The Grouping Genetic Algorithm (GGA) Applied to the Bin Balancing Problem

Brian W. Zulawinski
William F. Punch III
Erik D. Goodman

Genetic Algorithm Research and Applications Group
Michigan State University

Abstract

Emanuel Falkenauer published several papers on the Grouping Genetic Algorithm (GGA) [Falkenauer 92], [Falkenauer 94]. The GGA is a new representation proposed by Falkenauer as better suited for grouping problems than the classical representations and operators usually applied to grouping or reordering problems [Ding et al., 91]. Falkenauer applied the GGA to the Bin Packing Problem and other grouping problems. This paper discusses our application of a modified GGA to a new grouping problem, the Bin Balancing Problem. This paper shows the effectiveness of this approach on various Bin Balancing Problems. In particular, we obtained excellent results with as many as 4,000 objects selected for their difficulty of packing tightly. This requires as many as 8,000 loci with an alphabet size of 4,000 and requires 96,000 bits to store the chromosome, but searches out good solutions quite rapidly.

1.1 Representation

The GGA's chromosomes are composed of two parts: the object part and the group part. The object part has one gene for each object. For example, the first object (Object 0) is in the group specified by the first gene (Group A). The object part identifies which objects form which group. The group part contains all of the groups in any order. Crossover and mutation work with the group part directly, using the object part indirectly. The group part is necessary to allow epistatic linkage among the groups under crossover. Note that since the number of groups is not constant, the chromosomes have a varying length. In the example, there are three groups. The first group (Group B) is "closer" to the second group (Group A) than it is to the third group (Group C), in strength of linkage.

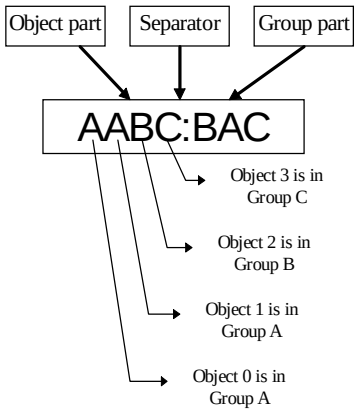


Figure 1

1.2 Crossover

The goal of the GGA's crossover is to pass possibly linked groups from parents to the children. Crossover works directly with the group part and indirectly with the object part.

Crossover is performed as follows [Falkenauer 94]:

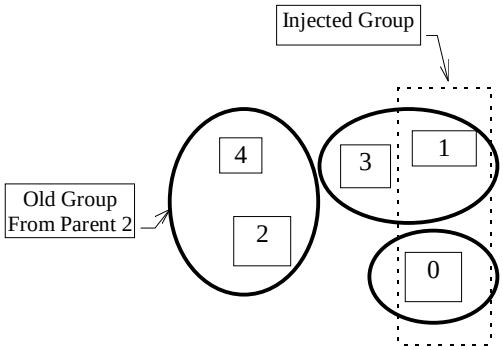
1) Select at random two crossing sites in the group part, delimiting the *crossing section*, in each of the two parents.
2) *Inject* the contents of the crossing section of Parent 1 at the first crossing site of the Parent 2. This means injecting some of the groups from Parent 1 into Parent 2.
3) Eliminate all items now occurring twice from the groups they were members of in Parent 2, so that the 'old' membership of these items gives way to the membership specified by the 'new' injected groups. Some of the 'old' groups coming from the second parent are altered: they do not contain all the same items anymore, since some of those items had to be eliminated. Remove these groups from the group part of Parent 2 and place objects belonging to these groups in a queue of objects lacking membership to a group.
4) Assign each object in the queue to a group. The algorithm used is domain dependent.
5) Steps 2) through 4) are performed on the two parents with their roles reversed in order to generate the second child.

Example: Groups from Parent 1 will be injected into Parent 2. Parent 2 is represented by small letters so that it can be differentiated from Parent 1.

(Parent 1) AABCC:ACB
(Parent 2) abcbc:cab

After step 1:

(Parent 1) AABCC:|A|CB
(Parent 2) abcbc:c|ab|



After step 2:

(Parent 1) AABCC:|A|CB
(Parent 2) abcbc:cAab

After step 3:

(Parent 1) AABCC:|A|CB
(Parent 2) AAc⊗c:cA
Queue = {object 2}

1.3 Mutation

Mutation in the GGA is performed as follows:

1) Select at random several groups to eliminate.
2) Place the objects in these groups into the queue.
3) Reinsert the objects in the queue into the solution. The algorithm used is domain dependent.

2.0 The GGA Applied to the Bin Packing Problem - Background for the Bin Balancing Problem

The Bin Packing Problem is a well-known NP-hard problem [Falkenauer 94] important to Operations Research.

Bin Packing Problem (BPP): Given identical bins of capacity *C* and *N* objects of sizes s_i such that each $s_i \leq C$, find a partition of the objects into the bins such that the number of bins, *M*, is minimized.

In order to address the Bin Packing Problem, Falkenauer makes use of several heuristics in the GGA. These heuristics are used to initialize the population and to assign "loose" objects in the queue to groups, in crossover and mutation. Without such heuristics, the GGA would not be able to address the BPP efficiently.

The heuristics in Falkenauer's BPP GGA are as follows:

1) **First Fit:** Randomize the objects. For each object, insert the object into the first bin in which it fits. If no such bin exists, start a new bin containing the object.
2) **First Fit Descending:** Sort the objects in descending order. For each object, insert the object into the first bin in which it fits. If no such bin exists, start a new bin containing the object.

Chromosomes are initialized using First Fit to produce an initial population of solutions that are valid, reasonably good, and different. First Fit Descending is used to assign the objects from the queue to groups.

Objective Function:
Falkenauer uses the following objective function for the BPP:

$$f_{BPP} = \frac{\sum_{i=1}^{M} (S_i / C)^K}{M}$$

M: number of groups
C: The capacity of the bins (given)
S_i: Size of the *i*th bin.
$K > 1$. Falkenauer uses $K = 2$.

3.0 The GGA applied to the Bin Balancing Problem

We have applied the GGA to a new grouping-type problem, the Bin Balancing Problem (BBP).

Bin Balancing Problem: Given *N* one-dimensional objects of various sizes, s_i, and *M* bins having the same variable size *C*, find a partition of the objects into the bins such that the bin size is minimized. The bin size is equal to the size of the largest partition.

The Bin Balancing Problem is closely related to the Bin Packing Problem. In the Bin Packing Problem, the size of the bins is fixed and the number of bins is to be minimized. In the Bin Balancing Problem, the number of bins is fixed and the size of the bins is to be minimized. The GGA crossover and mutation were done similarly to the way they were done in the Bin Packing Problem. First Fit Descending and First Fit cannot be used since the bin size is not fixed.

Representation: The representation was not changed.

Crossover: Crossover injects groups from Parent 1 into Parent 2. Objects belonging to an injected group will move to the injected group. Groups losing one or more objects to an injected group are removed. Other objects belonging to these groups are placed in a queue and are reinserted into the solution using a heuristic.

Crossover had to be modified in two ways for use in the Bin Balancing Problem.

- Crossover injects groups from Parent 1 into Parent 2. Any group in Parent 2 not affected by the injected groups will remain. The number of injected groups added to the number of unaffected groups can exceed the total number of groups. If this happens, groups are randomly selected and removed so that the total number of groups is equal to the (fixed) number of bins given. The objects from these groups are placed in the queue (with the others) to be reinserted into the solution.
- To keep the number of bins fixed, empty bins must be allowed to exist. The number of groups in the group part of the chromosome may exceed the number of groups represented (i.e. referenced) in the object part. Since the number of bins (groups) is fixed, the chromosomes no longer have variable length.

In Falkenauer's work on Bin Packing Problem, objects in the queue were reinserted into the solution using First Fit Descending. The population was initialized using First Fit. First Fit and First Fit Descending were replaced with *Loosest Fit* (LF) and *Loosest Fit Descending* (LFD), respectively, in the BBP. LF is used to initialize the population. LFD is used to assign objects in the queue to groups.

The following heuristics were used in the BBP:

1) **Loosest Fit (LF):** Randomize the objects. For each object, insert the object into the smallest group available. Recalculate the group size.
2) **Loosest Fit Descending (LFD):** Sort the objects in descending order. For each object, insert the object into the smallest group available. Recalculate the group size.

3.1 Implementation

The GGA was implemented using the **G**enetic **AL**gorithm **O**ptimized for **P**ortability and **P**arallelism **S**ystem (GALOPPS) Release 2.50 [Goodman 95], Michigan State University's Genetic Algorithm Research and Applications Group, http://isl.msu.edu/GA. Although this GA toolkit supports coarse-grain parallel subpopulations shown to produce superior solutions [Lin et al., 94], we have not yet applied them to this problem. Of course, fine-grain parallelism [Punch et al., 93] could be used to speed the computation on parallel or distributed hardware.

4.0 Objective Function for the Bin Balancing Problem

It is not obvious what would be a good objective function for the Bin Balancing Problem. The objective function, which is being maximized, should increase with decreasing bin size (size of the largest group). The bin size alone is not enough information to evaluate the entire fitness of the chromosome. In this section, several objective functions are tested and evaluated.

Objective Function 1

Objective Function 1 is similar to the objective function Falkenauer used for the BPP. The size of the largest group replaces the bin capacity. Chromosomes containing well filled bins and empty bins will be more fit than chromosomes containing evenly filled bins even if the bin sizes of the two chromosomes are the same. It was not initially clear that this is desirable.

$$f_{BBP} = \frac{\sum_{i=1}^{M} (S_i / S_L)^2}{M}$$

M: number of bins (given)
S_L: Size of the largest bin.
S_i: Size of the *i*th bin.

Objective Function 2

Since we want to minimize the size of the largest group, this objective function is a simple, obvious choice to test.

$$f_{BBP} = \frac{1}{S_L}$$

S_L: is the size of the largest bin.

Objective Function 3

This objective function tries to make all the groups equal sized. The objective function should encourage largest group size to become close to the smallest group size. This should minimize the largest group size.

$$f_{BBP} = \frac{1}{\sum_{i=1}^{M/2} (S_i - S_{M-i})}$$

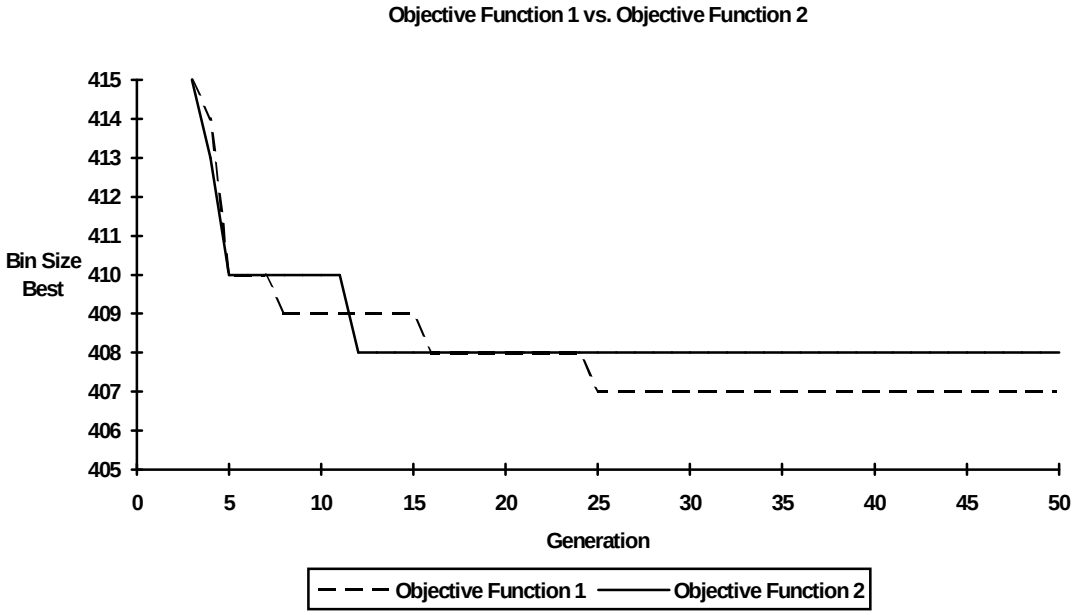
Groups are sorted in descending order.
M is the number of groups.

4.1 Performance Comparisons of the Objective Functions.

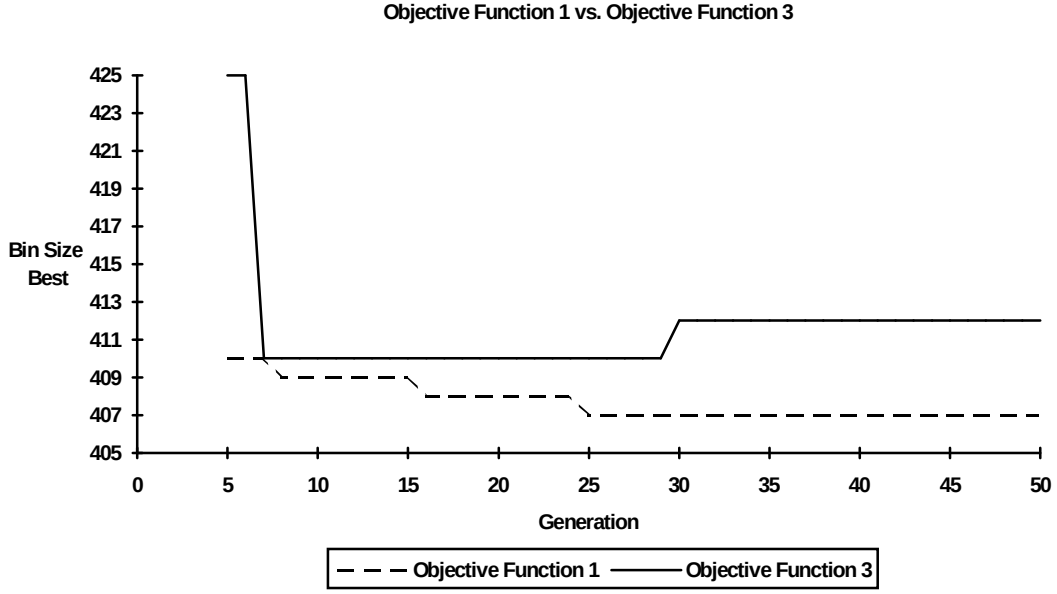
The objective functions were compared with the same parameters on a sample problem set. Objective function values from different objective functions cannot be directly compared. Since the objective functions are trying to minimize the bin size (the size of the largest group), the bin size of the best individual is plotted for each generation. Elitism was used to ensure that the chromosome with the highest fitness survives to the next generation. The runs were performed with the following parameters:

Objects	Groups	Pop Size	Pcrossover	Pmutation[1]	Linear scaling[2]
2000	500	100	0.5	0.002	1.1

[1] Pmutation is the probability that a given chromosome will be mutated.
[2] Linear scaling = (Scaled fitness of Best/Average fitness)



The above graph shows the relative performance of Objective Function 1 and Objective Function 2. Although it appears that the two objective functions perform similarly, a much higher quality of solution is required to make a slightly smaller bin size. Objective Function 1 performed significantly better than Objective Function 2.



The above graph shows the relative performance of Objective Function 1 and Objective Function 3. Objective Function 3 tried to make most of the middle-sized bins of the same size at the expense of having a larger bin size. Note the graph of the Objective Function 3 run actually increases at one point. This means than an individual had a higher fitness despite having a larger bin size (elitism was used). This is not a desirable characteristic for the objective function.

Since Objective Function 3 performed the best, it will be used for the remainder of the tests.

4.2 Results

Martello and Toth [Martello and Toth 92] tested their Bin Packing Algorithm on several domains and found the following to be the most difficult [Falkenauer 94]. They have the following characteristics that make them difficult for the Bin Balancing Problem as well: Almost no leeway, small object to bin ratio, and no very small objects. In order to test the effectiveness of the algorithm, the GGA was given the smallest possible number of groups to see how close the GGA can get to the optimal bin size.

Uniform: Objects sizes are integers chosen uniformly random from the range [20,100] and bin capacity C=150. Martello and Toth tested their procedure using various ranges and bin capacities, and found this one most difficult.

The runs were performed with the following parameters:

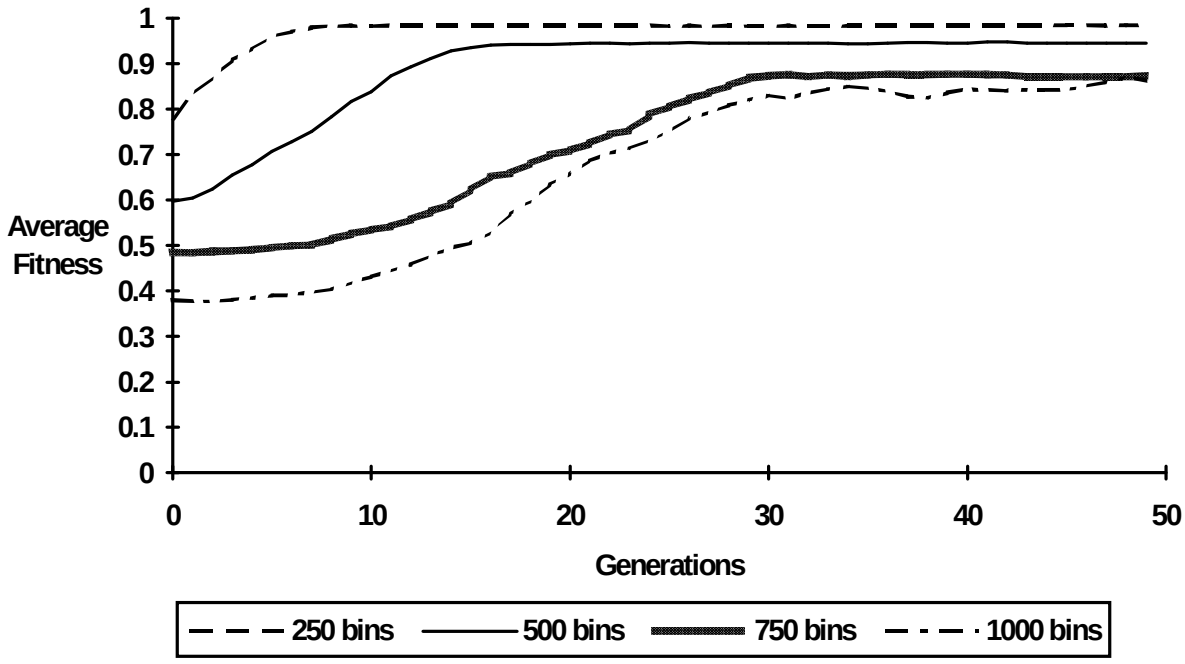
Pop Size	Pcrossover	Pmutation	Linear Scaling	Generations
100	0.5	0.002	1.2	300

Objects	Groups	Min Possible Bin Size	Leeway of Min Bin Size	Min Bin Size Achieved	Leeway Obtained
120	46	150	0.44%	156	4.45%
1000	399	150	0.14%	162	8.16%

5.0 The Effect of Larger Schemata

The GGA uses groups as schemata. The GGA's behavior changes relative to number of groups. With fewer groups, the average number of objects per group increases. Runs were performed with the same number of objects and the same parameters but with different number of bins.

Objects	Pop size	Pcross	Pmut	Obj Func	Scalemult
2000	100	0.5	0.002	1	1.1



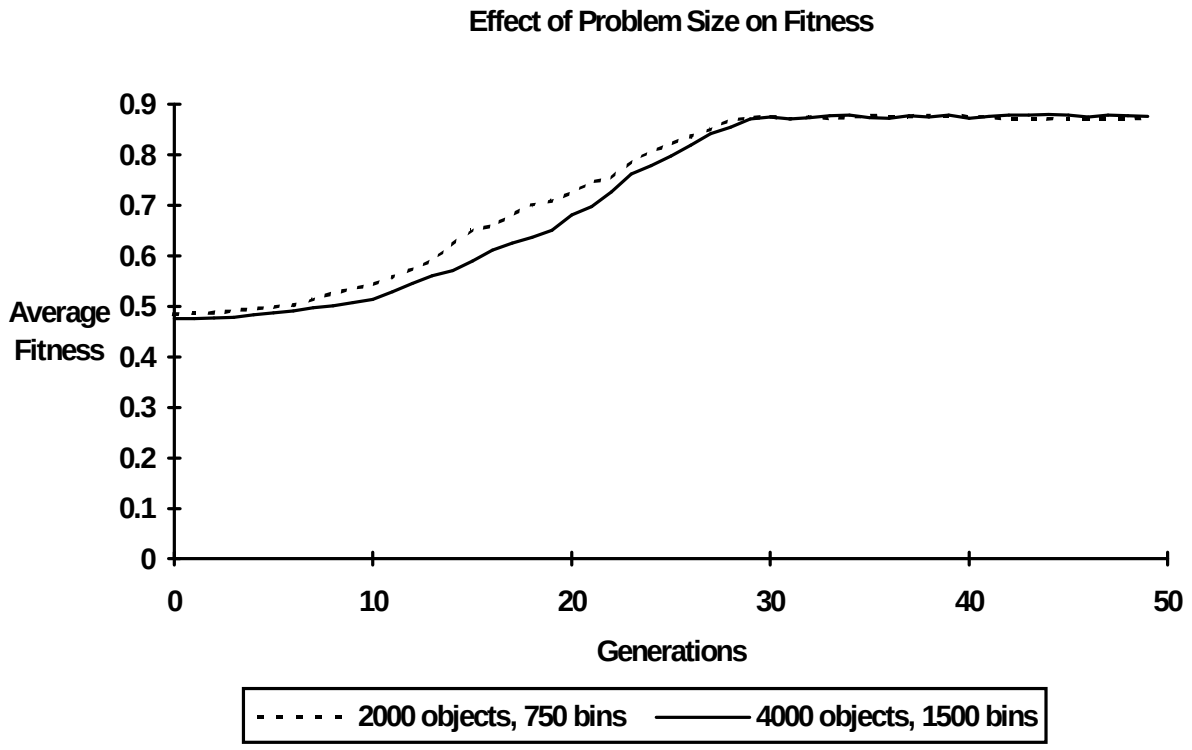
The initial fitness of the population becomes higher as the number of bins decreases. This can be seen from the higher fitness of the first generation on the graph. Since the GGA uses groups as building blocks, it seems that the good building blocks might become more difficult to find and preserve as they become larger. If this were true, runs with fewer bins would take longer to converge. The graph shows that the opposite is true. From the graph, the runs with fewer bins (more objects per bin) converge in fewer generations. The values to which the runs converge also increase as the number of bins decreases. This is expected since the fitness of solutions goes to 1 as the number of bins approaches 1. At the extreme, solutions always have fitness equal to 1 if the number of bins is equal to one. (When the number of bins is equal to 1, every object must be in the same group. Every solution is equivalent and has fitness = 1. The bin size must equal the sum of the object sizes. When the number of bins is greater than or equal to the number of objects, the LF initializes each chromosome to have each object in a separate bin. This arrangement is preserved under crossover and mutation. The size of the bin must be equal to the size of the largest object.)

The 1000 bin run is interesting since there is little progress made in the first seven or eight generations. This is due to the fact that one bad (large) group destroys the fitness of the entire chromosome. At approximately generation 10, the fitness starts climbing quickly and steadily. The several dips are caused by mutation. The fitness of individuals after mutation is often very low. This has a noticeable impact on the average fitness of the population.

6.0 Effect of the Problem Size

We explore the effect of larger problem sizes on the GGA by comparing two runs. For the second run, the number of objects and the number of bins were double that of the first run. The ratio of objects to bins is constant for the two runs.

Pop size	Pcross	Pmut	Linear Scaling
100	0.5	0.002	1.1

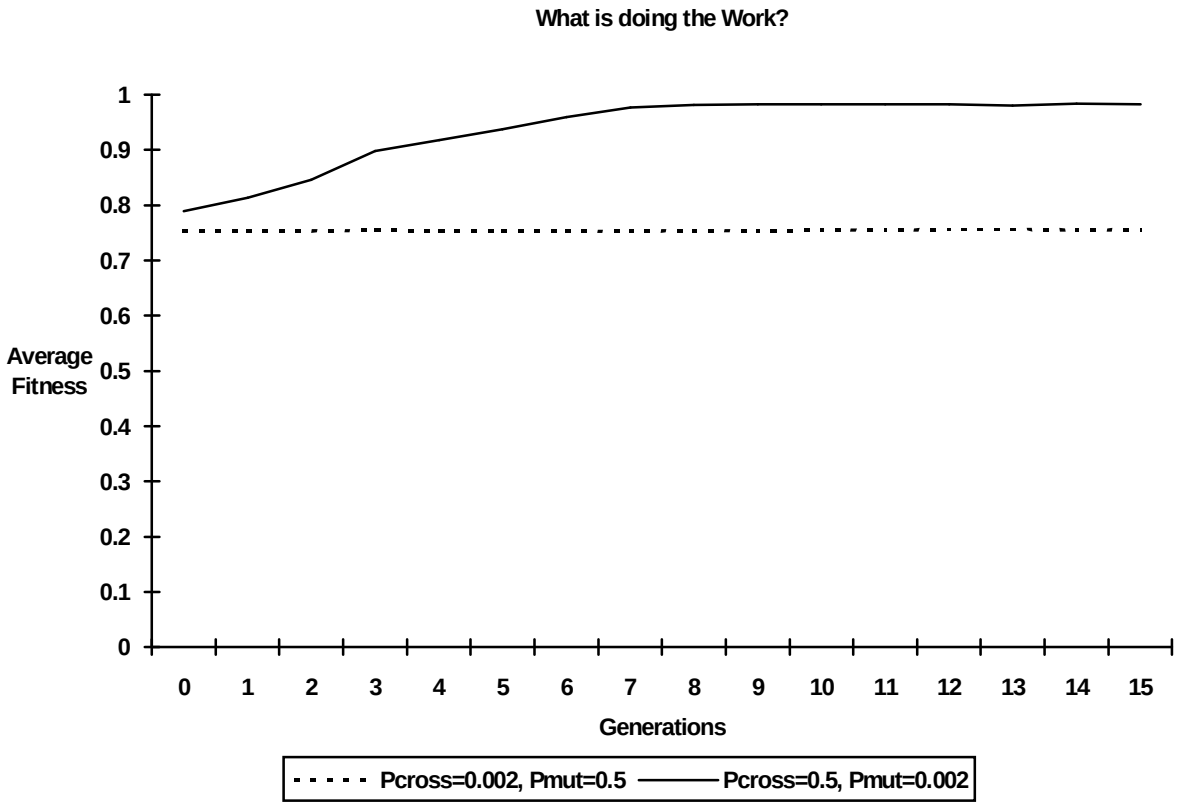


The run with more objects has a slightly lower average fitness during the first twenty-five generations. This is due to the fact that one poor (large) group can significantly lower the whole chromosome's fitness. As the number of groups in the chromosome increases, the probability that the chromosome contains a poor group increases. The two runs essentially converge to the same fitness in the same number of generations. The run with 4000 objects took considerably more CPU time.

7.0 What Is Doing the Work?

Crossover combines groups from each of two parents. "Loose" objects are then reinserted using Loosest Fit Descending Heuristic. The GGA works with groups as schemata. Are the results obtained due to preservation of schemata during crossover or from applying the Loosest Fit Descending many times? This can be determined by comparing the results of the GGA with a high crossover rate and a low mutation rate to the results of the GGA run with a low crossover rate and a high mutation rate. Mutation randomly picks several bins and eliminates them. Objects are replaced using the Loosest Fit Descending. Thus, using mutation alone is equivalent to applying the Loosest Fit Descending Heuristic many times.

Objects	Pop size	Bins	Linear Scaling
2000	100	250	1.06



The graph clearly shows that the use of the Loosest Fit Descending heuristic alone cannot produce good results. In the Mutation-only run, the average fitness of the population remained almost constant over many generations. The crossover run converged in a few generations at much higher average fitness. This shows that the Grouping Genetic Algorithm processes useful schemata relevant to grouping type domains.

8.0 Conclusion

We have successfully adapted Falkenauer's GGA to a new grouping-type domain, the Bin Balancing Problem. The GGA proved to be well-suited for this grouping-type problem and produced strong results. Since the Bin Balancing Problem has not been as well studied as the Bin Packing Problem, we lack an accepted heuristic method against which to compare the GGA, except for (for example) LFD, which it beats. We have proved through experiment that the GGA obtains results through the processing of schemata and not only by successively utilizing its heuristics. Ongoing work has already found a superior fitness function to the one reported here, but analysis of its performance is not yet complete. Use of coarse-grain parallelism is expected to yield still better results.

References

[Ding et al., 91] Ding H., El-Keib A. A. and Smith R. E. *Optimal Clustering of Power Networks Using Genetic Algorithms* TCGA Report No. 92001, March 5, 1992, University of Alabama, Tuscaloosa, AL.

[Falkenauer 94] Falkenauer, E., *A Hybrid Grouping Genetic Algorithm for Bin Packing*, Technical Report, Research Centre for Belgian Metalworking Industry, Brussels, Belgium.

[Falkenauer 92] Falkenauer, E., *A Genetic Algorithm for Bin Packing and Line Balancing*, Proceedings of the 1992 IEEE International Conference on Robotics and Automation.

[Goodman 95] Goodman E.D., *An Introduction to GALOPPS, The Genetic ALgorithm Optimized for Portability and Parallelism System*, Technical Report #95-01-01, Genetic Algorithm Research and Applications Group, Michigan State University.

[Lin et al., 94] Lin S., Punch W.F. and Goodman E.D., *Coarse Grain Genetic Algorithms, Categorization and New Approaches*, accepted to Parallel and Distributed Processing 1994.

[Martello and Toth, 89] Martello S. and Toth P., *Lower Bounds and Reduction Procedures for the Bin Packing Problem*,Discrete Applied Mathematics 28 (1990) pg 59-70.

[Punch et al., 93] Punch W.F., Goodman E.D., Pei M, Chia-Shun L., Hovland P. and Enbody R., *Further Research on Feature Selection and Classification Using Genetic Algorithms,* 5th International Conference on Genetic Algorithms, Urbana-Champaign, July 1993, pg 557.