) and is calculated 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 minus one and one inclusive, where minus one represents very dissimilar groups and one represents very similar groups. The Weighted-Kappa metric is used to measure how accurately a grouping method has reconstructed the original groupings, hence the two sets of groups the metric is applied to consist of a grouping methods result and the corresponding known grouping.

4.5 Experimental Results

In this section the results from the 15 different combinations of correlation search and grouping strategy are examined to see how they performed when averaged over the 6 datasets. Marginal statistics are then presented to see how the correlation searches and the grouping strategies performed irrespective of each other. The effect the different datasets had on the outcome is considered by looking at their marginal statistics. Finally the grouping results are discussed using three examples.

In tables 4.5 to 4.7, the abbreviations *Falk* stands for Falkenauer, *HC* stands for hill-climbing and *S&C* for Separate and Conquer, and *Ex* for the Exhaustive Search.

Measurement	GPV	PMX	Falk	HC	S&C
Grouping Metric	52.717	56.283	59.733	57.250	54.667
Weighted-Kappa Metric	0.639	0.658	0.700	0.670	0.620
Function Calls	232327.650	63442.750	31762.233	11666.667	427.667

Table 4.5: The 5 Grouping Strategies applied to the Random Bag *Q*

Measurement	GPV	PMX	Falk	HC	S&C
Grouping Metric	55.300	57.333	62.667	60.917	56.833
Weighted-Kappa Metric	0.670	0.698	0.734	0.719	0.655
Function Calls	232292.517	63445.100	31770.500	11666.667	400.667

Table 4.6: The 5 Grouping Strategies applied to the Evolutionary Program *Q*

Measurement	GPV	PMX	Falk	HC	S&C
Grouping Metric	58.900	61.017	65.500	63.433	61.333
Weighted-Kappa Metric	0.698	0.712	0.773	0.736	0.714
Function Calls	232237.367	63421.600	31755.000	11666.667	411.667

Table 4.7: The 5 Grouping Strategies applied to the Exhaustive Search *Q*

It can be seen from the results of the 15 different methods that whilst there is a lot of variation in the number of calls to the grouping function, the metrics (and in particular the Weighted-Kappa metric) do not vary a great deal. 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 Weighted-Kappa metric: the method that seems to perform best over all the datasets is the Falkenauer method, although the hill-climb method finds almost as good a solution, it takes fewer function calls. However, as the marginal statistics will show, the Falkenauer method performs significantly better when averaged over the entire 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.

4.5.1 A Note Regarding RB and EP

Table 4.8 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**	**Exhaustive Search**	**Random Bag**	**Evolutionary Programming**
Dataset 1	150	0.592	0.547	0.527
Dataset 2	64	0.536	0.402	0.488
Dataset 3	64	0.694	0.659	0.629
Dataset 4	64	0.641	0.569	0.575
Dataset 5	64	0.548	0.497	0.509
Dataset 6	64	0.625	0.558	0.568
Average	N/A	0.606	0.539	0.549

Table 4.8: The Top *r* Correlations for Three Search Methods

As expected, the exhaustive search performs the best, followed by the evolutionary program and then Random Bag. It should be noted that these results only apply to the situation where there are a large number of correlation calls made (*c* is large). It has been shown in [Swift1999b] and chapter 3 that the EP method significantly outperforms the RB method for smaller values of *c*. Based on the extensive analysis and experiments performed so far it is recommended that if *c* is more than 30% of *s* then the exhaustive search method should be used. If this is not the case then the EP method should be used.

4.5.2 Marginal Statistics

In order to explore more fully the effect of the different correlation searches, grouping strategies and datasets, various marginal statistics were calculated. 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 4.9 to 4.11 below and each table is discussed in the next section.

Measurement	Average Ex	Average EP	Average RB
Grouping Metric	62.191	59.000	56.451
Weighted-Kappa Metric	0.729	0.704	0.665
Function Calls	82712.634	82735.329	82741.967

Table 4.9: Averaging over Correlation Search

The correlation summary statistics 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 Weighted-Kappa metric which measures the distance between the original groupings and the discovered groupings are very similar for all correlation search methods (approximately 0.7). 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.

Measurement	Averages				
	GPV	PMX	Falk	HC	S&C
Grouping Metric	55.639	58.211	62.633	60.533	57.611
Weighted-Kappa Metric	0.669	0.689	0.736	0.708	0.663
Function Calls	232285.844	63436.483	31762.578	11666.667	413.333

Table 4.10: Averaging over Grouping Strategy

The best grouping strategies, as shown by the grouping summary statistics, 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 effective crossover developed by Falkenauer. The other GA methods used less effective 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.

Dataset	Measurement		
	Grouping Metric	Weighted-Kappa Metric	Function Calls
1	109.268	0.729	151560.350
2	35.439	0.488	68981.024
3	53.691	0.795	68968.073
4	51.268	0.691	68948.707
5	50.333	0.740	68984.780
6	55.285	0.755	68936.927

Table 4.11: Averaging over Datasets

Looking at the dataset statistics, it appears that dataset 3 (the purely VAR data) produced the highest Weighted-Kappa metric rest and also 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 dominated 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 Weighted-Kappa 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.

4.5.3 Sample of Groupings

Table 4.12 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 that 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. 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. For example, DBN structure 3 in appendix H contains 10 variables and these consist of variables 13-22 in dataset 1.

Grouping Method	Original MTS Groupings	Discovered Groupings
EP /	0 1 2	0 6
Falkenauer		1
		2
Dataset 1	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 21 22	13
		14 15 20 21 22
		16 17 18 19
	23 24 25 26 27 28 29 30 31 32	23 24 25 26 27 28 29 30 31 32
	33 34 35 36 37 38 39 40 41 42	33 34 35 36 37 38 39 40 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
Exhaustive /	0 1 2	0 1 2
Falkenauer	3 4 5 6 7	3 4 5 6 7
		8
Dataset 5	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
Rand Bag /	0 1 2 3 4 5 6 7 8 9	0 1 2 3 5 7 8 9
Falkenauer		4
		6
Dataset 3	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 4.12: Some Falkenauer Grouping Results along with the Original Groupings

However if how the EP/ Falkenauer algorithm grouped these variables is examined, it can be seen that variables 16-19 have been placed in their own group. In some respect it has split the network into relatively independent structures such as the chain of nodes consisting of variables 16-19.

4.6 Visual Field Data Experiment

A further set of experiments was carried out using the EP/Falkenauer combination on the visual field dataset. The algorithms were applied to each patient's time series and the resulting groupings for each patient are analysed. The groups are compared between patients, using the Weighted-Kappa metric. The intention is to ascertain if there are any similarities between groups based upon a similar level of severity of the condition. The aim is to group the visual field variables in order to analyse similarities between groups using the Weighted-Kappa metric.

Table 4.13 displays the EP and Falkenauer parameters. Within this table, the first four GA parameters have been described in section 2.3.1. As described in chapter 3, *Search Space* is how many possible correlations exist within each patient dataset, *c* is the maximum number of correlation calls permitted (a fraction of *Search Space*), *r* is the number of true dependencies being looked for (out of *R*) and *R* is the number of correlations being search for (the length of *Q*).

Method	Parameter	Value
EP	Population	150
	Generations	150
	Crossover Rate	1.0
	Mutation Rate	0.1
Falkenauer	Search Space	31350
	c	8664
	r	200
	R	1000

Table 4.13: EP and Falkenauer Parameters for the Visual Field Dataset

The patients are reordered according to average sensitivity; this being a rough measure of how far the condition has progressed for a patient. The new patient 1 corresponds to the lowest average sensitivity and the new patient 82 the highest. Figure 4.2 shows the results of the Weighted-Kappa metric applied to all patient pairings. In figure 4.2, there are ten scales, white to black. White corresponds to the highest metric results and black to the lowest. The remaining eight grey scales correspond to metric scores somewhere in between, i.e. lighter means a higher metric score. The white diagonal is where a patient's grouping is compared with itself, the figure is symmetric about this diagonal, i.e. patient *a* and *b* are compared above the line, whilst below it is patient *b* with *a*. The graph goes from light in the bottom right hand corner to a darker shade in the top left hand corner.

This indicates that those patients with high sensitivity have similar groups; and it was observed that these groups tend to be smaller in size. Those with lower sensitivity tend not to have any similar groupings to any other patient; and it was observed that their groups tend to be quite large.

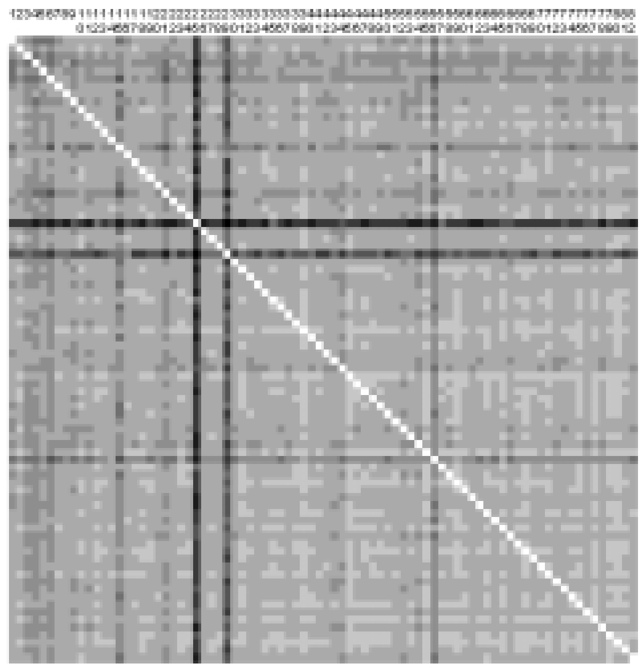


Figure 4.2 Comparison of Patient Groupings using the Metric

This can be explained by those with good sensitivity and hence good vision being more consistent during the visual field tests, which would result in similar grouping between those points that are related. With the low sensitivity (and low vision) patients, the results of the visual field test tend to be more variable. This is due to each field point corresponding to a large section of the retina; hence any glaucoma damage may only affect part of a point. The visual field test does not consistently test the same spot, but somewhere in the same vicinity, i.e. the visual field test light source does not fill the test point exactly, and hence may be testing different collections of cells on each test. Additionally, the deterioration does not follow exactly the same path for each patient: two patients may have the condition with the same severity but have totally differing parts of the eye affected, and hence different groupings.

In short, the visual field data generates groups that make good sense: the groups are low in size and similar for patients with good vision, and large in size and varied for those with low vision.

4.7 Concluding Remarks

In this chapter a framework for decomposing high dimension MTS into lower dimension MTS based on the correlation between the variables has been outlined. This can be very useful in problems where the high dimensionality of a MTS prevents certain algorithms from being applied, for example the modelling of VAR processes. These 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. Concrete practical recommendations on the parameter estimation for the correlation search step of the methodology have also been provided.

For the visual field data, the groups obtained make an ideal starting point for modelling the visual field deterioration through statistical models such as the Vector Autoregressive Process. Such a model has a large number of parameters, which is proportional to the square of the number of variables being modelled. Decomposing the visual field data into several smaller and highly related subsets of variables will make this model easier to deal with.

5. The VARGA Method

Chapter 3 presented several methods for locating relationships over time between variables within a high dimensional MTS, and showed that an evolutionary programming method performed the best. Chapter 4 presented a number of methods for using these correlations to group the variables into mutually exclusive lower dimensional MTS, and showed that a genetic algorithm based method was the best choice. The next step is the short term forecasting for each of these MTS subsets, which is the subject of this chapter, and is based upon work outlined in [Swift1999a] and [Swift2002].

5.1 Statistical Multivariate Time Series Modelling

Much research has been done on the analysis of MTS data in both the statistical and artificial intelligence communities. Statistical MTS modelling methods include the Vector Autoregressive process, the Vector Autoregressive Moving Average process, and other non-linear and Bayesian approaches [Casdagli1992], while various Artificial Intelligence (AI) methods have been developed for different purposes including dependence detection in MTS of categorical data [Oates1999], knowledge-based temporal abstraction [Shahar1996, Shahar1997], and forecasting [Baldi1999, Weigend1994].

However, one area that has been largely overlooked is MTS in which the data set consists of a large number of variables but with a small number of observations. There are inherent difficulties in using traditional statistical techniques to model this type of MTS. The normal tension glaucoma visual field dataset is one such MTS. As previously stated in chapter 2, a Vector Autoregressive Process has been deemed suitable for the modelling and forecasting of this dataset. The standard statistical methods for fitting a VAR process to a set of data often consist of two steps: order selection (determining a suitable p) and parameter estimation (calculating the matrices A_i from the data) as described in section 2.1.

The process of parameter estimation and order determination is difficult with short high dimensional MTS as discussed in section 2.1.3. This chapter presents a method that attempts to address these problems.

5.2 Genetic Algorithms and Time Series

One possible way of locating the order and parameters of a VAR process suitable for modelling short MTS datasets is through the use of a suitably adapted genetic algorithm. Within the field of time series modelling, GAs have been used with various levels of success. For example, in [Bearse1998] a binary genetic algorithm is used to find VAR subset models, however it is assumed the basic VAR process has been located through some other means; in [Rolf1997] a GA is used to find the parameters of a univariate ARMA model and in [Shi1997] a GA is used to find the parameters of a univariate exponential AR process. Very little work has been done with the VAR process using GAs, and where GAs have been applied, concentration has been on univariate and non-linear modelling. In chapter 6 other possible ways for parameter estimation and order selection are considered; however in the rest of this chapter the GA based approach will be introduced.

5.3 VARGA-v1

The VARGA (*Vector AutoRegressive Genetic Algorithm*) paradigm is essentially where an nVAR(p) process for a given set of data is represented by a selection of p possible $n \times n$ candidate matrices; a genetic algorithm based method is applied to a population of candidate solutions over subsequent generations to improve their suitability to fit the data being modelled. Such a genetic algorithm can be used to find the parameters and order of a VAR process without making any of the assumptions outlined in section 2.1.3. In addition, VARGA reduces the length restriction to $T>p$, which makes it possible to model many short multivariate time series. Note that this is the theoretical minimum length an MTS could be, given some constant order p. VARGA-v1, presented in [Swift1999a], is the first attempt at this idea.

Algorithm 5.1 describes the VARGA-v1 algorithm. Details of the genetic operators will be described in the following sections.

Algorithm 5.1: The VARGA-v1 Algorithm

```
1)   Input:   The VARGA-v1 parameters from table 5.2
              The fitness function from equation 5.2
2)   Create POPULATION random chromosomes of order UI(1,MAXP)
3)   Sort population ascending according to fitness
4)   For g = 1 to GENERATIONS do
5)            Crossover population
6)            Mutate population's genes
7)            Mutate population's order
8)            Sort population in ascending order according to fitness
9)            Select the new population
10)  Next g
11)  Output:  The best VAR process is the chromosome from the final population with the
              smallest fitness score
```

The VARGA algorithm is essentially the same as the standard Holland genetic algorithm. However, crossover is different and there are two mutation operators.

5.3.1 Representation

The chromosome representation is a list of (n×n) matrices, whose elements are integers ranging between [0,GENESIZE), where GENESIZE is an implementation constant. A simple scaling is done to map each value between the predefined parameter limits. Each matrix's order in the list corresponds to the equivalent coefficient matrix for the VAR process being represented. This chromosome representation is shown in Figure 5.1. The range of permissible orders (candidate values for p) for each VAR process is restricted to the range [1,MAXORDER], where MAXORDER is the maximum order (the value p can take) of the VAR process being searched for.

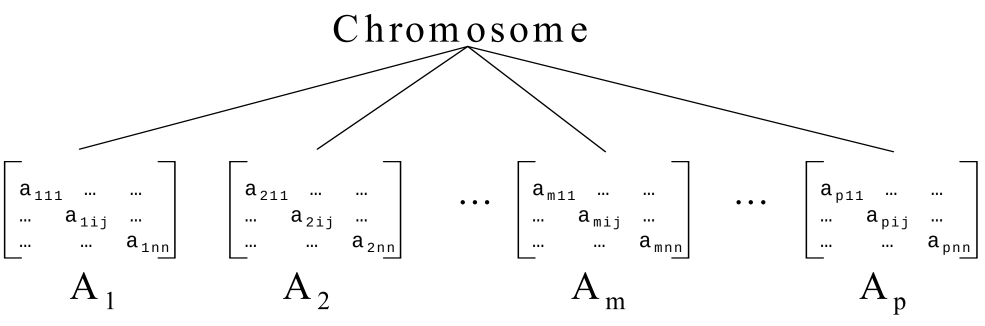


Figure 5.1: Chromosome Representation for VARGA method

5.3.2 Fitness

As with a normal GA, VARGA-v1 needs a suitable fitness function to evaluate candidate solutions to the problem. This fitness will rate a potential solution against a given dataset using some evaluation criteria. A VAR process that is optimised to forecast for a specific dataset will be searched for. One step ahead forecasting will be concentrated on, although any number of steps ahead could be used. The level of accuracy for VARGA (the fitness function) is defined in equations 5.1 and 5.2.

The fitness of a chromosome, the value ε within equation 5.2, is determined by the sum of the magnitudes of the noise vectors computed by performing a series of one-step ahead forecasts from $t=T_0,..,T$.

$$\underline{\hat{\varepsilon}}(t) = \underline{x}(t) - \sum_{i=1}^{p} \hat{A}_i \cdot \underline{x}(t-i) \tag{5.1}$$

$$\varepsilon = \sum_{t=T_0}^{T} \sum_{j=1}^{n} \left| \hat{\varepsilon}_j(t) \right| \tag{5.2}$$

where $\underline{\hat{\varepsilon}}(t)$ is the estimation of the noise vector, $\hat{A}_i$ is the estimation of the *ith* parameter matrix, ε is a scalar that represents the level of noise, and T_0 is defined as MAXORDER+1.

Absolute value error rather than mean squared error is used because mean squared error can exaggerate the effects of outliers, especially when the time series is short [Armstrong1992, Chatfield1992]. The model with the smallest ε value is deemed the most suitable model for forecasting since it is assumed that the best estimation for any unobserved noise vector is the zero vector.

With a GA it is traditional to maximise the fitness, hence the negative forecast error was used for the fitness function, i.e. $-\varepsilon$. The value of ε is computed for all possible one step ahead forecasts for a given set of data, starting at the T_0th recorded set of observations. If the order of the VAR process in question was used, then there would be a total of T-p forecasts available for a pth order VAR process. Hence with a length 20 MTS, a VAR(1) process would be evaluated on 19 forecasts, and a VAR(5) process on 15 forecasts. It can be clearly seen that even if these two VAR processes had equal error at all forecast points, then the VAR(1) process would have 19/15 (1.267) times the forecast error of the VAR(5) process. Hence the fitness function would be biased for larger order models. It would be a simple step to scale the fitness error by the number of forecasts used to construct it, but this would still mean that smaller order models were evaluated on data that larger order models could not be. By setting the first point that forecast accuracy is evaluated on to be the T_0th observation in the MTS, all order VAR processes are evaluated on the same number of forecasts and the same set of data. Table 5.1 demonstrates this for a length 20 VAR(3) process with MAXORDER=5. The forecast difference (the absolute value of the observed minus the predicted) is summed over all of the 15 forecasts and used for the fitness (fitness = negative forecast error).

Data Start	**Data End**	**Forecast**
(MAXORDER+1-*p*) 3	(MAXORDER+0) 5	(MAXORDER+1) 6
(MAXORDER+2-*p*) 4	(MAXORDER+1) 6	(MAXORDER+2) 7
…	…	…
(MAXORDER+14-*p*) 16	(MAXORDER+13) 18	(MAXORDER+14) 19
(MAXORDER+15-*p*) 17	(MAXORDER+14) 19	(MAXORDER+15) 20

Table 5.1: Example Fitness Function

5.3.3 Initial Population

The initial population (step 2 in algorithm 5.1) is generated according to algorithm 5.2.

Algorithm 5.2: VARGA-v1 Initial Population

```
1)  Input:  The VARGA-v1 parameters from table 5.2
            Population – an empty list of chromosomes
            n – the number of variables in the MTS
2)  For i = 1 to POPULATION
3)          Set ORDER = UI(1,MAXORDER)
4)          Create a new chromosome C of order ORDER
5)          For j= 1 to n
6)                  For k = 1 to n
7)                          For p = 1 to ORDER
8)                                  Set C[$a_{pjk}$] = UR(MINGENE,MAXGENE)
9)                          Next p
10)                 Next k
11)         Next j
12)         Add C to the Population
13) Next i
14) Output: Population – containing POPULATION random chromosomes
```

Here, $x[a_{ijk}]$ refers to the *jth,kth* element of the *ith* parameter matrix of chromosome x (see figure 5.1) and MINGENE and MAXGENE are the lower and upper limits this value can take respectively. The population is created by constructing POPULATION random chromosomes; where the order is randomly generated and the corresponding parameter matrix elements are randomly generated.

5.3.4 Crossover

The **Crossover** operator is described in algorithm 5.3. Each chromosome has a chance, say CROSSOVERRATE, of being selected to produce offspring. Once the parents have been selected, the number in the *breeding stock* is forced to be even. If the number is odd then there is a 50% chance that either a random chromosome is added into the stock, or one selected for breeding is removed. The next step is to pair up randomly each potential parent. For each pair of parents, a random parameter matrix, a random row index and a random column index is chosen. Parameters above this row and column are exchanged between the two selected matrices of each parent creating two new children.

Algorithm 5.3: VARGA-v1 Crossover Operator

```
1)   Input:  The VARGA-v1 parameters from table 5.2
             Population – an empty list of chromosomes
             n – the number of variables in the MTS
2)   Randomly select CROSSOVERRATE proportion of the population for breeding
3)   Randomly pair up the breeding stock
4)   For each parent pair c, d do
5)       Set x = c, Set y = d
6)       Set i = UI(1,order of x), Set j = UI(1,order of y)
7)       Set ROW = UI(1,n), Set COL = UI(1,n)
8)       For r = 1 to n
9)           for c = 1 to n
10)              If r < ROW and c < COL Then
11)                  Set x[$a_{irs}$] = d[$a_{jrs}$]
12)                  Set y[$a_{jrs}$] = c[$a_{irs}$]
13)              End If
14)          Next c
15)      Next r
16)      Add x, y back to Population
17)  Continue
18)  Output: Population – increased in size through breeding
```

5.3.5 Mutation

There are two forms of mutation in VARGA-v1, namely gene mutation and order mutation. Gene mutation is defined in algorithm 5.4 and order mutation in figure 5.2.

Algorithm 5.4: VARGA-v1 Gene Mutation

```
1)  Input:  The VARGA-v1 parameters from table 5.2
            Population – an empty list of chromosomes
            n – the number of variables in the MTS
2)  Each gene of every chromosome has a GENEMUTATIONRATE chance of mutating
3)  For each gene that mutates do
4)      Set v = [(gene value + UI(1,GENESIZE)) modulo GENESIZE]
5)      Set Gene value to v
6)  Continue
7)  Output: Population – increased in size through gene mutation
```

Each gene (matrix element) of each chromosome is given a chance (GENEMUTATIONRATE) to mutate. If a gene mutates then it is incremented by a random value between 1 and its maximum value (GENESIZE). The modulo part of step 4 in algorithm 5.4 ensures that this new value does not exceed the permissible gene limits.

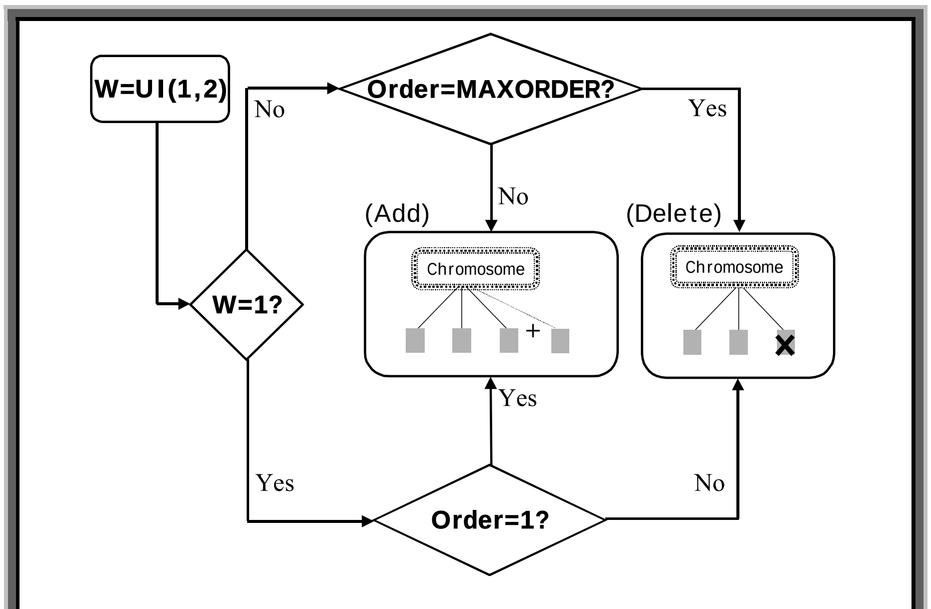


Figure 5.2: VARGA-v1 Order Mutation

Order mutation is detailed in figure 5.2 and works by giving each chromosome in the population a chance of having its order mutate. If its order does mutate then there is a 50% chance that it either increases or decreases. The order is not allowed to be below one or above some preset limit (MAXORDER). If the order increases then a new random parameter matrix is added as the new highest order matrix. If the order is to decrease then the highest order parameter matrix is deleted.

5.3.6 Selection

Selection of the new population is exactly the same as the roulette wheel method [Holland1975], however the reciprocal of the fitness score (equation 5.2) of each chromosome is used. This is because the score represents the residual noise; the lower the score, the better the model being represented.

Parameter	**Meaning**	**Value**
n	MTS dimensionality, nerve fibre bundle 12	9
POPULATION	Population size, constant	10
GENERATIONS	Number of generations; crossover will not be so effective: only a portion of each chromosome is crossed over	5000
Selection	The roulette wheel and elitism	1
ORDERMUTATIONRATE	Order mutation rate; if order = 1 then add a matrix, if order = MAXORDER then delete	5%
GENEMUTATIONRATE	Gene mutation rate, after crossover, including the parents; the population best is not mutated	0.5%
CROSSOVERRATE	Crossover Rate, percentage of population allowed to breed, uniform and one point	100%
Chromosome Size	Order$\times n \times n$	Order×81
GENESIZE	Gene limits, mapped onto the [-1.25,1.25] interval, the range for each a_{ijk}	[0,20000] (Integer)
MAXORDER	Maximum order; as *MaxLag* in chapter 3	5
Fitness	Negative, nearest to zero the better.	Real

Table 5.2: Genetic Algorithm Parameters for VARGA-v1

5.3.7 Parameters

For the VARGA-v1 method, the parameters for the GA are listed in Table 5.2.

5.3.8 Evaluation

For the purpose of this method, nine of the 76 points of the visual field data, corresponding to nerve fibre bundle twelve (see section 2.2.2), are considered. Bundle twelve has been chosen for two reasons. The first is that the nine points are one of the largest in number for any of the nerve fibre bundles (equal in size to nerve fibre bundle five). The second is that glaucoma damage tends to originate from the blind spot, and then move through these points to the visual periphery, thus usually affecting nerve fibre bundle twelve. To show that a multivariate model is appropriate, it is necessary to show that the variables being modelled have a strong interdependency. The correlations between all points, with a time lag of up to five time units, have been calculated by using the visual field records of the patients. An average was taken to give a single correlation value (this averaging is described in section 3.1.2). Table 5.3 shows these for all of the points, and then for nerve fibre bundle twelve (NFB 12). Both Pearson's and Spearman's Correlation Coefficients are used for this comparison. Clearly it can be seen from the table that on average the points within bundle twelve have a higher correlation with each other than with other points.

Statistic	**Pearson**		**Spearman**	
	All	**NFB 12**	**All**	**NFB 12**
Minimum	0.015	0.049	0.016	0.056
Maximum	0.874	0.661	0.898	0.714
Mean	0.204	0.306	0.205	0.294
Variance	0.024	0.027	0.022	0.024

Table 5.3: Correlation Coefficient Comparison

For this dataset, there is no missing data. It is assumed that each test is spaced evenly in time, i.e. the time gap between subsequent tests is a constant. The data itself is a continuous variable. The dataset contains information on 82 patients' right eyes tested approximately every six months for between five and 22 years. Therefore, the length of time series corresponding to some of the patients' visual field tests can be rather short. All patients had been diagnosed and are undergoing treatment for Normal Tension Glaucoma, and were representative of the population.

In this section the models found using the VARGA-v1 method are compared with those produced by a conventional way of finding a VAR process, in this case the solution of the Yule-Walker equations using S-Plus. To provide more insight into the accuracy of the VARGA-v1 method, it is further compared with the results from two other techniques: *Holt-Winters* forecasting and the *Noise Model*.

The VAR Process in S-Plus

S-Plus [Mathsoft1997] has an easy-to-use function for finding the best-fit VAR(p) process for a given dataset. Each patient's visual field results give a model that is rated according to equation 5.2. Since S-Plus uses "*Whittles Recursion*" [Whittle1984], a limit on the minimum length T of a time series with n variables is constrained by inequality 5.3.

$$T \geq n(p+1) \tag{5.3}$$

Holt-Winters Forecasting Method

Despite the fact that the dataset is multivariate, it is worth treating it as univariate to see if the assumptions about point clustering (by nerve fibre bundles) are accurate. The Holt-Winters (HW) forecasting method is a simple way of predicting the next value in a univariate time-series. For the visual field dataset, it is assumed that there is no seasonal effect, and that only one step ahead forecasts are needed. The HW Method is described in section 2.1.2.

Parameter	Meaning	Value
POPULATION	Constant	50
GENERATIONS	Crossover will not be so effective since the chromosome size is quite small	1000
Selection	As VARGA-v1	5
MUTATIONRATE	After Crossover, including parents; if a gene mutates, add UI(0,99) then modulo by 100, this is detailed within the section on VARGA-v1	0.5%
CROSSOVERRATE	Percentage of population allowed to breed, uniform and one point	80%
Chromosome Size	Three genes for each HW parameter	12
GENESIZE	Each HW parameter is created from three consecutive genes	[0,100] (Integer)
Fitness	Nearest to zero the better, calculated in a similar manner to that in equation 5.2	-ve, Real

Table 5.4: Genetic Algorithm Parameters for the HW method

There are various ways of finding the values of the required parameters, (α, $\hat{\gamma}$, L_0 and T_0), as suggested in [Chatfield1988b], but for simplicity sake, a genetic algorithm will be used to estimate them. This approach has been done in successfully in [Agapie1996] and [Agapie1997], however a binary GA was used. In this thesis, a standard Holland GA was implemented, with the modifications and parameter values listed in table 5.4; the notable difference is that the genes are not binary. The *fitness* for the HW method is rated in a similar way to a VAR process, but the residuals from a one step ahead forecast are summed for each point. A visual field case is treated as nine univariate forecasts, one for each point within nerve fibre bundle twelve. Each chromosome consists of twelve genes which are integers ranging between zero and 99. Three genes represent each parameter. The values of the three genes are then scaled accordingly, i.e. for parameters α, $\hat{\gamma}$, between zero and one, and for L_0, T_0 between ±100.

The Noise Model

The *Noise Model* is defined in equation 5.4, and will refer to an *n*VAR(*0*) process, i.e. one generated by noise. Note that equation 5.4 is equivalent to equation 5.1 with *p*=0, i.e. a VAR(0) process. A forecast for the noise model is defined in equation 5.5. This model is very easy to construct and simulate, and will be used as a simple benchmark with which to rate other VAR processes against. A VAR process that is designed to model a particular MTS is expected to perform much better than the corresponding noise model, since the noise model simply assumes that any time series observations were totally randomly generated, with no structure and reliance on previous observations.

$$\underline{x}(t) = \underline{\varepsilon}(t) \tag{5.4}$$

$$\underline{x}(t) = \underline{0} \tag{5.5}$$

5.3.9 Experimental Results

This section describes the results of the experiments of applying the four methods to nerve fibre bundle twelve of the visual field data.

S-Plus rejected (could not provide a result for) several of the time series due to matrix inversion problems and numerical instability. Further more, the restriction described in inequality 5.3 meant that the dataset for the experiments had to be reduced significantly. If an order of at least one is under consideration (p>0) and since there are nine variables (n=9), the time series length must be at least 18 (T≥18). This restriction reduced the dataset further; and as a result, only 34 patients' visual field records were usable. VARGA-v1 was run once on each of the patients, and figures 5.3 to 5.6 display the results for four methods over the 34 patients' visual field tests.

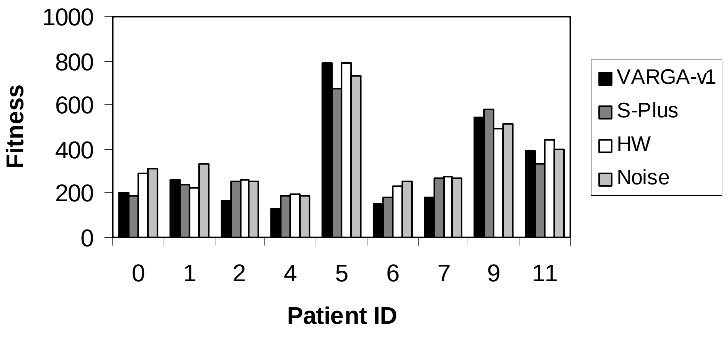


Figure 5.3: VARGA-v1 Results for Patients 0-8

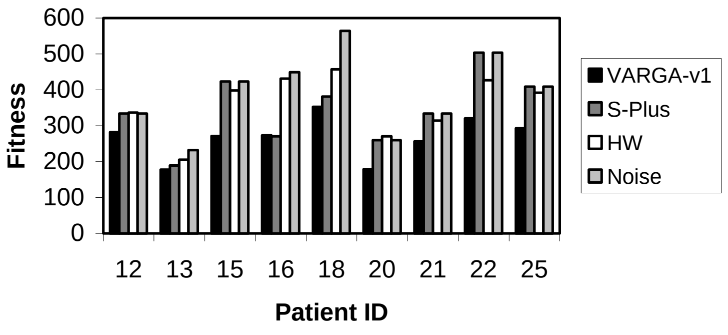


Figure 5.4: VARGA-v1 Results for Patients 9-17

Table 5.5 summarises these results, and lists the order of the best models. Note that the lower the fitness, the better. From this table the following can be observed. Firstly, VARGA has the best performance, followed by S-Plus, Holt-Winters, and the noise process. Secondly, VARGA has a more consistent set of results for the order than the VAR process in S-Plus. Thirdly, VARGA located models of only order one and two whilst the order ranged between zero and three with S-Plus.

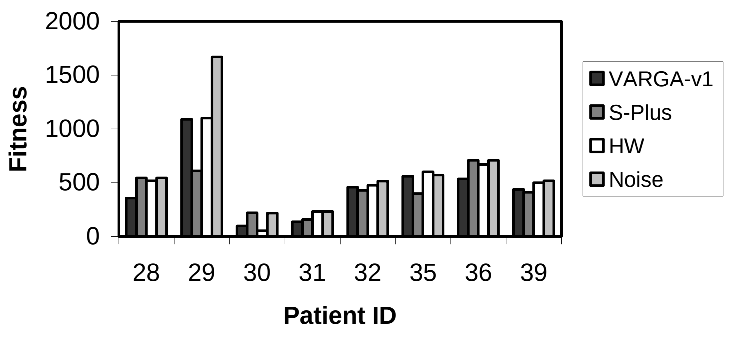


Figure 5.5: VARGA-v1 Results for Patients 18-25

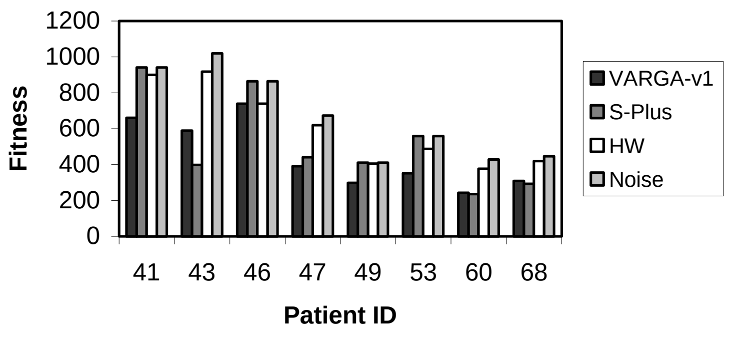


Figure 5.6: VARGA-v1 Results for Patients 26-33

Method	Order (*number* of *order*)	Average Fitness
VARGA v1	31 of 1, 3 of 2	367.060
S-Plus	15 of 0, 16 of 1, 2 of 2, 1 of 3	401.183
HW	N/A	454.420
Noise	34 of 0	502.364

Table 5.5: VARGA-v1 Results Summary

It could be argued that the VARGA method is biased towards finding a low order model, since the search space increases each time the order of the model increases. However tests have been run where the VARGA method is forced to search for a VAR(3) process (thus behaving like a conventional GA) where the number of generations is increased to compensate for the number of variables being found. The results here still showed that a VAR(1) or VAR(2) still fit the data better. From this finding, further work could be done to more accurately fit a VAR(1) or VAR(2) process, reducing the need to search for the model order.

It is curious that S-Plus found VAR(0) processes where it could have found higher order processes in some cases. These can be seen in Figures 5.3 to 5.6 where the score of S-Plus is the same as the noise model, e.g. patient IDs 15 and 53. This suggests that the order selection method is flawed and/or unreliable.

With the HW method, the results show that a multivariate method (VARGA-v1 or S-Plus) is more accurate than a series of univariate models (the HW method applied on each of the nine points) for the visual field dataset. To give this method more credence, a full search for the parameters (this takes a very long time) has shown that the GA based method locates them very accurately (within 1% of their actual values to four decimal places). Thus the methods poor performance is not due to the GA being unable to locate a good set of parameters but more likely due to the data being multivariate.

It is worth noting that for the patient case where the difference between the errors for the VAR process found by VARGA-v1 and the noise process is the most different (patient ID 30); the error (one step forecast errors) is reduced by only 34.9%. This could indicate that the visual field data has a very large noise term. This is confirmed when the actual data is viewed. The visual field history for patient ID 30 is given in Figures 5.7 to 5.9. As would be expected, visual field sensitivity deteriorates down to zero. However the graph seems to show that some of the points get better, which is impossible with the disease glaucoma. This can be explained by the fact that each visual field point corresponds to an area on the retina, and that the exact retina cells being tested for damage are not always the same when the point undergoes another test. This is due to there being a limit on the accuracy of the test machine. Hence the data seems to contain a large element of noise. These could be treated as outliers, but how these could be dealt with is a difficult problem, especially with time series data.

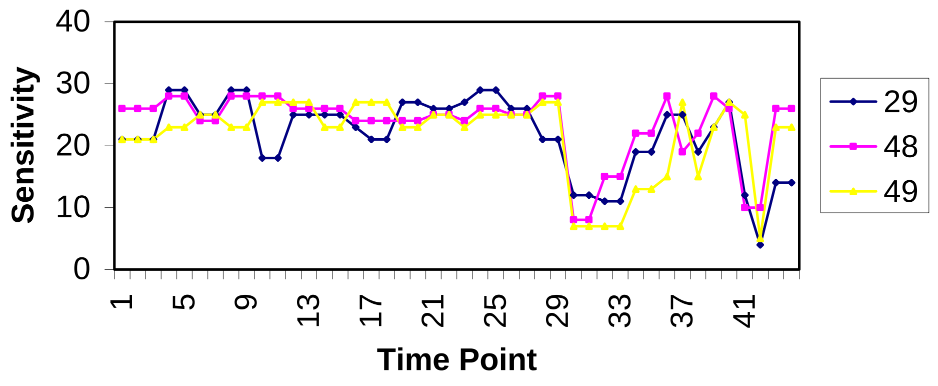


Figure 5.7: Points 29, 48 and 49 for a Patient

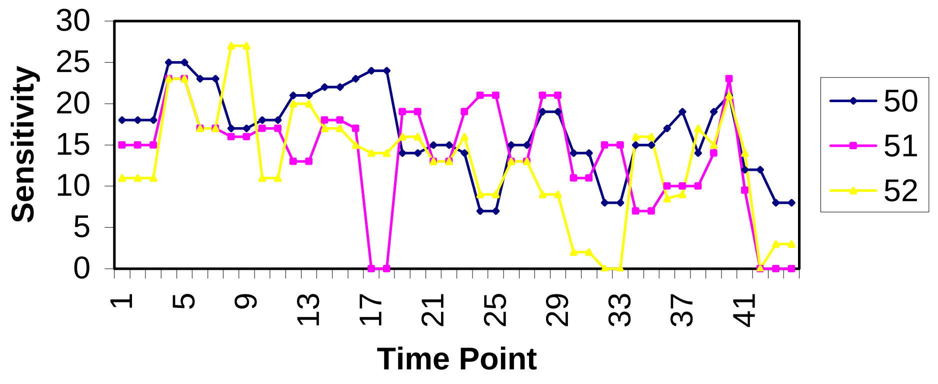


Figure 5.8: Points 50, 51 and 52 for a Patient

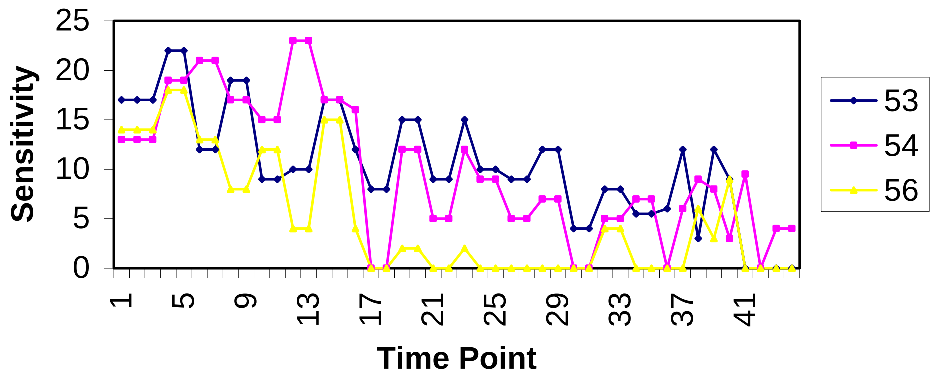


Figure 5.9: Points 53, 54 and 56 for a Patient

5.4 The VARGA-v2 Method

Section 5.3 presented the first version of VARGA. This method performs well versus the VAR process in S-Plus. However VARGA suffers from the following drawbacks:

i) There is a slight bias towards low order models, for example when a parameter matrix is added (through order mutation); there is a better chance of the fitness becoming worse than improving. (This fact was confirmed through experimentation).

ii) The representation limits the accuracy of the parameter matrix elements, i.e. the natural number mapping leads to a fixed level of decimal places.

iii) The *Crossover* operator only acts on part of a chromosome.

iv) The *survival* operator uses the inverse of the fitness function to proportionally select individuals to the next generation. Section 5.4.6 looks at the implications of this method and suggests an alternative.

v) A random individual is often worse than the corresponding noise model.

The following sections describe VARGA-v2, which is the next generation of the VARGA method. The changes made in this method are aimed at reducing the problems listed above and additionally the VARGA methodology is tested more rigorously. This section is a summary of work presented in [Swift2002].

The VARGA-v2 algorithm is also a standard genetic algorithm, but has refinements to deal with matrices, variable length chromosomes, and length variation as with VARGA-v1. Algorithm 5.5 details the VARGA-v2 method. Note that in this implementation of VARGA, only the children are mutated and there is an additional crossover operator.

Algorithm 5.5: The VARGA-v2 Algorithm

```
1)   Input:   The VARGA-v2 parameters from table 5.7
              The fitness function from equation 5.2
2)   Create POPULATION chromosomes of size random UI(1,MAXORDER), (the order p)
3)   Sort the population descending according to fitness
4)   For g = 1 to GENERATIONS do
5)            CROSSOVER the populations genes and add to list X1
6)            CROSSOVER the populations size (order) and to list X2
7)            MUTATE X1s genes and MUTATE X2s genes
8)            MUTATE X1s size (order) and add back to the population
9)            MUTATE X2s size (order) and add back to the population
10)           Sort the population in ascending order according to fitness
11)           SELECT the new population (size = POPULATION)
12)  Next g
13)  Output: The best VAR process is the chromosome from the final population with the
              smallest fitness score
```

Details of the initial population of candidate solutions and the genetic operators are described in the following sections.

5.4.1 Representation

The representation is similar to that of VARGA-v1, however with a slight change, here each matrix element is a bound real number. The elements are limited between some fixed bounds, denoted as [MINGENE,MAXGENE]. The intension here is to address drawback (ii) as described previously in this section.

5.4.2 Fitness

This is identical to the fitness function of VARGA-v1 (see section 5.3.2).

5.4.3 Initial Population

With a GA, the initial population is usually generated randomly. With VARGA-v2 the initial population is seeded with the noise model. Seeding [Michalewicz1996, Sharif1998] is where the initial population for a genetic algorithm has some or all of its chromosomes set to some predefined representation, as opposed to an entirely random selection. Usually these seeds come from expert knowledge or some other fast approximate search method, for example hill-climbing.

A set of random order (within the limits) individuals whose genes are all zero matrices comprise of half of the initial population. The other half is randomly generated within the specified constraints (section 5.4.7 contains the details of all of the VARGA-v2 parameters). The justification for this is that experimentation has found that a random VAR(p) process is likely to have a worse fitness when compared with the noise model for a given MTS. This is due to the accumulation of forecast errors through there being specified relationships where there should be none. Hence starting the search from the noise model saves VARGA-v2 from trying to improve from a very poor starting point, thus addressing drawback (v) as described previously in this section.

Having the initial population 50% noise model and 50% random VAR(p) processes allows some random genes to get mixed with the zeros of the noise model, otherwise the only change from a 100% noise model population would be through *gene mutation*. *Crossover* would simply swap zeros between matrices during the initial generations.

5.4.4 Crossover

Two crossover methods are used; *Order Crossover*, which acts on whole matrices, and *Gene Crossover*, which acts on the matrix elements. Both methods are designed to be applied to as much of a chromosome as possible, in order to make this stage more efficient hence addressing drawback (iii) as described previously in this section.

Order Crossover is where matrices from two parents are copied to form children, in a manner similar to uniform crossover (see section 2.3.1). Each member of the current population has a chance of being selected for breeding (ORDERCROSSOVERRATE). The breeding subset of the population is paired up randomly (*parents*), and two children are produced from each set of parents. The following steps construct these children, formally defined in algorithm 5.6 and figure 5.10:

i) The children are initialised by making a copy of each parent.

ii) For all of the matrices in both children, there is a 50% chance that the two children swap a matrix for a particular order.

iii) If the two children are of different sizes (differing order), then this is only performed for the matrices that have a corresponding order.

Algorithm 5.6: VARGA-v2 Order Crossover

```
1)    Input:   Two Parents P1 and P2
2)    Set MINORDER = Min(Order(P1),Order(P2))
3)    Child1 = P1, Child2 = P2
4)    For i = 1 to MINORDER
5)            If UI(0,1) = 1 then Swap Matrix i of Child1 and Matrix i of Child2
6)    Next i
7)    Output: Two children, Child1 and Child2
```

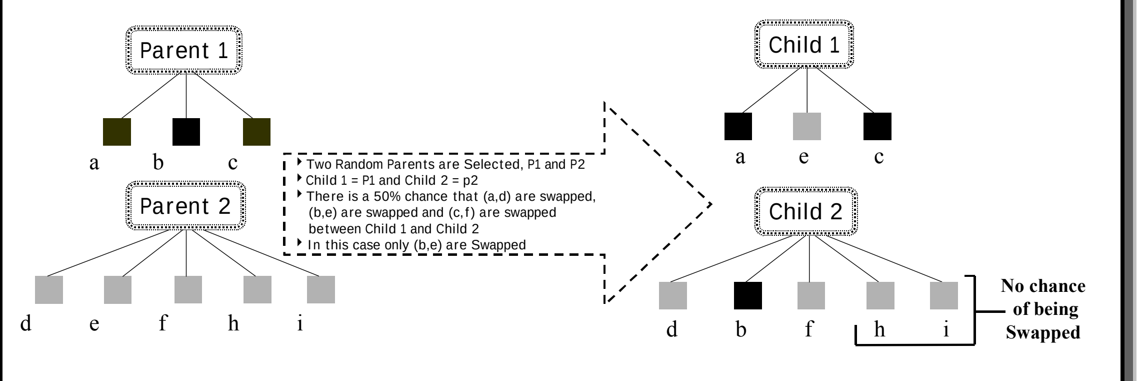

Parent 1
a
b
c
Parent 2
d
e
f
h
i
Two Random Parents are Selected, P1 and P2
Child 1 = P1 and Child 2 = p2
There is a 50% chance that (a,d) are swapped, (b,e) are swapped and (c,f) are swapped between Child 1 and Child 2
In this case only (b,e) are Swapped
Child 1
a
e
c
Child 2
d
b
f
h
i
No chance of being Swapped

Figure 5.10: VARGA-v2 Order Crossover

Min(x,y) returns the smaller number of x and y. *Order*(z) returns the order of the VAR process represented by chromosome z, i.e. the number of matrices in z.

Gene Crossover is very similar to *order crossover*. However matrix elements are swapped, and not whole matrices. If one of the parents is longer than the other then there is a 50% chance that this crossover will be offset by the difference in size. For example if parent one is of order three and parent two of order five, either elements from matrices 1, 2 and 3 would be uniformly crossed over from both parents, or 1, 2 and 3 of parent one with 3, 4 and 5 of parent two. The rational behind this is to ensure that only matrices that correspond to the same lag are crossed over. The procedure is described in algorithm 5.7 and shown in figure 5.11.

Algorithm 5.7: VARGA-v2 Gene Crossover

```
1)  Input:  Two Parents P1 and P2
            n – the dimensionality of the MTS
2)  If Order(P1) > Order(P2) then
3)          Set Child2 = P1, Set Child1 = P2
4)  Else
5)          Set Child2 = P2, Set Child2 = P1
6)  End if
7)  Set MINORDER = Order(Child1)
8)  Set Offset = (Order(Child2) – MINORDER) × UI(0,1)
9)  For i = 1 to MINORDER
10)         For j = 1 to n
11)                 For k = 1 to n
12)                         If UI(0,1) = 1 then
13)                                 Set Temp = Child1(i,j,k)
14)                                 Set Child1(i,j,k) = Child2(i+Offset,j,k)
15)                                 Set Child2(i+offset,j,k) = Temp
16)                         End if
17)                 Next k
18)         Next j
19) Next i
20) Output: Two children, Child1 and Child2
```

where *Childx*(*i*,*j*,*k*) is the matrix element a_{ijk} of child number *x* and the rest of the terms are defined above. Again each parent has a chance of producing offspring in this way where the chance of a member of the population being chosen for *gene crossover* is GENECROSSOVERRATE.

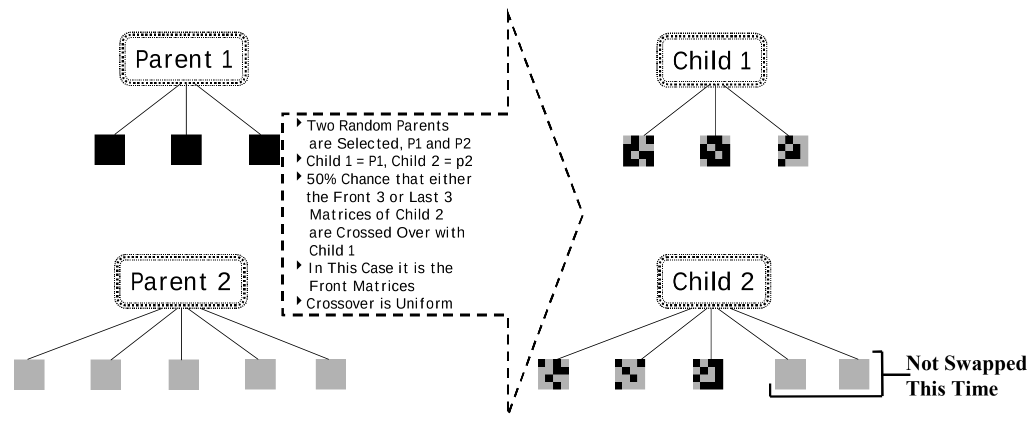

Parent 1
Parent 2
Two Random Parents are Selected, P1 and P2
Child 1 = P1, Child 2 = p2
50% Chance that either the Front 3 or Last 3 Matrices of Child 2 are Crossed Over with Child 1
In This Case it is the Front Matrices
Crossover is Uniform
Child 1
Child 2
Not Swapped This Time

Figure 5.11: VARGA-v2 Gene Crossover

5.4.5 Mutation

Two mutation operators are used, gene mutation and order mutation. Gene mutation changes the elements of a matrix and order mutation either deletes or adds a new parameter matrix. These are similar procedures to the ones used in VARGA-v1.

Gene Mutation. When an individual is deemed to *gene mutate* (step 7 in the VARGA-v2 algorithm), each element of each matrix (denoted a_{ijk}) is given a chance to mutate (GENEMUTATIONRATE). If a matrix element mutates, then it is changed according to a random Gaussian distribution. Algorithm 5.8 defines this more formally. In this algorithm, a check is made for each possible matrix element that UR(0.0,1.0) is less than the *gene mutation* rate parameter (GENEMUTATIONRATE). The two conditions (steps 6 and 7 of algorithm 5.8) check that the mutation has not allowed an element to deviate from the limits.

Algorithm 5.8: VARGA-v2 Gene Mutation

```
1)  Input:  The VARGA-v2 parameters from table 5.7
            Population – an empty list of chromosomes
            n – the number of variables in the MTS
2)  Each gene of every chromosome has a GENEMUTATIONRATE chance of mutating
3)  For each gene that mutates do
4)          Set v = gene value
5)          Set New_Value = N(v,σ)
6)          If New_Value < MINGENE then Set New_Value = MINGENE
7)          If New_Value > MAXGENE then Set New_Value = MAXGENE
8)  Continue
9)  Output: Population – increased in size through gene mutation
```

MINGENE and MAXGENE are application dependent, and are the minimum and maximum values a gene can take; the sum of these two parameters is usually zero. The standard deviation, σ, is chosen so that a mutated value is likely still to be within in the gene boundaries, i.e. [MINGENE,MAXGENE].

Order Mutation is responsible for altering the size of a chromosome and is similar to the version used in VARGA-v1. In steps 8 and 9 of algorithm 5.5 each chromosome has a chance of undergoing *order mutation* (ORDERMUTATIONRATE). If it does, then there is a 50% chance that either it increases in size, or decreases in size. Additions and deletions are always performed on the highest order matrix. Any new matrix that is added consists of zeros, thus ensuring that the fitness of an individual does not change when its order increases. The rational behind this is that if a random matrix was added, the order increase is more likely to reduce the fitness of the chromosome, rather than improve it. Hence a considerable part of the search space would be unlikely to be explored by VARGA-v2. The order of an individual is not allowed to exceed the specified limits. This procedure is very similar to the one detailed in figure 5.2. This new form of order mutation addresses drawback (i) as described previously in this section.

5.4.6 Selection

The selection operator is the *ranked survival* described in [Baker1985]. This is the same as the roulette wheel technique, but the chromosome's rank is used, rather than its fitness. This is because the task of finding a suitable VAR process is a minimisation problem; hence the roulette wheel would favour the worst in a population rather than the best as can be seen in the example in table 5.6, the column *RW*(*Fitness*). VARGA-v1 uses the reciprocal of fitness, the column *RW*(1/*Fitness*), see drawback (iv) as described previously in this section. From table 5.6 it can be seen that using the reciprocal of fitness results in an overly large bias towards better chromosomes.

Fitness	RW(Fitness)	RW(1/Fitness)	Rank
-10	0.018	0.341	0.182
-20	0.036	0.171	0.164
-30	0.055	0.114	0.145
-40	0.073	0.085	0.127
-50	0.091	0.068	0.109
-60	0.109	0.057	0.091
-70	0.127	0.049	0.073
-80	0.145	0.043	0.055
-90	0.164	0.038	0.036
-100	0.182	0.034	0.018

Table 5.6: VARGA Survival Comparison

Whatever the fitness, the *Rank* method produces a more gradual set of probabilities. Tests with fitness other than the example in table 5.6 have shown that the *Rank* method produces slightly better results. With *ranked survival*, equal ranking is not dealt with, since it will be very unlikely due to the fitness being real. *Elitism* (see section 2.3.1) is also used, which is the process of selecting a predefined number of the best chromosomes for survival, before the rest of the next population is selected.

5.4.7 Parameters

Table 5.7 details the parameters for the implementation of VARGA-v2.

Parameter	**Meaning**	**Value**
POPULATION	The size of the population	100
GENERATIONS	The number of generations the algorithm is run for	1000
ELITISM	The number of the fittest individuals guaranteed survival	1
ORDERCROSSOVERRATE	The chance of a chromosome being chosen for Order Crossover	0.500
GENECROSSOVERRATE	The chance of a chromosome being chosen for Gene Crossover	1.000
ORDERMUTATIONRATE	The chance of a chromosome having its order mutate	0.050
GENEMUTATIONRATE	The chance of a chromosomes gene mutating	0.005
MINGENE	The minimum value a gene can take	-1.250
MAXGENE	The maximum value a gene can take	1.250
MAXORDER	The maximum possible order of a VAR process	5
σ	The standard deviation for equation 4	0.700

Table 5.7: Genetic Algorithm Parameters for VARGA-v2

5.4.8 Population Dynamics

The population grows through the application of genetic operators as in figure 5.12. In this figure, the node at the top of the tree (*Population*) represents the starting size of the population. Two sets of children are produced through the application of *order crossover* and *gene crossover*. The mutation operators are then applied, *order* and then *gene*, to these children, where order mutation is applied after *gene mutation*. The population will increase by approximately:

ORDERCROSSOVERRATE + GENECROSSOVERRATE

The two mutation rates will not affect the increase in the population, since they are only applied to the children. Note that the *selection* operator reduces the population back to the size it was before any of the operators were applied.

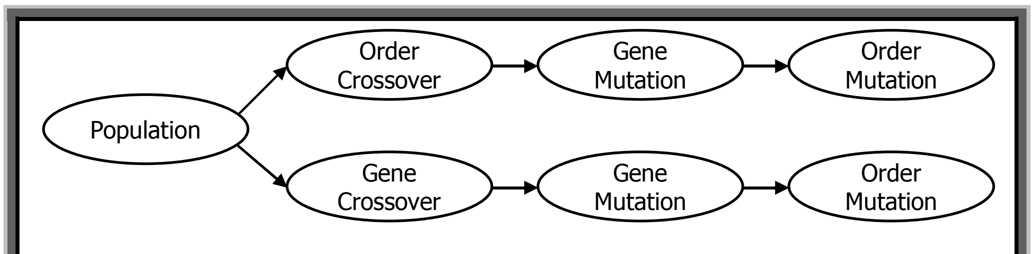


Figure 5.12: VARGA-v2 Population Growth though Operators

5.4.9 Evaluation

The proposed algorithm is evaluated against the Yule-Walker equations as implemented by S-Plus (version 2000) and the noise model. However the univariate method: the *Holt-Winters Forecasting* method will not be used (as was the case in [Swift1999a) since this was found to perform poorly. This is hardly surprising since the visual field has been clearly demonstrated to be multivariate.

The evaluation will consist of three criteria. The first is the one step ahead forecast accuracy of the methods, where the same fitness function is used to evaluate all of the methods. The second evaluates the methods based on the Weighted-Kappa metric indicating a level of agreement between observed and predicted values within an experiment where the level of agreement is in terms of a positive change, negative change or no change in deterioration. For example, a forecast error of, say, 1.0 might be considered reasonably accurate, but if each forecast were in the opposite direction, e.g. indicating an increase when a decrease actually happened, then the associated model would be of limited use. The last evaluation is on the operators that make up VARGA-v2 itself. The intention is to demonstrate the effectiveness of these operators. Nerve fibre bundle twelve is used again.

Forecast Accuracy

As was the problem described in section 5.3.9, only 34 patient's records were used in the evaluation. VARGA-v2, the noise model and the Yule-Walker equations were then run on each of the 34 patients. With the VARGA-v2 results, these were performed ten times, and then the average is taken, since genetic algorithms are stochastic and could produce an extreme result. This was added to test the new version of VARGA more thoroughly. The forecast accuracy of the Yule-Walker equations and the noise model was evaluated in the same way as with VARGA-v1. Patients are identified by a number between 0 and 81, even though only 34 patients are used in the experiments.

The experimental results are split into two groups, which are described below. Average forecast error (within figures 5.13 and 5.15) is the value obtained by dividing the forecast error by the number of forecast points. This is defined in equation 5.6.

$$\text{Average Forecast Error} = \frac{\varepsilon}{n(T - \text{MAXORDER})} \tag{5.6}$$

where ε is the total forecast errors defined in equation 5.2, n is the number of variables in the MTS, i.e. 9, and T is the length of an MTS for a patient. Note that the lower the average forecast error, the better and that the errors are for a one step ahead forecast.

Cases where the S-Plus Order is Zero. Figure 5.13 shows the results where the order of S-Plus and the noise model are equal, i.e. where S-Plus thought that the most likely model to fit the data was a zero order VAR Process. The total number of patients for which this is the case is 15 out of the 34. Here VARGA-v2 clearly does much better than the noise model in all cases, ranging from 170% to 600% better, with an average improvement of 327%. For all of these 15 patients, if inequality 5.3 is applied, the maximum order can only be 1. However S-Plus decided that a VAR(0) process was suitable, even though it had the option of choosing a VAR(1) process. This inappropriate choice of order is down to the AIC metric (used for order selection by S-Plus) failing to work adequately on a short MTS.

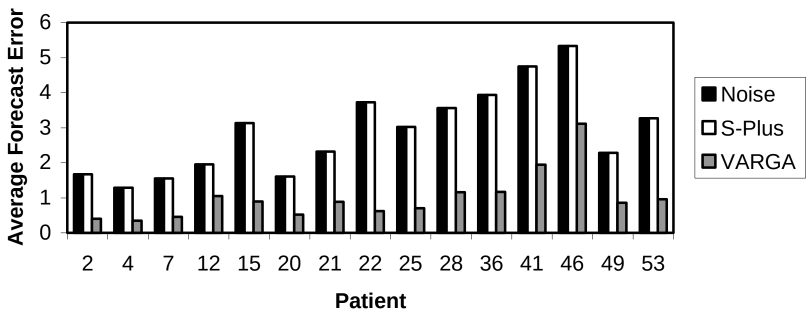


Figure 5.13: Forecast Error – S-Plus and Noise Equal

Figure 5.14 shows the order VARGA-v2 selected for the same patients. Note that the order is fractional in some cases as it has been averaged over the ten runs. Here the order is between three and five, with the average being about four.

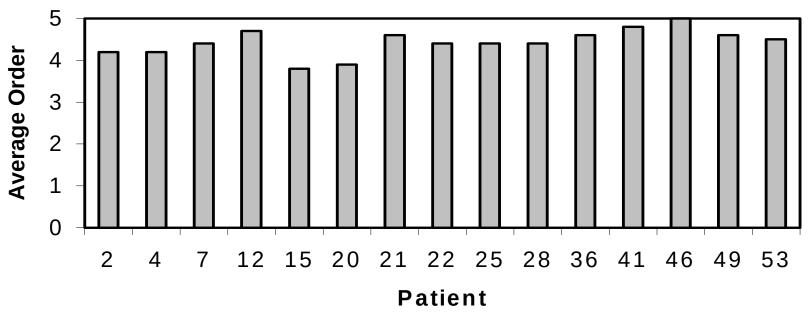


Figure 5.14: VARGA-v2 Order – S-Plus and Noise Equal

Cases where the S-Plus Order is Greater than Zero. Figure 5.15 summarises the results for those patients where S-Plus produced models with an order greater than zero (p>0). Here VARGA-v2 does better than S-Plus in 18 out of 19 cases, improving on forecasting accuracy between 200% and 450%. In the single case that S-Plus is better than VARGA-v2 (patient number 29), S-Plus produces an average forecast error about 70% of the VARGA-v2 method.

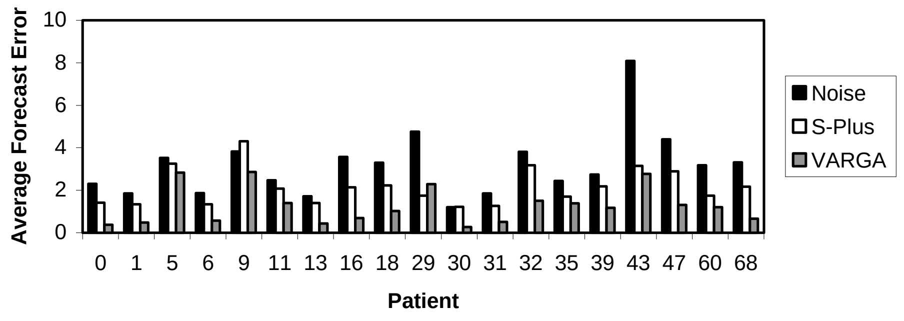


Figure 5.15: Forecast Error – S-Plus and Noise not Equal

Method	Number of Order
Noise	19 of 0
S-Plus	16 of 1, 2 of 2, 1 of 3
VARGA	1 of 3.8, 1 of 4, 1 of 4.2, 2 of 4.3, 2 of 4.4, 3 of 4.5, 2 of 4.6, 3 of 4.8, 4 of 5

Table 5.8: Order for Cases where S-Plus Order ≠ 0

The order results are summarised in table 5.8. The order found by VARGA-v2 seems to be on average between 3.5 and 5, suggesting that an order of four would be most suitable for the data. S-Plus, however, chooses lower orders because of size restrictions and problems applying the AIC metric on short time series.

All Cases Together. Table 5.9 summarises the forecast accuracy for all three methods on all of the 34 patients, including VARGA-v1; this shows that VARGA-v2 performs much better than the other three methods. The average sensitivity at each point in the visual field over all of the test results from the 34 patients is **16.907**. In table 5.9 the average forecast error per point (*By Point Value*) is given for the three the methods. Clearly VARGA-v2 is several times better than the other methods. *By Point Percentage* distributes the error by the average sensitivity and then lists the results as a percentage.

Measure	Noise	S-Plus	VARGA-v1	VARGA-v2
By Point Value (BPV)	3.048	2.475	2.209	1.144
By Point Percentage (BPV / 16.907)×100%	18.028%	14.639%	13.066%	6.766%

Table 5.9: Average Forecast Error by Point

Weighted-Kappa Results

Tables 5.11 and 5.12 show the results of the evaluation of the Weighted-Kappa metric run on the 34 patients for S-Plus and VARGA-v2. Table 5.10 shows how the 34 sets of results per method can be divided into strength categories according to appendix J.

Agreement Strength	Weighted-Kappa Count	
	S-Plus	VARGA-v2
Poor	19 (4)	0
Fair	6 (6)	1
Moderate	8 (8)	7
Good	1 (1)	18
Very Good	0	8

Table 5.10: Weighted-Kappa Strength Count Results by Category

Table 5.10 shows clearly that the majority of the VARGA-v2 results strongly agree with the deterioration trends within the data, a total of 26 out of 34, i.e. about 76%. With the S-Plus, the values in parentheses are those results with an order greater than zero. Hence it can be clearly seen that there is very poor agreement when S-Plus chooses the noise model. However, even when comparing *non-noise model* results, VARGA-v2 performs better.

Table 5.11 shows summary statistics over all of the runs. S-Plus is listed twice, once for when the selected order was zero, and then again for results where the corresponding order was not zero (i.e. not the noise model).

Summary Statistic	Weighted-Kappa Count		
	S-Plus (Order=0)	S-Plus (Order>0)	VARGA
Mean	0.201	0.359	0.684
Standard Deviation	0.222	0.173	0.153
Median	0.091	0.380	0.741
Minimum	0.000	0.026	0.219
Maximum	0.616	0.616	0.857

Table 5.11: Weighted-Kappa Strength Count Results by Value

Note that the VARGA results were averaged over the ten runs, and then summary statistics taken from these patient averages. Table 5.11 shows clearly that the VARGA-v2 one step ahead forecasts on average give a good level of agreement with the observed visual fields. The S-Plus results (order=0) are very poor, suggesting very little agreement with the original data's trends. However the S-Plus results (excluding the results where the order is zero) are much better, again showing that the noise model is the poorer choice given a selection of orders. On the other hand, the results for VARGA-v2 have a higher average for the Weighted-Kappa metric (almost twice as good as the average results for S-Plus), and have a lower standard deviation, indicating a more stable set of results. VARGA-v2 would therefore be the better choice based on this metric because its forecasts agree strongly with the trends in the original data.

Operator Experiments

The final experiment was to see how effective the various operators are. There are four operators defined in VARGA-v2, and it would be useful to see if some of them can be replaced by combinations of the others. An experiment was designed, executed and analysed towards this end. It was performed on a total of eight patients for a total number of 100,000 fitness function calls. Setting a limit on the fitness function calls is a fairer way of ensuring all of the experiments are fairly matched since each combination of operators result in a different number of offspring being generated. Fixing the number of generations would bias the experiments towards the ones that use the most number of operators. Only eight patients were chosen since each experiment is repeated ten times; hence it would take a very long time to complete on all of the patients. Eight patients were selected, which is roughly 10% of the total number of available patients, because fifteen operator experiments are run ten times on each patient, which would be a very time consuming exercise.

VARGA-v2 was executed a total of ten times with each different combination of operators. With the four operators, there are a total of sixteen possible combinations. However the case where all of the operators are zero is ignored, because VARGA would simply just run selection on each population, and not add or change any individuals, thus not increasing the number of times the fitness function was called. For the operator experiments, the VARGA-v2 parameters listed in table 5.7 are used, except for GENERATIONS, GENEMUTATIONRATE, ORDERMUTATIONRATE, GENECROSSOVERRATE and ORDERCROSSOVERRATE. GENERATIONS is determined by the number of fitness function calls (100,000) and the particular operator experiment in question. The rate parameters are defined in table 5.12. Within this table, each of the 15 experiments is given an identifier, called *Run* in column one. The binary digits (in column one) are used to indicate whether an operator has a chance of being applied. Where a *0* is indicated there is no chance, and where a *1* is indicated there is a chance according to the rates in table 5.12.

For example, in experiment number 5 (the shaded row in table 5.12) the operator *mask* is 0101. This means that the first operator and the third operator (according to the left to right ordering in table 5.12) have no chance of being applied. The remaining two operators have a chance of being applied as indicated in the respective columns of the table.

Experiment (Run)	**Gene Mutation Rate**	**Order Mutation Rate**	**Gene Crossover Rate**	**Order Crossover Rate**
0001 (1)	0.0%	0.0%	0.0%	50.0%
0010 (2)	0.0%	0.0%	100.0%	0.0%
0011 (3)	0.0%	0.0%	100.0%	50.0%
0100 (4)	0.0%	5.0%	0.0%	0.0%
0101 (5)	0.0%	5.0%	0.0%	50.0%
0110 (6)	0.0%	5.0%	100.0%	0.0%
0111 (7)	0.0%	5.0%	100.0%	50.0%
1000 (8)	0.5%	0.0%	0.0%	0.0%
1001 (9)	0.5%	0.0%	0.0%	50.0%
1010 (10)	0.5%	0.0%	100.0%	0.0%
1011 (11)	0.5%	0.0%	100.0%	50.0%
1100 (12)	0.5%	5.0%	0.0%	0.0%
1101 (13)	0.5%	5.0%	0.0%	50.0%
1110 (14)	0.5%	5.0%	100.0%	0.0%
1111 (15)	0.5%	5.0%	100.0%	50.0%

Table 5.12: Operator Application Rates by Experiment

There are a total of 1200[1] experiments for 100,000 fitness function calls. Each experiment is averaged over the combinations of operators. This is quite a normal step when evaluating a genetic algorithm, since it is a stochastic algorithm, and any particular result could be unrepresentative of its overall performance. The results for these experiments have been summarised in table 5.13. The generations average has been listed as an indication of their typical values under the different operator combinations. The algorithm is terminated when the number of fitness function evaluations exceeds 100,000 calls. In this table 5.13 *Gen.* refers to the number of generations that correspond to 100,000 function evaluations; the figures displayed are averages over the 1200 experiments.

[1]15 sets of operator experiments by 8 patients by 10 repeats per patient = 15×8×10 = 1200

Operators			**Fitness**		**Order**		**Weighted-Kappa**	
Run		**Gen.**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
0001	(1)	1979	-350.236	164.591	3.1	1.343	0.004	0.010
0010	(2)	999	-348.178	163.949	3.1	1.478	0.004	0.020
0011	(3)	665	-348.112	163.570	3.1	1.322	0.011	0.045
0100	(4)	19982	-350.236	164.591	2.8	1.303	0.004	0.010
0101	(5)	1978	-350.236	164.591	3.2	1.420	0.004	0.010
0110	(6)	999	-346.198	162.427	2.9	1.389	0.008	0.021
0111	(7)	664	-348.304	163.638	3.2	1.406	0.006	0.016
1000	(8)	1322	-194.298	181.108	4.0	0.981	0.517	0.221
1001	(9)	1979	-183.067	179.185	3.6	0.919	0.565	0.241
1010	(10)	999	-181.923	184.869	3.2	0.906	0.560	0.246
1011	(11)	664	-173.251	179.076	3.7	0.941	0.589	0.259
1100	(12)	1066	-193.859	179.828	5.0	0.112	0.510	0.219
1101	(13)	1978	-169.234	174.577	5.0	0.191	0.595	0.242
1110	(14)	999	-169.746	174.301	4.0	0.787	0.609	0.245
1111	(15)	664	-166.865	174.621	4.5	0.693	0.613	0.244

Table 5.13: Operator Experiment Results

Note that for the first seven (*Run*=1..7) of these experiments the fitness is very similar (in fact almost equal to the average of the corresponding noise models). This is because the initial populations were set to a selection of random order VAR processes where each matrix element was either zero (the noise model) or a random value within the gene element limits (50% chance of one or the other). This demonstrates that a random individual is nearly always worse than the noise model, since the final best individual is the noise model or very similar to it. The fitness is unlikely to increase beyond this point because selection ensures a zero element population before the operators (that are applied) can make any useful changes. Once this *zero-element-individual* state is achieved (or very close to it), a change is impossible in most cases. *Order mutation* may remove a matrix, which in these cases would not change fitness. If it adds a matrix then it is a zeroed matrix: again resulting in no fitness change. The *crossover* operators merely rearrange genes around, and do not alter their values.

If the first seven sets of results are ignored, then it can be clearly seen that all of the operators acting together are more effective than any other combination. It is worth noting that the best case (*Run*=15) and the worst (non-noise model) case (*Run*=8) differ by about 16.4%, the Weighted-Kappa values differ by about 18.4%. These are the two cases where either all of the operators are used, or just the gene mutation operator is used. These two figures show there is a significant improvement in results, when more genetic operators are utilised. The standard deviations for both the fitness and the Weighted-Kappa metric reduce slightly when more operators are used, indicating a slightly more consistent set of results.

The order column demonstrates a level of agreement between those experiments that have a low forecast error, being between 3 and 5. The standard deviation for the order reduces very significantly, by about 41.6%, between experiment 8 and experiment 15, which indicates that a more consistent order selection can be achieved when more of the genetic operators are used together. Finally, it would be fair to conclude from the observations about the runs 1..7 and from the runs 8..15 that the *gene mutation* operator is the most powerful operator, but works better when combined with the other operators.

5.5 Discussion

The previous two sections have presented methods for learning a VAR process from a given set of multivariate time series. This is achieved through new representations and associated crossover and mutation operators for a genetic algorithm. The results clearly show that the VARGA-v1 and VARGA-v2 models provide better methods for fitting a VAR process than the conventional statistical methods.

As demonstrated with the visual field data, the VARGA models can be applied to multivariate time series datasets where there are a small number of observations. This gives the method a wider range of applications than the standard statistical methods (e.g. than with the Yule-Walker equations). VARGA has been found to be a promising method for modelling *multivariate* time series data. The following summarises the observations and conclusions draw from the experiments regarding VARGA-v2 (the better of the two VARGA methods).

i) For short multivariate time series, conventional methods for order determination can be limited, e.g. the AIC metric. Parameter estimation methods can also fall down, i.e. matrix inversion errors and numerical instability, especially when modelling a short MTS.

ii) Although the performance of the Yule-Walker equations is better when the time series is long; VARGA-v2 still gives a better set of results for 33 cases out of 34. Further, for those applications involving a short MTS such as the glaucoma visual field data, the Yule-Walker equations (implemented in S-Plus) either cannot model them to a sufficiently high degree of accuracy or cannot model them at all.

iii) VARGA-v2's superior performance might be explained in part by the fact that approximately 60% of the resulting parameter matrices are zero. VARGA-v2 assumes that any relationships between variables are zero until proved otherwise, i.e. when a zero changes through mutation and the corresponding fitness improves. The standard statistical methods assume that there is some degree of relationship between all variables at all time lags, which appears to be counter-intuitive.

5.6 Concluding Remarks

In this chapter, VARGA, a method for learning a Vector Autoregressive process from a short and high dimensional MTS has been presented. This is achieved through a real valued matrix based representation and appropriate crossover and mutation operators for a genetic algorithm. The results clearly show that the VARGA models provide a better method for fitting a VAR process to a short MTS than the conventional statistical methods. However there are still a number of shortfalls with the proposed new methods, which shall be addressed by the further development of the VARGA paradigm in chapter 6.

6. Sparse-VARGA

In the previous chapters the difficulties in modelling short high dimensional MTS have been identified and these difficulties were addressed in the context of predicting visual deterioration in glaucoma patients using a genetic algorithm approach. The idea behind VARGA is that an nVAR(p) process could be represented as a chromosome of p $n \times n$ matrices. Suitably modified genetic operators are then applied over subsequent generations improving the population's fitness and thus finding the required parameters. However this representation can be inefficient in terms of algorithm speed. In this chapter this work is extended in three key areas: improving algorithm efficiency, ensuring accurate order selection and conducting an extensive evaluation.

6.1 The Sparse-VARGA Method

VAR subset models [Lütkepohl1993] have been used for short term forecasting. A VAR subset model is a VAR process that has one or more of its parameters set to zero, however deciding which parameters to set to zero is a difficult task. In [Bearse1998] a binary GA is used towards these ends, where the binary chromosome represents which of the parameters are set to zero, i.e. the chromosome consists of n^2p *genes* where each '*1*' means that the corresponding element in a parameter matrix is set to zero, and a '*0*' means it is left unchanged. However this method still relies upon the initial parameters and order being determined by some other method. It is proposed that all of these steps can be performed as one whole process, i.e. order determination, parameter computation and VAR subset selection. VAR processes that are highly sparse (lots of zeros for parameters) are a lot easier to work with than a densely populated VAR process.

Sparse Matrix Theory [Zlatev1991] suggests that if most of a matrix contains zeros, then one should only store the values that are not zero as opposed to all of the elements, i.e. assume that all elements are zero unless indicated otherwise. It would be more efficient to store the parameter matrices using a sparse representation; this is the idea behind *Sparse*-VARGA. This idea was also motivated from a similar representation introduced in [Ghozeil1996, Tucker1999, Swift1999b]. Sparse-VARGA stores the non-zero elements of each parameter matrix of a MAXORDER VAR process as a variable length unordered list, where MAXORDER is the maximum lag (number of parameter matrices) any VAR process under consideration can have. The Sparse-VARGA algorithm is similar to a standard genetic algorithm but has enhancements to deal with variable length chromosomes. The algorithm for Sparse-VARGA is given in algorithm 6.1 below.

Algorithm 6.1: Sparse-VARGA

```
1)    Input:  The Sparse-VARGA Parameters from table 6.2
              The fitness function from equation 5.2
2)    Create POPULATION chromosomes
3)    Sort the population in descending order according to fitness
4)    For g = 1 to GENERATIONS
5)            SHUFFLE population to create Children
6)                          Clone the population
7)                    MUTATE the Clones and add back to the population
8)            Add the Children back to the population
9)            Sort the population in descending order according to fitness
10)           SELECT the new population
11)                           Next g
12)   Output: The best VAR process; which is the chromosome from the final population with
              the largest fitness score
```

All of the operators described above are similar to those of a standard genetic algorithm (see chapter 2), where the *SHUFFLE* operator is analogous to crossover. The rest of this section explains each part of this algorithm.

6.1.1 Representation

A chromosome is an unordered variable length list of genes; each chromosome will range in size from one gene to some upper limit (say MAXSIZE). A gene is a tuple (*lag*,*row*,*col*,*value*), where *lag* is the particular parameter matrix (assuming there are

1,..,MAXORDER parameter matrices), *row* is the row of a parameter matrix, *col* is the column of the same matrix and *value* is the value the particular coefficient defined by (*lag*,*row*,*col*) takes; hence $a_{(lag)(row)(col)}=value$. GENESIZE (≤MAXSIZE) will be defined as the number of genes in a given chromosome. In order to reconstruct the parameter matrices, the genes of the *chromosome* are used to fill a set of matrices. It is assumed that all of the possible parameter matrices have zero for each element unless there is a gene that indicates otherwise. Each gene element is bounded as in table 6.1.

Element	**Type**	**Minimum**	**Maximum**
Lag	Integer	1	MAXORDER
Row	Integer	1	n
Col	Integer	1	n
Value	Real	-1.25	1.25

Table 6.1: Gene Part Types and Limits

Figure 6.1 shows the representation for Sparse-VARGA.

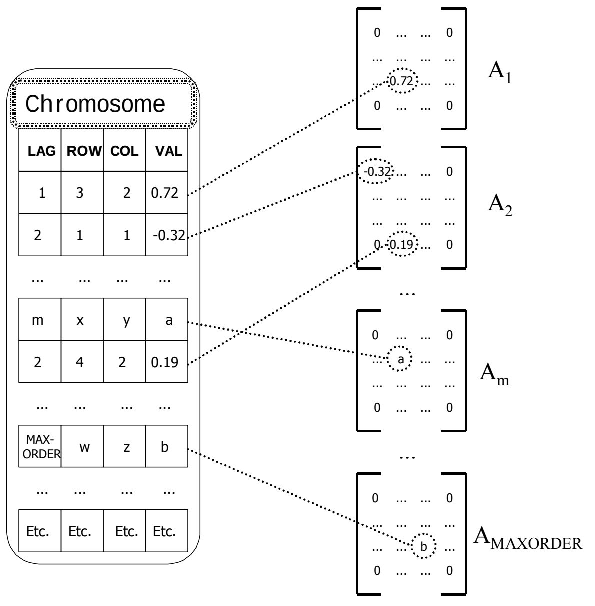


Figure 6.1: Sparse-VARGA Representation

Duplicates gene handling is described in the later sections. A duplicate is where two genes have the same value for the *lag*, *row* and *col* and parts of the gene. This could occur during the *SHUFFLE* and/or *MUTATION* part of the algorithm or even during the creation of the initial population.

6.1.2 Fitness

As with a normal GA, Sparse-VARGA needs a suitable fitness function to evaluate candidate solutions to the problem. This fitness will rate a potential solution against a given dataset using some evaluation criteria. The fitness for Sparse-VARGA is the same as the fitness for VARGA-v1 and VARGA-v2, defined in equation 5.2.

6.1.3 Initial Population

If the Yule-Walker equations can be solved for a given order for a given time series, then the results could be a good starting point for a VAR process specialised for short term forecasting. The *Noise Process* is added for good measure, and is included because this proved to be a good benchmark to compare with any other model from the experiments in chapter 5. Therefore Sparse-VARGA will be seeded with a selection of *chromosomes* that are known to represent a solution that is likely to be better than a randomly generated chromosome.

As mentioned above, there are two choices for selecting the *seeds*. The first represents the noise model, and the second are the results of the Yule-Walker equations run for order 1,...,MAXORDER. Note that for the Yule-Walker seeds it is assumed that the value for MAXORDER is less than the intended population size. Another choice must be made, and that is whether to pad out the seeds with zeros, i.e. genes where the *value* part is set to zero. This would enable *gene mutation* to affect parts of the solution that would not normally be available to it. Three sets of initial populations are generated as detailed below.

***Seeded Sparse*-VARGA (SSV)**. Chromosomes are created by solving the *Yule-Walker Equations* for an order ranging from one to MAXORDER and then using the results as a seed. Each of these seeds is padded out with zero genes where elements are not indicated, creating MAXORDER number of 100% dense matrices. The rest of the population consists of 50% chance of a noise or random individual. Each noise seed has a 50% chance of being padded out with zeros as above.

***Sparse*-VARGA-*Padding* (SVP)**. Each individual has a 50% chance of being a noise seed or a randomly generated individual. Each chromosome has a 50% chance of being padded out with zero *genes* as above.

***Sparse*-VARGA-*No-Padding* (SVNP)**. Each individual has a 50% chance of being a noise seed or a randomly generated individual. No individual is padded out with zeros.

These are the three variants on seeding that have been found to produce good results (from many test experiments), and will be further evaluated for effectiveness. The algorithms for creating these seeds are described in Appendix L.

When creating the initial population, special provision is made to ensure that there are no duplicates genes within a chromosome. For example, with a randomly created individual, a gene is only added if it is not already in the chromosome. Duplicate handling for the initial population is also described in Appendix L.

6.1.4 Shuffle

Shuffle is an operator similar to the *Uniform Crossover* operator described in [Syswerda1989]. Here all of the genes of both parents are placed into a single collection, which is then randomly shuffled. The first and the second gene element from this collection are then removed to create two single gene *chromosomes*, the starting point for two new children. The remaining genes in the collection are then divided between these two new children by deciding randomly where each one goes (50% chance for each child). Figure 6.2 demonstrates this.

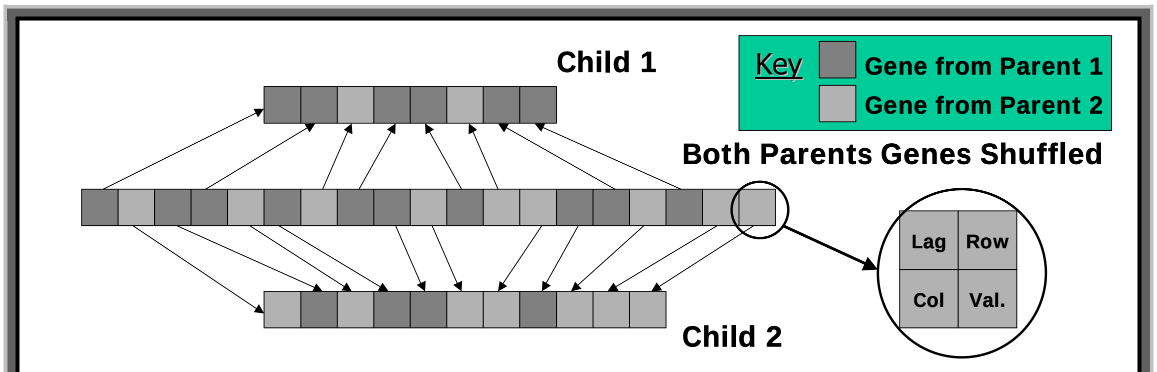


Figure 6.2: Sparse-VARGA Shuffle Operator

The representation would seem not to adhere to the *Schema Theorem* [Goldberg1989, Holland1975] since it has no sense of ordering. For example, given two *chromosomes x* and *y* where *y* is simply *x* in a different order, each would effectively be the same VAR process; this is the reason why the original version of *Uniform Crossover* (or any other standard crossover operator) will not work. However, this operator will effectively randomly divide the genes of two *chromosomes* between two new children, and when combined with the *survival* operator, will result in children being produced that contain useful genes from both parents.

As previously mentioned, this operator could result in duplicate genes. This could occur if the two parents have the same gene, i.e. one where the *lag*, *row*, and *col* part are the same. Note that each parent is assumed to have a unique set of genes before the operator is applied. During the *Shuffle* operation, before a gene is added to a child, a check is made to see if it is already contained in that child. If this is the case, the gene that came from the fittest parent is retained only, i.e. the *value* part of the gene is set to the corresponding *value* from the gene from the best parent.

6.1.5 Mutation

Mutation in the Sparse-VARGA algorithm is a two-stage process, i.e. *Gene Mutation* and *Size Mutation*. However there is a slight variation. A chromosome can only undergo either *gene mutation* or *size mutation* with a 50% chance of either occurring. Note that only the parents mutate, and they will always mutate to create a new individual.

Size Mutation causes a chromosome to increase or decrease in size by the addition or deletion of a number of genes. A chromosome has a 50% chance of losing some genes (effectively setting specified parameters to zero), or gaining new genes (effectively setting zeros to values). Note that if a gene is removed then it is randomly selected from all of the genes within the chromosome. The size of a chromosome is restricted between one and some predefined upper limit, say MAXSIZE. Figure 6.3 shows this process (for each addition or deletion).

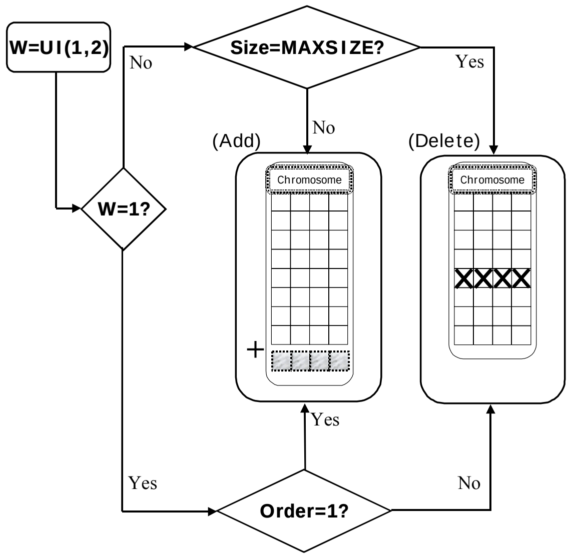


Figure 6.3: Sparse-VARGA Size Mutation

The number of additions or deletions is computed according to equation 6.3.

$$\mu = {GENESIZE}/{NSIZEMU} \tag{6.1}$$

$$\sigma = {GENESIZE}/{NSIZESTD} \tag{6.2}$$

$$N_{Mute} = Ceil(|N(\mu,\sigma)| + 1) \tag{6.3}$$

Here GENESIZE is the number of genes in the chromosome, NSIZEMU and NSIZESTD are application dependent parameters, *Ceil* is a function that returns the nearest integer that is greater than the specified parameter, and N_{Mute} is the number of size mutations. Deleting a gene from a chromosome will not result in any duplicates, but the addition of a random gene could do. To prevent this occurring, a random gene is repeatedly generated until it can be added to a chromosome without duplication.

Gene Mutation mutates a single gene of an individual. The mutation only happens to the *value* part of the gene. Algorithm 6.2 describes this procedure.

Algorithm 6.2: Gene Element Mutation

```
1)    Input:  CHROME – a Chromosome
              GENESIZE – the size of a Chromosome
              MAXGENE, MINGENE – Gene limits
2)    Set i = UI(1,GENESIZE), Set a = Gene i of CHROME
3)    Set New_Value = N(a,σ)
4)    If New_Value > MAXGENE then Set New_Value = MAXGENE
5)    If New_Value < MINGENE then Set New_Value = MINGENE
6)    Set Gene i of CHROME to New_Value
7)    Output: CHROME, the mutated Chromosome
```

where σ is an application dependent parameter that is a standard deviation for the Normally distributed *Creep Mutation* [Goldberg1990], and MINGENE and MAXGENE are the bounds an element of a VAR parameter matrix can take. Gene mutation will not result in any duplicate genes since it only changes the *value* part of a gene. This mutation is similar to that of VARGA-v2 as discussed in section 5.4.5.

6.1.6 Population Dynamics

The population grows through the application of genetic operators as in figure 6.4. One set of children is produced through the application of the *shuffle* operator. The mutation operators are applied, *size* and *gene*, to the old population consisting of the parents. Each parent has a chance of breeding (*ShuffleRate*).

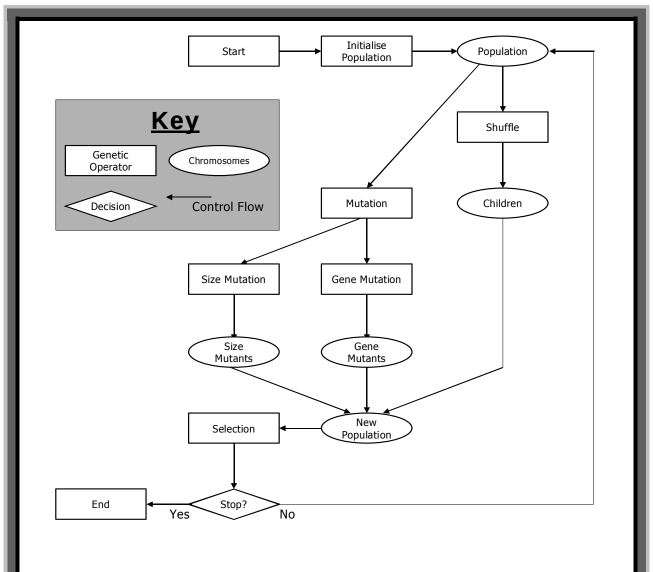


Figure 6.4: Sparse-VARGA Population Growth

The population will increase by a factor equal to 1+*ShuffleRate*. Note that the *selection* operator reduces the population back to the size it was before the *shuffle* and mutation operators were applied.

6.1.7 Selection

The selection operator is the *ranked survival* described in section 5.4.6, and used for the same reasons. *Elitism* [Dejong1975] is also used, which is the process of selecting a predefined number of the best *chromosomes* for survival, before the rest of the next population is selected.

6.1.8 Parameters

Parameter	**Meaning**	**Value**
POPULATION	The size of the population	10
GENERATIONS	The number of generations the algorithm is run for	~600
ELITISM	The number of the fittest individuals guaranteed survival	2
ShuffleRate	The chance of a chromosome becoming a parent	0.650
MINGENE	The minimum value a gene (*value*) can take	-1.250
MAXGENE	The maximum value a gene (*value*) can take	1.250
MAXORDER	The maximum possible order of a VAR process	5
MAXSIZE	The maximum number of genes a chromosome can have	$n^2 \times$MAXORDER
NSIZEMU	The mean scaling constant in equation 6.1	20
NSIZESTD	The standard deviation scaling constant in equation 6.2	20
σ	The standard deviation for algorithm 6.2	0.400

Table 6.2: Sparse-VARGA Parameters

Table 6.2 details the parameters of Sparse-VARGA. Note that GENERATIONS is an approximate figure, since all of the experiments were executed for a fixed number of fitness function evaluations.

6.2 Evaluation

The models found using the three Sparse-VARGA methods are compared with those produced by a conventional way of finding a VAR process, i.e. the solution of the *Yule-Walker* equations and a hill-climbing based method. All of the methods will be evaluated on three separate criteria. These are *Forecast Error*, *Weighted-Kappa* and *Complexity*.

Forecast error is used since it is a direct way of indicating what the accuracy of each method is, i.e. how closely each one matches the original time series. Weighted-Kappa provides a measure of whether the predicted sequence follows the *direction of change* of the original sequence. The Weighted-Kappa metric is described in appendix J. Finally, an indication of complexity will put each model into perspective, for example if one method is 0.1% more accurate than another but one thousand times slower in executing, then the less accurate but faster model is probably the better choice in many real world applications. The evaluation will be further spilt up into two distinct sections. The *forecast error* and the *Weighted-Kappa* metric are application dependent whilst the discussion of the performance of the Yule-Walker equations and *Complexity* are application independent.

6.2.1 Methods to be Contrasted

In chapter 5 S-Plus was used to solve the Yule-Walker equations. However, as noted above, there can be some problems with locating the order, and there can be some matrix inversion problems. To tackle these problems, a modified version of the Yule-Walker equations has been implemented external to S-Plus and order selection will be ignored. The Yule-Walker equations will be applied to the test data for all possible lags, and then the model with the lowest forecast error will be chosen. As with previous work, the noise model will also be implemented to put any results into context. Additionally a hill-climbing based search will be used to locate the parameters for a VAR(MAXORDER) process.

The Yule-Walker Equations

The method described by Lütkepohl in [Lütkepohl1993] will be used. Essentially, a relationship between the parameter matrices and the auto-covariance function of a VAR process exists under certain circumstances. This relationship can be exploited and rearranged to produce a set of linear matrix equations called the Yule-Walker equations. These equations are defined in chapter 2, along with a more detailed explanation of how they are derived.

Hill-Climbing

As described in chapter 4, the hill-climbing algorithm is a local search based method that iteratively improves a solution by making a single step from one point in the search space to an improved point. Usually, the starting point is randomly chosen, and then a sequence of one-step (usually random) changes is made. After each change, if the new point is better than the previous one, then the old one is forgotten, and any further changes are made to this new solution. Algorithm 6.3 describes the hill-climb procedure for learning the VAR process.

Definition 6.1. *Columns*(X) returns the number of columns the matrix *X* has.

Definition 6.2. *Rows*(X) returns the number of rows matrix *X* has.

Definition 6.3. *A*(*i*,*j*) refers to the *ith,jth* element of matrix *A*.

Algorithm 6.3: The Seeded Hill-Climbing Algorithm

```
1)   Input:  MTS of length (rows) T by n (columns), MAXCALLS – the number of algorithm
             iterations
             MINGENE, MAXGENE – The limits for a parameter matrix element
2)   Set BEST = -∞
3)   For P = 0 to MAXORDER
4)           Solve the Yule-Walker equations for CASE of Order P,
             returning A(1) to A(P) VAR parameter matrices if solvable
5)           If Solvable then
6)                   Compute the fitness according to equation 5.2 on A(1)..A(P) (FITNESS)
7)                   If FITNESS > BEST then
8)                           Set BEST = FITNESS
9)                           Set B = [A(1)..A(P)]
10)                  End if
11)          End if
12)  Next P
13)  Set b = n by 1 vector of zeros
14)  While Columns(B) < n
15)          Set B = [B b]
16)  End While
17)  Let A = B, a = BEST
18)  For i = 1 to MAXCALLS
19)          Set B = A
20)          Set ROW = UI(1,n)
21)          Set COL = UI(1,n×MAXORDER)
22)          Set VALUE = UR(MINGENE,MAXGENE)
23)          Set B(ROW,COL) = VALUE
23)          Separate B into A(1)..A(MAXORDER)
25)          Compute the fitness (FITNESS) according to equation 5.2 on
             A(1)..A(MAXORDER)
26)          If (FITNESS > a) then
27)                  Set A = B, Set a = FITNESS
28)          End if
29)  Next i
30)  Output: A=[A(1)..A(MAXORDER)] – the best VAR process for CASE based on equation
             6.2 (as far as Hill-Climbing can compute) and a – the forecast error for A
```

Definition 6.4. The notation $C = [A\ B]$ will be referred to as matrix concatenation and is the construction of a matrix C that is *Rows*(A) by (*Columns*(A)+*Columns*(B)) in size, where $C(i,j)=A(i,j)$ if $i \leq$*Columns*(A) otherwise $C(i,j) = B(i,j$-*Columns*(A)); for all valid i and j. Note that the number of rows in both matrices must be equal. If the operation is applied to more than two matrices, e.g. $[A\ B\ C]$ or $[A_1 ... A_p]$ then all of the columns of all of the matrices are concatenated in a similar manner to the two matrix case.

Lines 2-12 set up the starting point to be the best (fittest) VAR process with an order between 0 (the noise model) and MAXORDER. Lines 13-16 make sure that the resultant parameter matrices (once combined) are padded out with zeros until it is $n \times (n \times$MAXORDER) in size. Lines 17-30 are the *Hill-Climbing* loop.

In the context of finding the parameters for a VAR process that best fits an MTS, the point in the search space will be represented by an $n \times (n \times$MAXORDER) matrix, which represents the VAR parameters matrices $[A_1..A_{MAXORDER}]$. Each step will consist of randomly changing one element within this matrix to a uniformly distributed real number between the specified bounds. The worth of a matrix will be the same as the Sparse-VARGA fitness function. To place the method on an even footing with the seeded versions of Sparse-VARGA, the hill-climbs starting point is chosen from the best out of the Yule-Walker equations and the noise model. If a VAR process of lower than MAXORDER is deemed the best, then the additional matrices will contain elements that are all zero.

6.2.2 Test Data

The set of *Normal Tension Glaucoma Visual Field* data will be used to evaluate the Sparse-VARGA method, particularly in the aspects of *forecast accuracy* and *direction of change*. In chapter 5, the visual field data were clearly demonstrated to be multivariate. The value of each of the 76 field points ranges exclusively between 0 and 60, and the length of this MTS is rather short. In this chapter all of the 16 groups (points within a nerve fibre bundle) will be considered to be a separate MTS for each patient; the number of variables in each MTS will be the number of points in each group.

Additionally each patient's visual field will be modelled as a single 76 variable MTS.

6.2.3 Application Dependent Results

In this section the results of running the following seven experiments are examined:

1) The Yule-Walker (YW76) equations on the 76 points for each patient

2) The Yule-Walker (YW16) equations for each nerve fibre bundle of each patient

3) The noise model (NOISE) for each nerve fibre bundle of each patient

4) *Seeded Sparse*-VARGA (SSV) for each nerve fibre bundle of each patient

5) *Sparse*-VARGA-*Padding* (SVP) for each nerve fibre bundle of each patient

6) *Sparse*-VARGA-*No Padding* (SVNP) for each nerve fibre bundle of each patient

7) Hill-climb (HC) for each nerve fibre bundle of each patient

Experiments 4-7 are stochastic, hence each will be run ten times, and then the results averaged. This gives seven lots of 1312 (82 patients by 16 nerve fibre bundles) sets of results (experiments 2-7) and one set of 82 results (experiment 1).

These will be presented as follows:

i) General comments about the results

ii) Results for experiments involving all methods over all groups and patients (1312 comparisons)

iii) Results for methods 2-7 by group (16 comparisons). Note that the YW76 methods results cannot be considered by group since there is only one MTS

iv) Results for all methods by patient (82 comparisons)

Within the results section, the *Mean* will refer to the sample mean, the *Median* value will be the middle value when the result in question is sorted, *StDev.* is the sample standard deviation, *Min.* is the minimum value and *Max.* is the maximum value.

General Comments

It is worth noting the following details regarding the experiments and the presentation of the results.

i) With the YW16 results, 145 out of 1312 experiments were not solvable (11.05%), i.e. there were no solvable equations for all lags over a nerve fibre bundle for a patient. However 729 out of the 6560 YW16 runs (1312 × 5 lags) were not solvable (11.11%), which resulted in 33 out 82 patients series containing a non-solvable set of YW16 equations for some order.

ii) With the YW76 results, 165 out of 410 experiments were not solvable (40.24%), i.e. there were no solvable equations for all lags over all points for a patient. This is equivalent to 33 patients out of 82, which are exactly the same patients as those mentioned above; this will be discussed in section 6.2.4.

iii) The noise model is an nVAR(0) process hence the forecast error (fitness) is the sum of the absolute values of each point from MAXORDER+1 to T.

iv) For both the forecast error (fitness) and Weighted-Kappa, standard summary statistics will be displayed. For methods 4-6, the size (number of genes) will be presented as an indication of the model's sparseness.

v) The fitness will be scaled according to equation 6.4 for each set of results, so that the results are not biased against those cases with a longer time series or with a higher dimensionality.

$$Scaled\ Fitness = \frac{Fitness}{n(T - \mathrm{MAXORDER})} \tag{6.4}$$

vi) For methods 4-7 the experiments were terminated after the evaluation of 10,000 fitness calls was exceeded. This figure was chosen as it allows for a reasonable chance of convergence (to a good solutions) and is not too large as to make the total number of experiments impractical to carry out.

By Experiment

Tables 6.3 to 6.6 show the results of all the experiments considered together. Table 6.3 provides summary statistics for the forecast error (fitness) for each experiment and table 6.4 displays the Weighted-Kappa metric. Table 6.5 shows the number of times each method performed the best with regards to forecast error, whilst table 6.6 displays the same for the Weighted-Kappa metric.

Method	Fitness/Forecast Error				
	Mean	Median	StDev.	Min.	Max.
YW76	3.212	4.592	1.713	0.915	9.789
YW16	0.849	0.738	0.806	0.000	5.641
NOISE	2.940	2.513	1.734	0.000	10.833
SSV	0.556	0.334	0.646	0.000	4.207
SVP	1.653	1.208	1.588	0.000	10.688
SVNP	1.406	1.163	1.032	0.000	6.022
HC	0.562	0.341	0.644	0.000	4.223

Table 6.3: Summary Statistics for the Seven Methods by Fitness

In table 6.3 the shaded cells represents the best performance for a given method using a given summary statistic. It can be clearly seen that the Seeded Sparse-VARGA (SSV) method is slightly better than the hill-climbing method: about 1.07% better in mean, and 2.05% better in median. The next best method is the *Yule-Walker Equations by nerve-fibre bundle* (YW16). Here the mean and median error is about 50% more than with the HC method. The non-padded and then the padded versions of Sparse-VARGA (SVNP and SVP) are next in performance, followed next by the noise model and then the *Yule-Walker equations on all 76 points* (YW76). The standard deviations are smallest for the SSV and HC method (about 35% smaller than the next best – SVNP) indicating that these two methods produce a more stable (consistent) set of forecasts. It is surprising that the YW76 method performs worse than the NOISE method. This perhaps suggests that modelling the visual fields as one large dimensional MTS is the wrong thing to do, and that splitting the variable up into sub-groups (maybe by nerve-fibre bundles or by the method suggested in chapter 4) will provide much better forecasts. In most cases (not YW76) the mean is larger than the median value, and the maximum value is very much larger than the mean (and the median). This would suggest that there are many small forecast errors, and a few very large ones. Figure 6.5 shows a histogram for the forecast error for the SSV method. This diagram clearly shows that many of the errors are less than 0.4.

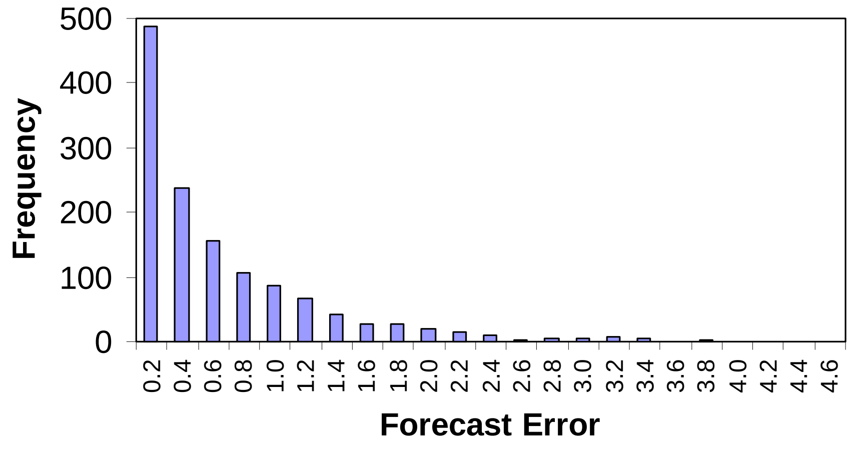


Figure 6.5: Distribution of Forecast Error for SSV Model

Table 6.4 contains the same information as table 6.3, but uses the Weighted-Kappa metric instead of forecast error (fitness). Note that the closer the Weighted-Kappa value is to one, the better (see appendix J). In table 6.4 the shaded cells represents the best performance for a given method using a given summary statistic. It can be clearly seen that the Seeded Sparse-VARGA (SSV) methods is better than the hill-climbing method: about 8.99% better in mean, and 10.34% better in median.

Method	**Weighted-Kappa**				
	Mean	**Median**	**StDev.**	**Min.**	**Max.**
YW76	0.267	0.320	0.256	-0.028	0.748
YW16	0.686	0.781	0.313	-0.032	1.000
NOISE	0.021	0.000	0.142	0.000	1.000
SSV	0.800	0.886	0.229	-0.093	1.000
SSP	0.561	0.557	0.218	-0.222	1.000
SSNP	0.561	0.562	0.217	-0.200	1.000
HC	0.734	0.803	0.209	-0.089	1.000

Table 6.4: Summary Statistics for the Seven Methods by Weighted-Kappa

Figure 6.6 shows a histogram for the results for the SSV method. It can be clearly seen that the majority of the values are greater than 0.7.

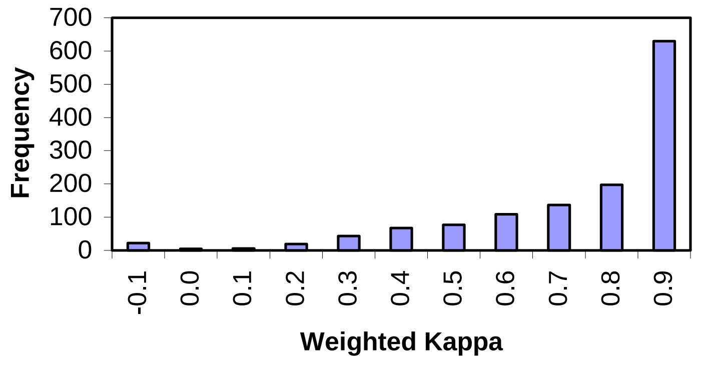


Figure 6.6: Distribution of Weighted-Kappa Metric for SSV Model

The next best performing method in terms of the Weighted-Kappa metric is the *Yule-Walker Equations by nerve-fibre bundle* (YW16). Here the mean and median error is about 7% and 3% respectively more than with the HC method. The non-padded and then the padded versions of Sparse-VARGA (SVNP and SVP) are next in performance, followed next by the *Yule-Walker equations on all 76 points* (YW76) and then the noise model (NOISE). Here the NOISE method does the worst since every single point forecast made by this model is the same, i.e. no change. Note that the standard deviation for the NOISE method is the lowest (all of the others are roughly the same) since it consistently gets a very low value for this metric due to its poor forecasts. Tables 6.5 and 6.6 look at all of the methods except YW76 and display the frequencies of how many times each method had the best performance out of the 1312 experiments. Table 6.5 is with respect to the forecast error of each method and table 6.6 is regarding the Weighted-Kappa metric.

Method Ranking by Fitness	**Count**	**Percentage**
HC=YW16	1	0.076
HC=SSV=YW16	1	0.076
HC=SSV	2	0.152
SSV=YW16	7	0.534
HC=SSV=SVNP=SVP=NOISE	17	1.296
SVNP	24	1.829
YW16	43	3.277
SVP	88	6.707
HC	263	20.046
SSV	866	66.006

Table 6.5: Method Ranking by Fitness

Almost two thirds of all of the experiments resulted in the Seeded Sparse-VARGA performing the best. The next best was the hill-climbing method, with a total number of better performances of about 20%. Surprisingly, the non-padded version of Sparse-VARGA (SVNP) was next in the rankings, with about 6.7% of the total number of experiments. The 17 cases where five of the methods drew correspond to 17 sets of MTS where all the visual field values are zero. With such a time series, the Yule-Walker equations will fail – see section 6.2.4 for a detailed discussion.

The Weighted-Kappa results as indicated in table 6.6 again show that the SSV method comes out on top, with about 57% of the highest values. It also comes equal second with the YW16 method, with a total of about 15%. This means that the SSV method is either the best or equal best in about 72% of the cases.

Method Ranking by Weighted-Kappa	Count	Percentage
SSV=SVNP	1	0.076
HC=SSV=SVP	1	0.076
SSV=SVNP=SVP=YW16	1	0.076
SSV=SVP=YW16	1	0.076
SSV=SVNP=SVP	2	0.152
SSV=SVNP=SVP=NOISE=YW16	4	0.305
SSV=SVNP=SVP=NOISE	4	0.305
HC=SSV=SVNP=SVP=NOISE=YW16	5	0.381
NOISE	7	0.534
NOISE=YW16	8	0.610
HC=SSV=SVNP=SVP=NOISE	17	1.296
SVNP	36	2.744
SVP	39	2.973
HC	72	5.488
YW16	158	12.043
SSV=YW16	200	15.244
SSV	756	57.622

Table 6.6: Method Ranking by Weighted-Kappa

It is easy to see how the YW16 method may better the HC, SVP and SVNP methods with respect to the Weighted-Kappa metric, even though it is a seed to some of these methods. This is due to a low forecast error not necessarily meaning that the change is in the right direction. Forecast error is computed as a deviation from the observed, with no consideration of sign, see equation 5.2. However in table 6.5 it can be clearly seen that the YW16 method has the best fitness (forecast error) in 43 cases (~3.2%). This should be impossible since the YW16 method should always be equal or worse than the SSV, SVNP, SSP and HC methods because of seeding and elitism. Note that the HC method uses a form of elitism: there is a population of one, no crossover and a one-gene mutation producing one offspring. When these 43 cases are examined it can be seen that the difference in method fitness is 10^{-10} or lower; i.e. the Yule-Walker equations provided an almost perfect fit. With the four stochastic methods (that should have performed at least as well as the YW16 method), each is averaged over 10 runs. This combined with scaling the error as in equation 6.4 above, results in some cases having a small rounding error. In fact the results from these methods are all the same as expected, when they are compared to a very large number of decimal places.

By Group

In this section, the forecast error and Weighted-Kappa results will be examined by nerve fibre bundle. However, only the mean will be used as a summary statistic, because since there are 16 MTS groups, tables 6.3 to 6.6 and figures 6.5 and 6.6 would have to be repeated 16 times. In tables 6.7, 6.8 and 6.9, the shaded row represents the method with the best performance for the metric in question.

In table 6.7 it can be clearly seen that the SSV method produces the lowest forecast error. However the SSV method is followed very closely by the HC method, which is better than SSV in 2 out of 16 groups. However the difference in forecast error in these two cases is very small. Table 6.8 splits the results in table 6.7 into groups and then displays the frequency that a method performed best or equal best. It can then be seen that the method that comes top for all of the groups is the SSV method followed by HC. However the HC method only performs the best in under a third of the cases.

Group	Method					
	HC	SSV	SVNP	SVP	NOISE	YW16
0	1.333	1.317	1.614	1.851	3.207	1.885
1	0.431	0.422	1.250	1.413	2.708	0.578
2	0.411	0.399	1.213	1.445	2.727	0.623
3	0.761	0.752	1.306	1.557	2.718	0.904
4	0.458	0.453	1.332	1.563	2.885	0.751
5	0.402	0.410	1.597	1.837	3.095	0.948
6	0.428	0.422	1.631	1.873	3.395	0.963
7	0.567	0.561	1.546	1.849	3.442	0.832
8	0.667	0.655	1.391	1.650	3.030	0.872
9	0.731	0.721	1.517	1.765	3.058	0.909
10	0.384	0.371	1.372	1.655	2.933	0.627
11	0.314	0.307	1.409	1.673	2.851	0.670
12	0.418	0.430	1.473	1.734	2.809	0.881
13	0.435	0.428	1.328	1.595	2.904	0.677
14	0.744	0.738	1.329	1.577	2.741	0.910
15	0.512	0.510	1.187	1.417	2.530	0.666

Table 6.7: Mean Forecast Error by Group

Methods	Groups (Nerve-fibre Bundles)																Totals
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
HC	16	17	10	21	17	24	17	12	16	15	12	14	31	10	16	15	263
HC=SSV	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2
HC=SSV=SVNOPAD =SVPAD=NOISE	2	0	4	0	0	4	3	2	1	0	0	0	0	0	0	1	17
HC=SSV=YW16	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
HC=YW16	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
SSV	44	58	50	46	52	48	56	58	57	64	65	63	43	61	52	49	866
SSV=YW16	0	0	1	0	1	0	0	0	0	0	0	1	0	0	2	2	7
SVNOPAD	7	0	2	5	1	0	1	0	0	2	0	0	0	1	2	3	24
SVPAD	13	4	8	6	7	4	5	5	3	0	3	2	5	8	7	8	88
YW16	0	2	5	4	4	2	0	5	5	0	2	2	3	2	3	4	43
																	1312

Table 6.8: Method Forecast Error (Fitness) Rank by Group (Nerve-Fibre Bundle)

Methods	**Groups (Nerve-fibre Bundles)**																**Totals**
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	
HC	2	2	3	5	5	13	8	3	1	1	0	3	17	4	4	1	72
HC=SSV=SVNOPAD =SVPAD=NOISE	2	0	4	0	0	4	3	2	1	0	0	0	0	0	0	1	17
HC=SSV=SVNOPAD =SVPAD=NOISE=YW16	0	0	1	1	1	0	0	0	0	1	0	0	0	0	0	1	5
HC=SSV=SVPAD	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
NOISE	2	0	1	1	2	1	0	0	0	0	0	0	0	0	0	0	7
NOISE=YW16	3	0	1	1	0	2	0	0	1	0	0	0	0	0	0	0	8
SSV	32	47	43	41	51	49	58	49	37	38	57	62	51	53	39	49	756
SSV=SVNOPAD	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
SSV=SVNOPAD=SVPAD	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	2
SSV=SVNOPAD =SVPAD=NOISE	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	4
SSV=SVNOPAD =SVPAD=NOISE=YW16	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0	0	4
SSV=SVNOPAD =SVPAD=YW16	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
SSV=SVPAD=YW16	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
SSV=YW16	3	16	12	17	11	6	5	14	20	24	12	11	7	10	18	14	200
SVNOPAD	13	2	2	5	3	0	1	0	0	2	0	0	1	2	2	3	36
SVPAD	10	1	5	2	1	2	2	2	3	1	0	0	0	2	6	2	39
YW16	13	14	9	7	6	4	5	12	18	15	11	5	6	10	12	11	158
																	1312

Table 6.9: Method Weighted-Kappa Rank by Group (Nerve-Fibre Bundle)

Table 6.9 is the same format as table 6.8, but the frequencies are given for the Weighted-Kappa metric. Within table 6.9 it can be seen that in all of the cases (groups), the SSV method performs the best (or equal best), i.e. for each group, it has the best Weighted-Kappa value out of the 82 patients. None of the other methods come near to its performance.

Table 6.10 shows the average chromosome size for the methods. In this table the chromosome sizes for the SV, SVP, SVNP and YW16 methods are listed. *Group* is the nerve fibre bundle that the size pertains to, *Max.* is the maximum possible chromosome size, *Best* is the size of the fittest chromosome, averaged over all of the experiment for a particular group, and *Average* corresponds to the average of the populations average size over each generation. Note that the maximum chromosome size (*Max.*) is the size of the HC method. For the YW16 method, the size is computed by multiplying the average lag by the number of variables in the MTS squared, n^2. Note that the noise model has a size of zero. It can be seen that the best size for the Sparse-*VARGA* methods is usually smaller than the average size. The models in descending order of size are as follows: HC, YW16, SSV, SVP and SVNP. Note that technically the noise model is the smallest. The implications of the size will be discussed in section 6.2.4.

Group		**Max.**	**SSV**		**SVP**		**SVP**		**YW16**
ID	**n**		**Best**	**Average**	**Best**	**Average**	**Best**	**Average**	
0	2	20	15.3	15.0	11.4	9.8	11.2	9.4	19.1
1	4	80	74.0	74.2	25.5	19.4	24.9	18.6	63.4
2	4	80	67.2	67.2	23.0	18.1	22.5	17.2	62.6
3	3	45	37.0	36.9	18.6	14.7	18.3	14.2	41.8
4	5	125	98.3	98.3	26.2	19.8	25.7	19.1	81.9
5	9	405	330.2	330.4	43.2	33.1	39.9	29.3	155.9
6	7	245	204.7	204.8	34.4	26.2	33.1	24.5	128.7
7	4	80	73.7	74.0	25.6	19.8	25.1	19.1	61.5
8	3	45	42.9	42.9	21.7	16.9	21.3	16.3	41.6
9	3	45	43.3	43.3	20.8	16.1	20.7	15.8	41.1
10	4	80	77.2	77.6	26.0	19.6	25.7	19.1	64.2
11	7	245	236.5	237.2	35.3	25.7	34.8	25.0	119.4
12	9	405	360.9	360.2	38.2	27.3	37.9	26.7	163.1
13	5	125	104.7	105.1	28.0	20.9	27.4	20.1	83.2
14	3	45	39.1	39.1	19.8	15.5	19.6	15.1	41.0
15	4	80	71.9	72.1	24.5	18.8	24.1	18.2	64.2
Average:		100.0%	87.8%	87.8%	30.3%	23.6%	29.7%	22.8%	73.1%

Table 6.10: Method Model Size

By Patient

The final analysis of the results is from a patient perspective. Here the results are averaged by patient, giving 82 sets of results. Appendix M lists both the forecast error and Weighted-Kappa metric averaged over each patient. The column *Best* refers to which method provided the best average for a given patient. Any Yule-Walker equation estimate that could not be computed is given a penalising forecast error of 100.00 or a Weighted-Kappa of zero. Table 6.11 summarises these results, showing the number of times each method performs the best for each metric. Full details can be found in Appendix M.

It can be clearly seen that the SSV method performs best. However it is interesting to note that the YW76 method performs third best. This will be considered further in section 6.2.4.

Method	Count of 1st Rank	
	Forecast	Weighted-Kappa
HC	11 (13.5%)	1 (1.2%)
NOISE	0 (0.0%)	0 (0.0%)
SSV	69 (84.1%)	80 (97.6%)
SVNP	0 (0.0%)	0 (0.0%)
SVP	0 (0.0%)	0 (0.0%)
YW16	0 (0.0%)	0 (0.0%)
YW76	2 (2.4%)	1 (1.2%)

Table 6.11: Summary of Method Ranking by Patient

Discussion

Forecast Error (Fitness) Results. When looking at the forecast error results as a whole (table 6.3) it initially looks like there is little difference between the HC and SSV methods. However it is only when these results are considered by looking at the frequencies of which performs best from a nerve-fibre bundle (tables 6.7, 6.8 and 6.9) and a patient perspective (table 6.11), that it can be seen that the SSV method clearly performs better. This suggests that there are many low forecast error results, but a few very large error results for this method.

One unexpected result is that the YW76 method performs the best (when considered by patient with the other method) for 2 out of 82 patients (table 6.11). If all of the patients where the YW76 method could not provide an answer are ignored and the remaining patient's average point sensitivity is correlated with the point forecast error for the YW76 method using *Spearman's Rank Correlation Coefficient*, a value of –0.417 is obtained. This is significant to the 1% level, suggesting there is evidence to support that as the average point sensitivity increases, the YW76 point forecast error decreases. For the two cases where the YW76 method performs the best, the patients are ranked 13 and 23 out of the 82 in terms of average point sensitivity.

This could be due to relationships existing between points in patients with better sensitivity that lie outside the nerve fibre bundle arrangements. The YW76 method could model these relationships whilst all of the other methods are restricted to modelling the visual field points according to the layout of each nerve fibre bundle.

Weighted-Kappa Results. The results for the SSV method are much better when comparing the directional agreement between observed and predicted deterioration using the Weighted-Kappa metric (tables 6.4 and 6.6). It is worth noting that this agreement is as important as forecast error, since the consequences of altering medication or performing surgery is quite severe. Most glaucoma clinicians make decisions based upon whether the visual fields are seen to be improving, deteriorating or remaining stable. Forecast error can give an indication of how accurate the forecasts are, but not if they were in the right direction of change.

6.2.4 Application Independent Results

In this section the results from performing complexity analysis of the methods and the performance issues arising from using the Yule-Walker equations are examined.

Complexity

No attempt to perform a formal analysis of algorithm efficiency will be made, but instead the focus will be on the most expensive part of all of the methods: the forecast error evaluation. The evaluation of the forecast error (fitness function) has a computation complexity in terms of arithmetic operations documented in table 6.12.

Method	Complexity	Equation
YW16, YW76	$(T-\mathrm{MAXORDER})\times\left(\mathrm{O}(n)+\mathrm{O}(n^2 lag)\right)$	(6.5)
SSV, SVP, SVNP	$(T-\mathrm{MAXORDER})\times\left(\mathrm{O}(n)+\mathrm{O}(GeneSize)\right)$	(6.6)
HC	$(T-\mathrm{MAXORDER})\times\left(\mathrm{O}(n)+\mathrm{O}(n^2\mathrm{MAXORDER})\right)$	(6.7)
NOISE	$(T-\mathrm{MAXORDER})\times\mathrm{O}(n)$	(6.8)

Table 6.12: Forecast Error Computation Complexity by Method

In table 6.12, T is the length of the time series in question, MAXORDER is the maximum permissible order for a VAR process modelling the time series, O(·) is a function indicating the complexity of an operation, n is the number of variables in the time series, *lag* is the order of the VAR process being represented by a method and GENESIZE is the number of genes within a chromosome of Sparse-VARGA. The term (T-MAXORDER) represents the number of sets of forecast error that is evaluated, O(n) represents the computations involved in taking the difference between observed and predicted error, and the terms concerning O(n^2…) involve the application of equation 5.2, i.e. computing the forecast.

Note that the Sparse-VARGA methods can have their fitness calculated in a lower computation time than the other methods because of the nature of their representation, i.e. sparse matrices (see algorithm K.1 in Appendix K).

Table 6.12 puts all of the methods in perspective. Since the Yule-Walker equations (YW16 and YW76) and the noise model are used as seeds for the Sparse-*VARGA* based methods and the hill-climbing method, further analysis will only be done on the SSV, SVP, SVNP and HC methods. These four latter methods also iterate many times, whereas the three previous methods are deterministic. The relative complexity of the three Sparse-VARGA based methods will be compared with the hill-climb based search. Since the term (T-MAXORDER) is common to all of the methods it can be ignored; also the term O(n) is less than any term involving O(n^2) and hence can also be ignored. This leaves only the term O(n^2MAXORDER) for the hill-climb and the term O(GENESIZE) for the Sparse-VARGA based methods. If the average size figure from table 6.10 is used for GENESIZE then a graph can be drawn for the four methods for each nerve fibre bundle. Note that 1≤GENESIZE≤n^2MAXORDER is always true. The Sparse-VARGA algorithm ensures there is at least one gene in a chromosome (but this may be zero valued) and that the *lag* part of any gene cannot exceed MAXORDER, therefore for an n-dimensional MTS, n^2MAXORDER is the maximum number of genes, i.e. a gene for each parameter of all the $n\times n$ parameter matrices.

Although the Sparse-VARGA family of applications appears to be less complex than the HC method, it would be interesting to see how they compare in practice. If the average chromosome length from table 6.10 is used for the value of GENESIZE, an indication of the relative complexity of the Sparse-VARGA and HC methods can be seen as in figure 6.7. In this figure, the HC method's size corresponds to the maximum size at all times, hence the relative complexity being equal to n^2MAXORDER, where n is dependent upon the dimensionality of each nerve fibre bundle. It is quite clear that the SVP and SVNP methods have a tendency to create small (sparse) models, and hence have a very fast and efficient forecast evaluation (less than 20% of the maximum). However the SSV method seems to have an average of about 87.8% the maximum.

Note that the non-padded version of Sparse-VARGA (SVNP) produces slightly smaller models than the padded version (SVP), and also performs moderately worse (in terms of forecast error and Weighted-Kappa).

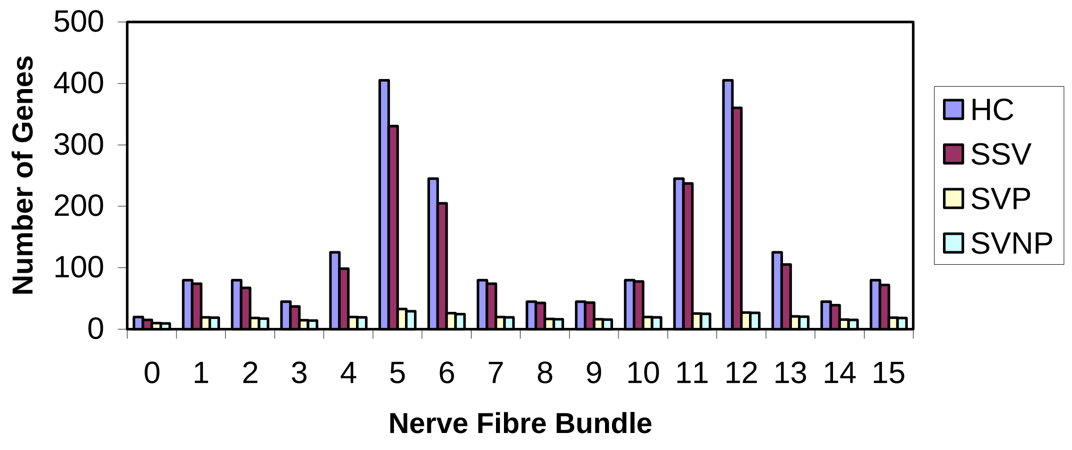


Figure 6.7: Relative Method Complexity

Finally, it is worth noting that the YW16 and YW76 methods produce models that are 100% populated, but the order is less than the maximum specified order. The SSV, SVP and SVNP methods produce models that are of the maximum order (they do not have to, but do so in most of the cases). Densely populated matrices mean that the Yule-Walker results are difficult to interpret, since there are too many parameters to visually inspect. This is not the case with the SVP and SVNP methods, since the parameter matrices (especially in higher dimensionality models) are highly sparse, thus relationships are easier to identify (this is true to some extent with the SSV method results). This could explain why the YW method can be improved upon by creating VAR subset models. These are models where some of the parameters of a VAR model are set to zero to improve the accuracy of the model. It has been acknowledged that the results of the statistical time series methods might not always return the desired results because techniques such as correlation (used in the *$\Gamma(h)$* function), and matrix inversion can be subject to inherent inaccuracies and biases, especially when the time series is short.

Performance of the Yule-Walker Equations

For short length MTS the Yule-Walker equations are often not solvable, for example in about 11% of the visual field cases (145 cases), i.e. the YW method could not find a solution for 145 multivariate time series over all permissible lags.

For a matrix B to be invertible (a requirement for the Yule-Walker equations to be solvable):

i) The columns and rows of B are linearly independent.

ii) For a given vector $\underline{b}$, $B\underline{x} = \underline{b}$ has a unique solution.

Note that other conditions not pertinent to this method are omitted; see [Stewart1998] for a full list. Therefore the Yule-Walker equations produce either a set of equations that do not have a unique solution, or a matrix being inverted has two or more rows or columns that are linearly dependent. Note that with the visual field data, 17 out of the 145 cases are where all of the values in the time series are zero, hence it is not the case that the YW method's failures are due to the time series being completely zero, i.e. total lack of vision on part of the eye (a whole group). More investigation is needed to see why this is the case, but it is suggested that this is likely to happen when the time series is short and high dimensional.

For example, with the visual field dataset, there are a large number of variables in nerve fibre bundle 5, which is where glaucoma usually originates. Hence there is a good chance that one or more of the variables will be mostly zero as the condition progresses (a blind spot developing). Hence a column of the matrices in equation 2.8 could easily be zero. This will mean that condition (ii) above will be violated, since there will not be a unique solution (possibly none at all). The patients that the Yule-Walker equations fail on are those that have a lower average sensitivity. This is in cases where the condition is terminal, i.e. certain visual field points have a sensitivity of zero, indicating total blindness in part of the eye. The YW76 method fails whenever there is a failure with any associated YW16 method because a column of zeros in a nerve fibre bundle will be a column of zeros in the corresponding 76 variable MTS.

6.3 Testing the Methodology

This section combines all of the work presented in the previous three chapters and this chapter. Within chapter 3, correlation mining was introduced and then tested with promising results. Within chapter 4, the problem of grouping MTS was described, a solution suggested and it was further shown that when combined with the correlation mining procedure, very good results could be achieved on simulated data. Within chapter 5, the problems of modelling short and high-dimensional MTS is addressed, using the VAR process to short term forecast the visual field dataset. Once this technique had been shown to work well, further improvements were made to extend the method into modelling VAR subset models, again with very good results. This section bridges the techniques from the correlation mining and grouping chapters with the modelling work, showing that the combined methodology works well the visual field data. The choice of methods is simply the techniques that provided the best results in the previous work, a more thorough analysis using all of the methods presented would be too time consuming to be practical. The best temporal correlation mining method, grouping method and forecasting method were selected for evaluation. This was the evolutionary programming method from chapter 3, Falkenauer's grouping genetic algorithm from chapter 4 and Seeded Sparse-VARGA from this chapter.

The dataset used for the experiments is a subset of visual field data from ten patients who all have 20 tests. Each set is split up into an 18 length time series for training, and the remaining two tests are for testing purposes. This set of patients has been chosen since it is the largest number of patients who have the greatest number of tests (see appendix A). The number of correlations to be searched for needs to be specified, and as in chapter 3, this was set at 200 for the parameter *r*. The reason for choosing this value is to allow for enough correlations to potentially reconstruct the nerve fibre bundles, i.e. the layout is used as a guideline for possible grouping arrangements. The evolutionary programming method was applied using Spearman's rank correlation coefficient, which proved to produce slightly better results than using Pearson's. The evolutionary programming correlation mining method was executed once on each patient and the resultant set of 200 correlations passed to the grouping genetic algorithm as the input matrix *Q*. Once the groups had been determined for a patient, the univariate groups (those groups containing a single member) were stripped out. Seeded Sparse-VARGA was then run on each group using the parameters from table 6.2. For each patient the *whole* procedure was performed ten time and the results averaged.

Table 6.13 below is a short summary of the results of short-term forecasting after applying the whole methodology to both test and training datasets. The columns *Mean*, *Median* and *StDev*. have their usual meaning and apply to the forecast error, and WK is the Weighted-Kappa metric applied to the directional accuracy of each point forecast.

Patient ID	**Training Data**				**Test Data**			
	Mean	**Median**	**StDev.**	**WK**	**Mean**	**Median**	**StDev.**	**WK**
0	1.094	0.429	1.897	0.524	1.271	0.738	1.577	0.246
6	1.587	1.057	1.863	0.105	1.300	0.998	1.151	0.374
9	3.730	1.581	6.906	0.026	1.257	1.003	0.936	0.000
13	0.745	0.365	0.920	0.698	1.673	1.283	1.293	-0.035
15	1.252	0.852	1.443	0.361	1.205	0.628	1.191	0.071
22	1.232	0.370	2.138	0.484	1.332	0.969	1.445	0.021
25	1.129	0.170	2.124	0.481	1.031	0.366	1.136	0.333
32	1.288	0.702	1.583	0.659	1.399	1.155	1.187	0.167
60	1.644	1.266	1.419	0.160	1.740	1.684	1.148	0.081
68	1.423	0.814	1.737	0.492	1.626	1.184	1.227	0.122
Average:	1.512	0.761	2.203	0.400	1.383	1.001	1.229	0.138

Table 6.13: Methodology Results for Ten Patient Subset

From table 6.13 it can be seen that for 9 out of the 10 patients the mean forecast error lies in the range of [0.7,1.7] for the training data, and for 10 out 10 patients the mean error was in the [1.0,1.8] range for the test data. It is interesting to note that patient ID 9 has a very large mean forecast error for the training data, but a low forecast error for the test data. If patient ID 9's training error is ignored then the average across the remaining 9 patients mean forecast error is reduced from 1.512 to 1.266, which is lower (by 8.5%) than the corresponding error in the testing samples. This mean error seems consistent across both the training and test datasets. It can also be seen that the median error increases although the standard deviation of the errors decrease from training to test datasets. However the average Weighted-Kappa rating is *Moderate* in the training dataset but drops two categories to *Poor* within the test dataset.

6.4 Concluding Remarks

A method called Sparse-VARGA has been presented which improves upon the Yule-Walker method and works where this method cannot. It has been shown that by integrating the order selection process into the whole procedure, better short-term forecasts can be produced, and the resultant parameter matrices are a better aid to explaining relationships between variables. Lastly the method has been demonstrated to be a good model for forecasting Normal Tension Glaucoma deterioration.

Sparse-VARGA is a novel method in that it views a VAR(P) process not as a series of densely populated matrices, but as a collection of parameters that are spread over a series of zero matrices from order p=1 to some maximum order, p=MAXORDER. The advantage of this approach is that there is scope for handling much more complicated types of the VAR process family. For example, Sparse-VARGA can identify VAR subset models and models where some of the parameter matrices are zero, e.g. a VAR(3) process where only the first and third parameter matrices are populated. Additionally Sparse-VARGA introduces some new operators that exploit this representation. This approach has advantages over simply storing a VAR process as a series of matrices, as in chapter 5. That representation had the problem of having to store a vast number of parameters as the dimensionality of the time series increases, along with the previously mentioned order bias problems. The sparse matrix representation of Sparse-VARGA has the additional advantage of having a fitness function where the computational complexity is proportional to the number of genes in the chromosome.

The extensive results show that Sparse-VARGA performs better than the comparative methods with regards to *forecasting accuracy* and *direction of change*. In addition, the method has better computational complexity than the hill-climbing method. Clearly Sparse-VARGA is an appropriate method for modelling short length high dimensional multivariate time series, with very promising results on the visual field dataset. The results of the visual field dataset has also been shown to have a low forecast error which is consistent across both training and test datasets for a subset of the patients. However there are many other datasets that are both short in length and high in dimensionality to which this approach could be applied. For example work could be carried out to see how the methods presented in this chapter can be applied to the analysis of virus gene expression data [Jenner2001].

Finally, the method can be improved in the following areas. Firstly, seeding techniques and strategies will be studied to try to produce models with a smaller number of parameters. Secondly, a variety of fitness functions could be investigated to learn models that can forecast for several steps ahead, rather than just one; and lastly attempts will be made to extending this work by developing spatio-temporal models [Pfeifer1980a, Pfeifer1980b]. These will be discussed further in the next chapter.

7. Conclusions

Model selection is arguably the most important and most difficult aspect of model building, and yet is the one where there is least help [Chatfield1995, Hand1994]. This situation is even worse for modelling short multivariate time series data. Such datasets are common in many fields such as medicine, finance and science, and any advance in modelling this kind of data would be beneficial. There is little work in this important area. This thesis presented some of the important first steps. Novel algorithms for decomposing high-dimensional MTS have been developed. New methods have been provided for modelling such MTS data which traditional statistical methods have found difficult to deal with. This has led to a three-step computational framework for modelling this type of data. In addition to visual field deterioration prediction, the methods have found successful applications in several areas, including bioinformatics (gene expression data) [Kellam2001] and industry (user allocation to email servers) [Counsell2001].

This thesis has presented a methodology which is orientated towards the modelling and forecasting of high dimensional, short length time series datasets, and has demonstrated its worth on an appropriate and important real-world medical application along with a variety of synthetic datasets. Chapter 3 presented a fast approximate way of locating relationships in an MTS through a novel use of evolutionary programming and correlations. This correlation list was then used in chapter 4 to decompose the MTS into a subset of smaller dimensionality MTS that have strong relationships between members and weak relationships between non-members. Finally the sub-MTS found in using the best method from chapter 4 are used as the basis of a suitably modified genetic algorithm which found the order and parameters of a VAR process in a single step as presented in chapter 5. In chapter 6 an advanced method for finding an evolutionary computation based VAR process is presented, and all of the parts presented in the previous chapters were combined into a single methodology. It was demonstrated that all of the elements work efficiently together producing excellent results when combined.

This chapter is organised as follows. Firstly, the main results presented in this thesis are summarised. Then the main contributions and limitations of the work are described. Finally an in-depth look at the possibilities for further research and the possible improvement of the methods are presented.

7.1 Contributions

This section is an overview of the main results presented in this thesis.

7.1.1 Correlation Mining

The main goal of the correlation mining procedure was to locate a specified number of correlations from an MTS in as short a time as possible but as accurately as possible. Several methods were extensively evaluated and it was concluded that an evolutionary computation based method was the best choice of algorithm for this type of problem. The basis of the method was to maintain a population of candidate best correlations over the specified time lags and to improve them using evolutionary programming with self-adapting parameters. This method converged to a higher average at a faster rate of convergence than two search based methods; and compared favourably to the exhaustive search method. The evolutionary programming method reached an average that approached that of the exhaustive search in a fraction of the time.

Another main result from this technique is a selection of correlations depicting relationships between visual field variables as shown in figure 3.7. The correlations correspond to relationships that would be expected to exist as a result of previous work in ophthalmology. The results in chapter 3 correspond to using the Fisher z-transformation to combine correlations to get a view of the relationships for the condition as opposed to those for an individual patient.

The correlation mining method provides a quick and approximate way of locating a predefined number of correlations from an MTS, over a specified maximum time lag. The method is particularly appropriate to applications where the speed of retrieving these correlations is essential.

7.1.2 Variable Grouping

The grouping genetic algorithm provides a method for partitioning a number of variables in mutually exclusive and highly related subsets. This approach is novel in that it uses a grouping specific crossover, combined with a metric that is based upon cross-correlations, thus providing a fast and efficient way of solving such a variable grouping problem. Within chapter 4, five grouping methods are proposed and these are combined with three of the correlation mining techniques from chapter 3. This combination of 15 methods was applied to six synthetic datasets that have a known structure, and the results were extensively analysed to determine the best combination of methods. The best grouping method used Falkenauer's grouping genetic algorithm to decompose the MTS.

Detailed analysis was also performed on the correlation mining procedure and how the method parameters affect the performance of the grouping algorithms. The analysis links the confidence of getting the number of correlations sought for with respect to how long the algorithm is allowed to execute for. This provides a useful guideline to how many correlations should be located based upon the expected group size.

The grouping part of the methodology, when applied to the six pre-determined groups, (a mixture of VAR processes and DBN's) produced very promising results. All of the proposed search based methods for MTS grouping performed well, demonstrating that the strategy of using an approximate correlation search was sound. The only discrepancies seemed to be the odd variable being assigned to a group on its own and a DBN being split into two sub-models. However, with the latter problem, the split was at a logical point (see section 4.5 and appendix H).

When the grouping method is applied to the visual field dataset promising results were obtained in that the points within the found groups tend to be those that would be expected. The comparison between the groups generated for each patient again showed promising results, those with good sensitivity and hence good vision are more consistent during the visual field tests, resulting in similar grouping between those points which are related. With the low sensitivity (and low vision) patients the results of the visual field test tend to be more variable and hence there are dissimilar groupings.

7.1.3 VAR Model Fitting

With VARGA-v1 it was demonstrated that the visual field data was clearly multivariate and that a VAR process is a suitable model for forecasting the next test results for each patient. Because of the order and parameter fitting problems which the traditional statistical methods of VAR model fitting have with short high-dimensional multivariate time series, it was shown that a modified genetic algorithm can be used to locate both the order and parameters in a single step and that the forecast accuracy results are very promising.

VARGA-v2 developed this idea further, with more accurate results. Another evaluation criteria was added at this stage in the development of the methodology: the Weighted-Kappa metric. This metric measured whether a forecast was in the correct direction, e.g. forecasting a decrease when one was observed. VARGA-v2 provided highly accurate results both for forecast accuracy and directional accuracy.

The final version of VARGA, Seeded Sparse-VARGA performed better than all of the rival methods it was compared against, including the previous two versions of VARGA. In addition to the forecast accuracy and Weighted-Kappa metric, Seeded Sparse-VARGA was also evaluated on algorithm complexity. The results were once again promising; Seeded Sparse-VARGA performed better on all accounts.

Seeded Sparse-VARGA is a unique and interesting way of defining a VAR subset model using sparse matrices. This method does not find all of the parameters of a VAR process (which can be a very large number) but assumes they are all zero unless information is available to show otherwise. The method can be tailored to find VAR process which meet pre-defined criteria. The method introduces a new representation along with associated crossover and mutation operators. The method has the advantage of being able to find a suitable VAR process regardless of whether the seeding is successful or not, i.e. whether the Yule-Walker equations can be evaluated or not. Seeded Sparse-VARGA has proved to be a fast and accurate way of finding VAR processes (or VAR subset models) to suit a particular purpose in a fast and accurate manner. Order determination does not prove to be the problem as can be the case when applying the standard statistical methods to short and high-dimensional time series.

7.1.4 Summary

Much of the work in decomposing MTS variables has been restricted to the use of distance matrices to cluster similar variables. This type of approach does not explicitly take temporal relationships between variables into consideration. The proposed grouping methodology has presented a principal way of utilising information regarding variable dependencies over time and allowed a variety of heuristic search methods such as evolutionary computation methods to be investigated. It is believed that the methodology should find successful applications beyond those implemented within this thesis.

VARGA and Sparse-VARGA are probably the first methods capable of modelling short MTS. Both improve upon the traditional Yule-Walker method and work where this method cannot. It has been shown that by integrating the order selection process into the whole procedure, better short term forecasts can be produced and the resultant parameter matrices are a better aid to explaining relationships between variables. The sparse matrix representation of Sparse-VARGA has the additional advantage of having a fitness function where the computational complexity is proportional to the number of genes in the chromosome.

7.2 Limitations

As with all new ideas, there may be "teething" problems. The methods in this thesis are no exception. The limitations of the methods are listed below.

Correlation Mining – Search Space. The procedure is of limited use if the number of calls to the correlation function is near to the number when the whole search space is explored, i.e. c is almost as large as s. However, the purpose of the method is to explore only a fraction of the search space. The implications are that if R and r are large when compared to s and the level of confidence required is also high, then the method may not be appropriate.

Grouping – *RankSize*. The value for the *RankSize* is a user defined parameter which needs to be specified in order to perform grouping. Chapter 4 provided insight into how this parameter relates to the correlation mining algorithms results, and suggests values for R/s and r/R. However these guidelines affect the algorithms efficiency rather than accuracy. However chapter 3 and chapter 6 demonstrate that good results can be gained by using these parameter heuristics. It is worth noting that this parameter can play a role in indirectly influencing the resulting number of groups, and in the visual field application, the number of groups appears related to patient's average sensitivity (chapter 4). It is therefore thought that a relationship exists where *RankSize* can be expressed as a simple function of patient's average sensitivity. This could be investigated perhaps as a future research opportunity with an aim of determining this parameter on an individual patient level, hopefully resulting in even better forecast results than those shown in chapter 6.

***Seeded Sparse*-VARGA – Dimensionality**. The performance of Seeded Sparse-VARGA is directly related to the dimensionality of the MTS that it is modelling. If the number of variables is large, say 100 plus, then the method may take a long time to complete, especially if the best selection of models are very dense.

The Methodology – Univariates. Apart from the *RankSize* problems mentioned above, there is the consideration of what to do with the univariate variables. As described in previous chapters the grouping method, through working on correlations, may not group all of the variables. This may occur when there are no significant correlations relating to one or more variables. If expert knowledge is available then, e.g. nerve fibre bundle layout or gene function classifications, then a simple regrouping procedure could be applied to place a variable into the most likely group. The Holt-Winters method with perhaps variable time step consideration [Wright1986] could be applied to the modelling stage for single variables.

7.3 Further Work

This section looks at the possible extensions to the work presented in this thesis.

7.3.1 Correlation Mining

One of the main possibilities for expanding the temporal correlation mining is the extension of the method to handle spatial-temporal datasets. Here, instead of relating variables based on cross-correlation, space-time correlations could be used instead. Variables could not only be correlated in time, but also correlated by their spatial dependencies on other variables. With the visual field dataset, currently each visual field variable is simply enumerated, thus some of the spatial information about how the variables are positioned together is lost. It is thought that because the glaucoma condition spreads through the retina in a definable way (out of the blind spot to periphery points – rather like an oil slick), this way of relating inter-point dependencies might prove more accurate.

7.3.2 Variable Grouping

There are four further research opportunities that could be explored in order to improve the variable grouping procedure.

i) One important addition to the representation of the grouping genetic algorithm would be to look at placing limits on group sizes, for example, if a minimum group size of two was specified, then the univariate problem mentioned in section 7.2 would not arise.

ii) The specification of maximum group sizes, perhaps through expert knowledge, would limit the search space significantly. To tackle this problem, either a change of representation would be needed or the operators would have to be redesigned to ensure the size constraints were adhered to.

iii) As previously mentioned, determining the parameter *RankSize* within the grouping problem is an essential task. Similar to the work carried out in chapter 4, simulations on datasets of a given length, dimensionality and time lag could be carried out to see if there is a way to model the correlation distribution against these parameters. As used in chapter 3, the *Fisher z-transformation* can be used to model the correlation sample distribution of a bivariate set of data of known length. It might be possible to extend this to the multivariate case.

iv) Another improvement to the grouping genetic algorithm would be to look at seeding (see chapters 5 and 6 where seeding was used successfully within the variants of VARGA). Possible seeds could be the results of the Separate and Conquer (S&C) algorithm as described in chapter 4 or any domain knowledge, such as the nerve fibre bundle arrangements as described in chapter 2.

7.3.3 VAR Model Fitting

The VARGA idea can be extended in a number of ways.

As mentioned in previous chapters, *Space-Time* series models [Pfeifer1980a, Pfeifer1980b] might be more applicable to the visual field dataset since the visual field points have a spatial arrangement. Once such process is the Space-Time version of the VAR process, the STAR process as shown in equation 7.1.

$$\underline{x}(t) = \sum_{i=1}^{p} \sum_{j=0}^{\Theta_i} A_{ij} W^{(j)} \underline{x}(t-i) + \underline{\varepsilon}(t) \tag{7.1}$$

where	$\underline{x}(t)$	Visual Field Data Vector at time *t*
	p	Autoregressive factor
	Θ_i	Autoregressive spatial lag at time lag *i*
	A_{ij}	Autoregressive parameters
	$W^{(j)}$	Weighting Matrix for lag *j*
	$\underline{\varepsilon}(t)$	Noise Vector at time *t*

With the STAR model, there are more parameters than with a VAR process, and not only does the time lag need to be identified, but also the spatial lags at each time lag. Additionally a weighting matrix needs to be constructed which models the spatial dependencies within the data. It is thought that a spatial version of VARGA could be developed to handle all of these parameters in one procedure. Additionally a sparse version could also be developed to reduce the number of parameters.

In order to improve the effectiveness of VARGA's crossover and shuffle operators the new operator could be adapted to mimic the *Blend Crossover* (BLX) operators' way of dealing with real numbers [Eshelman1993]. This new operator would work like uniform crossover but, instead of swapping two genes, with values say *a* and *b* where $a < b$, a single child is created where the resultant gene is a random number in the interval $[a-\delta_1, b+\delta_2]$ where δ_1, δ_2 are functions of *a*, *b* and the parents' fitness. BLX has been demonstrated to allow the formation of real valued versions of schema called *Interval Schema*. It is thought that this version of crossover could improve the convergence rate of all of the VARGA methods. Other techniques to improve performance such as *Niche Methods* [Mahfoud1995] could also be looked into.

A best candidate solution with the VARGA methods is one that has the lowest one step ahead forecast errors. However, there are other criteria that could be used to rate a good model. Some examples are the Weighted-Kappa metric on directional accuracy, the number of parameters in the model and larger step ahead forecasts. Multi-objective genetic algorithms [Deb2001] are a form of GA which can cope with comparing candidate solutions on more than one fitness function; hence the techniques in this field could be applied to the VARGA methods. The intention would be that models located using multiple objectives would be better generalised to a problem than one specifically tuned for one step ahead forecasts.

Finally, as detailed in appendix D, there might be a way to reconstruct a VAR process from its eigenvalues. An nVAR(p) process has n^2p parameters but only np eigenvalues. Hence by adapting VARGA to search for a set of eigenvalues rather than parameters, the search space would be reduced by a factor of n. However the development of such a method may need to address some challenging issues. Since each eigenvalue is a complex number therefore representations and operators would have to be adapted to deal with this property. More insight into this problem could be gained from examining the literature on the *inverse eigenvalue* problem [Chu1998].

Appendix A – Visual Field Data Tables

This appendix describes the normal tension glaucoma visual field dataset used in this thesis.

A.1 Entity Relationship Diagram

Figure A.1 shows the entity relationship diagram for the normal tension glaucoma visual field dataset.

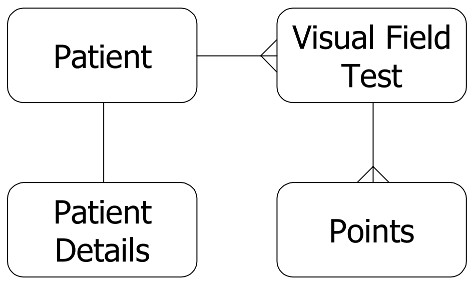


Figure A.1: Entity Relationship Diagram for the VF Dataset

A.2 Entity Attributes

Tables A.1 to A.4 described the attributes of each entity.

Patient			
Field Name	**Type**	**Specification**	**Description**
Patient ID	Integer	Unique, ≥0	A unique identifier for each patient
Number of Tests	Integer	≥1	How many visual field tests a patient has had

Table A.1: Patient Table Attributes

Patient Details			
Field Name	**Type**	**Specification**	**Description**
Patient ID	Integer	Unique, ≥0	A unique identifier for each patient
Age at 1st Test	Integer	≥0	What age the patient was when their first Visual Field was carried out; in years

Table A.2: Patient Details Table Attributes

Visual Field Test			
Field Name	**Type**	**Specification**	**Description**
Patient ID	Integer	Unique, ≥0	A unique identifier for each patient
VF Test ID	Integer	Unique, ≥0	A unique identifier for each Visual Field Test
Test Date	Date	DD-MM-YYYY	The date that the test was carried out on; in day, month, year format
Test Time	Time	HH:MM:SS	The time the test was carried out; in hour, minute, second format

Table A.3: Visual Field Test Table Attributes

Point			
Field Name	**Type**	**Specification**	**Description**
VF Test ID	Integer	Unique, ≥0	A unique identifier for each Visual Field Test
Point Location	Integer	[1,76]	A particular test location point
Test Point Value	Real	[0,60]	The sensitivity at the corresponding test location

Table A.4: Point Table Attributes

A.3 Dataset Comments

The database tables are not in fully 3rd normal form in order to speed up data access and query construction. For each entry in the *Patient* table there is one corresponding record in the *Patient Details* table and one or more records in the *Visual Field Test* table (the exact number is equal to the relevant entry for the field *Number of Tests* in the *Patient* table). For each record in the *Visual Field Test* table there are 76 corresponding records in the *Point* table, i.e. one for each test location of the *Central Threshold 30-2* test.

A total of 280 patient records were originally available, but the 82 patients used in this thesis correspond had taken ten or more tests. Figure A.2 shows a histogram of how many patients have had how many tests, where *Count* is the number of tests and *Frequency* is how many patients undertook the corresponding number of tests.

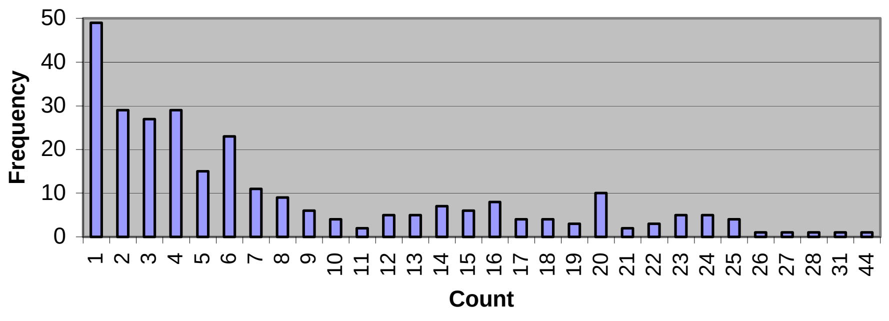


Figure A.2: Frequency of Patient Time Series Length

Appendix B – VAR Process Stationality and Stability

In section 2.1 stationary and non-stationary VAR processes were introduced. A VAR process is stationary if the mean and variance is independent of time. In practice, this means that the value of the variables tends to approach a limit (or limits) as time tends to infinity. The assumption of stationality for a VAR process is essential in this work since the intention is to verify the short term forecasting accuracy of certain VAR processes, applied to a dataset, the stationality condition would be essential, since the consequences of a non-stationary process becoming zero or infinity would prevent any forecast comparison. Another family of VAR processes is *stable* and *non-stable.* Stable processes are easier to generate than stationary processes and all stable processes are stationary. For any VAR(1) process with parameter matrix $A_1=A$ to be stable then all of the eigenvalues of the matrix A must be strictly less than one in modulus. The proof and further details of this property can be found in [Lütkepohl1993].

Figure B.1 shows the VAR(1) representation of any VAR(p) process.

$$\underline{x}(t) = \sum_{i=1}^{p} A_i \underline{x}(t-i) + \underline{\varepsilon}(t),\ \underline{x}(t) \text{ is a } n\text{VAR}(p) \text{ process}$$

$$\text{Let } \underline{y}(t) = A\underline{y}(t-1) + E(t),\ \underline{y}(t) \text{ is a } (np)\text{VAR}(1) \text{ process}$$

$$\text{Let } \underline{y}(t) = \begin{bmatrix} \underline{x}(t) \\ \underline{x}(t-1) \\ \underline{x}(t-2) \\ \cdots \\ \cdots \\ \underline{x}(t-p) \\ \underline{x}(t-p+1) \end{bmatrix} \quad A = \begin{bmatrix} A_1 & A_2 & \dots & \dots & \dots & \dots & A_p \\ I_n & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & I_n & 0 & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & 0 & I_n & 0 \end{bmatrix} \quad \underline{E}_t = \begin{bmatrix} \underline{\varepsilon}_t \\ \underline{0} \\ \underline{0} \\ \cdots \\ \cdots \\ \underline{0} \\ \underline{0} \end{bmatrix}$$

$$(np) \text{ by } 1 \qquad\qquad (np) \text{ by } (np) \qquad\qquad (np) \text{ by } 1$$

$$\text{then } \underline{y}(t) \text{ is equivalent to } \underline{x}(t)$$

Figure B.1: VAR(1) representation of a VAR(p) Process

It is easier to ensure Stability than Stationality, since there is a well defined and strict mathematical property associated with all stable processes as described above. Hence it is possible to generate VAR test data that are stable. Since stability is only defined for VAR(1) processes it may be necessary to convert any higher order process to the VAR(1) representation as in figure B.1 to test for stability.

Appendix C – Gershgorin Circle Theorem

The following is an extract from [Gourlay1973] which describes the *Gershgorin Circle* method of placing bounds on eigenvalues.

Theorem C.1. Let A be an $n \times n$ matrix. Let a_{ij} be the element from the *ith* row and *jth* column of the matrix A, where $1 \le i,j \le n$. Let C_i be discs in the complex plane with centres a_{ii} and radii $R_i = \sum_{\substack{k=1 \\ k \ne i}}^{n} |a_{ik}|, 1 \le i \le n$. Let D be defined as follows, $D = \bigcup_{i=1}^{n} C_i$, then all of the eigenvalues of A lie within D.

Proof. Let $\{\lambda_1,..,\lambda_n\}$ be the eigenvalues of A and $\{\underline{x}_1,..,\underline{x}_n\}$ be the corresponding eigenvectors. Let x_{ij} be the jth element of $\underline{x}_i$ and let each $\underline{x}_i$ be normalised such that $\max(|x_{ij}|) = 1, 1 \le j \le n.$

By definition $A\underline{x}_i = \lambda_i \underline{x}_i$ Hence

$$(\lambda_i - a_{jj})x_{ij} = \sum_{\substack{k=1 \\ k \ne j}}^{n} a_{jk}x_{ik}, 1 \le j \le n$$

Since there exists an x_{ir} such that $|x_{ir}|=1$, $1 \le r \le n$.

$$(\lambda_i - a_{rr})x_{ir} = \sum_{\substack{k=1 \\ k \ne r}}^{n} a_{rk}x_{ik}$$

$$|(\lambda_i - a_{rr})x_{ir}| = \left| \sum_{\substack{k=1 \\ k \ne r}}^{n} a_{rk}x_{ik} \right|$$

$$|(\lambda_i - a_{rr})||x_{ir}| = \sum_{\substack{k=1 \\ k \ne r}}^{n} |a_{rk}||x_{ik}|$$

$$|(\lambda_i - a_{rr})| \le \sum_{\substack{k=1 \\ k \ne r}}^{n} |a_{rk}||x_{ik}|$$

$$|(\lambda_i - a_{rr})| \le \sum_{\substack{k=1 \\ k \ne r}}^{n} |a_{rk}| = R_r$$

So the eigenvalue λ_r lies in the disc C_r. Since λ_r is arbitrary then all of the eigenvalues of A must lie in the union of all such discs, i.e. D. ■

For example, if

$$A = \begin{bmatrix} 7 & 1 & 1 & 1 \\ 1 & 5i & 1 & 1 \\ 0 & 1 & 1 & -3 \\ 1 & 2 & i & 0 \end{bmatrix}$$

then figure C.1 shows the Gerschgorin discs , and the eigenvalues are listed.

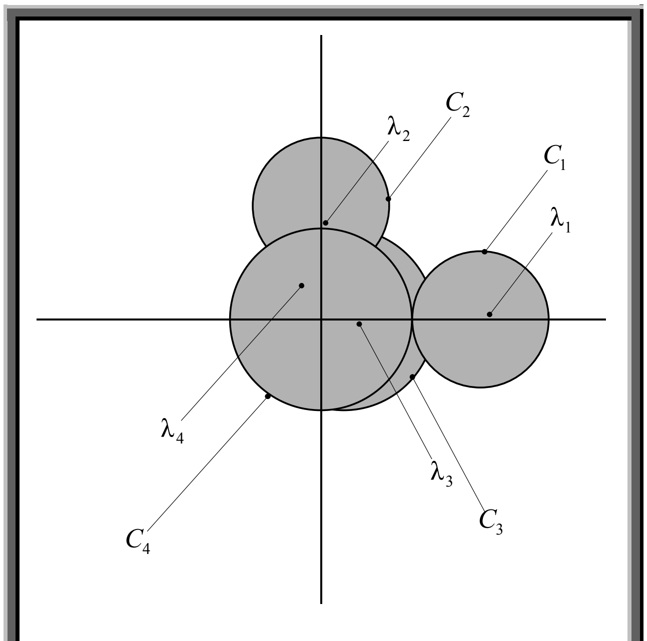


Figure C.1: Gerschgorin Discs Example

The discs are as follows:

$$C_1 : |z-7| \le 3$$
$$C_2 : |z-5i| \le 3$$
$$C_3 : |z-1| \le 4$$
$$C_4 : |z| \le 4$$

Here z is complex

The eigenvalues of A are:

$$\lambda_1 = 7.206 + 0.090i$$
$$\lambda_2 = 0.180 + 4.280i$$
$$\lambda_3 = 1.943 - 0.095i$$
$$\lambda_4 = -1.329 + 1.725i$$

Appendix D – Algebraically Building a Stable VAR Process

The VAR(1) representation of an *n*VAR(*p*) process is as follows, (see appendix B):

$$A = \begin{bmatrix} A_1 & A_2 & \dots & \dots & \dots & \dots & A_p \\ I_n & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & I_n & 0 & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & 0 & I_n & 0 \end{bmatrix}$$ where A is an np by np matrix.

The eigenvalues of A must all be strictly less than one in modulus for the process to be stable. Let $\lambda_1,...,\lambda_{np}$ be the eigenvalues of A and $\underline{x}_1,...,\underline{x}_{np}$ be the corresponding eigenvectors, then:

$$\begin{bmatrix} A_1 & A_2 & \dots & \dots & \dots & \dots & A_p \\ I_n & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & I_n & 0 & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & 0 & I_n & 0 \end{bmatrix} \begin{bmatrix} x_{i1} \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ x_{i(np)} \end{bmatrix} = \lambda_i \begin{bmatrix} x_{i1} \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ x_{i(np)} \end{bmatrix}$$ where x_{ij} is the *j*th element of $\underline{x}_i$.

Let $\underline{y}_{i1},...,\underline{y}_{ip}$ be n dimensional vectors defined as follows:

$$\begin{bmatrix} \underline{y}_{i1} \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \underline{y}_{ip} \end{bmatrix} = \begin{bmatrix} x_{i1} \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ x_{i(np)} \end{bmatrix}$$ the *k*th element of $\underline{y}_{ij}$, y_{ijk}, is equal to the $(n(j\text{-}1)+k)$th element of $\underline{x}_i$.

Therefore,

$$\begin{bmatrix} A_1 & A_2 & \dots & \dots & \dots & \dots & A_p \\ I_n & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & I_n & 0 & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & 0 & I_n & 0 \end{bmatrix} \begin{bmatrix} \underline{y}_{i1} \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \underline{y}_{ip} \end{bmatrix} = \lambda_i \begin{bmatrix} \underline{y}_{i1} \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \underline{y}_{ip} \end{bmatrix}$$

Therefore,

$A_1\underline{y}_{i1} + A_2\underline{y}_{i2} + \dots + A_p\underline{y}_{ip} = \lambda_i\underline{y}_{i1}$

$\underline{y}_{i1} = \lambda_i\underline{y}_{i2}$

$\underline{y}_{i2} = \lambda_i\underline{y}_{i3}$

$\cdots$

$\underline{y}_{i(p-1)} = \lambda_i\underline{y}_{ip}$

Therefore,

$$\underline{y}_{ij} = \lambda_i^{p-j}\underline{y}_{ip} \tag{D.1}$$

Hence,

$$A_1\lambda_i^{p-1}\underline{y}_{ip} + A_2\lambda_i^{p-2}\underline{y}_{ip} + \dots + A_p\underline{y}_{ip} = \lambda_i^p\underline{y}_{ip} \tag{D.2}$$

If it assumed that $\lambda_1,\dots,\lambda_{np}$ and $\underline{x}_1,\dots,\underline{x}_{np}$ are known, then there are n^2p unknown parameters (each a_{ijk}) and np lots of n equations (by taking each row of equation D.2 as a separate equation). This forms a set of linear equations that can be solved using any of the standard techniques.

The values for $\lambda_1,\dots,\lambda_{np}$ can be simply set to a unique real random number in the interval (-1,1) (excluding values close to zero). However care must be taken in choosing $\underline{x}_1,\dots,\underline{x}_{np}$ since as shown above, the elements are highly inter-dependent. Since $\underline{y}_{i1},\dots,\underline{y}_{i(p-1)}$ depend on the value of $\underline{y}_{ip}$, these vectors can be set to random real vectors, and then the rest computed according to equation D.1. Note further that only real values for the eigenvalues and eigenvectors are allowed to ensure that the elements of $A_1,\dots,A_p$ are real.

Appendix E – VAR Process Data Generation

This appendix discusses the various ways in which test VAR processes can be constructed.

In order to verify the methods introduced in this thesis, it is necessary to generate some synthetic datasets, rather than just working on the visual field data. This is because the underlying process that generated the data will be known, and hence assessment of method accuracy will be easier. Stable VAR processes are used for data generation since all stable VAR processes are stationary. A non-stationary process's mean would rapidly approach infinity, and thus be of no use as a test dataset.

E.1 Data Generation

A nVAR(p) has n^2p parameters, and its corresponding VAR(1) process has np eigenvalues. When generating some test data it is assumed that the number of variables and the order are known.

The task in hand is to choose the set of parameters (n^2p) such that the np corresponding eigenvalues (of the VAR(1) representation) are less than one in modulus. Two methods are presented for this purpose, method one involves a genetic algorithm, and method two involves enforcing stability by mathematically working through from a predefined set of eigenvalues that lie in the specified range.

VAR Process Generation through a Genetic Algorithm

One way of locating a stable VAR process would be to generate a population of random VAR processes of a specified dimensionality and order, convert each to its VAR(1) representation, solve the eigenvalues for each VAR(1), score on how close to zero in modulus they are (the eigenvalues), and then iterate through a series of generations using a genetic algorithm. Figure E.1 is a schematic diagram of this method.

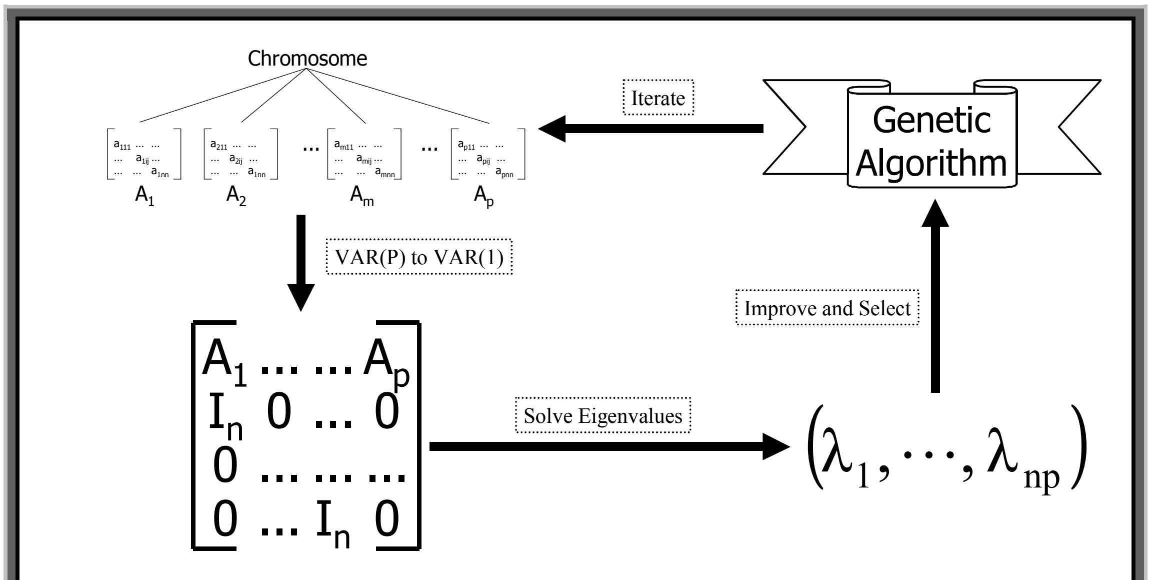


Figure E.1: Sketch Algorithm for VAR Process Generation

However there are certain problems with this method. Namely, the evaluation of the eigenvalues of a matrix is no easy task, especially if the matrix is non-symmetric. It is possible that the eigenvalues will be complex, which significantly increases the difficulty of the problem. An alternative would be to find an approximate or upper limit on the eigenvalues.

For a stable VAR(1) process, all the eigenvalues have modulus less than one. Therefore over all of the possible *Gerschgorin Circles* (see appendix C), the distance to the point furthest away from the origin should be as small as possible. This is defined in equation E.1.

$$M_\lambda = \underset{i=1}{\overset{n}{MAX}}\left(|C_i + R_i|, |C_i - R_i|\right) \tag{E.1}$$

Here M_λ is the distance to the point furthest away from the origin that lies in or on one of the disks. A genetic algorithm can use this value for a fitness function, using the representation for VARGA-v2 (see chapter 5). Crossover is also the same, as is gene mutation. However there is no need for the size mutation operator since the order is fixed.

VAR Process Generation through Algebra

If the VAR(1) representation of a matrix is examined and certain assumptions are made, it is possible to reverse the process and to specify a set of eigenvalues (plus some additional parameters), and then build the VAR(1) process representation. Appendix D details this procedure.

From appendix D all that is needed are *np* lots of *n* elements for the eigenvectors, the remaining values being calculated from equation D.1. These are just required for a solution and are not as important as the eigenvalues. So a solution would be to choose random values for the eigenvectors, and then assuming that all of the eigenvalues are less than one in modulus, the corresponding VAR(1) representation can be reconstructed.

E.2 Comparing Both Methods

Method one, involving a genetic algorithm, is not exact. All of the eigenvalues for a given VAR(1) representation could be less than one in modulus, but the limit specified by equation D.1 could be above one. This means that there is no way of knowing exactly when to terminate the genetic algorithm. From experience, if the genetic algorithm is terminated when the fitness is less than one, the eigenvalues are all very small in modulus, which in turn makes all of the parameters in the VAR(1) process very small as well. Experience showed that terminating the genetic algorithm before the fitness was below one would result sometimes in the desired set of eigenvalues.

If this was not the case but the maximum eigenvalue modulus was nearing one, then the genetic algorithm generations were increased slightly to compensate. This interactive way of using the method provided the set of parameter matrices, and usually resulted in the maximum modulus eigenvalue being just under one. Between 5 and 10 iterations were needed depending on the dimensionality and order of the VAR(*p*) process, i.e. the size of the VAR(1) representation.

The algebra method would automatically produce a VAR(1) process that was stable, once the restrictions on the eigenvectors were applied. This could be achieved through generating the *np* eigenvalues and then the n^2p associated eigenvector elements, according to the constraints. However, to ensure that the VAR process parameters are real, the eigenvectors and eigenvalues are also restricted to being real. A complex VAR process is likely to produce a complex time series, which would complicate many of the methods applied to such a time series, for example when calculating correlation coefficients.

To summarise, the GA method produces VAR(1) processes that could cover the entire range of stable VAR processes, however the procedure is interactive and not exact. The algebraic method produces automatically a random VAR(1) process or a VAR(1) process with specified eigenvalues if required, but can only produce a specific type of stable VAR(1) process.

E.3 Random Data Generation From a Specified VAR(p) Process

Assuming that the order, the dimensionality and the corresponding parameters of a VAR(p) process are available, along with the specified length of the time series, both procedures will generate a set of random VAR(p) data. This data will then be centred around a zero mean.

Appendix F – Proofs for the Grouping Metric

Proposition F.1. When there are no correlations, then $Q = \phi$.

Proof. 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*. ■

Proposition F.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.

Proof. It therefore follows that the value for *h(g_i)* will be $\frac{k_i(k_i-1)}{2}$ using the same logic. By using equation 4.1:

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

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.

It is assumed that $1 \le k_1 \le k_2 \le \ldots k_m$. If $k_1' = k_1 + k_2$ 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. ■

Proposition F.3. If the data generating the correlations came from a mixed set of MTS observations, then *f*(*G*) is maximised when the groups match the partitioning of the MTS.

Proof. For a given grouping arrangement *G* and correlation set *Q*

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

$$\max(h(g_i)) = \max\left(\sum_{\substack{\forall a,b \\ a \neq b \\ 1 \leq a < b \leq k_i}} L(g_{ia}, g_{ib}) \right)$$

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. ■

Appendix G – The Lilliefors' Test Results

Table G.1 displays the results of applying Lilliefors' test on a small sample of simulations as described in section 4.3.

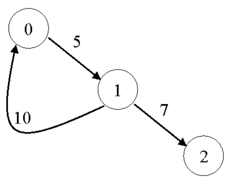
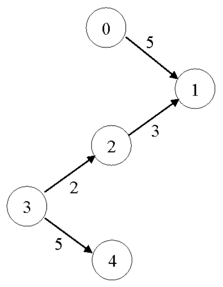
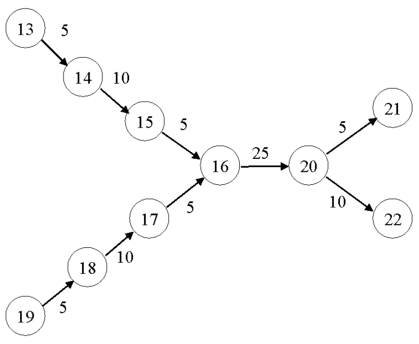
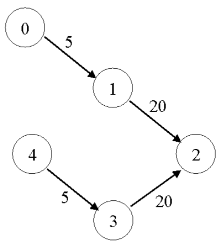
R	***c***	***s***	μ	σ	$\overline{\mu}$	$\overline{\sigma}$	D_{max}	$\frac{1.031}{\sqrt{v}}$	$D_{max} < \frac{1.031}{\sqrt{v}}$
50	657	4600	6.590	2.393	6.665	2.365	0.106	0.144	True
60	780	3900	10.823	2.939	10.909	3.152	0.079	0.132	True
70	357	3570	6.665	2.407	6.667	2.211	0.097	0.122	True
80	1062	7440	10.687	3.052	10.659	2.840	0.086	0.115	True
90	918	4590	16.293	3.639	16.364	3.629	0.069	0.108	True
100	1583	9500	15.400	3.589	15.382	3.420	0.070	0.103	True
110	1452	14520	10.418	3.058	10.476	2.846	0.082	0.098	True
120	1851	12960	15.972	3.743	15.997	3.476	0.067	0.094	True
130	1841	11050	19.95	4.138	19.993	3.896	0.060	0.090	True
140	2963	17780	21.539	4.222	21.536	4.055	0.059	0.087	True
150	3930	19650	27.127	4.767	27.273	4.581	0.052	0.084	True
160	2272	22720	15.222	3.708	15.238	3.640	0.071	0.081	True
170	4386	21930	30.896	5.007	30.909	4.898	0.053	0.079	True
180	1854	18540	17.158	3.878	17.143	3.957	0.065	0.077	True
190	988	9880	18.063	4.106	18.095	4.116	0.065	0.075	True

Table G.1: Lilliefors' Test Results

Within this table, $\mu, \sigma, \overline{\mu}$ and $\overline{\sigma}$ are listed as examples for the section on Genetic Programming.

Appendix H – MTS Dataset Generation

Below is a selection of DBN structures that were used to generate the DBN datasets. Numbers associated with nodes represent variables and numbers associated with links represent time lags.

	
DBN 1	DBN 2
 13 5 14 10 15 5 16 25 20 5 21 10 22 17 5 18 10 19 5	
DBN 3	DBN 4

Below is a selection of VAR process parameters that were used to generate the VAR datasets.

2 Variable VAR(2)				
	P=1 (A_1)		P=2 (A_2)	
	1	2	1	2
1	-0.52	0.02	0.23	-0.22
2	0.28	-0.17	-0.41	-0.11

3 Variable VAR(5)

	P=1 (A_1)			P=2 (A_2)			P=3 (A_3)			P=4 (A_4)			P=5 (A_5)		
	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.10	0.02	-0.20	0.15	-0.04	0.24	0.06	-0.08	0.31	-0.07	0.03	-0.04	-0.05	0.19	0.15
2	-0.16	-0.02	-0.02	0.01	-0.40	0.10	-0.28	0.02	-0.24	-0.08	-0.01	-0.04	0.12	0.06	0.19
3	0.03	0.55	-0.18	0.08	-0.02	0.04	-0.09	0.00	0.19	0.05	0.26	0.01	-0.12	-0.02	-0.23

7 Variable VAR(3)

	P=1 (A_1)							P=2 (A_2)							P=3 (A_3)						
	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7
1	0.13	-0.32	-0.21	0.02	-0.22	-0.06	0.00	-0.29	0.11	0.30	-0.09	0.02	-0.04	-0.33	-0.05	0.17	-0.03	0.22	0.04	0.02	-0.25
2	-0.40	-0.63	0.02	-0.10	0.01	0.02	-0.22	0.05	0.16	0.58	0.05	-0.02	0.10	-0.24	-0.02	0.00	0.08	0.22	0.00	0.16	-0.01
3	0.11	-0.08	-0.10	0.03	-0.09	0.06	0.12	-0.12	-0.21	0.29	-0.17	0.08	-0.17	-0.26	0.09	-0.28	0.02	0.02	-0.03	-0.69	0.09
4	-0.11	-0.18	-0.42	-0.22	0.02	-0.13	0.12	-0.05	0.38	0.06	-0.14	-0.07	0.08	0.11	0.02	-0.27	-0.01	-0.08	0.32	0.02	0.06
5	-0.07	-0.01	0.09	0.31	0.08	-0.07	0.00	0.07	-0.61	-0.02	0.04	-0.29	0.01	-0.16	0.07	0.30	-0.41	-0.13	0.06	-0.15	0.06
6	0.06	0.27	-0.07	-0.19	0.16	-0.20	0.01	0.21	0.02	0.04	0.19	0.18	0.44	-0.04	0.06	0.18	-0.34	0.19	-0.05	-0.04	0.06
7	0.08	0.21	-0.32	0.07	-0.11	0.03	-0.41	-0.03	-0.33	-0.12	0.09	0.00	0.55	0.19	0.14	-0.02	0.02	0.16	0.04	-0.15	0.04

Appendix I – Genetic Programming Details

Table I.1 displays the genetic programming (symbolic regression) parameters used in the experiments described in section 4.3.3.

Operator	Description	Value
Population	Constant Population	100
Generations	Number of iterations	μ =1,000 σ =50,000
Crossover	Percentage of population allowed to breed	0.75
Prune Mutation Rate	Cut down a sub-tree to a random terminal node	0.25
Add Sub-tree Mutation Rate	Replace a sub-tree for a new random sub-tree (size varies)	0.25
Change Node Mutation Rate	Change an operator to a new random operator or a terminal symbol to a new random terminal symbol	0.25
Survival	Simply select the top "Population" after new individuals have been added through Crossover and Mutation	Deterministic and Extinctive

Table I.1: Genetic Programming Parameters

I.1 μ and σ Results

The test dataset of parameters for the problems consisted of the same 150 records generated by the simulation experiments. This was divided into two halves, one for the training set and the other for the verification set. The records were numbered from 1 to 150; μ was trained on the even records and σ on the odd records. The magnitude of the values for *R*, *s* and *c* increased as the record identifier (ID) increased.

The results are listed in table I.2.

Result	μ		σ	
Value for function (1)	$\frac{2cR^2}{2Rs+c(R+4)}$		$\frac{R+8}{63}+\frac{11c}{s}$	
Approximation to be used (2)	$\frac{2cR}{2s+c}$		$\frac{R}{63}+\frac{11c}{s}$	
	(1)	(2)	(1)	(2)
Fitness for Training Set (Excluding Nodes)	-0.104	-0.240	-0.791	-2.096
Sum of Absolute Error for Training Set	2.220	3.194	6.088	11.351
Average % Error for Training Set	0.203	0.251	2.391	4.479
Fitness for Testing Set (Excluding Nodes)	-1.038	-0.249	-0.842	-2.114
Sum of Absolute Error for Testing Set	2.236	3.200	6.263	11.401
Average % Error for Testing Set	0.815	1.112	2.460	4.518

Table I.2: μ and σ Results

Appendix J – The Weighted-Kappa Metric

The *Weighted-Kappa* metric [Altman1997] is used to rate agreement between the classification decisions made by two or more observers. The decisions are classified into a number of classes $\{class_1,...,class_N\}$ where the classes are ordered, i.e. a classification by the two observers of $class_1$ and $class_2$ is a better agreement than $class_1$ and $class_4$ etc.

J.1 Calculating Weighted-Kappa

An $N{\times}N$ table of counts is constructed according to figure J.1 for a set of classifications. Rows are indexed according to one observer's classification and columns by the other observer's classification. $Row(i)$ and $Col(i)$ are row and column totals respectively, and $Count_{ij}$ is the count for a particular combination of classifications. The sum of all of the cells $N_k = \sum_{i=1}^{N}\sum_{j=1}^{N} Count_{ij} = \sum_{i=1}^{N} Row(i) = \sum_{i=1}^{N} Col(i)$.

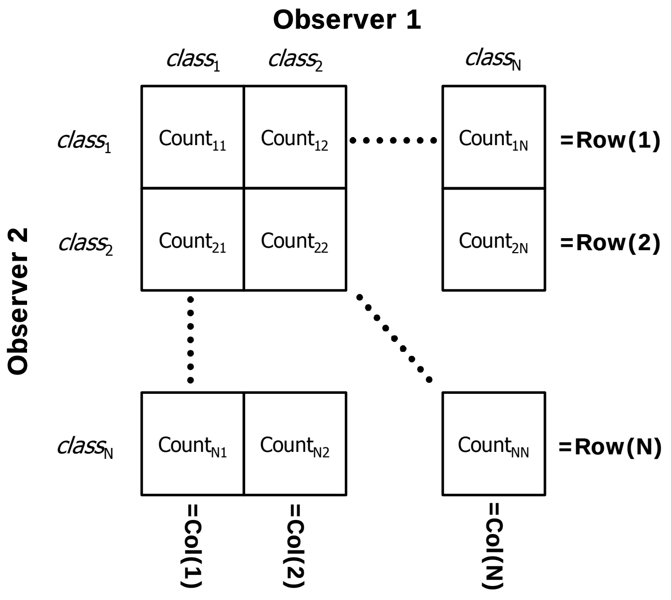


Figure J.1: The Construction of the Weighted-Kappa Count Table

The weighted-kappa metric, K_w, is calculated as follows:

i) Compute $w_{ij} = 1 - \frac{|i-j|}{N-1}$ where $1 \le i, j \le N$

ii) Compute $p_{o(w)} = \frac{1}{N_k} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} Count_{ij}$

iii) Compute $p_{e(w)} = \frac{1}{N_k^2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} Row(i) Col(j)$

iv) Then $K_w = \frac{p_{o(w)} - p_{e(w)}}{1 - p_{e(w)}}$

The two parameters within this calculation, $p_{o(w)}$ and $p_{e(w)}$, represent the observed weighted proportional agreement and the expected weighted proportional agreement. The expected weighted proportional agreement is an indication of what totals would be expected by chance. Table J.1 is the suggested interpretation of *Kappa* values to indicate the strength of agreement between two observers [Altman1997], which can be used as an approximate interpretation for *Weighted-Kappa*. Although the *Weighted-Kappa* values are normally higher than those of *Kappa*, it is only going to be used in the assessment of the agreement strength between two methods according to their respective categories.

Kappa (*K*)	**Agreement Strength**
$-1 \le K \le 0$	Very Poor
$0 < K \le 0.2$	Poor
$0.2 < K \le 0.4$	Fair
$0.4 < K \le 0.6$	Moderate
$0.6 < K \le 0.8$	Good
$0.8 < K \le 1.0$	Very Good

Table J.1: The Kappa Guideline

J.2 Visual Field Test Results

In the case where the two observers are the recorded visual field test results and the predicted results from a VAR process the classification decisions from each observer for each point $x_i(t)$ can be divided into three states as in table J.2. Within this table, *direction of change* refers to the shorthand description for whether the change is worsening the condition (-), not altering at all (*0*) or improving (+). This classification is based on the change between a point measured at two successive tests, i.e. the direction of change.

Direction of Change (Class)	**Definition**
Deterioration (-)	$x_i(t) - x_i(t-1) < -\Omega$
No change (0)	$\lvert x_i(t) - x_i(t-1) \rvert \le \Omega$
Improvement (+)	$x_i(t) - x_i(t-1) > \Omega$

Table J.2: Visual Field Change Classification

The constant Ω gives a margin of error for defining equality, since the visual fields are treated as real numbers (they are integers ranging between 0 and 60). This makes it very unlikely that any pair of observed and predicted points will be equal. In this thesis $\Omega = 0.5$ has been chosen. The Weighted-Kappa metric can then be calculated as above, where a category of – has an index of 1, a category of *0* has an index of 2 and a category of + has an index of 3.

The same number of one step ahead forecasts are performed as with the VARGA fitness function (see chapter 5), but the differences between forecasts are considered, i.e. the direction of change, giving a total of $n(T - \text{MAXORDER - 1}) = \sum_{i=0}^{2}\sum_{j=0}^{2} Count_{ij}$ possible classifications (N_k).

J.3 Comparisons of Grouping

The weighted-kappa metric can be used to compare the similarities between two grouping arrangements. It is assumed that the grouping arrangements being compared are between groupings containing the same number of variables.

If n variables are being grouped then a set V is defined as being the set of all possible variable indices, $V = \{1,...,n\}$. A grouping arrangement $G = \{g_1,...,g_m\}$ where $g_i \subseteq V$; $\bigcup_{i=1}^{m} g_i = V$ and $g_i \cap g_j = \varphi, 1 \le i < j \le m$. Let the function $\pi(G,i)$ return the group index of which group, g_j, variable index i is located, where $1 \le i \le n$ and $1 \le j \le m$.

Given two grouping arrangements, G_1 and G_2, then the weighted-kappa metric between these two groups is computed according to algorithm J.1.

Algorithm J.1: WK(G_1,G_2)

```
1)   Input:  G_1, G_2 – two grouping arrangements
             n – the number of variables in the two groupings
2)   Set Count = a 2 by 2 initially zero matrix
3)   For i = 1 to n-1
4)       For j = i+1 to n
5)           If $\pi(G_1,i) = \pi(G_1,j)$ then a = 0 else a = 1
6)           If $\pi(G_2,i) = \pi(G_2,j)$ then b = 0 else b = 1
7)           Set $Count_{ab} = Count_{ab} + 1$
8)       Next j
9)   Next i
10)  Output: The Weighted-Kappa metric computed using Count
```

Essentially algorithm J.1 works by considering all unique pairing of variables as a classification decision, where there are two classes; that the pair of variables are in the same group or the pair of variables are in different groups.

Appendix K – Algorithm for Sparse-VARGA Fitness Calculation

Algorithm K.1 is the algorithm for computing the Sparse-VARGA fitness function. It works by computing the forecast error using sparse matrix arithmetic, i.e. only performing matrix multiplication for the known elements, all the others are assumed to be zero.

Algorithm K.1: ComputeFitness(CHROME,CASE)

```
1)    Input:  Chromosome CHROME of size GENESIZE
              Visual Field CASE of size NTESTS (T) by NPOINTS (n)
              (n – size of a nerve fibre bundle group)
              Maximum VAR Process Order MAXORDER
2)    Set FITNESS = 0
3)    Set ERROR = a zero vector of length NPOINTS
4)    For t = MAXORDER to NTESTS
5)        For i = 1 to NPOINTS
6)            let ERROR(i) = CASE(t,i)
7)        next i
8)        For i = 1 to GENESIZE
9)            Set GENE = ith gene of CHROME
10)           Set ROW = GENE.row
11)           Set COL = GENE.col
12)           Set LAG = GENE.lag
13)           Set VALUE = GENE.value
14)           ERROR(ROW) = ERROR(ROW) – VALUE×CASE(t-LAG,COL)
15)       Next i
16)       For i=1 to NPOINTS
17)           FITNESS = FITNESS - |ERROR(i)|
18)       Next i
19)   Next t
20)   Output: FITNESS – the fitness of chromosome CHROME
```

Appendix L – Algorithms for the Seeding of Sparse-VARGA

This Appendix details five algorithms that are used in the seeding of *Sparse*-VARGA. These are summarised in table L.1. A listing of each algorithm is then given.

Algorithm	Description
CreateYuleWalkerSeed	Creates a *chromosome* that represents a VAR process of a given order for a given dataset
CreateNoiseSeed	Creates a *chromosome* that represents the *Noise Model* for a given dataset
CreateRandomChromosome	Creates a *chromosome* that represents a random *n*VAR(*p*) process
PadAChromosomeWithZeros	Fills a *chromosome* up with zero genes
ConvertToChromosome	Converts an *n* by $n \times p$ matrix to a *chromosome*

Table L.1: Sparse-VARGA Seeding Algorithms

Algorithm L.1: CreateYuleWalkerSeed(P,CASE)

```
1)  Input:  P – Order of required VAR process
            Visual Field CASE of size NTESTS (T) by NPOINTS (n)
            (n – size of a nerve fibre bundle group)
2)  Solve the Yule-Walker equations for CASE of Order P, returning A(1) to A(P) VAR
    parameter matrices if solvable A(i,j,k) is the jth,kth element of A(i)
3)  Set CHROME to an empty Chromosome (i.e. no genes)
4)  If Solvable then
5)          Set B = [A(1)..A(P)], a NPOINTS by NPOINTS×MAXORDER matrix
6)          Set CHROME = ConvertToChromosome(B) (see Appendix C.5)
7)  End if
8)  Output: CHROME – A Yule-Walker seeded Chromosome for CASE of Order P if the
            equations were solvable. An empty Chromosome, otherwise; i.e. no genes
```

Algorithm L.2: CreateNoiseSeed(NPOINTS,MAXORDER)

```
1)   Input:  NPOINTS – Size of a parameter matrix
             MAXORDER – Maximum size of the respective VAR process
2)   Set CHROME to an empty Chromosome (i.e. no genes)
3)   Let GENE be a gene
4)   Set GENE.row = UI(1,NPOINTS)
5)   Set GENE.col = UI(1,NPOINTS)
6)   Set GENE.lag = UI(1,MAXORDER)
7)   Set GENE.value = 0.0
8)   Insert GENE into CHROME
9)   Output: CHROME – A Noise Model seeded Chromosome for CASE
```

Algorithm L.3: CreateRandomChromosome(P,CASE)

```
1)   Input:  NPOINTS (n) – Size of a parameter matrix
             MAXORDER – Maximum size of the VAR process
             MINGENE, MAXGENE – The limits for a parameter matrix element
             MAXSIZE – The maximum size of a chromosome
2)   Set CHROME to an empty Chromosome (i.e. no genes), Let SIZE = UI(1,MAXSIZE)
3)   For i = 1 to SIZE
4)           Let GENE be a gene
5)           Set GENE.row = UI(1,NPOINTS)
6)           Set GENE.col = UI(1,NPOINTS)
7)           Set GENE.lag = UI(1,MAXORDER)
8)           Set GENE.value = UR(MINGENE,MAXGENE)
9)           Insert GENE into CHROME
10)  Next i
11)  Output: CHROME – A random Chromosome representing a NPOINTS dimensional VAR
             process of Order between 1 and MAXORDER inclusive
```

Algorithm L.4: PadAChromosomeWithZeros(CHROME)

```
1)    Input:  CHROME – A Chromosome representing an NPOINTS (n) dimensional VAR
              process of maximum Order MAXORDER, minimum Order 1, and number of
              genes GENESIZE. It is assumed that the genes in CHROME are unique
2)    Set B = An NPOINTS by (NPOINTS×MAXORDER) matrix with all elements zero
      B(i,j) is the ith, jth element of B
3)    For i = 1 to GENESIZE
4)        Set GENE be the ith gene of CHROME
5)        Set ROW = GENE.row, Set COL = GENE.col, Set LAG = GENE.lag
6)        Set VALUE = GENE.value
7)        Set B(ROW,NPOINTS×(LAG-1)+COL) = VALUE
8)    Next i
9)    Set CHROME = ConvertToChromosome(B) (see algorithm L.5)
10)   Output: CHROME – A random Chromosome representing a NPOINTS dimensional VAR
              process of Order between 1 and MAXORDER inclusive
```

Algorithm L.5: ConvertToChromosome(MATRIX)

```
1)    Input:  MATRIX – An NPOINTS by NPOINTS×P real values matrix
              MATRIX(i,j) is the ith,jth element of MATRIX
2)    Set CHROME to an empty Chromosome (i.e. no genes)
3)    For i = 1 to P
4)        For j = 1 to NPOINTS
5)            For k = 1 to NPOINTS
6)                Set GENE to a new gene
7)                Set GENE.row = j
8)                Set GENE.col = k
9)                Set GENE.lag = i
10)               Set GENE.value = MATRIX(j,(i-1)×NPOINTS+k)
11)               Insert GENE into CHROME
12)           Next k
13)       Next j
14)   Next i
15)   Output: CHROME – A Chromosome representing the matrix MATRIX
```

Appendix M – Method Error by Patient

This Appendix displays the two ways of rating the error of each method, i.e. forecast error and *Weighted-Kappa*, averaged for each patient.

M.1 Forecast Error

Patient	HC	NOISE	SSV	SVNP	SVP	YW16	YW76	Best
0	0.921	4.473	0.909	2.266	2.635	1.352	3.960	SSV
1	1.160	4.404	1.138	2.190	3.498	1.452	4.133	SSV
2	0.752	2.826	0.744	1.587	2.131	0.917	2.380	SSV
3	1.317	6.114	1.306	3.846	2.738	2.362	4.869	SSV
4	0.235	1.358	0.233	0.750	0.933	0.294	1.305	SSV
5	1.543	3.445	1.532	2.911	5.265	1.671	1.289	YW76
6	0.556	1.712	0.555	1.039	1.229	44.284	100.000	SSV
7	0.616	1.995	0.608	1.203	1.804	0.771	2.469	SSV
8	0.305	3.173	0.303	0.942	0.505	19.372	100.000	SSV
9	1.117	4.387	1.100	3.555	4.219	1.452	0.946	YW76
10	0.361	3.738	0.349	1.652	1.416	0.791	3.677	SSV
11	0.854	2.399	0.842	1.815	2.580	1.024	3.880	SSV
12	1.081	3.104	1.069	1.920	2.910	1.352	2.347	SSV
13	0.452	1.829	0.445	1.021	1.207	0.619	0.915	SSV
14	0.133	2.437	0.128	0.637	0.408	0.431	3.356	SSV
15	0.671	3.193	0.657	1.651	1.940	0.971	4.292	SSV
16	0.717	3.688	0.699	1.964	2.159	0.979	2.883	SSV
17	0.803	4.958	0.840	2.566	2.240	13.359	100.000	HC
18	0.880	2.878	0.898	1.553	2.312	50.586	100.000	HC
19	0.192	2.685	0.170	0.597	0.349	0.526	1.873	SSV
20	0.594	2.644	0.587	1.522	2.168	0.738	1.719	SSV
21	0.464	3.039	0.457	1.539	1.934	6.757	100.000	SSV
22	0.738	2.644	0.736	1.290	1.509	25.607	100.000	SSV
23	0.078	2.709	0.074	0.417	0.161	0.625	2.069	SSV
24	0.188	2.080	0.182	0.857	0.743	0.361	5.441	SSV
25	0.708	2.647	0.704	1.447	1.755	38.064	100.000	SSV
26	0.244	1.525	0.236	0.696	0.716	0.355	1.227	SSV
27	0.238	2.195	0.229	0.825	0.713	0.457	4.022	SSV
28	0.522	2.008	0.518	1.209	1.615	37.982	100.000	SSV
29	1.681	4.468	1.667	2.440	7.436	1.819	3.478	SSV
30	0.850	2.346	0.848	1.366	2.178	13.447	100.000	SSV
31	0.554	2.780	0.536	1.472	1.625	0.801	2.989	SSV
32	0.845	3.802	0.839	2.233	2.628	1.127	2.542	SSV
33	0.139	2.358	0.135	0.373	0.170	6.764	100.000	SSV
34	1.144	2.992	1.158	1.692	2.551	26.058	100.000	HC
35	1.581	3.302	1.572	2.535	5.183	1.706	3.702	SSV
36	1.088	2.848	1.083	1.696	2.680	50.936	100.000	SSV
37	0.374	2.229	0.371	0.978	0.757	56.736	100.000	SSV
38	0.352	2.031	0.345	0.785	0.618	0.669	6.066	SSV
39	0.872	2.470	0.870	1.483	2.453	25.798	100.000	SSV
40	0.459	1.275	0.475	0.666	0.942	81.425	100.000	HC

Table M.1: Forecast Error (Fitness) by Patient (#0-40)

Patient	HC	NOISE	SSV	SVNP	SVP	YW16	YW76	Best
41	0.986	3.369	0.976	2.090	3.628	7.338	100.000	SSV
42	0.521	4.139	0.516	1.941	1.700	19.385	100.000	SSV
43	1.130	8.122	1.109	4.521	5.012	1.349	2.111	SSV
44	0.208	2.343	0.199	0.630	0.356	13.124	100.000	SSV
45	0.171	1.925	0.171	0.627	0.455	0.457	1.702	HC
46	1.205	4.250	1.191	3.113	4.445	1.430	3.810	SSV
47	0.559	2.778	0.552	1.530	2.072	0.697	4.828	SSV
48	0.075	1.555	0.075	0.499	0.311	0.257	2.006	HC
49	0.765	2.268	0.763	1.362	2.166	13.404	100.000	SSV
50	0.535	2.955	0.521	1.522	1.562	6.891	100.000	SSV
51	0.104	3.063	0.099	0.417	0.157	0.615	4.172	SSV
52	0.141	1.681	0.132	0.597	0.473	0.325	1.543	SSV
53	1.034	3.618	1.024	2.032	3.003	1.304	6.208	SSV
54	0.450	2.449	0.446	1.102	1.131	0.712	3.600	SSV
55	0.448	3.797	0.434	1.355	0.965	1.177	6.109	SSV
56	0.344	3.222	0.331	1.288	1.132	0.707	9.789	SSV
57	0.848	3.297	0.836	1.816	2.575	13.421	100.000	SSV
58	0.392	2.761	0.384	1.340	1.256	0.640	2.784	SSV
59	0.332	1.633	0.316	0.618	0.469	44.178	100.000	SSV
60	0.380	2.470	0.369	1.318	1.576	0.604	4.590	SSV
61	0.432	1.766	0.460	0.700	0.549	68.875	100.000	HC
62	0.529	3.825	0.547	1.140	0.639	32.054	100.000	HC
63	0.450	3.370	0.438	1.132	0.803	6.957	100.000	SSV
64	0.374	3.043	0.373	1.369	1.294	25.334	100.000	SSV
65	0.445	4.439	0.442	2.068	1.962	0.766	4.595	SSV
66	0.264	3.789	0.255	1.572	0.997	6.747	100.000	SSV
67	0.528	2.923	0.518	1.479	1.549	0.819	3.060	SSV
68	0.677	3.568	0.659	1.847	2.184	0.987	4.006	SSV
69	0.414	3.265	0.388	1.177	0.866	7.106	100.000	SSV
70	0.108	3.166	0.101	0.479	0.185	31.671	100.000	SSV
71	0.275	2.126	0.266	0.935	0.821	25.303	100.000	SSV
72	0.097	1.949	0.089	0.584	0.366	0.311	3.397	SSV
73	0.202	1.839	0.195	0.709	0.573	0.331	2.845	SSV
74	0.290	3.158	0.281	1.314	1.228	0.471	2.809	SSV
75	0.419	3.614	0.420	1.331	0.967	13.113	100.000	HC
76	0.665	2.536	0.700	1.344	1.174	62.750	100.000	HC
77	0.280	2.829	0.264	0.880	0.483	0.756	1.432	SSV
78	0.379	3.508	0.389	1.260	0.904	31.749	100.000	HC
79	0.095	1.886	0.061	0.190	0.073	0.614	2.075	SSV
80	0.114	1.877	0.104	0.586	0.366	0.291	0.959	SSV
81	0.066	1.688	0.061	0.292	0.135	0.355	1.137	SSV

Table M.2: Forecast Error (Fitness) by Patient (#41-81)

M.2 Weighted-Kappa

Patient	HC	NOISE	SSV	SVNP	SVP	YW16	YW76	Best
0	0.722	0.000	0.798	0.451	0.449	0.746	0.446	SSV
1	0.562	0.000	0.617	0.317	0.309	0.550	0.336	SSV
2	0.734	0.000	0.788	0.485	0.481	0.758	0.518	SSV
3	0.544	0.000	0.609	0.475	0.469	0.458	0.417	SSV
4	0.797	0.000	0.878	0.552	0.556	0.861	0.436	SSV
5	0.551	0.000	0.614	0.259	0.259	0.586	0.700	YW76
6	0.552	0.000	0.571	0.406	0.402	0.371	0.000	SSV
7	0.726	0.000	0.777	0.494	0.493	0.745	0.446	SSV
8	0.742	0.000	0.827	0.615	0.608	0.577	0.000	SSV
9	0.667	0.000	0.760	0.407	0.407	0.696	0.691	SSV
10	0.803	0.000	0.890	0.561	0.549	0.817	0.422	SSV
11	0.678	0.000	0.749	0.357	0.358	0.699	0.368	SSV
12	0.617	0.000	0.662	0.383	0.368	0.598	0.474	SSV
13	0.756	0.000	0.835	0.574	0.578	0.785	0.748	SSV
14	0.879	0.000	0.971	0.819	0.814	0.911	0.164	SSV
15	0.710	0.000	0.790	0.451	0.445	0.732	0.372	SSV
16	0.776	0.000	0.848	0.531	0.536	0.812	0.601	SSV
17	0.702	0.000	0.780	0.550	0.545	0.678	0.000	SSV
18	0.563	0.063	0.579	0.416	0.423	0.376	0.000	SSV
19	0.824	0.000	0.925	0.756	0.752	0.795	0.506	SSV
20	0.750	0.000	0.803	0.486	0.483	0.770	0.582	SSV
21	0.785	0.000	0.863	0.516	0.511	0.811	0.000	SSV
22	0.671	0.000	0.716	0.501	0.498	0.537	0.000	SSV
23	0.906	0.000	0.994	0.898	0.902	0.867	0.539	SSV
24	0.792	0.000	0.885	0.564	0.570	0.828	0.158	SSV
25	0.639	0.063	0.685	0.514	0.521	0.465	0.000	SSV
26	0.792	0.000	0.853	0.515	0.513	0.795	0.470	SSV
27	0.866	0.000	0.950	0.701	0.700	0.873	0.315	SSV
28	0.568	0.063	0.614	0.406	0.408	0.476	0.000	SSV
29	0.308	0.000	0.320	0.025	0.025	0.312	-0.028	SSV
30	0.334	0.250	0.357	0.222	0.222	0.303	0.000	SSV
31	0.806	0.000	0.868	0.548	0.542	0.817	0.424	SSV
32	0.722	0.000	0.794	0.423	0.433	0.751	0.548	SSV
33	0.915	0.000	0.986	0.909	0.912	0.805	0.000	SSV
34	0.566	0.000	0.595	0.446	0.445	0.483	0.000	SSV
35	0.579	0.000	0.616	0.294	0.301	0.593	0.407	SSV
36	0.604	0.188	0.629	0.501	0.506	0.295	0.000	SSV
37	0.727	0.313	0.726	0.560	0.579	0.330	0.000	HC
38	0.828	0.000	0.915	0.741	0.752	0.803	0.388	SSV
39	0.575	0.000	0.597	0.386	0.389	0.506	0.000	SSV
40	0.606	0.188	0.620	0.550	0.560	0.138	0.000	SSV

Table M.3: Weighted-Kappa by Patient (#0-40)

Patient	HC	NOISE	SSV	SVNP	SVP	YW16	YW76	Best
41	0.577	0.000	0.614	0.374	0.378	0.545	0.000	SSV
42	0.776	0.000	0.863	0.652	0.652	0.730	0.000	SSV
43	0.717	0.000	0.790	0.429	0.423	0.766	0.697	SSV
44	0.845	0.063	0.909	0.796	0.806	0.645	0.000	SSV
45	0.900	0.000	0.982	0.756	0.747	0.906	0.504	SSV
46	0.642	0.000	0.718	0.375	0.375	0.684	0.570	SSV
47	0.756	0.000	0.827	0.545	0.542	0.804	0.365	SSV
48	0.899	0.000	0.977	0.755	0.750	0.919	0.494	SSV
49	0.626	0.000	0.670	0.454	0.448	0.565	0.000	SSV
50	0.803	0.000	0.870	0.612	0.606	0.792	0.000	SSV
51	0.890	0.000	0.983	0.907	0.921	0.889	0.466	SSV
52	0.871	0.000	0.952	0.705	0.699	0.880	0.491	SSV
53	0.672	0.000	0.720	0.451	0.454	0.682	0.285	SSV
54	0.810	0.000	0.881	0.582	0.578	0.834	0.380	SSV
55	0.838	0.000	0.903	0.676	0.677	0.803	0.450	SSV
56	0.807	0.000	0.900	0.643	0.645	0.823	0.215	SSV
57	0.690	0.000	0.732	0.453	0.459	0.652	0.000	SSV
58	0.816	0.000	0.909	0.576	0.563	0.853	0.571	SSV
59	0.661	0.063	0.677	0.555	0.549	0.466	0.000	SSV
60	0.786	0.000	0.861	0.489	0.484	0.783	0.192	SSV
61	0.707	0.125	0.714	0.680	0.673	0.297	0.000	SSV
62	0.700	0.000	0.753	0.558	0.560	0.515	0.000	SSV
63	0.760	0.000	0.838	0.631	0.629	0.734	0.000	SSV
64	0.796	0.000	0.856	0.585	0.584	0.677	0.000	SSV
65	0.810	0.000	0.902	0.590	0.584	0.841	0.487	SSV
66	0.798	0.000	0.887	0.562	0.555	0.831	0.000	SSV
67	0.786	0.000	0.865	0.563	0.556	0.805	0.505	SSV
68	0.772	0.000	0.840	0.542	0.541	0.802	0.468	SSV
69	0.802	0.000	0.881	0.641	0.646	0.749	0.000	SSV
70	0.731	0.063	0.799	0.710	0.718	0.599	0.000	SSV
71	0.739	0.000	0.824	0.548	0.553	0.605	0.000	SSV
72	0.871	0.000	0.979	0.783	0.792	0.934	0.368	SSV
73	0.867	0.000	0.951	0.748	0.747	0.897	0.038	SSV
74	0.859	0.000	0.933	0.637	0.640	0.893	0.525	SSV
75	0.836	0.000	0.893	0.664	0.657	0.779	0.000	SSV
76	0.680	0.250	0.735	0.593	0.580	0.322	0.000	SSV
77	0.853	0.000	0.941	0.765	0.752	0.826	0.733	SSV
78	0.619	0.000	0.675	0.416	0.428	0.465	0.000	SSV
79	0.856	0.000	0.974	0.897	0.893	0.708	0.324	SSV
80	0.867	0.000	0.974	0.707	0.710	0.906	0.696	SSV
81	0.879	0.000	0.984	0.871	0.877	0.896	0.644	SSV

Table M.4: Weighted-Kappa by Patient (#41-81)

References

[Agapie1996] Agapie A., Agapie A., "Parameter Estimation in Time Series Forecasting using Genetic Algorithms", Proceeding of EUFIT'96 - The European Congress on Intelligent Techniques and Soft Computing, Aarchen, Germany, pp. 2155-2158, 1996

[Agapie1997] Agapie A., Agapie A., "Economic Forecasting Using Genetic Algorithms", Proceedings of the 5th Fuzzy Days international Conference in Computational intelligence: theory and applications, LNCS 1226, Reusch B., (ed.), Springer, pp. 543-544, 1997

[Aho1986] Aho A. V., Sethi R., Ullman J. D., "Compilers: Principles, Techniques and Tools", World student series edition, Addison-Wesley, 1986

[Akaike1971] Akaike H., "Autoregressive Model Fitting for Control", Annals of the Institute of Statistical Mathematics, 23, pp. 163-180, 1971

[Akaike1974] Akaike H., "A New Look at the Statistical Model Identification", IEEE Transactions on Automatic Control, AC-19, 6, pp. 716-723, 1974

[Altman1997] Altman D. G., "Practical Statistics for Medical Research", Chapman and Hall, 1997

[Armstrong1992] Armstrong J. S., Collopy F., "Error Measures For Generalizing About Forecasting Methods: Empirical Comparisons", International Journal of Forecasting, 8, pp. 69-80, 1992

[Åsman1992]	Åsman P., "Computer-Assisted Interpretation of Visual Fields in Glaucoma", Acta Ophthalmologica Supplement, 206, pp. 7-47, 1992
[Åsman1993]	Åsman P., Heijl A., "Arcuate Cluster Analysis in Glaucoma Perimetry", Journal of Glaucoma, 2, pp. 13-20, 1993
[Bäck1993a]	Bäck T., Rudolph G., Schwefel H.-P., "Evolutionary Programming and Evolution Strategies: Similarities and Differences", Fogel D. B., Atmar W., (eds.), Proceedings of the Second Annual Conference on Evolutionary Programming, Evolutionary Programming Society, San Diego CA, pp. 11-22, 1993
[Bäck1993b]	Bäck T., Schwefel H.-P., "An Overview of Evolutionary Algorithms for Parameter Optimization", Evolutionary Computation, 1, 1, pp. 1-23, 1993
[Bäck1996]	Bäck T., "Evolutionary Algorithms in Theory and Practice", Oxford University Press, 1996
[Bäck1997]	Bäck T., Hammel U., Schwefel H.-P., "Evolutionary Computation: Comments on the History and Current State", IEEE Transactions on Evolutionary Computation, 1, 1, 1997
[Baker1985]	Baker J. E., "Adaptive Selection Methods for Genetic Algorithms", Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, pp. 101-111, 1985

[Baldi1999]	Baldi P., Brunak S., Frasconi P., Pollastri G., Soda G., "Bidirectional Dynamics for Protein Secondary Structure Prediction", Proceedings of the IJCAI-99 Workshop on Neural, Symbolic and Reinforcement Methods for Sequence Learning, pp. 77-83, 1999
[Banzhaf1998]	Banzhaf W., Nordin P., Keller R. E., Francone F. D., "Genetic Programming – An Introduction", Morgan Kaufmann, 1998
[Barnett1994]	Barnett V., Lewis T., "Outliers in Statistical Data", John Wiley and Sons, New York, 3rd edition, 1994
[Bearse1998]	Bearse P. M., Bozdogan H., "Subset Selection in Vector Autoregressive Models Using the Genetic Algorithm with Informational Complexity as the Fitness Function", Systems Analysis, Modelling, and Simulation (SAMS), 31, pp. 61-91, 1998
[Birch1995]	Birch M. K., Wishart P. K., O'Donnell N. P., "Determining Progressive Visual Field Loss in Serial Humphrey Visual Fields", Ophthalmology, 102, pp. 1227-1235, 1995
[Blanton1993]	Blanton J. L., Wainwright R. L., "Multiple Vehicle Routing with Time and Capacity Constraints using Genetic Algorithms", Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 452-459, 1993
[Box1970]	Box G. E. P., Jenkins G. M., "Time Series Analysis - Forecasting and Control", Holden-Day, 1970

[Brigatti1996]	Brigatti L., Hoffman D., Caprioli J., “Neural Networks to Identify Glaucoma With Structural and Functional Measurements”, American Journal of Ophthalmology, 121, pp. 511-521, 1996
[Brigatti1997]	Brigatti L., Nouri-Mahdavi K., Weizman M., Caprioli J., “Automatic Detection of Glaucomatous Visual Field Progression With Neural Networks”, Arch. Ophthalmol., 115, pp. 725-728, 1997
[Buntine1996]	Buntine W., “A Guide to the Literature on Learning Probabilistic Networks from data”, IEEE Transactions on Knowledge and Data Engineering, 8, 2, pp. 195-210, 1996
[Casdagli1992]	Casdagli M., Eubank S., “Nonlinear Modeling and Forecasting”, Addison-Wesley, 1992
[Chatfield1988a]	Chatfield C., “What is the 'best' method of forecasting?”, Journal of Applied Statistics, 15, 1, pp. 19-38, 1988
[Chatfield1988b]	Chatfield C., Yar M., “Holt-Winters forecasting: some practical issues”, The Statistician, 37, pp. 129-140, 1988
[Chatfield1989]	Chatfield C., “The Analysis of Time Series – An Introduction”, Chapman and Hall, 4th edition, 1989
[Chatfield1992]	Chatfield C., “A commentary on error measures”, International Journal of Forecasting, 8, pp. 100-102 (99-111), 1992
[Chatfield1995]	Chatfield C., “Model Uncertainty, Data Mining and Statistical Inference” (with discussion), Journal of the Royal Statistical Society, Series A, 158, pp. 419-466, 1995

[Chauhan1988] Chauhan B. C., Henson D. B., Hobley A. J., “Cluster Analysis in Visual Field Quantification”, Documenta Ophthalmologica, 69, pp. 25-39, 1988

[Cheng1996] Cheng G., Liu X., Wu J., Jones B., “Establishing a Reliable Visual Function Test and Applying it to Screening Optic Nerve Disease in Onchocercal Communities”, International Journal of Bio-Medical Computing, 41, pp. 47-53, 1996

[Chu1998] Chu M., “Inverse Eigenvalue Problems”, SIAM Review, 40, 1, pp. 1-39, 1998

[Counsell2001] Counsell S., Liu X., McFall J., Swift S., Tucker A., “Using Evolutionary Algorithms to Tackle Large Scale Grouping Problems: an Application to Email Log File Data”, Proceedings of the Late-Breaking Papers of the Genetic and Evolutionary Computation Conference (GECCO-2001), 2001

[Crabb1996/97] Crabb D., Fitzke F., McNaught A., Hitchings R., “A Profile of the Spatial Dependence of Pointwise Sensitivity Across The Glaucomatous Visual Field”, Perimetry Update, pp. 301-310, 1996/1997

[Dagum1995] Dagum P., Galper A., Horvitz E., Seiver A., “Uncertain Reasoning and Forecasting”, International Journal of Forecasting 11, pp. 73-87, 1995

[Darwin1998] Darwin C., “The Origin of Species”, Suriano G., (ed.), Gramercy Books, 1998

[Deb2001] Deb K., “Multi-Objective Optimization Using Evolutionary Algorithms”, Wiley, 2001

[Dejong1975]	De Jong K. A., "An Analysis of the Behaviour of a Class of Genetic Adaptive Systems", Doctoral Thesis – University of Michigan, Dissertational Abstracts International, 36, 10, 5140B, 1975
[Dempster1977]	Dempster A. P., Laird N. M., Rubin, D., B., "Maximum-Likelihood from Incomplete Data via the EM Algorithm", Journal of the Royal Statistical Society (B), 39, pp. 1-38, 1977
[Diggle1994]	Diggle P. J., Liang K.-Y., Zeger S. L., "The Analysis of Longitudinal Data", Clarendon, 1994
[Dua1995]	Dua P., Ray S. C., "A BVAR Model for the Connecticut Economy", Journal of Forecasting: Special issues on the Vector Autoregressive Model, 14, pp. 167-180, 1995
[Eshelman1993]	Eshelman L. J., Schaffer J. D., "Real-Coded genetic Algorithms and Interval-Schemata", Editor: Whitley, L., D., Foundations of Genetic Algorithms, (2), Morgan Kaufmann, pp. 187-202, 1993
[Falkenauer1998]	Falkenauer E., "Genetic Algorithms and Grouping Problems", Wiley, 1998
[Faraway1998]	Faraway J., Chatfield C., ": Time series forecasting with neural networks: a comparative study using the airline data", Journal of the Royal Statistical Society: Applied Statistics (C), 42, 2, pp. 231-250, 1998
[Fildes1998]	Fildes R., Hibon M., Makridakis S., Meade, N., "Generalising about univariate forecasting methods: further empirical evidence", International Journal of Forecasting, 14, 3, pp. 339-358, 1998

[Fisher1921]	Fisher R. A., “On the ‘Probable Error’ of a Coefficient of Correlation Deduced from a Small Sample”, Metron, 1, pp. 4-32, 1921
[Fitzke1995]	Fitzke F. W., Crabb D. P., McNaught A. I., Edgar D. F., Hitchings R. A., “Image Processing of Computerised Visual Field Data”, British Journal of Ophthalmology, 79, pp. 207-212, 1995
[Fitzke1996]	Fitzke F. W., Hitchings R. A., Poinoosawmy D., McNaught A. I., Crabb D. P., “Analysis of Visual Field Progression in Glaucoma”, British Journal of Ophthalmology, 80, pp. 40-48, 1996
[Fogel1995]	Fogel D. B., “Evolutionary Computation – Toward a New Philosophy of Machine Intelligence”, IEEE Press, 1995
[Friedman1998]	Friedman N., Murphy K., Russell S., “Learning the Structure of Dynamic Probabilistic Networks”, Proceedings of the 14th Conference on Uncertainty in AI, pp. 139-147, 1998
[Garey1979]	Garey M., Johnson D., “Computers and Intractability – A Guide to the Theory of NP-Completeness”, W. H. Freeman, San Francisco, 1979
[Ghozeil1996]	Ghozeil A., Fogel D. B., “Discovering Patterns in Spatial Data using Evolutionary Programming”, Genetic Programming 1996, Proceedings of the First Annual Conference, Cambridge, MA, MIT Press, pp. 521-527, 1996
[Gilks1996]	Gilks W. R., Richardson S., Spiegelhalter D. J., “Markov Chain Monte Carlo in Practice”, Chapman and Hall, 1996

[Goldbaum1994] Goldbaum M. H., Sample P. A., White H., Côté B., Raphaelian P., Fechtner R. D., Weinreb R. N., "Interpretation of Automated Perimetry for Glaucoma by Neural Network", Investigative Ophthalmology and Visual Science, 35, 9, pp. 3362-3373, 1994

[Goldberg1985] Goldberg D. E., Lingle R., "Alleles, Loci, and The Travelling Salesman Problem", Proceedings of an International Conference on Genetic Algorithms and Their Applications, pp. 154-159, 1985

[Goldberg1989] Goldberg D. E., "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, 1989

[Goldberg1990] Goldberg D. E., "Real-Coded Genetic Algorithms, Virtual Alphabets, and Blocking", University of Illinois at Urbana-Champaign, Technical Report no. 90001, 1990

[Goldberg1991] Goldberg D. E., Deb K., "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms", Editor: Rawlins G. J. E., Foundations of Genetic Algorithms, (1), Morgan Kaufmann, pp. 69-93, 1991

[Gourlay1973] Gourlay A. R., Watson G. A., "Computational Methods for Matrix Eigenproblems", John Wiley and Sons, 1973

[Goutte1999] Goutte C., Tofte P., Rostrup E., Nielsen F. Å., Hansen L. K., "On Clustering fMRI Time Series", NeuroImage, 9, pp. 298-310, 1999

[Graefe1857] Von Graefe A., "Über die Iridectomie bei Glaucom und über den Glaukomatöse Process", Arch. Ophthalmol., 3, pp. 456-650, 1857

[Hackworth1999]	Hackworth T., “India and Pakistan, a classic ‘Richardson’ Arms Race: A Genetic Algorithmic approach”, Gecco99: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann, pp. 1543-1550, 1999
[Haley1987]	Haley M. J., (editor), “The Field Analyzer Primer”, Allergan Humphrey, 1987
[Hand1994]	Hand D. J., “Deconstructing Statistical Questions” (with discussion), Journal of the Royal Statistical Society, Series A, 157, pp. 317-356, 1994
[Hand2001]	Hand D., Mannila H., Smyth P., “Principles of Data Mining”, MIT Press, 2001
[Hannan1970]	Hannan E. J., “Multiple Time Series”, Wiley, 1970
[Haykin1994]	Haykin S., “Neural Networks – A Comprehensive Foundation”, MacMillan, New York, 1994
[Heckerman1996]	Heckerman D., “A Tutorial on Learning with Bayesian Networks”, Technical Report MSR-TR-95-06, Microsoft Corporation, Redmond USA, 1996
[Heijl1988/89a]	Heijl A., Lindgren A., Lindgren G., “Inter-Point Correlations of Deviations of Threshold Values in Normal and Glaucomatous Visual Fields”, Perimetry Update, pp. 177-183, 1988/89
[Heijl1988/89b]	Heijl A., Åsman P., “Clustering of Depressed Points in the Normal Visual Field”, Perimetry Update, pp. 185-189, 1988/89

[Henson1997]	Henson D. B., Spenceley S. E., Bull D. R., “Artificial Neural Network Analysis of Noisy Visual Field Data in Glaucoma”, Artificial Intelligence in Medicine, 10, pp. 99-113, 1997
[Hitchings2000]	Hitchings R. A., “Glaucoma”, BMJ Publishing Group 2000, 2000
[Holden1995]	Holden K., “Vector Autoregression Modelling and Forecasting”, Journal of Forecasting: Special issues on the Vector Autoregressive Model, 14, pp. 159-166, 1995
[Holland1975]	Holland J. H., “Adaptation in Natural and Artificial Systems”, Ann Arbor, The University of Michigan Press, 1975
[Hollwich1985]	Hollwich F., “Ophthalmology: A Short Textbook”, Thieme, 2nd revised edition, 1985
[Hurvich1989]	Hurvich C. M., Tsai C., “Regression and Time Series Model Selection in Small Samples”, Biometrika, 76, 2, pp. 297-307, 1989
[Jain1999]	Jain A. K., Murty M. N., Flynn P. J., “Data Clustering: A Review”, ACM Computing Surveys, 31, 3, pp. 264-323, 1999
[Jenner2001]	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, 2, pp. 891-902, 2001

[Jones1984]	Jones R. H., “Fitting Multivariate Models to Unequally Spaced Data”, Time Series Analysis of Irregularly Observed Data, Editor: Parzen E., Lectures Notes in Statistics, 25, Springer-Verlag, pp. 158-188, 1984
[Kellam2001]	Kellam P., Liu X., Martin N., Orengo C., Swift S., Tucker A., “A Framework for Modelling Short, High-Dimensional Multivariate Time Series: Preliminary Results in Virus Gene Expression Data Analysis”, The Proceedings of the Fourth International Conference on Intelligent Data Analysis (IDA-2001), Hoffmann F., Hand D. J., Adams N., Fisher D., Guimaraes G., (eds.), LNCS 2189, Springer, pp. 218-227, 2001.
[Kohonen1989]	Kohonen T., “Self-Organization and Associative Memory”, Springer-Verlag, 1989
[Koza1992]	Koza J., “Genetic Programming: On the Programming of Computers by Natural Selection”, MIT Press, 1992
[Lilliefors1967]	Lilliefors H. W., “On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown”, Journal of the American Statistical Association, 62, 399-402, 1967
[Liu1994]	Liu X., Cheng G., Wu J., “Identifying the Measurement Noise in Glaucomatous Testing: an Artificial Neural Network Approach”, Artificial Intelligence in Medicine, 6, pp. 401-16, 1994
[Lockhart2000]	B. Lockhart, Winzeler E., “Genomics, Gene Expression and DNA Arrays”, Nature, 405, pp. 827-836, 2000

[Lush1931]	Lush J. L., "Predicting Gains in Feeder Cattle and Pigs", Journal of Agricultural Research, 42, pp. 853-881, 1931
[Lütkepohl1993]	Lütkepohl H., "Introduction to Multivariate Time Series Analysis", Springer-Verlag, 2nd edition, 1993
[Mahfoud1995]	Mahfoud S. W., "Niching methods for Genetic Algorithms", University of Illinois at Urbana-Champaign, Technical Report no. 90001, a dissertation from the University of Illinois, 1995
[Mahmoud1984]	Mahmoud E., "Accuracy in Forecasting: a Survey", Journal of Forecasting, 3, pp. 139-159, 1984
[Mandava1993]	Mandava S., Zulauf M., Zeyen T., Caprioli J., "An Evaluation of Clusters in the Glaucomatous Visual Field", American Journal of Ophthalmology 116, pp. 684-691, 1993
[MathSoft1997]	"S-Plus 4 Guide to Statistics", MathSoft, 1997
[McNaught1995]	McNaught A. I., Crabb D. P., Fitzke F. W., Hitchings R. A., "Modelling Series of Visual Fields to Detect Progression in Normal Tension Glaucoma", Graefes Archive for Clinical and Experimental Ophthalmology, 233, pp. 750-755, 1995
[McNaught1996]	McNaught A. I., Crabb D. P., Fitzke F. W., Hitchings R. A., "Visual Field Progression: Comparison of Humphrey Statpac 2 and Pointwise Linear Regression Analysis", Graefes Archive for Clinical and Experimental Ophthalmology, 234, pp. 411-418, 1996

[Michalewicz1995] Michalewicz Z., “Genetic Algorithms, Numerical Optimization, and Constraints”, Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 151-158, 1995

[Michalewicz1996] Michalewicz Z., “Genetic Algorithms + Data Structures = Evolution Programs”, Springer, 3rd edition, 1996

[Michalewicz1998] Michalewicz Z., Fogel D. B., “How To Solve It: Modern Heuristics”, Springer, 1998

[Mirkin1999] Mirkin B., “Concept Learning and Feature Selection Based on Square-Error Clustering”, Machine Learning 35, pp. 25-39, 1999

[Mosteller1977] Mosteller F., Tukey J., “Data Analysis and Regression”, Addison-Wesley, 1977

[Neumaier] Neumaier A., Schneider T., “Multivariate Autoregressive and Ornstein-Uhlenbeck Processes: Estimates for Order, Parameters, Spectral Information, and Confidence Regions”, submitted to the ACM Transactions on Mathematical Software

[Nouri-Mahdavi1997] Nouri-Mahdavi K., Brigatti L., Weizman M., Caprioli J., “Comparison of Methods to Detect Visual Field Progression in Glaucoma”, Ophthalmology, 104, pp. 1228-1236, 1997

[Numata1998] Numata M., Sugawara K., Yoshihara I., Abe K., “Time Series Prediction by Genetic Programming”, Genetic Programming 1998 Conference: Late Breaking Papers, Edited by Koza J., pp. 176-179, 1998

[Oates1996]	Oates T., Cohen P. R., "Searching for Structure in Multiple Streams of Data", Proceedings of the 10th International Conference in Machine Learning, 1996
[Oates1999]	Oates T., Schmill M., Cohen P. R., "Efficient Mining of Statistical Dependencies", Proceedings of 16th IJCAI, 1999
[Pearl1988]	Pearl J., "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference", Morgan Kaufmann, 1988
[Pearson1896]	Pearson K., "Mathematical contributions to the Theory of Evolution. – III. Regression, Heredity and Panmixia", Philosophical Transactions of the Royal Society of London, Series A, Containing Papers of a Mathematical or Physical Character, Vol. 187, pp. 253-318, 1896
[Pena1987]	Pena D., Box G., "Identifying a Simplifying Structure in Time Series", Journal of American Statistical Association 82, pp. 836-843, 1987
[Pfeifer1980a]	Pfeifer E. P., Deutsch S. J., "A Three-Stage Iterative Procedure for Space-Time Modeling", Technometrics, 22, 1, pp. 35-47, 1980
[Pfeifer1980b]	Pfeifer E. P., Deutsch S. J., "Identification and Interpretation of First Order Space-Time ARMA Models", Technometrics, 22, 3, pp. 397-408, 1980
[Quinn1980]	Quinn B. G., "Order Determination for a Multivariate Autoregression", Journal of the Royal Statistical Society (B), 42, 2, pp. 182-185, 1980

[Rolf1997] Rolf S., Sprave J., Urfer W., "Model Identification and Parameter Estimation of ARMA Models by means of Evolutionary Algorithms", Conference on Computational Intelligence for Financial Engineering (CIFEr), Crowne Plaza Manhattan, New York City, 1997

[Russell1995] Russell S., Norvig P., "Artificial Intelligence, A Modern Approach", Prentice Hall, pp.111-112, 1995

[Schwarz1978] Schwarz G., "Estimating the Dimension of a Model", The Annals of Statistics, 6, 2, pp. 461-464, 1978

[Shahar1996] Shahar Y., Musen M. A., "Knowledge-Based Temporal Abstraction in Clinical Domains", Artificial Intelligence in Medicine, 8, pp. 267-298, 1996

[Shahar1997] Shahar Y., "A Framework for Knowledge-Based Temporal Abstraction", Artificial Intelligence, 90, pp. 79-133, 1997

[Sharif1998] Sharif A. M., Barrett A. N., "Seeding a genetic Population for Mesh Optimisation and Evaluation", Genetic Programming 1998 Conference: Late Breaking Papers, Koza J., (ed.), pp. 195-200, 1998

[Shi1997] Shi Z., Aoyama H., "Estimation of the Exponential Auto-Regressive Time Series Model by using the Genetic Algorithm", Journal of Sound and Vibration, 205, pp. 309-321, 1997

[Snedecor1967] Snedecor G., Cochran, W., "Statistical Methods", Iowa State University Press, 6th edition, 1967

[Spearman1904] Spearman C., “The Proof and Measurement of Association Between Two Things”, The American Journal of Psychology, 15, pp. 73-101, 1904

[Spenceley1994] Spenceley S. E., Henson D. B., Bull D. R., “Visual Field Analysis using Artificial Neural Networks”, Ophthalmic and Physiological Optics, 14, pp. 239-248, 1994

[Spenceley1996] Spenceley S. E., Henson D. B., Bull D. R., “Spatial Classification of Glaucomotas Visual Field Loss”, British Journal of Ophthalmology, 80, pp. 526-531, 1996

[Steeg1998] Steeg E., Robinson W. D., Willis E., “Coincidence Detection: A Fast Method for Discovering Higher-Order Correlations in Multidimensional Data”, Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, New York, pp. 112-120, 1998

[Stewart1998] Stewart G. W., “Matrix Algorithms Volume 1, Basic decompositions”, Philadelphia: Society for Industrial and Applied Mathematics, 1998

[Swift1999a] Swift S., Liu X., “Modelling and Forecasting of Glaucomatous Visual Fields Using Genetic Algorithms”, Gecco99: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann, pp. 1731-1737, 1999

[Swift1999b] Swift S., Tucker A., Liu X., “Evolutionary Computation to Search for Strongly Correlated Variables in High-Dimensional Time-Series”, Proceedings of Intelligent Data Analysis 99 (IDA-99), Hand D. J., Kok J. N., Berthold M. R., (eds.), LNCS 1642, Springer-Verlag, pp. 51-62, 1999

[Swift2001]	Swift S., Tucker A., Martin N., Liu X., “Grouping Multivariate Time Series Variables: Applications to Chemical Process and Visual Field Data”, Knowledge Based Systems Journal, Elsevier, 14, pp. 147-154, 2001
[Swift2002]	Swift S., Liu X., “Predicting Glaucomatous Visual Field Deterioration Through Short Multivariate Time Series Modelling”, Artificial Intelligence in Medicine, 24, pp. 5-24, 2002
[Syswerda1989]	Syswerda G., “Uniform Crossover in Genetic Algorithms”, Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 10-19, 1989
[Tucker1999]	Tucker A., Liu X., “Extending Evolutionary Programming to the Learning of Dynamic Bayesian Networks”, Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann, pp. 923-929, 1999
[Tucker2001a]	Tucker A., Liu X., Ogden-Swift A., “Evolutionary learning of dynamic probabilistic models with large time lags”, The International Journal of Intelligent Systems, 16, 5, Wiley, pp. 621-646, 2001
[Tucker2001b]	Tucker A., Swift S., Liu X., “Grouping Multivariate Time Series via Correlation”, IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics, 31, 2, pp. 235-245, 2001
[Ungar1995]	Ungar L. H., “Forecasting”, in “The Handbook of Brain Theory and Neural Networks”, Arbib M. A., (ed.), MIT Press, pp. 399-403, 1995

[Venugopal1992]	Venugopal V., Narendran T. T., “Cell formation in manufacturing systems through simulated annealing: An experimental evaluation”, European Journal of Operational Research, 63, pp. 409-422, 1992
[Viswanathan1997]	Viswanathan A. C., Fitzke F. W., Hitchings R. A., “Early Detection of Visual Field Progression in Glaucoma: A Comparison of PROGRESSOR and STATPAC2”, British Journal of Ophthalmology, 81, pp. 1037-1042, 1997
[Webb1995]	Webb R. H., “Forecasts of Inflation from VAR Models”, Journal of Forecasting: Special issues on the Vector Autoregressive Model, 14, pp. 267-285, 1995
[Weigend1994]	Weigend A. S., Garshenfeld N. A., Time Series Prediction, Addison-Wesley, 1994
[Whittaker1990]	Whittaker J., “Graphical Models in Applied Multivariate Statistics”, Wiley, 1990
[Whittle1984]	Whittle P., “Prediction and Regulation”, Basil Blackwell, 2nd edition, 1984
[Wild1993]	Wild J. M., Hutchings N., Hussey M. K., Flanagan J. G., Trope G. E., “Pointwise Topographical and Longitudinal Modeling of the Visual Field in Glaucoma”, Investigative Ophthalmology and Visual Science, 34, 6, pp. 1907-1916, 1993
[Wild1997]	Wild J. M., Hutchings N., Hussey M. K., Flanagan J. G., Trope G. E., “Pointwise Univariate Linear Regression of Perimetric sensitivity against Follow-up Time in Glaucoma”, Ophthalmology, 104, pp. 808-815, 1997

[Wright1986] Wright D. J., “Forecasting Data Published at irregular Time Intervals Using an Extension of Holt’s Method”, Management Science, 32, 4, pp. 499-510, 1986

[Wyatt1995] Wyatt J. C., “Hospital information management: the need for clinical leadership”, British Medical Journal, 311, pp. 175-178, 1995

[Zirilli1997] Zirilli J. S., “Finacial Prediction using Neural Networks”, Thomson Computer Press, 1997

[Zlatev1991] Zlatev Z., “Computational Methods for General Sparse Matrices”, Kluwer Academic Publishers, 1991