Revisiting the Restricted Growth Function Genetic Algorithm for Grouping Problems

R. Lewis and E. Pullin

School of Mathematics, Cardiff University, CF24 4AG, Wales.
Email: lewisR9@cf.ac.uk, emma_pullin@msn.com

April, 2011

Abstract: An overview of the Restricted Growth Function Genetic Algorithm is given. Empirically we show that the algorithm exhibits poor performance and is consistently outperformed on a range of problems by two very basic evolutionary algorithms with blind operators.

Keywords: Grouping, genetic algorithms, restricted growth functions, bin packing, equal piles, graph colouring.

1 Introduction

Grouping problems involve partitioning a set of n items into an exhaustive set of mutually exclusive subsets (groups), subject to certain constraints being met. Examples include cutting and packing problems (Wäscher et al., 2007), timetabling problems (Lewis and Paechter, 2007), routing problems (Pankratz, 2005), graph colouring problems (Galinier and Hao, 1999; Malaguti et al., 2008) and balancing problems (Falkenauer, 1998), all of which are known to be NP-hard in their general forms.

Over the past two decades an important research area has been to understand whether evolutionary algorithms (EAs) might be applied to grouping problems and, in particular, whether representation schemes and recombination operators exploiting their underlying structures can be designed. This issue is especially relevant because it has previously been argued that "traditional" evolutionary operators and representations are inappropriate with such problems because they are seen to indiscriminately break up the underlying building blocks of such problems (Falkenauer, 1998; Ross et al., 1998). To illustrate, consider an approach that uses an item-based encoding where a chromosome such as `23112` represents a solution in which item-1 is in group-2, item-2 is in group-3, an so on. First, observe that this encoding exhibits degeneracy and contradicts the Principle of Minimum Redundancy (Radcliffe, 1991) since, by simply relabeling the groups, a candidate solution using m groups might be represented by $m!$ different chromosomes. Second, it is also proposed that the application of "traditional" recombination operators (such as 1-, 2-, and n-point crossover) is inappropriate with such representations as they typically pass context-specific

information out of context, often leading to nothing more than random jumps within the search space and, depending on the problem at hand, perhaps even creating illegal offspring.

In his book, Falkenauer (1998) presents compelling arguments regarding the above and ultimately suggests that it is the *groups of items* that constitute the underlying building blocks of grouping problems. In his case this proposal motivates the design of the Grouping Genetic Algorithm (GGA), which features operators that encourage complete groups of items to be passed from parents to their progeny. In particular, the crossover operator of the GGA ensures that if particular groupings of items are seen to be present in two parents, then their offspring will also inherit these groups in their entirety (though these could then be altered via mutation). In further work, Falkenauer has also hybridised the GGA with local search methods, producing high-quality results for the bin packing and equal piles problems (Falkenauer, 1998). More recently, Brown and Sumichrast (2005) have also demonstrated the superiority of the GGA over "traditional" EAs of the type mentioned above using a collection of different grouping problems.

One drawback of the GGA, however, is that there is no "one-size-fits-all" scheme. This is because its operators need to be augmented with problem-specific constructive heuristics that are used for repairing solutions during a run. Lewis and Paechter (2007) have shown that GGA operators also experience difficulties when the problem at hand involves a small number of large groups (as opposed to many small groups), because offspring are less likely to inherit complete groups of items from both parents. In these cases the GGA recombination operator is thus seen to be more of a large mutation operator, because complete groups (building blocks) tend only to be inherited from one parent, with the remainder being formed via the chosen repair operator.

In a previous issue of this journal, Tucker et al. (2005) introduced what appears to be a more flexible alternative to the GGA, the so-called the Restricted Growth Function Genetic Algorithm (RGFGA) (see also (Tucker et al., 2007)). This method involves a specialised representation which ensures that each partition of the n items is only representable by one distinct chromosome, thus strictly adhering to the Principle of Minimum Redundancy. In addition, specialised evolutionary operators appropriate for this representation are defined. In their work, Tucker et al. compare the RGFGA to the GGA using two example problems: the grouping of variables in multivariate time series, and the one dimensional bin-packing problem. We observe, however, that although the authors claim to have gained "favourable" results for the latter problem, this comparison has been made under a fitness function of the authors' own design, and it is not clear how these results compare to other well-known approaches appearing in the literature, including the GGA itself. In this paper we seek to clarify this issue and demonstrate that the RGFGA actually produces rather poor solutions to the bin packing problem, being outperformed not only by the GGA, but also in many cases by a simple single-parse greedy heuristic. We also examine the capabilities of the RGFGA more generally by considering two further grouping problems and comparing its results against two very basic EAs

suffering from the drawbacks the RGFGA is claimed to remedy. Surprisingly in these experiments the RGFGA also demonstrates inferior performance in many cases, and we investigate the reasons as to why

2 The Restricted Growth Function Genetic Algorithm (RGFGA)

The main idea behind Tucker et al.'s algorithm is to use a method of solution representation, the Restricted Growth Function (RGF), that features a one-to-one correspondence between the set of RGFs and the set of partitions of $[n]$. Consequently, an RGF is defined as a function $f : [n] \rightarrow [n]$ such that:

$$f(1) = 1 \quad (1)$$

$$f(i+1) \leq \max\{f(1), \ldots, f(i)\} + 1. \quad (2)$$

An RGF is thus an array of n integers, with $f(i)$ representing the value (group) of the ith item. We might, therefore, consider the RGF as the canonical form of a partition of $[n]$, where the groups have been labelled such that, by definition, item-1 occurs in group-1, item-2 occurs in group-1 or 2, and so on.

In defining a recombination operator for use with RGFs, Tucker et al. make use of the following notation.

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

Definition 2 *Let f and g be two RGFs. The* distance*, based on Hamming distances, between f and g is defined:*

$$H(f, g) = \sum_{i=1}^{n} |f(i) - g(i)|. \quad (3)$$

Definition 3 *Let f and g be two RGFs. Define $\overline{fg} : [n] \rightarrow [n]$, where:*

$$\overline{fg}(i) = \max\{f(i), g(i)\} \quad (4)$$

The recombination operator of Tucker et al. operates by taking two RGFs f and g (s.t. $f \leq g$) and creates a path between these that crosses $\overline{fg}$. The resultant offspring are then two randomly selected points along this path. An example of this process is provided in fig. 1. Observe that the path from f to $\overline{fg}$ is created by incrementing the appropriate variables one-by-one from left to right in the array. The same process is then also used to form the path between g and $\overline{fg}$. It is proved by Tucker et al. that all members of this path are RGFs. Also note that the total length of this path including parents (and thus the number of possible offspring) is $H(f, g) + 1$.

Regarding mutation with the RGFGA, Tucker et al. suggest that this can come in three forms: 1) Move a randomly selected item into another randomly

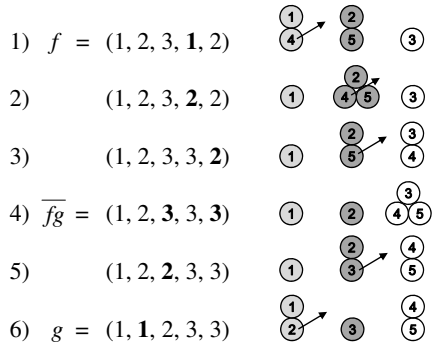

Figure 1: Example RGFGA crossover with $n = 5$ and $H(f, g) = 5$ and illustration of the resultant groupings.

selected group; 2) Merge two randomly selected groups into one; and 3) Split a randomly selected group into two groups. Further details on these can be found in their publication, but it should be noted that a chromosome resulting from these actions may not be an RGF. Thus a procedure is applied that relabels the groups to restore the RGF property. Depending on the objectives of the grouping problem under consideration, only some of these mutation schemes might be appropriate (see below).

3 Experimental Setup

The three test problems considered in our experiments are as follows:

The Bin Packing Problem (BPP) assumes a "weight" w_i for each item $1 \leq i \leq n$, and a constant C. The objective is to partition the items into a set of mutually exclusive groups $\mathcal{U} = \{U_1, \ldots, U_{|\mathcal{U}|}\}$ such that $\forall U_i \in \mathcal{U}$, $W(U_i) = \sum_{j \in U_i} w_j \leq C$, with $|\mathcal{U}|$ being minimised. Note that Falkenauer's GGA for this problem operates such that no group (bin) is ever able to exceed its capacity C. However, this is not the case for the RGFGA and a different fitness function is proposed by Tucker et al.; namely:

$$F = \sum_{\{i \leq |\mathcal{U}| : W(U_i) \leq C\}} \left(\frac{W(U_i)}{C} \right)^2 \tag{5}$$

This function, which is to be maximised, does not consider groups whose total weight exceeds C and, unlike other approaches, raises the possibility of final solutions having some of their groups over-filled, making them infeasible.

Similarly to (Falkenauer, 1998; Tucker et al., 2005) a suite of 80 "uniform" BPP instances are used in our tests. These are generated such that weights

of items are uniformly distributed between 20 and 100, with $C = 150$. For each instance the minimum number of groups needed χ is also specified and we use this to generate our initial population (see below). However, as with Tucker et al.'s experiments the number of groups is allowed to vary during a run because of the behaviour of the three mutation schemes, which are each applied with probability 1/3.

The Equal Piles Problem (EPP) assumes a weight w_i for each item $1 \leq i \leq n$, and takes a constant k. The objective is to partition the items into k groups $\mathcal{U} = \{U_1, \ldots, U_k\}$ such that the weights of each group are equal. We use the fitness function $F = \sum_{i=1}^{k} \sqrt{|W(U_i) - \mu|}$, where $\mu = (\sum_{j=1}^{n} w_j)/k$ is the mean weight of each group in an optimal solution (Falkenauer, 1998). This function is to be minimised, and a zero fitness implies a solution with equi-weighted groups.

Ten problem instances are used here, which were kindly supplied by Ben Paechter of Edinburgh Napier University. In all cases $k = 10$, and zero-fitness solutions are known to exist. Note that unlike the BPP formulation the number of groups is fixed for the EPP and so only mutation scheme 1) is used here.

The Graph k-Colouring Problem (GCOL) takes a simple undirected graph $G = (V, E)$, with $|V| = n$ and a positive integer k. The objective is to partition V into k groups $\mathcal{U} = \{U_1, \ldots, U_k\}$ such that no group contains a pair of adjacent vertices. Thus an appropriate fitness function for minimisation is simply $F = \sum_{i=1}^{k} c(i)$, where $c(i)$ gives the number of edges between pairs of vertices in each group U_i.

Five benchmark instances are used for our tests, details of which are given in Table 2. The k-values for these instances are taken from (Galinier and Hao, 1999) and it is known that zero-fitness solutions are achievable in all cases. As with the EPP, only mutation scheme 1) is used here.

In our experiments with the RGFGA, each chromosome in the initial population was produced by assigning items to a random group between 1 and χ (or for the latter two problems, 1 and k). These chromosomes were then relabeled to meet the RGF criterion. Each iteration of the algorithm then consisted of (a) two parents being selected via binary tournament selection, (b) the creation of two offspring using the RGFGA crossover operator, (c) a mutation of each offspring, and (d) insertion of the offspring into the population using binary tournament replacement. In all cases a population of size 100 was used, and the best final solution was recorded. As with Tucker et al., computational effort was gauged by monitoring the number of calls to the problem data (weight checks for the BPP and EPP, adjacency checks for GCOL) and very high cut-off points were used, giving some notion of excess time (see Table 2).

As mentioned earlier, for comparative purposes two control EAs are also used in our experiments, which differ to the RGFGA only in their methods of representation and offspring production. These are summarised as follows:

Table 1: Number of bins used in solutions from the RGFGA, FFD and GGA using the 80 uniform BPP instances. Entries in parenthesis indicate solutions seen to be infeasible due to at least one bin being over-filled.

	Instance[a]																			
	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15	#16	#17	#18	#19	#20	Av±SD[b]
$n = 120$																					
RGFGA	48	49	46	50	50	49	49	50	51	47	53	49	49	49	50	49	52	53	50	50	49.65±1.68
FFD	49	49	47	50	50	49	49	50	51	47	52	50	49	49	50	49	52	53	50	50	49.75±1.44
GGA[c]	48	49	46	49	50	48	48	49	50	46	52	49	48	49	50	48	52	52	49	49	**49.05±1.63**
$n = 250$																					
RGFGA	102	102	104	102	104	104	103	106	108	103	108	104	108	104	102	108	99	102	102	104	103.95±2.44
FFD	100	101	104	101	102	104	103	105	107	102	106	103	107	104	101	107	99	101	102	103	**103.10±2.32**
GGA	99	100	102	100	101	102	102	104	106	101	105	101	106	103	100	106	97	100	100	102	**101.85±2.43**
$n = 500$																					
RGFGA	206	210	209	212	213	214	216	212	204	210	207	209	206	202	212	208	209	205	210	205	208.95±3.53
FFD	201	204	205	207	209	207	210	207	199	204	202	203	202	198	206	204	205	201	205	199	**203.90±3.22**
GGA	198	202	203	205	206	206	208	205	197	202	200	200	199	196	204	201	202	199	202	196	**201.55±3.37**
$n = 1000$																					
RGFGA	(435)	443	441	(447)	430	435	428	436	435	432	434	430	422	425	427	436	436	438	434	(433)	433.85±5.82
FFD	403	411	416	416	402	404	399	408	404	404	404	405	398	401	400	408	407	409	403	406	**405.40±4.80**
GGA	400	407	412	413	398	400	396	404	400	399	400	402	393	397	396	404	404	405	400	400	**401.50±4.94**

[a]Problems instances are available at people.brunel.ac.uk/∼mastjjb/jeb/info.html. Results are taken from one run on each instance.

[b]Entries in bold indicate sample means that are significantly different to the RGFGA's at the 1% level according to a paired two-tailed t-test.

[c]GGA implemented according to the operators and parameters specified by Falkenauer (1994). In all cases the RGFGA and GGA were executed for 2.5×10^{10} calls.

Algorithm B: employs the item-based encoding mentioned in Section 1 and uses a classical n-point crossover, where each gene (item) is inherited from either parent with probability 0.5.

Algorithm C: also employs the item-based encoding but uses no crossover. Thus pairs of offspring are created by simply copying the two selected parents before applying mutation.

Algorithms B and C thus represent the "traditional" EAs mentioned earlier and, in particular, do not perform any relabeling, meaning they may well suffer from the various pitfalls described in Section 1.

4 Results

Our first set of experiments considers the BPP and compares the performance of the RGFGA to two well-known methods, the GGA (Falkenauer, 1994), and the first-fit decreasing (FFD) heuristic. The latter method is a classical single-parse algorithm for the BPP that operates by sorting the items into decreasing order of weight and then simply inserting these one-by-one into the first bin seen to have sufficient capacity (see for example (Dosa, 2007)).

It is obvious from the results in Table 1 that the RGFGA is outperformed by the GGA. The GGA has found higher quality solutions in seventy-two of the eighty problems, with the remaining eight (which all belong to the set of smallest problems) being of equal quality. We also see that the gap in quality widens as problem size is increased, with the RGFGA requiring an average of over thirty extra bins for instances with $n = 1000$.

Table 2: Results Summary for the RGFGA and Algorithms B and C.

Problem Description[d]	Init.	End Fitness [a,b]			Additional Information[c]		
		RGFGA	Alg. B	Alg. C	RGFGA	Alg. B	Alg. C
BPP, $n = 120$, ($\times 20$)	14.64	47.60 ± 1.50	47.47 ± 1.46	47.47 ± 1.48	49.65 (20)	49.85 (20)	49.90 (20)
BPP, $n = 250$, ($\times 20$)	27.14	98.63 ± 2.22	98.40 ± 2.12	98.55 ± 2.17	103.95 (20)	104.40 (20)	104.10 (20)
BPP, $n = 500$, ($\times 20$)	48.43	193.63 ± 3.24	$\mathbf{195.07 \pm 3.30}$	$\mathbf{195.64 \pm 3.13}$	208.95 (20)	**207.00 (20)**	**206.25 (20)**
BPP, $n = 1000$, ($\times 20$)	91.07	373.68 ± 4.65	$\mathbf{389.45 \pm 4.74}$	$\mathbf{390.40 \pm 4.81}$	433.85 (17)	**411.80 (20)**	**410.65 (20)**
EPP (#1), $n = 100$	226.47	1.10 ± 1.10	0.29 ± 0.66	0.00 ± 0.00	35	**80**	**100**
EPP (#2), $n = 100$	330.13	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	100	100	100
EPP (#3), $n = 100$	425.09	11.09 ± 6.84	6.78 ± 3.73	4.81 ± 4.77	0	0	10
EPP (#4), $n = 100$	884.97	1.67 ± 0.58	1.46 ± 0.58	0.52 ± 0.53	0	5	**50**
EPP (#5), $n = 100$	218.24	2.03 ± 4.84	0.00 ± 0.00	3.29 ± 5.69	85	100	75
EPP (#6), $n = 500$	1386.76	0.67 ± 0.83	0.10 ± 0.30	0.00 ± 0.00	50	**90**	**100**
EPP (#7), $n = 500$	333.07	0.17 ± 0.75	0.00 ± 0.00	0.00 ± 0.00	95	100	100
EPP (#8), $n = 500$	1042.06	1.11 ± 0.81	0.00 ± 0.00	0.00 ± 0.00	25	**100**	**100**
EPP (#9), $n = 500$	1037.95	0.94 ± 0.72	0.00 ± 0.00	0.25 ± 0.43	25	**100**	**75**
EPP (#10), $n = 500$	356.63	0.49 ± 1.47	0.00 ± 0.00	0.00 ± 0.00	90	100	100
GCOL, $n = 250, k = 28$	507.55	27.75 ± 3.42	$\mathbf{19.10 \pm 2.93}$	$\mathbf{18.65 \pm 1.96}$	0	0	0
GCOL, $n = 500, k = 48$	1223.15	128.65 ± 5.73	$\mathbf{56.05 \pm 3.53}$	$\mathbf{69.65 \pm 3.34}$	0	0	0
GCOL, $n = 1000, k = 83$	2882.05	517.85 ± 12.50	$\mathbf{208.15 \pm 9.88}$	$\mathbf{289.65 \pm 10.82}$	0	0	0
GCOL, $n = 300, k = 31$	635.50	41.90 ± 3.22	$\mathbf{23.65 \pm 3.13}$	$\mathbf{26.10 \pm 3.75}$	0	0	0
GCOL, $n = 1000, k = 83$	2844.15	502.05 ± 10.31	$\mathbf{189.75 \pm 7.13}$	$\mathbf{269.95 \pm 10.80}$	0	0	0

[a]Mean $\pm$ standard deviation. For the BPP, samples are formed from one run on 20 instances; else they are formed from 20 runs on each instance. Cut-off points of 2.5×10^{10}, 2.5×10^{10}, and 10^{11} calls were used for the BPP, EPP and GCOL respectively.

[b]Entries in bold indicate situations where Algorithm B or C's statistics are seen to be significantly different to the RGFGA's at the 1% level. For the BPP and GCOL this is due to a paired two-tailed t-test; for the EPP, this is due to McNemar's test on the success/failure of an algorithm in producing zero-cost solutions.

[c]For the BPP, the mean number of bins used in final feasible solutions is given, together with the number of runs where feasibility was achieved (in parenthesis). For the EPP and GCOL we give the percentage of runs where zero-fitness solutions were achieved.

[d]The GCOL instances are available online at mat.gsia.cmu.edu/COLOR/instances.html. The instances used are, respectively, DSJC250.5.col, DSJC500.5.col, DSJC1000.5.col, flat300_28_0.col, and flat1000_76_0.col.

Perhaps even more surprising, the results also indicate that the RGFGA is outperformed by the FFD heuristic. Specifically, the solutions produced by FFD are superior to the RGFGA for all of the forty larger instances ($n \geq 500$), and fifteen of the twenty 250-item instances (with the remaining five being of equal quality). However, for $n = 120$, no significant difference in the means of the RGFGA and FFD is observed.

We should also note that the results of the GGA, though clearly superior to the RGFGA, are by no means the best results available for this set of benchmarks: for example, better results for the larger instances are reported due to the hybrid-GGA of Falkenauer (1998), the ant colony optimisation-based approach of Levine and Ducatelle (2003) and the hill-climbing approach of Lewis (2009).

In our next set of experiments, we consider all three grouping problems outlined in Section 3 and examine the fitness of solutions achieved by the RGFGA compared to the two "traditional" EAs, Algorithms B and C. The results of these trials are summarised in Table 2, where we recall that for the BPP we seek to maximise fitness, while for the EPP and GCOL the aim is minimisation.

The results in Table 2 demonstrate that in many cases Algorithms B and C produce solutions with fitnesses that are significantly better than the RGFGA. However, the opposite is not true: in no cases has the RGFGA produced sig-

nificantly better results than either control algorithm.[1] Nevertheless, to bring these results into context we make note of two points. First, for the BPP all three algorithms' final results are equalled or significantly bettered by FFD; second, none of the algorithms have come close to achieving optimal solutions for GCOL, despite the fact that such solutions are known to exist. Thus although the results of Algorithms B and C can be considered rather poor (as we might expect according to the arguments of Falkenauer (1998)), the results of the RGFGA have often shown to be significantly worse still.

To explore the RGFGA's apparent lack of performance, one factor to consider is the way in which population diversity varies during a run. As noted earlier, given two parents, the number of possible offspring that can be produced by the RGFGA crossover is related to the parents' Hamming distance. On the other hand, when using n-point crossover if two parents contain different values in x of their n genes, then the number of possible offspring 2^x will usually be much higher than this figure (although, of course, many of these could be illegal and/or represent the same groupings of items). Could it be that the RGFGA crossover, with its smaller number of possible offspring, causes diversity to be lost more quickly, leading to premature convergence?

Figure 2 suggests the answer to this question to be negative. In these graphs, we use a diversity metric specialised for grouping problems due to Lewis and Paechter (2007), calculated as follows. Given a population of p individuals, each representing a partition of the n items, $\mathcal{U}_i = \{U_{i,1}, \ldots, U_{i,|\mathcal{U}_i|}\}$, $1 \leq i \leq p$,

$$D = p \left(\frac{|\bigcup_{i=1}^{p} \mathcal{U}_i|}{\sum_{i=1}^{p} |\mathcal{U}_i|} \right). \tag{6}$$

This diversity metric thus compares the number of *different* item groupings in the population to the *total* number of groupings. Consequently a fully diverse population (containing no grouping of items more than once) gives $D = p$, since $|\bigcup_{i=1}^{p} \mathcal{U}_i| = \sum_{i=1}^{p} |\mathcal{U}_i|$; on the other hand, a fully homogeneous population implies $\sum_{i=1}^{p} |\mathcal{U}_i| = p\,(|\bigcup_{i=1}^{p} \mathcal{U}_i|)$, giving $D = 1$.

As we might expect, the graphs in fig. 2 indicate that Algorithm C exhibits the most rapid loss of diversity, since offspring are produced by simply *copying* and mutating parents; however as noted above, the solutions produced by Algorithm C tend to be of higher quality than the RGFGA. On the other hand, Algorithm B's diversity tends to fall more slowly than the RGFGA in initial stages due to the random jumps initiated by the n-point crossover. However, like Algorithm C, B's results also seem offer an improvement on the RGFGA. This suggests that the diversity levels experienced by the RGFGA are not critical in its comparative lack of performance.

Perhaps more telling patterns are revealed in fig. 3. In the three left-hand graphs we show the relative frequencies of the difference in fitness between each offspring and the mean fitness of its two parents for the BPP, EPP and GCOL respectively. In each case these distributions were generated using the first

[1] Note that the RGFGA's end fitness values for the BPP are consistent with those reported by Tucker et al. (2005).

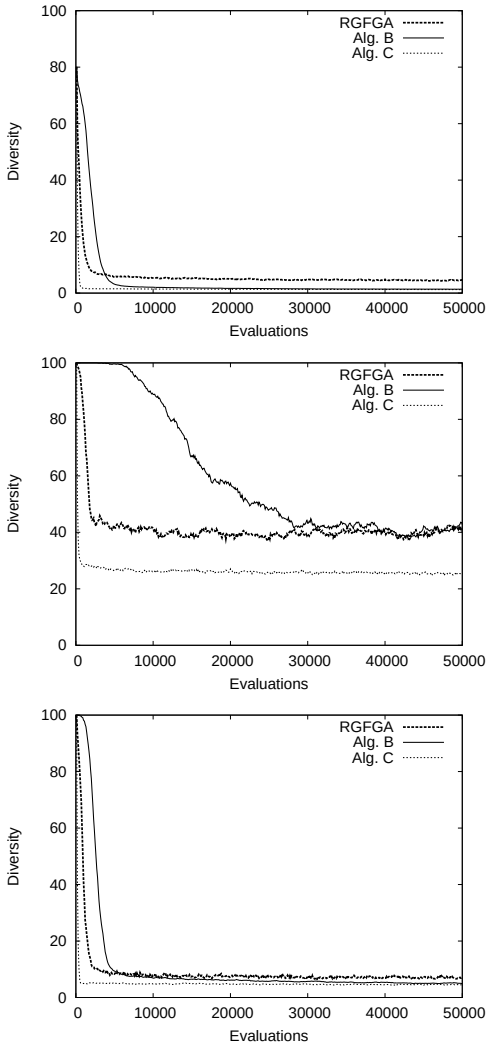

Figure 2: Changes in diversity using BPP, $n = 1000$ (top left); EPP #9 (top right); and GCOL $n = 1000$, $k = 83$ (bottom). The BPP figure is the average of one run on each of the 20 instances; the remaining figures are the mean of 20 runs on a single instance.

$500,000$ offspring generated in a run — improvements are thus characterised by positive values with the BPP and negative values for the EPP and GCOL. For all problems the distributions of the three algorithms indicate that, on average, offspring are inferior to their parents. However as we might expect, Algorithm C's distributions feature the lowest spreads (and means closest to

zero) since the offspring it produces tend to be very similar to the parents. Algorithm B's distributions are also quite similar to C's, though its spreads are higher due to the larger changes induced by the n-point crossover, which also causes a greater proportion of offspring to assume worse fitness values than their parents. However, the RGFGA features different shaped distributions to these, particularly due to the presence of long tails on the left side for the BPP, and on the right side for the EPP and GCOL. These tails have the effect of pulling the mean away from zero and increasing the spread. They also indicate that compared to Algorithms B and C, offspring produced via the RGFGA crossover are inferior to their parents more regularly, and that the degradation caused by this operator tends to be to a greater degree.

To explain the RGFGA crossover's susceptibility for producing unfit offspring, we now refer back to the illustrative example in fig. 1. Note that while chromosomes f and g define solutions featuring groups that are relatively equisized, the construction of paths from both f and g to $\overline{fg}$ involves iteratively moving items rightwards. In other words, as we progress along these paths there is a bias for items to accumulate in the higher indexed groups. However, there is generally no benefit in doing this — indeed, this tendency may offer a distinct disadvantage in many grouping problems including those considered here, where some sort of suitable "balance" of the items across groups is desirable. The right-hand graphs of fig. 3 illustrate the cumulative effects of this feature, where we indeed observe that the RGFGA produces substantially inferior performance during the course of a run compared to the other two algorithms.[2]

5 Conclusions and Discussion

Problems that require items to be grouped are common in computing and mathematics, and the potential of evolutionary-based algorithms in this domain is an interesting and on-going area of research. Indeed, at the time of writing it is arguable that such algorithms, when hybridised with suitable local search procedures, even produce state-of-the-art results in some cases (Galinier and Hao, 1999; Malaguti et al., 2008).

On first glance, the RGFGA of Tucker et al. (2005) is appealing, particularly because it elegantly copes with the issue of degeneracy and also offers a set of general-purpose evolutionary operators. However, we cannot concur with the authors' claim that the RGFGA offers "favourable" results, particularly as it is consistently outperformed by traditional evolutionary algorithms exposed to the very difficulties the RGFGA purports to remedy.

In further experiments, we also looked at the performance of a modified version of Algorithm B that, after performing n-point crossover and mutation, relabeled offspring in order to satisfy the RGF criterion. However our tests with the BPP, EPP, and GCOL indicate that this modified method actually

[2]Note that the shape of the RGFGA's curve in fig. 3 (top-right) is consistent with the equivalent graph in (Tucker et al., 2005).

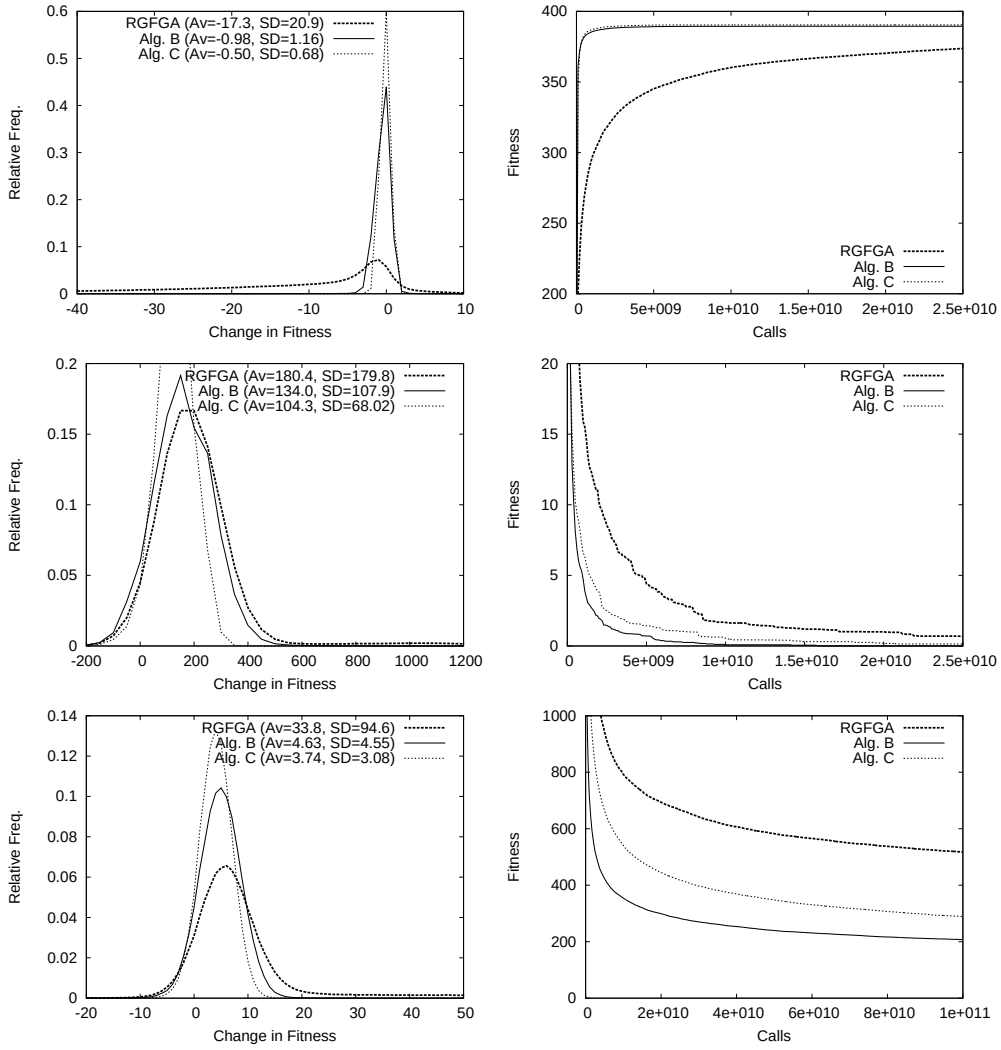

Figure 3: (left, top to bottom) Relative frequencies of the difference in fitness between each offspring and the mean fitness of their parents for the BPP, EPP and GCOL respectively; (right, top to bottom) Best-so-far graphs for the BPP, $n = 1000$; EPP #9; and GCOL $n = 1000$, $k = 83$. Each plot is the average of 20 runs.

produces inferior performance compared to the original version of B, seemingly due to difficulties it experiences in allowing item groupings to be propagated during evolution. A demonstration of why such difficulties exist is given in fig. 4. Chromosome a) in this figure is an RGF, while chromosome b) represents the

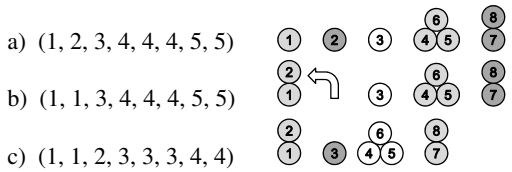

Figure 4: Illustration of the issues caused by a relabeling process.

same solution except that a single item has been moved to a different group. As a result of this change, b) no longer meets the RGF criterion and so it is necessary to relabel the groups to restore this property, as is the case with chromosome c). However, despite the fact that chromosomes a) and c) feature four identical item-groupings, we see that the RGF relabeling process has resulted in these being allocated different group numbers. This means that unlike an n-point crossover between a) and b), where groups 3, 4, and 5 would be inherited in their entirety, the recombination of a) and c) is unlikely to produce offspring featuring these complete groupings. Thus, although the strategy of relabeling groups ensures a one-to-one relationship between chromosomes and groupings, in this case it is also inhibiting the ability of crossover to combine the underlying substructures existing in the parents, compromising what is a desirable attribute in a recombination operator.

References

Brown, E. and Sumichrast, R. (2005). Evaluating performance advantages of grouping genetic algorithms. *Engineering Applications of Artificial Intelligence*, 18:1–12.

Dosa, G. (2007). The tight bound of first fit decreasing bin-packing algorithm is $\text{ffd}(i) \leq 11/9\ \text{opt}(i) + 6/9$. In Chen, B., Paterson, M., and Zhang, G., editors, *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, volume 4614 of *Lecture Notes in Computer Science*, pages 1–11. Springer, Berlin.

Falkenauer, E. (1994). A new representation and operators for genetic algorithms applied to grouping problems. *Evolutionary Computation*, 2(2):123–144.

Falkenauer, E. (1998). *Genetic Algorithms and Grouping Problems*. John Wiley and Sons.

Galinier, P. and Hao, J.-K. (1999). Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3:379–397.

Levine, J. and Ducatelle, F. (2003). Ant colony optimisation and local search for bin packing and cutting stock problems. *Journal of the Operational Research Society*, 55(12)(7):705–716.

Lewis, R. (2009). A general-purpose hill-climbing method for order independent minimum grouping problems: A case study in graph colouring and bin packing. *Computers and Operations Research*, 36(7):2295–2310.

Lewis, R. and Paechter, B. (2007). Finding feasible timetables using group based operators. *IEEE Transactions on Evolutionary Computation*, 11(3):397–413.

Malaguti, E., Monaci, M., and Toth, P. (2008). A metaheuristic approach for the vertex coloring problem. *INFORMS Journal on Computing*, 20(2):302–316.

Pankratz, G. (2005). Dynamic vehicle routing by means of a genetic algorithm. *International Journal of Physical Distribution and Logistics Management*, 35(5):362–383.

Radcliffe, N. J. (1991). Forma analysis and random respectful recombination. In R.K., B. and B, B. L., editors, *the Fourth International Conference on Genetic Algorithms*, pages 222–229, San Marco CA. Morgan Kaufmann.

Ross, P., Hart, E., and Corne, D. (1998). Some observations about ga-based exam timetabling. In Burke, E. and Carter, M., editors, *Practice and Theory of Automated Timetabling (PATAT) II*, volume 1408, pages 115–129. Springer-Verlag, Berlin.

Tucker, A., Crampton, J., and Swift, S. (2005). RGFGA: An efficient representation and crossover for grouping genetic algorithms. *Evolutionary Computation*, 13(4):477–499.

Tucker, A., Swift, S., and Crampton, J. (2007). Efficiency updates for the restricted growth function ga for grouping problems. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 1536–1536, New York, NY, USA. ACM.

Wäscher, G., Haußner, H., and Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183:1109–1130.