RGFGA: An Efficient Representation and Crossover for Grouping Genetic Algorithms

Allan Tucker allan.tucker@brunel.ac.uk
Department of Information Systems and Computing,
Brunel University, Uxbridge, Middlesex, UB8 3PH, UK

Jason Crampton jason.crampton@rhul.ac.uk
Department of Mathematics, Royal Holloway,
University of London, Egham, Surrey, TW20 0EX, UK

Stephen Swift stephen.swift@brunel.ac.uk
Department of Information Systems and Computing,
Brunel University, Uxbridge, Middlesex, UB8 3PH, UK

Abstract

There is substantial research into genetic algorithms that are used to group large numbers of objects into mutually exclusive subsets based upon some fitness function. However, nearly all methods involve degeneracy to some degree.

We introduce a new representation for grouping genetic algorithms, the *restricted growth function genetic algorithm*, that effectively removes all degeneracy, resulting in a more efficient search. A new crossover operator is also described that exploits a measure of similarity between chromosomes in a population. Using several synthetic datasets, we compare the performance of our representation and crossover with another well known state-of-the-art GA method, a strawman optimisation method and a well-established statistical clustering algorithm, with encouraging results.

Keywords

Grouping, Genetic algorithms, Restricted growth functions, Hill climbing, Multivariate time series, Bin packing.

1 Introduction

Problems that require the partitioning of a set of variables in order to compute a solution are typically **NP**-hard. Examples include the *bin packing problem* [Garey and Johnson, 1979] and the *line balancing problem* [Sacerdoti, 1977]. Hence, researchers have focused on producing heuristic methods for finding appropriate partitions. Algorithms that compute approximate solutions for grouping problems include statistical clustering algorithms [Jain et al., 1999], and optimisation algorithms such as hill climbing and evolutionary algorithms.

Falkenauer's *grouping genetic algorithm* (GGA), has been designed for dealing with grouping problems [Falkenauer, 1999]. In our previous work, we adapted the PMX crossover operator, developed for ordering problems [Goldberg and Lingle, 1985], and the GGA to deal with grouping *multivariate time series* (MTS) variables [Tucker et al., 2001b].

When dealing with an n dimensional Multivariate Time Series (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. Therefore we are interested in developing methods to decompose a high-dimensional MTS into groups of smaller MTSs, such that the dependency between variables within the same group is high, but very low with variables in another group. We define an MTS as a series of observations, $x_i(t)$; $[i = 1, \ldots, n; t = 1, \ldots, T]$, made sequentially through time where i indexes the measurements made at each time point t.

Many of the representations used in optimisation algorithms including those in the GAs described above suffer from *degeneracy*. Degeneracy occurs according to [Radcliffe and Surry, 1995] when multiple chromosomes represent the same solution. This is not the same as *redundancy* which is defined as the amount of excess information in the chromosome. Degeneracy can lead to inefficient coverage of the search space as the same configuration of groups are repeatedly explored. A key component of good design is the minimisation of degeneracy [Radcliffe, 1991]. However, there has been much debate in the GA community as to whether degeneracy and neutrality (which is where fitness does not change despite alterations in genotype) are beneficial. It appears that the class of search space may affect this. For example, in [Hancock, 1992] degeneracy was found to aid the efficiency of a GA through the advantages of providing multiple ways to solve the same problem. This result was found when applied to learning neural network architectures where the *permutation problem* leads to the same hidden node configurations being defined in a number of different orders. However, in [Smith et al., 2001] neutrality was found to be less useful when dealing with variable length genotypes. Furthermore, [Falkenauer, 1996] has found that representations with less degeneracy result in more efficient GAs with respect to grouping problems. In this paper a new representation is introduced for grouping genetic algorithms that effectively removes all degeneracy. It is used to investigate whether the removal of degeneracy results in more efficient search when applied to grouping problems compared to other methods.

In the next section different grouping methods are described, including statistical clustering techniques, the hill climbing procedure and different GA representations. In Section 3 we introduce the new proposed GA as well as the representation, a similarity metric, and a set of crossover and mutation operators. There follows in Section 4 a description of the set of experiments designed to compare efficiency of the different methods when applied to the grouping of variables in MTS data and the bin packing problem. This involves outlining the datasets used in the study and the results obtained, including learning curves. Finally, the results of the experiments are discussed and conclusions are drawn in Section 5.

2 Methods

Any algorithm that applies a global search for the optimal clusters will run in time exponential to the size of the problem space, and so a heuristic or approximate procedure is normally required to cope with most real world problems [Michalewicz and Fogel, 2000]. In this section we consider some of these methods including statistical clustering, hill climbing and genetic grouping algorithms.

Hereafter we assume that the problem at hand is to create a partition $\mathcal{G}$ of the set $\{1, \ldots, n\}$ such that $\phi(\mathcal{G})$ is maximised, where ϕ is some fitness function. We denote the set $\{1, \ldots, n\}$ by $[n]$ and the set difference operator by $\setminus$.

2.1 Statistical Clustering

Much research has been done on statistical clustering and there are many different heuristic algorithms available, perhaps the most common being *k-means* [McQueen, 1967], *partitioning around medoids* (PAM) [Kaufman and Rousseeuw, 1990] and *hierarchical clustering* [Ward, 1963]. Nearly all of these algorithms make use of a starting allocation of variables (for example, based upon random points in the dataspace or upon the most correlated variables), and therefore contain bias in their search. Many are also prone to getting stuck in local maxima during the search. In this study, PAM is compared to the other methods for grouping variables. This method is chosen as it is a well established method and has generally been found the most reliable and efficient by the authors [Swift et al., 2004].

2.1.1 Partitioning Around Medoids

The PAM algorithm creates a partition into m sets by computing m *medoids*. A medoid is a representative member for each of the m sets in the partition. Informally, a medoid is the 'most central point' of a set (with respect to some distance metric).[1] The algorithm seeks to minimise the sum of the distances between the medoid and the other members of its set. The algorithm first builds a partition by selecting m medoids; the second phase of the algorithm seeks to improve this initial selection by evaluating the effect of swapping a non-medoid for a medoid, and accepting the swap if it improves the overall fitness of the partition.

```
Input: $d$ where $d(x,y)$ = distance between variables $x$
and $y$
-------------------build phase---------------------
medoids = $\emptyset$

for $k = 1$ to $m$ {
    add $c_k$ to medoids where $\sum_{j=1}^{k}\sum_{x\in\langle c_j\rangle} d(x,c_j)$ is minimised
}
-------------------swap phase----------------------
for $c \in$ medoids
{
    for $x \notin$ medoids
    {
        if replacing $x$ with $c$ improves fitness then
            medoids = medoids $\cup \{x\} \setminus \{c\}$
    }
}
Output: medoids
```

Figure 1: Pseudo-code for the PAM algorithm

One disadvantage of this method is that m is part of the input to the algorithm. In other words, PAM needs to be run for all possible values of m and the best selected. Clearly, the algorithm is computationally expensive if all possible swaps are evaluated (and all values of m are considered).

2.2 Hill Climbing

Hill Climbing (HC) solves a search or optimisation problem by making a sequence of small random changes to a random solution to the problem [Michalewicz and Fogel, 2000]. In this study, we use mutation operators to apply these small changes which are described later in this section. After each change a check is made to see if the new solution is better than the current solution, and if so, subsequent changes are made to this improved solution point in the search space. (Typically the HC algorithm makes a fixed user-specified number of changes before terminating.) It is well known that a significant drawback of this algorithm is its propensity for finding solutions that are optimal in a neighborhood (so-called *local maxima*) rather than solutions that are optimal within the entire search space.

[1] An alternative way of thinking about a partition defined by medoids is to consider the following equivalence relation: given a set of medoids $C = \{c_1,\ldots,c_k\}$, we say $x_i \sim x_j$ if x_i and x_j have the same closest medoid. It can easily be seen that this is an equivalence relation. It is well known that the set of equivalence classes is a partition.

2.3 Falkenauer's Grouping Genetic Algorithm

Falkenauer's representation [Falkenauer, 1999] contains a vector of integers where the index of the chromosome represents the variable and the value of the chromosome represents the group to which that variable is assigned. The representation also has an extra part on the chromosome which represents each group without any information about their contents. The second part of the chromosome is simply a list of the groups found in the first part. Crossover is only applied to this part of the chromosome as will be detailed below.

2.3.1 Representation

Falkenauer's grouping genetic algorithm (GGA) represents a partition of $[n]$ as $f : R$, where $f : [n] \rightarrow [n]$ is a function with range $R \subseteq [n]$.[2] We will write f as a list, where the ith position in the list denotes $f(i)$. We will write R as a list containing the elements in the range of f. Hence $[1, 2, 3, 4, 1, 1, 1, 1, 1] : [1, 2, 3, 4]$, for example, represents the partition $\{\{1, 5, 6, 7, 8, 9\}, \{2\}, \{3\}, \{4\}\}$. Note that there is redundancy in this representation: it is not necessary to include R. However, R is used to define the crossover operation in GGA. Note also that this representation is not unique: $[2, 3, 4, 1, 2, 2, 2, 2, 2] : [2, 1, 4, 3]$ represents the same partition as the example above. Indeed any permutation of $\{1, 2, 3, 4\}$ gives rise to a chromosome that represents an identical partition.

2.3.2 Crossover

Let $f_1 : R_1$ and $f_2 : R_2$ be two parent chromosomes and choose $S_1 \subset R_1$ and $S_2 \subset R_2$ at random. The crossover operation creates two children $c_1 : R_1 \cup S_2 \setminus S_1$ and $c_2 : R_2 \cup S_1 \setminus S_2$ as described below. Informally, child 1 is created in the following way:

1. Elements that are mapped into S_2 by f_2 are inserted into child 1
2. Remaining elements are mapped into $R_1 \setminus S_1$ by f_1
3. Any undefined elements are assigned a group at random

Child 2 is created in a similar fashion, reversing the use of f_1 and f_2. More formally the children are created in the following way:

Child 1

1. For all $x \in [n]$ such that $f_2(x) \in S_2$, define $c_1(x) = f_2(x)$
2. For all $x \in [n] \setminus S_2$ such that $f_1(x) \in S_1$, define $c_1(x) = f_1(x)$
3. For all $x \in [n]$ such that $c_1(x)$ is still undefined, $c_1(x) = i$, where i is chosen at random from $R_1 \cup S_2 \setminus S_1$

Child 2

1. For all $x \in [n]$ such that $f_1(x) \in S_1$, define $c_2(x) = f_1(x)$
2. For all $x \in [n] \setminus S_1$ such that $f_2(x) \in S_2$, define $c_2(x) = f_2(x)$
3. For all $x \in [n]$ such that $c_2(x)$ is still undefined, $c_2(x) = i$, where i is chosen at random from $R_2 \cup S_1 \setminus S_2$

In practice, S_1 and S_2 are chosen by picking two pairs of random values that each identify the beginning and end of a sublist in R_1 and R_2. Essentially, S_1 and S_2 are crossing sections; the values associated with each of S_1 and S_2 are injected into the respective parents.

[2]In other words, for every $r \in R$ there exists $i \in [n]$ such that $f(i) = r$.

3 The Restricted Growth Function Genetic Algorithm

In this section we introduce the concept of a *restricted growth function* (RGF) [Er, 1988, Kaye, 1976, Proskurowski et al., 1998] and describe our *restricted growth function genetic algorithm* (RGFGA). We define an RGF and demonstrate its ability to represent a candidate set of groupings without degeneracy. Then a distance measure for the representation is defined. This is used in the description of the crossover operator, which exploits the representation and metric in order to build a 'path' between two parent chromosomes. Finally a set of mutation operators is described and a 'clean up' procedure is introduced to ensure each mutation results in a valid RGF.

A *restricted growth function* is a function $f : [n] \to [n]$ such that

$$f(1) = 1 \tag{1}$$
$$f(i+1) \leqslant \max\{f(1), \ldots, f(i)\} + 1. \tag{2}$$

Note that there is a one-to-one correspondence between the set of RGFs and the set of partitions of $[n]$. In particular, the RGF represents a partition into $m \leqslant n$ groups, where 1 by convention belongs to the first group, i belongs to the $f(i)$th group, and $\max\{f(1), \ldots, f(n)\} = m$. The fact that there is a one-to-one correspondence means that there is no degeneracy in the representation of a partition using an RGF.

Henceforth we represent a RGF f as a list of n integers, the interpretation being that the ith element of the list is the value of $f(i)$. We write $\Uparrow f(i)$ to denote $\max\{f(1), \ldots, f(i)\}$ and $\Uparrow f$ to denote $\Uparrow f(n) = \max\{f(1), \ldots, f(n)\}$.

We now introduce the idea of 'distance' between two RGFs based upon the Hamming distance [Hamming, 1950], two methods for deriving new RGFs from existing ones and prove some elementary results.

Definition 3.1 *Let f and g be two RGFs. We say $f \leqslant g$ if $f(i) \leqslant g(i)$, $1 \leqslant i \leqslant n$. We write $f < g$ if $f \leqslant g$ and $f \neq g$.*

Definition 3.2 *Let f and g be two RGFs. We define*

$$h(f,g) = \sum_{i=1}^{n} \left(f(i) - g(i)\right), \tag{3}$$
$$H(f,g) = \sum_{i=1}^{n} |f(i) - g(i)|. \tag{4}$$

Proposition 3.1 *Let f and g be RGFs. Then*

$$h(f,g) \leqslant H(f,g) \leqslant \frac{1}{2}n(n-1).$$

Proof The minimum value of $f(i)$ is 1 and the maximum value of $g(i)$ is i. Hence

$$H(f,g) \leqslant \sum_{i=1}^{n} i - 1 = \frac{1}{2}n(n-1).$$

The left hand inequality is immediate. ■

Definition 3.3 *Let f and g be RGFs. Define $\overline{fg} : [n] \to [n]$, where*

$$\overline{fg}(i) = \max\{f(i), g(i)\}$$

Definition 3.4 *Let f and g be RGFs such that $f \neq g$ and let j be the smallest integer such that $f(j) < g(j)$, $2 \leqslant j \leqslant n$ and k be the largest integer such that $f(k) > g(k)$. We define the functions*

$$(f{\uparrow}g)(i) = \begin{cases} f(i)+1 & \textit{if } i = j \textit{ and } f(j) < g(j), \\ f(i) & \textit{otherwise}; \end{cases}$$

and

$$(f{\downarrow}g)(i) = \begin{cases} f(i)-1 & \textit{if } i = k \textit{ and } f(k) > g(k), \\ f(i) & \textit{otherwise}. \end{cases}$$

	Function	**Partition**
f	$[1,2,3,1,4,1,2,5]$	$\{\{1,4,6\},\{2,7\},\{3\},\{5\},\{8\}\}$
g	$[1,2,2,1,3,3,1,4]$	$\{\{1,4,7\},\{2,3\},\{5,6\},\{8\}\}$
$\overline{fg}$	$[1,2,3,1,4,3,2,5]$	$\{\{1,4\},\{2,7\},\{3,6\},\{5\},\{8\}\}$
$(f{\uparrow}g)$	$[1,2,3,1,4,2,2,5]$	$\{\{1,4\},\{2,6,7\},\{3\},\{5\},\{8\}\}$
$(f{\downarrow}g)$	$[1,2,3,1,4,1,2,4]$	$\{\{1,4,6\},\{2,7\},\{3\},\{5,8\}\}$

Table 1: Examples of RGFs

Table 1 shows examples of RGFs f, g, $\overline{fg}$, $(f{\uparrow}g)$ and $(f{\downarrow}g)$ and the corresponding partitions. Informally, $\overline{fg}$ is an RGF that simply takes the larger of each of the corresponding pairs of values of f and g, and $(f{\uparrow}g)$ is an RGF that is one step closer to g than f is. More formally, we have the following results.

Proposition 3.2 *Let f and g be RGFs. Then $\overline{fg}$ is an RGF and $\overline{fg} = \overline{gf}$.*

Proof First note that $\overline{fg}(1) = 1$ since $f(1) = g(1) = 1$. Now

$$\begin{aligned} \overline{fg}(i) &\leqslant \max\left\{\Uparrow\overline{fg}(i-1), f(i), g(i)\right\} & (5) \\ &\leqslant \max\{\Uparrow f(i-1), \Uparrow g(i-1), f(i), g(i)\} & (6) \\ &\leqslant \max\{\Uparrow f(i-1)+1, \Uparrow g(i-1)+1\} & (7) \\ &\leqslant \Uparrow\overline{fg}(i-1)+1 & (8) \end{aligned}$$

That is, $\overline{fg}$ is an RGF. Clearly $\overline{fg} = \overline{gf}$. ■

Proposition 3.3 *Let f and g be RGFs such that $f \neq g$. Then $(f{\uparrow}g)$ is an RGF and*

$$H((f{\uparrow}g), g) = H(f,g) - 1.$$

Proof We first note that $(f{\uparrow}g)(1) = 1$. Let j be the smallest integer such that $f(j) \neq g(j)$. Now, since $g(j-1) = f(j-1)$ and g is an RGF, we have

$$g(j) \leqslant \Uparrow f(j-1)+1 \quad \text{and hence} \quad f(j) \leqslant \Uparrow f(j-1). \qquad (9)$$

We also have

$$\Uparrow\overline{fg}(j-1) = \Uparrow f(j-1). \qquad (10)$$

By definition,

$$\begin{aligned} (f{\uparrow}g)(j) &= f(j)+1 & (11) \\ &\leqslant \Uparrow f(j-1)+1 \quad \text{by (9)} & (12) \\ &= \Uparrow(f{\uparrow}g)(j-1)+1 \quad \text{by (10)}. & (13) \end{aligned}$$

That is, $(f{\uparrow}g)$ is an RGF. Furthermore,

$$\begin{aligned}
H((f{\uparrow}g),g) &= \sum_{i=1}^{n} |(f{\uparrow}g)(i) - g(i)| \\
&= \sum_{i\neq j} |(f{\uparrow}g)(i) - g(i)| + |(f{\uparrow}g)(j) - g(j)| \\
&= \sum_{i\neq j} |f(i) - g(i)| + |f(j) + 1 - g(j)| \quad \text{by definition of } (f{\uparrow}g) \\
&= \sum_{i\neq j} |f(i) - g(i)| + g(j) - f(j) - 1 \quad \text{since } f(j) + 1 - g(j) \leqslant 0 \\
&= \sum_{i\neq j} |f(i) - g(i)| + |f(j) - g(j)| - 1 \\
&= \sum_{i=1}^{n} |f(i) - g(i)| - 1 \\
&= H(f,g) - 1
\end{aligned}$$

■

A result analogous to Proposition 3.3 can be proved for $(f{\downarrow}g)$.

3.1 Crossover

Let f and g be two distinct RGFs. If $f < g$, then $(f{\uparrow}g)$ exists. Clearly, $f \leqslant \overline{fg}$. Hence, by Proposition 3.3, if $f \neq \overline{fg}$ then $(f{\uparrow}\overline{fg})$ is an RGF that is one step closer (in terms of the distance function H) to $\overline{fg}$ than f is. Hence, by repeating this construction, there exists a finite sequence of RGFs, $f_1, \ldots, f_k$ such that $f_1 = f$, $f_i = (f_{i-1}{\uparrow}\overline{fg})$ and $f_k = \overline{fg}$. Similarly, there exists a finite sequence of RGFs, $g_1, \ldots, g_l$ such that $g_1 = \overline{fg}$, $g_i = (g_{i-1}{\downarrow}g)$, and $g_l = g$. We say the sequence of functions $f_1, \ldots, f_k, g_1, \ldots, g_l$ is a *path* from f to g.

The crossover operator works by building a path between two parent chromosomes f and g. The children are two randomly chosen points (RGFs) on this path. Previously, Reeves has explored the use of *path relinking* between local maxima to investigate the *big valley* nature of some fitness landscapes, where local maxima were clustered around the global maxima and not dispersed uniformly over the landscape [Reeves and Yamada, 1998]. RGFGA crossover makes this assumption and experiments support this where we explore the fitness landscape along such paths, and observe numerous other maxima between local optima. See Figure 2 for an example fitness path between two local optima on the MTS data with 500 variables. Future work will explore other methods for selecting children along these paths. Note that if the hamming distance between two parents is less than 3 then two new children cannot be created along the path. In this case copies of each parent are created before applying mutation.

Example 3.1 *Let* $f = [1,2,3,1,4,1,2,5]$ *and* $g = [1,2,2,1,3,3,1,4]$. *Then* $\overline{fg} = [1,2,3,1,4,3,2,5]$ *and the path between* f *and* g *is*

$$\begin{aligned}
f_1 = f &= [1,2,3,1,4,1,2,5]\,, \\
f_2 &= [1,2,3,1,4,2,2,5]\,, \\
f_3 = g_1 = \overline{fg} &= [1,2,3,1,4,3,2,5]\,, \\
g_2 &= [1,2,3,1,4,3,2,4]\,, \\
g_3 &= [1,2,3,1,4,3,1,4]\,, \\
g_4 &= [1,2,3,1,3,3,1,4]\,, \\
g_5 = g &= [1,2,2,1,3,3,1,4]\,.
\end{aligned}$$

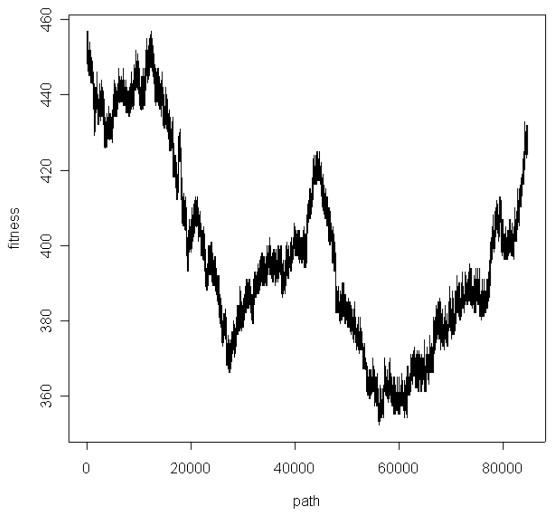


Figure 2: A typical feature landscape between two local optima. This one was calculated using the path generated during RGF crossover for 500 variable MTS grouping.

3.2 Mutation

Three mutation operators are investigated which have one of the following effects:

- an element is moved from one group to another,
- two groups are merged,
- a group is split into two groups.

These mutations have been chosen as they are typically used in conjunction with Falkenauer's crossover. Before describing the implementation of these mutation operators in detail we will consider a motivating example. Consider the function $[1, 2, 3, 1, 4, 1, 2, 5]$ and suppose that we wish to move 2 from the group containing 2 and 7 (currently labelled group 2) to the group containing 1, 4 and 6 (currently labelled as group 1). Simply changing the labelling of the second position from a 2 to a 1 is not sufficient because we obtain the chromosome $[1, 1, 3, 1, 4, 1, 2, 5]$, which does not represent a RGF. Hence we must re-label parts of the chromosome.

3.2.1 Validating Mutated Chromosomes

Let $f : [n] \to \mathbb{N}$ be some mapping from $[n]$ into the set of natural numbers. Then we can convert f into a RGF $f' : [n] \to [n]$ using a re-labelling algorithm. (As before we represent f as a list of n values where the ith place is equal to $f(i)$. Clearly not every function is a RGF.)

The algorithm requires two traversals of the list. In the first traversal, a mapping between the values of $f(i)$ and $f'(i)$ is created. We record each new value that f takes and associate it with the next available integer. This enables us, in the second traversal, to replace the values of $f(i)$ with the values of $f'(i)$. An example will illustrate the procedure.

Example 3.2 *Let $f = [3, 8, 10, 3, 1, 3, 8, 5]$. The first traversal results in the following mapping.*

$f(i)$	3	8	10	1	5
$f'(i)$	1	2	3	4	5

We now replace every occurrence of 3 with 1, every occurrence of 8 with 2, etc., to obtain $f' = [1, 2, 3, 1, 4, 1, 2, 5]$.

Clearly the time taken to execute the re-labelling procedure is proportional to the number of elements in the list. There is a risk that this relabelling will cause some possible mutations to result in large changes to the chromosome with very little change in fitness. However, as the chromosome is a legal RGF prior to mutation it is thought that the changes made by one relabelling procedure will generally be small. The mutation operators are now easy to describe.

3.2.2 Move

Moving variable i into group g. Given a chromosome representing an RGF f:

1. Pick a position i at random ($1 \leqslant i \leqslant n$) and a value g at random ($1 \leqslant g \leqslant \Uparrow f$);
2. Set $f(i) = g$;
3. Apply the re-labelling algorithm.

3.2.3 Merge

Merge groups g_1 and g_2 into a single group g_2. Given a chromosome representing a RGF f:

1. Pick values g_1 and g_2 at random ($1 \leqslant g_1, g_2 \leqslant \Uparrow f$);
2. For all i such that $f(i) = g_1$, set $f(i) = g_2$;
3. Apply the re-labelling algorithm.

3.2.4 Split

Split group g into two groups. Given a chromosome representing a RGF f:

1. Pick a value g at random ($1 \leqslant g \leqslant \Uparrow f$);
2. For all i such that $f(i) = g$, set $f(i)$ at random to either g or to $\Uparrow f + 1$;
3. Apply the re-labelling algorithm.

4 Empirical Results

In order to assess the efficiency of the RGFGA, the methods described in the previous two sections were applied to the problem of grouping variables in a number of multivariate time series and the well known bin packing problem [Garey and Johnson, 1979]. The data are described in the next section followed by a description of the fitness functions used.

4.1 Grouping MTS

As we briefly discussed in the introduction, there has been little work on the grouping of variables in an MTS. There are many statistical MTS modelling methods used for forecasting, such as the *vector auto-regressive* (VAR) [Lutkepohl, 1993], as well as other linear, non-linear and Bayesian systems [Pole et al., 1994]. Typically, these methods need to establish a large number of parameters, this number being a function of dimensionality and MTS length. Therefore, when dealing with an MTS with a large number of variables, it is desirable to model the data as a group of smaller MTS models as opposed to a single one. To use the VAR process, for example, requires at least n^2p parameters, where p is the order of the VAR process and n is the number of variables in the data set. Alternatively, suppose we are trying to learn *dynamic Bayesian network* (DBN) models [Friedman et al., 1998] from an MTS which has very high dimensionality, n, and large possible time lags, in order to explain MTSs. Then the number of possible candidate networks will be $2^{n^2 MaxLag}$ where $MaxLag$ is the maximum time lag [Tucker et al., 2001a].

Decomposing the data into smaller dimensional time-series that are independent to some degree would narrow the search space a great deal allowing the speedier production of MTS models. Therefore we are interested in finding out how to decompose a high-dimensional MTS into groups of smaller MTSs, where the dependency between variables within the same group is high, but very low with variables in another group. Note that this is different from dimensionality reduction techniques such as *principal component analysis* or *factor analysis* which make some sort of multivariate transformation of the data [Morrison, 1990].

4.2 The Bin Packing Problem

The bin packing problem [Garey and Johnson, 1979] is defined as follows: Given a set of N items, with the size of the ith item equal to s_i, and a set of bins each of capacity c, what is the minimum number of bins that can be used to pack the items such that no bin's capacity is exceeded?

4.3 Synthetic Data

The efficiency of the RGFGA is tested on a number of synthetic datasets. Firstly, some have been generated using a VAR process [Lutkepohl, 1993]. In order to generate groups of highly dependent variables, a number of time series were generated from different VAR processes. These groups had a number of different dimensions and were amalgamated to create three different multivariate time series. Secondly, bin packing data from the OR library [Falkenauer, 1996, Beasley, 1990] is used to explore the efficiency of RGFGA. A breakdown of these datasets is shown in Table 2 where n is the number of variables (dimension), l is the length of the MTS and k is the number of MTS used to generate the data, i.e. the number of groups, or the smallest known number of bins to pack the objects into without overflowing. The MTS data and its composition is available from `http://people.brunel.ac.uk/∼cssrajt/` and the bin packing data from `www.brunel.ac.uk/depts/ma/research/jeb/info.html`.

Table 2: Breakdown of the Synthetic Datasets

Dataset	**MTS1**	**MTS2**	**MTS3**	**BP1**	**BP2**	**BP3**
n	50	250	500	250	500	1000
l	1000	1000	1000	NA	NA	NA
k	14	26	35	48	198	399

4.4 Fitness Function

4.4.1 MTS Grouping Fitness

Each pair of variables in a multivariate time series may be related, where the strength of that relationship may depend on the time lag between observations for the respective variables. Our fitness function was first used by [Tucker et al., 2001b] and makes use of Pearson's Correlation Coefficient [Pearson and Lee, 1903]. For each pair of variables x_i and x_j ($1 \leqslant i, j \leqslant n$) and each time lag t ($1 \leqslant t \leqslant T$), we evaluate the correlation coefficient between the data sets

$$\{x_i(0), x_i(1) \ldots, x_i(l-t)\} \quad \text{and} \quad \{x_j(t), x_j(t+1), \ldots, x_j(l)\},$$

which we denote by $\rho(x_i, x_j, t)$. We then construct an $n \times n$ matrix, C, where

$$C_{ij} = \begin{cases} \max\{\rho(x_i, x_j, 0), \rho(x_i, x_j, 1), \ldots, \rho(x_i, x_j, T)\} & \text{if } i \neq j, \\ 0 & \text{otherwise.} \end{cases}$$

Given C_{ij} and α, where $0 < \alpha \leqslant 1$, we construct the $n \times n$ matrix C^α, such that

$$C^\alpha_{ij} = \begin{cases} 1 & \text{if } C_{ij} > \alpha, \\ 0 & \text{if } i = j, \\ -1 & \text{otherwise.} \end{cases}$$

In other words, if there exists a time lag t such that $\rho(x_i, x_j, t)$ exceeds some threshold value (α), then we assign a value of 1 in the matrix for that pair of variables.

Given a partition $\mathcal{G}$ of $[n]$, we define the fitness of $\mathcal{G}$ to be

$$\phi(\mathcal{G}) = \sum_{G \in \mathcal{G}} \sum_{x_i, x_j \in G} C^\alpha_{ij}.$$

Informally, a high value of $\phi(\mathcal{G})$ is obtained when the number of variables that are correlated in each $G \in \mathcal{G}$ is large and the number of variables that are not correlated is small.

4.4.2 Bin Packing Fitness

The RGF f represents a partition of the set $[n]$. In the context of the bin-packing problem, we interpret $f(i)$ as the index of the bin in which item i has been placed. Hence the contents of bin j is determined by $\{i \in [n] : f(i) = j\}$. Let

$$S_j = \sum_{\{i \in [n] : f(i) = j\}} s_i.$$

In other words, S_j is the sum of the sizes of items allocated to bin j by the function f. We define the fitness of f to be

$$\phi(f) = \sum_{\{i \leqslant \Uparrow(f) : S_i \leqslant c\}} \left(\frac{S_i}{c}\right)^2.$$

This definition of fitness is based on that used by Falkenauer in his work on the bin-packing problem [Falkenauer, 1996]. Intuitively, the fitness of f improves as the contents of each bin approaches the capacity of the bin. Note that the contribution of bins that overflow are ignored. If each bin contains exactly its capacity then $\phi(f) = \Uparrow(f)$.

4.5 The Algorithms

For both Falkenauer and RGFGA we used the following genetic algorithm set up: Parents are randomly selected for crossover using a uniform distribution over all individuals. The resulting children are added to the population. Mutation is applied to all individuals in the swollen population. Selection involves removing individuals with the lowest fitness until the original population size is reached. The population size is set to 50, crossover rate is 0.5 and mutation rate is 0.05. These parameters were found in general to suit both representations. For the MTS fitness function, α was set to 0.5. If α is equal to 0, then the function is maximised when all variables are placed into the same group (that is, a single large group). Alternatively, when α is equal to 1, the function is maximised when each variable is placed into its own group. A value for α should lie somewhere between the 0 and 1 so as not to skew the scoring function and will depend upon the distribution of correlations between variables. Previously we have found a suitable general value for α to be 0.5 [Swift et al., 2004]. Future work will involve a more thorough sensitivity analysis of these parameters.

For hill climbing, each mutation operator is individually applied and tested for fitness improvement.

4.6 Method Comparison

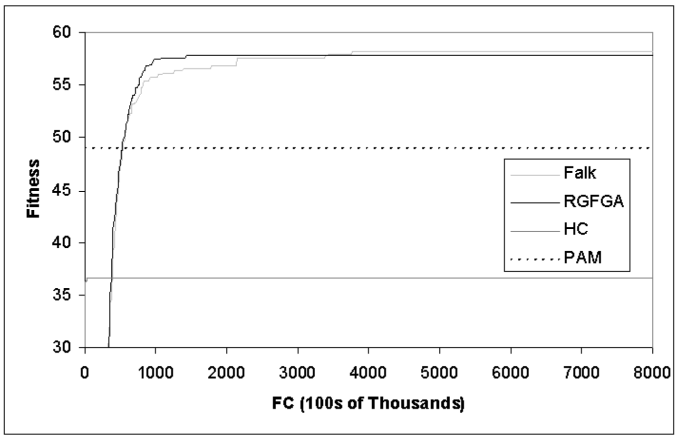
Falkenauer's GGA, the hill climbing and the proposed RGFGA were applied to each synthetic dataset in order to compare the efficiency of each algorithm as the number of variables and groups increased. The fitness of the best individual was measured every 100000 calls to the correlation matrix, C, for the MTS data and every 5000 calls to the bin packing data (as the MTS fitness function requires a lot more calls to the data matrix). All experiments were repeated 10 times and the average value recorded to account for the stochastic nature of the algorithms. For the MTS data, the statistical clustering method PAM was also used as a strawman method of comparison. We also make use of a statistic known as Weighted Kappa (WK) [Altman, 1991] in order to compare the discovered groupings with the original groupings used to generate the MTS data. This metric is used commonly in medical statistics and is used to rate agreement between the classification decisions made by two observers. In this case observer 1 is the original MTS groupings and observer 2 is the discovered groupings. The value for WK always falls between -1 and 1, where 1 denotes perfect agreement. For the bin packing data we recorded the final number of bins that were required to pack the data without overflow as a measure of quality.

Figure 3a shows the average fitness curves of each method when applied to the 50 variable MTS dataset and Table 3 shows the mean final fitness and the standard deviation for each stochastic method as well as PAM. It can be seen that even on this relatively low number of variables HC

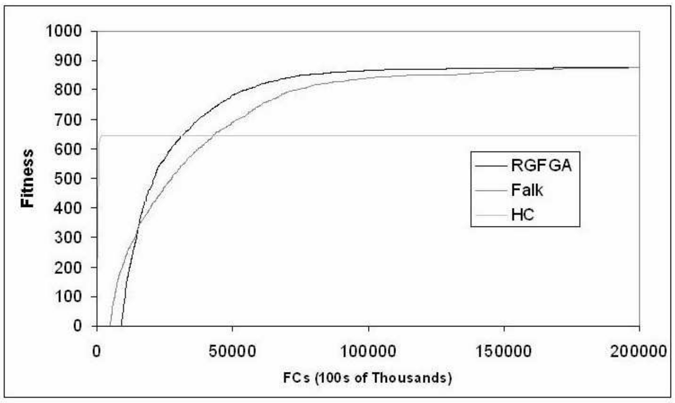
Table 3: Fitness of MTS Solution Analysis

Method	n=50	n=250	n=500
HC	36.6 (15.4)	644.8 (49.9)	2089.0 (59.2)
Falkenauer	58.2 (1.0)	889.8 (14.9)	2414.8 (31.8)
RGFGA	57.8 (0.4)	880.6 (17.4)	2551.8 (7.2)
PAM (given m)	49.0 (NA)	558.0 (NA)	1414.0 (NA)

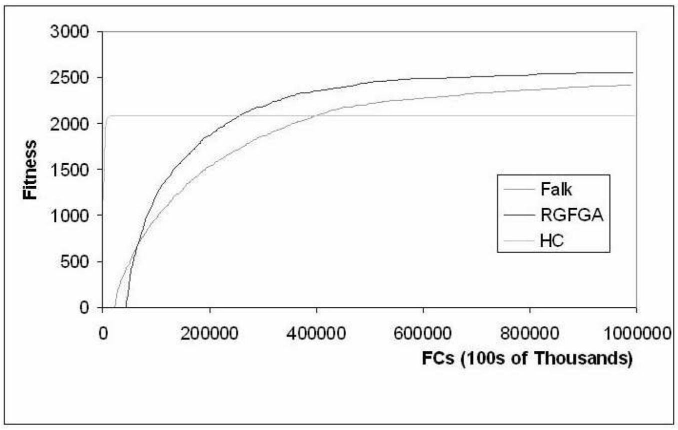
quickly hits a local maximum with an average fitness of 36.6. In comparison, Falkenauer and the proposed RGFGA do far better both converging with an average fitness of about 58, Falkenauer finishing with a slightly but not significantly higher fitness. PAM does better than HC but not as well as the GAs with a fitness of 49. PAM, however, requires the number of groups as input (m) and so has an unfair advantage being given a far smaller search space. The standard deviation between runs was highest for HC, followed by Falkenauer with RGFGA on the lowest standard



(a) 50 variable MTS



(b) 250 variable MTS

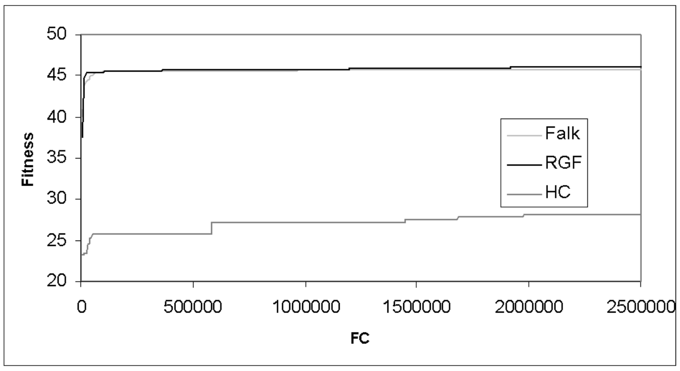


(c) 500 variable MTS

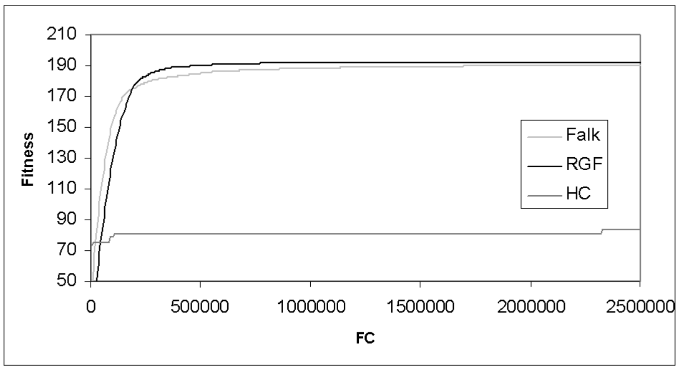
Figure 3: Comparison of the HC, PAM, GGA, and RGFGA methods for generating partitions on the MTS data

deviation. This suggests that RGFGA is more consistent than the other two methods. As PAM is deterministic, standard deviation is not applicable.

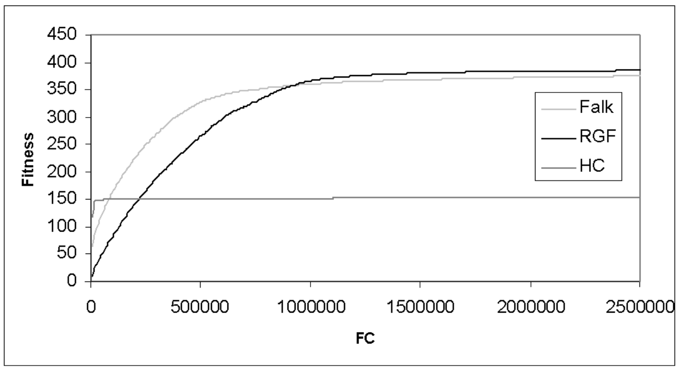
Figure 3b shows the average fitness curves for the methods on the 250 variable MTS. Again HC performs the worst and hits a local maximum of 644.8 on average. Now, however, Falkenauer



(a) 120 variable Bin Packing



(b) 500 variable Bin Packing



(c) 1000 variable Bin Packing

Figure 4: Comparison of the HC, GGA, and RGFGA methods for generating partitions on the bin packing data

and the RGFGA begin to show different learning curves. In the early generations, it appears that Falkenauer performs more efficiently with a steeper ascent. However, the RGFGA eventually overtakes Falkenauer's fitness before converging to a solution with a fitness of around 880. Note, again that Falkenauer's final fitness is slightly better than RGFGA's and the standard deviation (Table 3) is slightly lower though not by much. PAM now performs worse than all the stochastic methods with a fitness of 558, even when m is given.

Figure 3c shows the average fitness curves for the full 500 variables of the MTS data. This really shows that as the number of variables increase, the RGFGA performs more efficiently than the other methods even though Falkenauer still does slightly better during the earlier generations. This was an unexpected result as we expected RGFGA to be more efficient from the outset. A possible reason that RGFGA does not perform as well during earlier generations compared to Falkenauer is that Falkenauer's crossover is exploited more effectively in the early generations. However, as mutation becomes the dominant factor in exploring the search space, Falkenauer becomes less and less efficient whilst the RGFGA representation continues its efficient search with no degeneracy.

Table 3 shows that the average final fitness for RGFGA is clearly higher than Falkenauer (2414.8 and 2551.8, respectively). The consistency of RGFGA also appears to be much higher with a considerably lower standard deviation than the other methods.

Table 4: Fitness of Bin Packing Solution Analysis

Method	n=120	n=500	n=1000
HC	28.2 (1.6)	83.3 (5.5)	154.1 (14.6)
Falkenauer	46.1 (0.2)	190.8 (0.6)	380.1 (0.6)
RGFGA	46.2 (0.3)	191.9 (0.9)	386.5 (0.8)

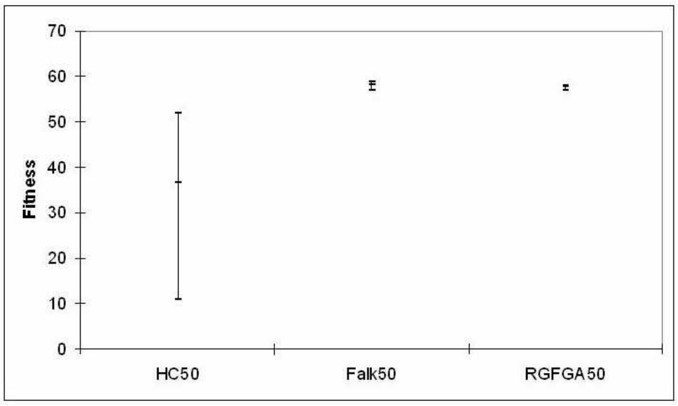
Figure 4 shows the equivalent curves for the bin packing data with 120, 500 and 1000 variables, whilst Table 4 shows the final fitnesses and standard deviations. Again HC hits a local maxima in all experiments whilst RGFGA and Falkenauer do similarly well on the lower dimensional 120 variable problem. For 500 and 1000 variables, Falkenauer appears to do well in the early generation but is overtaken by RGFGA which finishes with a higher mean fitness. However the standard deviation of the RGFGA final solutions are also slightly higher.

Figure 5 illustrates the spread of the final solution fitnesses for the MTS data. They show the mean, the maximum and the minimum for each method and dataset. It is apparent that for the smaller dimensional datasets Falkenauer and RGFGA both discover similarly fit solutions, and are considerably better than HC. However, on the larger dimensional dataset with 500 variables, it is obvious that RGFGA is more consistently finding high-fitness solutions with less variation than either of the other two methods.

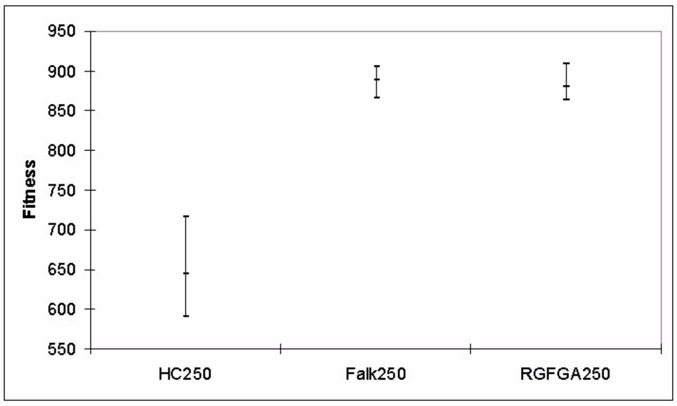
Figure 6 shows the equivalent results on the bin packing data. We have omitted the HC results as these were so much worse that they skewed the graphs. Once again it is clear that as N increases, the improvement in final fitness is more marked.

We now look at the quality of the discovered groupings compared to the original groups used to generate the MTS data by using the WK metric. Table 5 shows the mean and standard deviation of the WK scores for the groupings discovered for each method for the 50, 250 and 500 variable MTS. It is evident that the WK scores indicate that the discovered groupings closely reflect the fitness results. For smaller dimensional problems it seems that all methods perform well with high WK and low standard deviation, whereas for higher dimensional datasets RGFGA discovers groupings with higher WK and lower standard deviation, followed by Falkenauer's GGA, then PAM and HC suffering most, likely due to local maxima. Recall that m, the number of groups in the partition, is an input to PAM and so this method has an unfair advantage over the other ones.

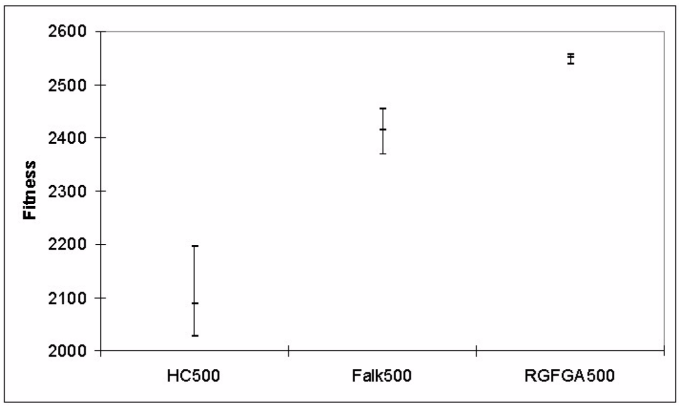
For the bin packing we used the number of bins used to pack the data in the final solution as a method for measuring the quality of the final solutions. Table 6 shows the means and standard deviations of each method. It is evident that they closely reflect the fitnesses of the solution with RGFGA resulting in better solutions (with a smaller number of bins) followed by Falkenauer and then HC. It should be noted, however, that for this set of data the standard deviation for RGFGA is slightly higher.



(a) 50 variable MTS



(b) 250 variable MTS

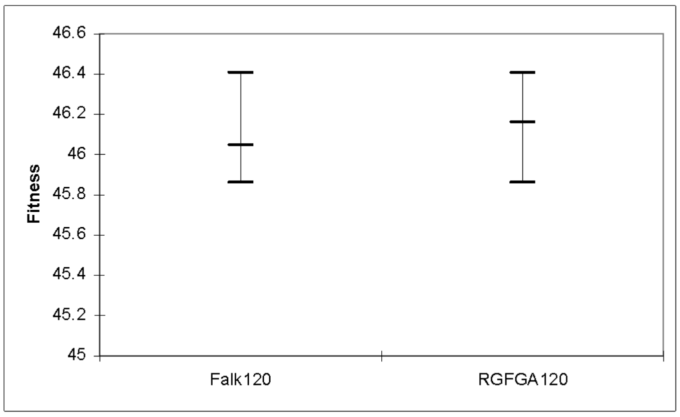


(c) 500 variable MTS

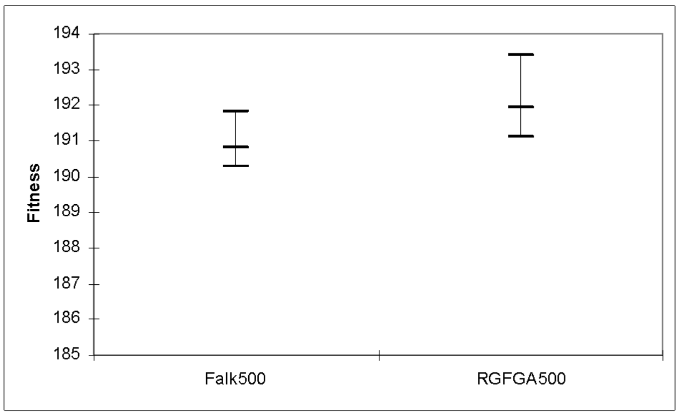
Figure 5: Comparison of fitness of final solutions on the MTS data

5 Conclusions and Future Work

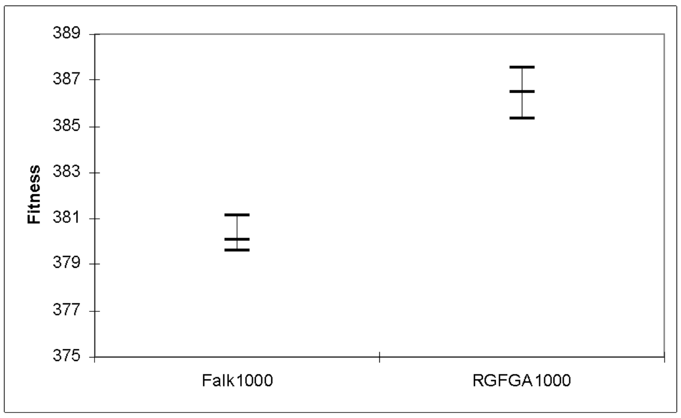
In this paper we have introduced a new representation for grouping genetic algorithms known as the Restricted Growth Function Genetic Algorithm (RGFGA). This avoids the problems of degeneracy, common in other representations. In addition, we have introduced a set of operators, including a form of crossover that only generates valid offspring, in contrast with crossover operators in other methods. We have carried out an empirical study to investigate the efficiency of



(a) 120 variable Bin Packing Data



(b) 500 variable Bin Packing Data



(c) 1000 variable Bin Packing Data

Figure 6: Comparison of fitness of final solutions on the bin packing data

our method compared to another state-of-the-art grouping genetic algorithm representation and a strawman hill climbing procedure, with favourable results.

The RGFGA crossover makes use of a Hamming distance metric which can also be used to measure overall population diversity. Figure 7 illustrates how this diversity varies during each experiment. It is interesting to note how rapidly the population diversity drops as many individuals begin to represent similar groupings. This is likely to be due to the nature of our crossover

Table 5: MTS Data Weighted Kappa Analysis

Method	n=50	n=250	n=500
HC	0.623 (0.042)	0.561 (0.032)	0.601 (0.074)
Falkenauer	0.707 (0.038)	0.576 (0.029)	0.614 (0.018)
RGFGA	0.748 (0.024)	0.626 (0.022)	0.656 (0.007)
PAM (given m)	0.816 (NA)	0.707 (NA)	0.638 (NA)

Table 6: Bin Packing Analysis

Method	n=120	n=500	n=1000
HC	58.2 (4.97)	245.20 (10.76)	494.00 (12.86)
Falkenauer	46.05 (0.21)	204.40 (0.89)	419.40 (0.89)
RGFGA	46.15 (0.26)	203.00 (1.22)	410.40 (1.14)

operator and could be limiting the speed of convergence of the RGFGA as the search comes to rely more and more upon mutation as generations pass. There are a number of methods that can be explored to help prevent early convergence of GAs. For example, *niching* [Mahfoud, 1995] involves maintaining sub-populations in order to generate multiple solutions. A limited amount of breeding between sub-populations is allowed but the idea is to allow several part-solutions to be preserved within the different populations. We intend to look into niching methods as well as investigate the potential of other operators that do not suffer from premature convergence whilst still generating valid RGF offspring. Several real-world datasets including medical data from visual field tests and biological data from gene expression experiments will also be employed to test the RGFGA as well as a parametric study of the representation.

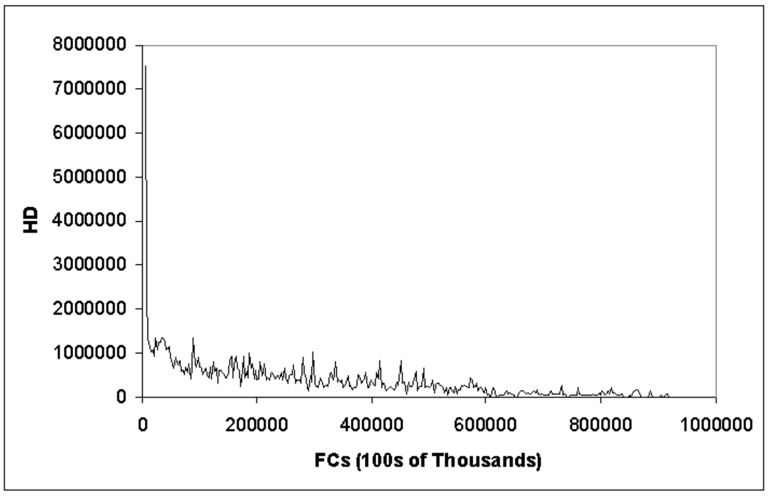


Figure 7: The Hamming distance (measure of diversity) within the population of RGFGA during learning for 500 variable MTS grouping.

6 Acknowledgements

The authors would like to thank the referees for their constructive comments of the manuscript during review.

References

[Altman, 1991] Altman, D. G. (1991). *Practical Statistics for Medical Research*. Chapman and Hall/CRC, Boca Raton, FL.

[Beasley, 1990] Beasley, J. (1990). Or-library: Distributing test problems by electronic mail. *European Journal on Operational Research*, 41:1069–1072.

[Er, 1988] Er, M. (1988). A fast algorithm for generating set partitions. *The Computer Journal*, 31(3):283–284.

[Falkenauer, 1996] Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2:5–30.

[Falkenauer, 1999] Falkenauer, E. (1999). *Genetic Algorithms and Grouping Problems*. John Wiley, New York, NY.

[Friedman et al., 1998] Friedman, N., Murphy, K., and Russell, S. (1998). Learning the structure of dynamic probabilistic networks. In Cooper, G. F. and Moral, S., editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 139–147.

[Garey and Johnson, 1979] Garey, M. and Johnson, D. (1979). *Computers and Intractability*. W. H. Freeman and Company, New York, NY.

[Goldberg and Lingle, 1985] Goldberg, D. and Lingle, R. (1985). Alleles, loci, and the travelling salesman problem. In *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 154–159.

[Hamming, 1950] Hamming, R. (1950). Error detecting and error correcting codes. *Bell Systems Technical Journal*, 29:147–160.

[Hancock, 1992] Hancock, P. (1992). Genetic algorithms and permutation problems: a comparison of recombination operators for neural net structure specification. In *Proceedings of COGANN-92: International Workshop on combinatorics of Genetic Algorithms and Neural Networks*, pages 108–122.

[Jain et al., 1999] Jain, A., Murty, M., and Flynn, P. (1999). Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323.

[Kaufman and Rousseeuw, 1990] Kaufman, L. and Rousseeuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, New York , NY.

[Kaye, 1976] Kaye, R. (1976). A Gray code for set partitions. *Information Processing Letters*, 5(6):171–173.

[Lutkepohl, 1993] Lutkepohl, H. (1993). *Introduction to Multivariate Time Series Analysis*. Springer-Verlag, London, United Kingdom.

[Mahfoud, 1995] Mahfoud, S. (1995). *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL.

[McQueen, 1967] McQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematics, Statistics and Probabiliity*, volume 1, pages 281–298.

[Michalewicz and Fogel, 2000] Michalewicz, Z. and Fogel, D. (2000). *How to Solve It: Modern Heuristics*. Springer Verlag, New York, NY.

[Morrison, 1990] Morrison, D. (1990). *Multivariate Statistical Methods*. McGraw-Hill, Inc., New York, NY, 3rd edition.

[Pearson and Lee, 1903] Pearson, K. and Lee, A. (1903). On the laws of inheritance in man: Inheritance of physical characters. *Biometrika*, 2(4):357–462.

[Pole et al., 1994] Pole, A., West, M., and Harrison, P. (1994). *Applied Bayesian Forecasting and Time Series Analysis*. Chapman-Hall, New York, NY.

[Proskurowski et al., 1998] Proskurowski, A., Ruskey, F., and Smith, M. (1998). Analysis of algorithms for listing equivalence classes of k-ary strings. *SIAM Journal on Discrete Mathematics*, 11(1):94–109.

[Radcliffe, 1991] Radcliffe, N. (1991). Equivalence class analysis of genetic algorithms. *Complex Systems*, 5:183–205.

[Radcliffe and Surry, 1995] Radcliffe, N. and Surry, P. (1995). Fitness variance of formae and performance prediction. In Whitley, D. and Vose, M., editors, *Foundations of genetic algorithms 3*, pages 51–72, San Mateo. Morgan Kaufmann.

[Reeves and Yamada, 1998] Reeves, C. and Yamada, T. (1998). Genetic algorithms, path relinking, and the flowshop sequencing problem. *Evolutionary Computation*, 6(1):45–60.

[Sacerdoti, 1977] Sacerdoti, E. (1977). *A Structure for Plans and Behavior*. Elsevier North-Holland, Amsterdam, Holland.

[Smith et al., 2001] Smith, T., Husbands, P., and O'Shea, M. (2001). Neutral networks and evolvability with complex genotype-phenotype mapping. In *Proceedings of Advances in Artificial Life, 6th European COnference, ECAL2001*, pages 272–281.

[Swift et al., 2004] Swift, S., Tucker, A., Vinciotti, V., Martin, N., Orengo, C., Liu, X., and Kellam, P. (2004). Consensus clustering and functional interpretation of data. *Genome Biology*, 5(11:R94).

[Tucker et al., 2001a] Tucker, A., Liu, X., and Ogden-Swift, A. (2001a). Evolutionary learning of dynamic probabilistic models with large time lags. *The International Journal of Intelligent Systems*, 16(5):621–646.

[Tucker et al., 2001b] Tucker, A., Swift, S., and Liu, X. (2001b). Grouping multivariate time series via correlation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 31:235–245.

[Ward, 1963] Ward, J. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244.