Enhancing the GA's Ability to Cope with Dynamic Environments

Mark Wineberg
Intelligent Systems Research Unit
School of Computer Science
Carleton University
Ottawa, Canada, K1S 5B6
wineberg@scs.carleton.ca
(613) 520-2600 x 1857

Franz Oppacher
Intelligent Systems Research Unit
School of Computer Science
Carleton University
Ottawa, Canada, K1S 5B6
oppacher@scs.carleton.ca
(613) 520-2600 x 3520

Abstract:

The Shifting Balance Genetic Algorithm (SBGA) is a pluggable module for a GA (or any other Evolutionary Algorithm) based on a modification of Sewall Wright's shifting balance theory. The SBGA is intended to enhance a GA's ability to adapt to a changing environment. Here we describe the detailed mechanisms required to implement the SBGA as well as an experiment that shows that the SBGA not only outperforms the GA in a difficult dynamic environment, but actually seems to thrive in such an environment.

1 INTRODUCTION

In (Oppacher, and Wineberg, 1999) we developed the "shifting balance" addition to evolutionary algorithms called the SBGA (Shifting Balance Genetic Algorithm). This is a pluggable module that can be added to any evolutionary algorithm (GA, ES or EP). In this paper, however, we will only concentrate on the GA.

Added to the main population of the GA are a series of smaller, helping populations called *colonies*. The main population, called the *core*, is responsible for exploring the area of the search space that seems most promising, i.e. performing exploitation, while the colonies explore the areas of the search space where the core population doesn't yet have a presence. For the colonies to maintain their distance from the core, they must have information as to how far they are away from the core. This requires a mathematically precise definition of distance from a chromosome to a population, which we have obtained from cluster analysis.

In this paper we will describe the detailed mechanisms by which the colonies are forced away from the core, and by which the migrants entering the core are integrated and exploited. Finally, we will present some new results that show that the SBGA not only outperforms the classical GA in dynamic environments, but that it actually thrives in such scenarios.

2 MOVING THE COLONY ELSEWHERE

2.1 INTRODUCTION

There are two main tasks to accomplish when keeping the colony away from the core: detecting when the colony is too close to the core, and actually moving the colony away when so detected. The detection mechanism will be dealt with in §2.3 and §2.4, while the procedure for moving the colony, if found too close to the core, will be developed below.

2.2 KEEPING COLONY AWAY FROM CORE

For the SBGA to perform as desired, a mechanism is needed that allows a population to relocate to other places in the search space, thus escaping from a local maximum. Previously, to escape a local optimum, most GA implementations invoked a blind restart. A less haphazard approach could be developed if we had a method of determining whether an individual from the colony is searching in the same area as the core. We could use this distance measure as a fitness function for the members in the colony when the colony becomes too close to the core. The members will then be selected for reproduction, not only according to their objective function value but also according to their 'distance' from the core. They will *evolve* into a new area of the search space.

Fortunately the distance from a member of a population to the entire population is a well-known concept in cluster analysis. In GA terminology such a distance is simply the average Hamming distance between the given member and all other members of the population. This concept is easily extended to encompass the distance from a single member of one population to an entire second population.

Since we intend the colony to follow the objective function's landscape, even as it moves away from the core, we are faced with a bi-objective optimization task. We handle this bi-objective optimization as follows:

The population for the next generation is split into two sections. The first section is selected under the objective function, and the second under the distance to the core. When filling a sub-population, the parents are selected from the previous generation's entire population. However, the fitness function used during the selection is the one associated with the sub-population being filled.

Notice that under this approach, when selection occurs, the two fitness criteria are kept separate thus avoiding the problem of averaging out into mediocrity. Furthermore, this approach avoids the possibly counterproductive crossover of a member selected under one criterion with a member selected under a wholly different criterion. Instead, crossover is only performed on chromosomes that have been selected under the same criteria. Like mates with like.

2.3 THE PERCENTAGE OVERLAP

2.3.1 Keeping the Proper Distance from the Core While Still Following the Fitness Landscape

In order to implement the above bi-objective approach to moving the colony away from the core, we need to determine the portion (κ) of the population governed by the objective function[1]. How should this parameter be set?

If we were to keep κ constant, say $\kappa = \frac{1}{2}$, a constant pressure would eventually drive the colony away from the core as far as possible. To state it mildly, this is not the behavior that one wants the system to have!

It is now obvious that a measure is needed to regulate the value of κ for the colony, keeping it small to zero when far from the core while increasing it when it is close. In other words, we want to measure the overlap between two populations. Since all we have to work with are distances, our definition of population overlap is perforce distance-based.

2.3.2 Finding the Percentage Population Overlap

Let *Core* be a core population of size n and *Colony* be a colony population of size m. Now, let x_i be the number of core members that are closer to the i^{th} member of the colony than they are to the core. This can be more formally stated as follows: let X_i be the multi-set corresponding to the colony member $d_i \in Colony$ such that

$$X_i = \{c \mid c \in Core, \text{dist}(c, d_i) \le \text{dist}(c, Core)\}$$

then we can define $x_i = |X_i|$.

Now the percentage of the core that are closer to the i^{th} member than they are to each other is $\frac{x_i}{n}$ and so the average, $\frac{1}{m}\sum_{i=1}^{m}\frac{x_i}{n}$, is the total fraction of the core that the colony is close to. This is what κ is supposed to measure! So, after factoring out the $\frac{1}{n}$, we obtain a formula for the split ratio:

$$\kappa_o = \frac{1}{m\ n}\sum_{i=1}^{m} x_i .$$

κ_o is called the percentage overlap.

The percentage overlap departs somewhat from our intuitive notions of the properties that an "overlap" should have. For example if we compare two identical populations we would think that they should completely overlap, i.e. that κ should be 1. However, in most cases we would find κ to be much smaller than 1. A percentage unary overlap would only occur if all members of the colony were at the center, i.e. all had zero distance to the core population. Any spread in the colony population will lower the percentage overlap. This means that κ, thus defined, is conservative in the sense that the resulting system follows the fitness landscape more readily than it pushes away from the core.

Unfortunately there is a major problem with the definition of κ as it stands. Since the distance from each colony member to each core member must be computed and treated separately (there is a comparison for each core member / colony member pair) instead of averaged, there will be $m\ n$ such comparisons. Each comparison takes $O(l)$ time, where l is the length of the chromosome. Consequently the algorithm to compute κ will have an $O(l\ m\ n)$ time complexity. Since this must be done for each colony, if there are k colonies the time complexity becomes $O(k\ l\ m\ n)$. Now the total population size is $N = k\ m + n$. Normal settings for k, m and n have $n \approx k\ m$ since the core group should be a lot larger than the colonies to do proper exploitation, yet the colonies must be large enough not to suffer excessively from random drift. Therefore $N = k\ m + n \approx 2n$, which means that $n \approx k\ m \approx \frac{N}{2}$. Consequently $k\ m\ n \approx \frac{N}{2}\ \frac{N}{2} \approx N^2$, and so the time complexity can now be seen to be $O(l\ N^2)$.

Since the time complexity of the GA is only $O(l\ N)$, if κ is calculated by the means of percentage overlap, the SBGA will have caused a slowdown in the performance of the GA. Another approach will have to be found.

[1] Obviously, (1 - k) would be the proportion of the size of the sub-population selected by the distance.

2.4 THE PERCENTAGE SIMILARITY

2.4.1 Finding the Percentage Similarity

Since the percentage population overlap is so computationally expensive, we will look for other methods to accomplish the task. We do not expect to find another method with exactly the same properties (otherwise it would probably have the same time complexity). However, as long as the new measure is conceptually similar to the percentage overlap, prevents the colony from being in the same area as the core, and has a time complexity no greater than that of the GA, it will be preferable.

Since the n comparisons between each colony member and all the core members is responsible for the large time complexity, let us weaken the previous method and just calculate the distance of a colony member to the core group as a whole. This distance can be computed for every colony member in $O(k\ l\ m)$ time. (The distance between a chromosome and a population, defined as the average Hamming distance from the chromosome to the members of the population, can be re-expressed as $Dist(chr,P) = \frac{1}{l}\sum_{k=1}^{l}(1 - f_{P|k}(g_k))$. Here g_k is the gene from chromosome *chr* at locus *k*, and $f_{P|k}(g)$ is the frequency of a gene at locus *k* across population *P*. Since the same gene frequencies can be used for any chromosome, it only needs to be computed once. Consequently, the above equation can be computed in $O(l)$ time for any chromosome in the colonies and hence in $O(k\ l\ m)$ time for every member in the colonies). Since we usually keep the total number of members in the colonies equal to the core population size, $O(k\ l\ m) \approx O(l\ n)$ thus keeping the total time complexity of the GA unchanged as required.

Now instead of comparing a core member's distance to the core with its distance to a colony member we compare its distance to the core with the colony member's distance to the core. If distance to the core of a core member is greater than the colony member's distance to the core it is deemed to be 'outside' the colony member. Conversely if its distance is less it is considered to be 'inside'. Since distance to a population is just the converse of similarity, a chromosome's having a smaller distance to a population is equivalent to its being more similar to that population. Consequently core members that are 'outside' of a colony member are less similar to the core than is that colony member.

As with the percentage overlap, we wish to know what fraction of the core population is to be considered too similar to the colony member. But instead of judging similarity by the number of core members that are too close to the colony member, we look at the number of core members with a greater distance to the core than the colony member's distance to the core as evidence of the colony member's similarity to the core.

We can now define r_i, which is the analog to x_i, as the number of core members whose distance to the core is greater than the colony member's distance to the core. By the same token, r_i can be thought of as the number of core members that are less similar to the core than the given colony member.

We will now present a formal definition of r_i. Let R_i be the multi-set corresponding to the colony member $d_i \quad Colony$ such that

$R_i = \{c \mid c \quad Core, \text{dist}(d_i, Core) \le \text{dist}(c, Core)\}$.

Then we can define $r_i = |R_i|$.

Now $\frac{r_i}{n}$ is the percentage of the core that is less similar to the core itself than is the i[th] member of the colony. We shall call this the *percentage similarity to the i[th] member*.

We can now average all the percentage similarities of all members together to produce the average percentage similarity of the colony, which is $\frac{1}{m}\sum_{i=1}^{m}\frac{r_i}{n}$. This average represents the total fraction of the core that, when compared with itself, is found to be less similar than the comparison of the colony to itself. This is the *percentage similarity*[2].

We can now use the percentage similarity as the new split ratio:

$$\kappa_s = \frac{1}{m\ n}\sum_{i=1}^{n} r_i .$$

2.4.2 An Algorithm for Computing the Population Similarity

To prepare for the computation of the percentage similarity, the distance to the core must first be computed for all members in all colonies as well as for all members in the core. Each member is then labeled as to whether it belongs to the core or a colony and then all of them are combined into a single population.

Now all of the r_i values for all of the colony members can be found. The combined population is first sorted by distance to the core and then traversed in reverse sort order. Each core member encountered during the traversal is tallied into a running count. This count therefore contains the number of core members that are further from the center of the core than those members of the colonies not yet seen, which is what the r_i values are. So, when a colony member is encountered the current running count is stored as its r_i value.

---

[2] In our previous paper (Oppacher and Wineberg, 1999) percentage similarity was called *containment*.

To handle ties, two different sort orders have to be created, producing two different r_i values for each member: one where ties are included and one where they are excluded. To include ties, the tied core members must be placed higher in the sort order than the colony member that they are tied with. Then when the array is traversed in descending order, the tied core members are encountered before the colony member and so are included in the colony member's r_i value. To exclude ties the opposite sort order is used. Now the colony member is encountered before the core members that it tied with. Therefore those core members will not be included in the colony members r_i value. Both inclusive and exclusive r_i values must be stored for each colony member.

Once the r_i values are found the percentage similarity for each colony can be calculated in short order. All members from a colony are canvassed to find their r_i values. These are then summed together into two numbers, R_{incl} and R_{excl}, which are averaged together to find R the true similarity sum for the colony. Finally, to arrive at the percentage similarity, this sum is first divided by m, the colony size, and then by n, the core size, as per the definition of κ_s.

2.4.3 The Time Complexity of the Percentage Similarity Algorithm

In the preparation stage, the distance to the core of all members in the system is computed in $O(l\ N)$, see 2.4.1, where $N = k\ m + n$, and all populations are combined into a single population in $O(N)$. Thus the total time complexity for the preparation stage is $O(l\ N)$.

When computing the r_i values, the complete population is first sorted, which takes $O(N \log N)$, and then iterated through to compute and store the r_i values, which takes linear time. So the time complexity of this stage is $O(N \log N)$.

Finally the r_i values are summed for each population, which is again a linear time computation, and the percentage similarity is computed, which is constant for each colony. So in total, the final stage takes linear time.

Overall, since $O(N \log N) < O(l\ n)$ (this is so because $\log N < l$ in order for the GA to be useful, otherwise enumeration could be used instead), the percentage similarity algorithm takes $O(l\ N)$ time to compute. As the GA also has a $O(l\ N)$ time complexity, computing the split ratio in this way only adds a constant overhead to the GA. This is in direct contrast with the percentage-overlap method, with its $O(l\ N^2)$ time complexity.

2.4.4 Summary

We have now finished our description of the mechanism that drives colonies to search in areas where the core isn't, while allowing the colonies to stay in promising areas of the search space.

The main component of the mechanism actively moves the colony away from the core. This is done in an evolutionary way by selecting parents for the colony's next generation based, in part, on the distance to the core.

To move away from the core while still following the fitness landscape demands a dynamic way to determine how much of the colony should be selected by distance to the core. This is done with the help of the concept of percentage similarity, which yields an algorithm with better time complexity than does the concept of percentage overlap.

3 MIGRATION FROM COLONY TO CORE

3.1 THE MIGRATION

The preceding section dealt with the problem of keeping the colonies away from the core so that they can explore novel territory. The mechanism that accomplishes this causes information flow, albeit indirectly, from the core to the colony. However, the information flow must go two ways. Not only do the colonies need to access information about the core, but the core must also receive information from the colonies. Otherwise the core would be independent from the colonies, and the colonies will not be aiding the search but only absorbing resources. The mechanism that closes this feedback loop between core and colonies is migration.

In the SBGA, the colony sends members, called *migrants*, back to the core. If a colony sends a member with great potential to the core it will after a few generations start to dominate the core and shift the location of the population to the area of gene space where the originating colony is located. The colony will then have to move into a new area because of the core avoidance mechanism, potentially into an area with a still larger local optimum, thus driving the system ever 'upwards'.

During migration the colony may send all of its members to the core or only some portion thereof. The colony members could be randomly selected, selected stochastically according to fitness, or, represent an elite subgroup. All of the above techniques have been used in the various island model parallel GAs, see (Pettey, 1987), (Tanese, 1989) and (Cohoon, 1991)). In the SBGA system as implemented, the colony members are chosen as an elite subgroup.

Since migration of the colony members disrupts the core group, time is given for the colony to evolve potentially useful members. The number of generations between the movement of colony members to the core is called the *migration interval*. To reduce the pressure on the core even more, immigration is staggered between the colonies. For example, if there are eight colonies and the

migration interval is four, two of the colonies send immigrants to the core each generation. However, for any given colony, four generations pass before colony members are allowed to migrate again.

3.2 THE INTEGRATION OF THE MIGRANTS

Just like all multiple-population based GAs, the SBGA needs a method to integrate the migrants arriving from the colony into the core's population. There are two different strategies used in the GA literature; they are briefly sketched below.

The first technique is to replace current members of the host population with the new immigrants ((Pettey, 1987) (Tanese, 1987) (Whitley, 1990)). The host members to be replaced can be selected by many means: randomly, based on fitness, or based on similarity using the Hamming distance (as done in Crowding).

The second approach is similar to the (μ, λ) technique of evolutionary strategies, but uses GA style selection and reproduction. The host population is temporarily enlarged by the migrants (analogous to μ) and then pared down again to its normal size (analogous to λ) after reproduction. In other words, selection for reproduction is performed on the extended population, host members plus immigrants, but there will only be λ offspring. This technique has been used by (Cohoon, 1991) in their GAPE system.

Since the purpose of the colonies is to add diversity to the core, a large fraction of the colony will be sent to the core, perhaps the entire population. Consequently, if a substantial number of colonies send migrants, the core will be inundated with new members. Since the replacement approach (the first technique of those presented above) incorporates the new members by replacing the old core members, all or most of the members of the core could be replaced by new immigrants. As the purpose is to add diversity to the core – not replace it!! – the (μ, λ) method of immigration absorption will be used.

The competition during migrant integration is very fierce. Since the core population is reduced to its normal size during reproduction, the selection pressure exerted on the bloated core exceeds that experienced by a normal GA population. If the migrants from the colonies have very poor fitness with respect to the core, they will not be selected during reproduction and will have no effect on the core at all. Any migrant that makes it into the core intact deserves to be there! However, since the selection is not deterministic, colony migrants do have a chance of being selected as a parent and so can influence the next core generation, if not intact, then in part.

4 THE BEHAVIOR OF THE SBGA IN DYNAMIC ENVIRONMENTS

4.1 PURPOSE

In our GECCO'99 paper, we showed that the SBGA outperforms the GA in both stationary and dynamic environments. Here we attempt to analyze its behavior in the dynamic environment in much greater detail in order to gain a deeper understanding why the SBGA performs as well as it does.

For GECCO'99 we found that the SBGA outperformed the GA in dynamic environments under the many settings of migration interval, dimensionality, and function speed that we looked at. However, in order to compute all of the possible combinations of settings, only 3 repetitions of each combination could be done. Since we learned that any setting will do, we will now run only one setting 111 times to achieve more detailed insights with statistical significance.

4.2 EXPERIMENTAL DESIGN

This experiment was done using the F2 function from the De Jong test suite composed with the one dimensional version of the Griewangk Function (F8). The combined function is called the F8F2 function, described in (Whitley, 1996), see Table 1. This minimization function is non-symmetric, linearly separable, increases in difficulty as the dimension increases, and has a known minimum at (1, 1). The solutions to F8F2 were encoded using Gray coding.

Table 1: The F8F2 fitness function

$F2 : f(x,y) = 100\left(x^2 - y\right)^2 + (1-x)^2$ $x, y \quad [-2.048, 2.047]$ Minimum when $x_1 = x_2 = 1$ (F2 = 0)
$F8 : f(x) = 1 + \frac{x^2}{4000} - \cos x \quad x \quad [-512, 511]$ Minimum when x = 0 (F8 = 0)
$F8F2(x_1,x_2,x_3,\ldots,x_n) = F8(F2(x_1,x_2)) + F8(F2(x_2,x_3)) + \ldots + F8(F2(x_{n-1},x_n)) + F8(F2(x_n, x_1))$ Minimum when $x_i = x_j = 1$ (F8F2 = 0)

To better understand how the SBGA would handle itself in a dynamic environment, we compared it to the GA running both algorithms on the F8F2 fitness functions of dimensionality 5, for 750 generations. For the first 150 generations, F8F2 was held stationary allowing both systems to converge on the global minimum. In the experiment convergence typically occurs by generation 90.

In the experiment, both the GA and SBGA were given the following (common) parameter settings: the probability of mutation = 0.006 per bit, and the probability of one-point crossover = 0.7. Linear rank selection with elitism was used. The slope was set to be as steep as possible (i.e. with Max = 2.0).

The SBGA was given 10 colonies of 100 members each. The colony size was chosen to reduce the amount of random drift, yet keep the colonies relatively small in size. The core group size was set to 1000, an amount equal to the size of all the colonies combined. During migration 25 elite members of a colony are sent to the core group. The immigration interval was set to 5 generations (so in every generation, only 2 colonies are sending members to the core).

The GA was given a population of 2000, equal to the total population of the SBGA system.

The environment undergoes a simple translation in phenotype space along the hyper-diagonal. This motion is kept at a constant speed for 600 generations, at which point the global minimum has moved from the point (1, 1) to the point (1.5, 1.5). Since the genome is 60 bits long, the environment changes, on average, approximately half a bit per generation

The best fitness value of the population is recorded every generation, as is the median of the population and the diversity[3] of the population.

5 RESULTS AND DISCUSSION

5.1 BEST OF THE GENERATION RESPONSE

The experiment has a stationary region for 150 generations, after which the environment moves along the hyperdiagonal at a constant rate. In the stationary region one expects an evolutionary system to rapidly close in on the global optimum (at the price of decreasing diversity), eventually either finding it, or approximating it. Once the environment changes, the SBGA because of its design should begin to track the global optimum sooner and with a greater accuracy than the GA, which should have experienced an extreme loss of diversity.

Figure 1 shows the fitness of the best individual for each generation for both the GA and the SBGA. Confidence bands are displayed around the mean curves, the SBGA's confidence band is in light grey, the GA's is in dark grey.

The confidence bands are calculated to take into account that the graph comprises 750 individual confidence intervals, one for each generation. In order to compensate for the effect of the multiplication axiom of probability theory which would reduce our confidence level from 95% for any individual confidence interval to practically zero for the entire band, we use Bonferroni Joint Confidence Invervals (Neter et. al., 1996, pp. 152-155).

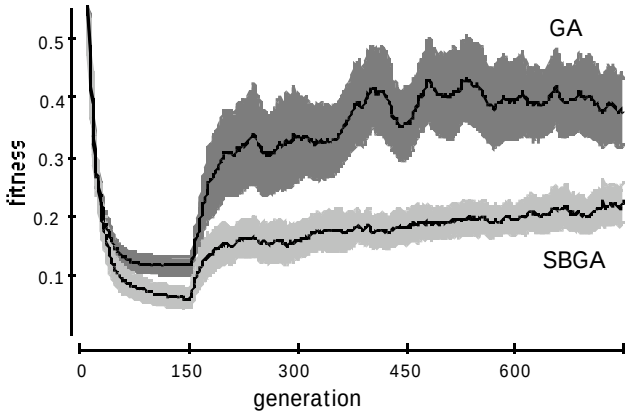

Figure 1: The Fitness of the Best Member Per Generation

Looking at the graph, during the first 150 generations, both the GA and SBGA converge rapidly to very low fitness values (remember F8F2 is a minimization problem) as expected. About generation 60, the two curves begin to diverge; the SBGA begins to outperform the GA. In fact the GA flatlines by generation 75 while the SBGA continues to improve even until the environment starts to move at generation 150.

Again, by looking at the graph after the environment has started to move, one can discern two stages. In the first stage, both systems drastically drop in fitness as a result of having converged. In the second stage, the systems recover somewhat, although they are still losing ground. However the slope of the GA is far steeper than that of the SBGA (until it plateaued) and the GA cannot track the global as closely. The fitness difference between the GA and the SBGA is far greater now than during the stationary stage.

5.2 MEDIAN RESPONSE

Figure 2 shows the median fitness value of the population for every generation (we have only recorded the median of the core population for the SBGA). Unlike figure 1 the 111 median fitness values at a given generation are not averaged, but rather the median is taken because we found that the results were not normally distributed; the distribution curves showed a large one sided tail which heavily skews the results of an average.

As a result of the use of the median instead of the average, one cannot use the standard confidence interval around the mean. Instead, we used the Thompson-Savur distribution-free confidence interval based on the sign test (Hollander and Wolfe, 1973, pp. 48-49). Again because we deal with 750 such confidence intervals, we must apply the Bonferonni correction to keep the confidence level up at 95%.

[3] The diversity of a population as used here is the average genetic entropy across the population, averaged across all loci, see (Wineberg and Oppacher, 1996)).

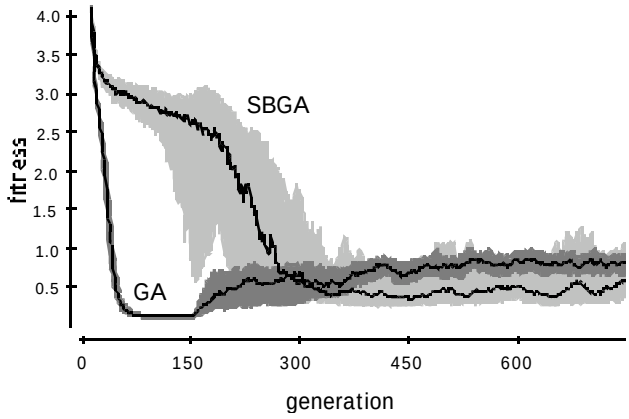

Figure 2: The Median Fitness of the Core Population Per Generation

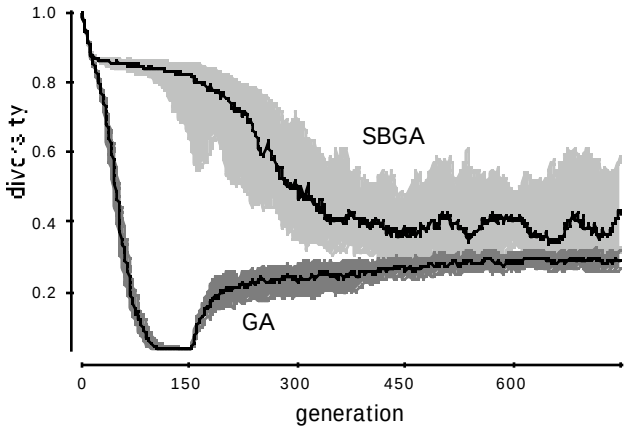

Figure 3: The Diversity of the Core Population Per Generation

Figure 2 shows that the median response of the GA population mirrors the curve for the best fitness found in figure 1. However, this is not the case for the SBGA: during the stationary phase, the fitness remains surprisingly high even though the best of the SBGA is lower than that of the GA. Furthermore, once the environment starts to move, the median fitness actually drops dramatically, even below that of the GA. This counterintuitive behavior (the SBGA seems to be doing worse on 'average' on the easier task) can be explained when one takes the effect of the shifting balance into account.

As the SBGA converges during the stationary phase, the core envelops the region in which the global minimum resides. The colonies, kept away from the core by the distancing mechanism, cannot but help to have poor members. These members are sent into the core, keeping the median very high. Once the environment moves, the core, due to its huge bulk, cannot respond quickly and so its members become very unfit. However, the small, quick moving colonies, which, by being outside the core are most likely already near the new location of the optimum, will send highly fit members back to the core. These members will begin to dominate the core, thus reducing the median.

It is this very fact that shows that the benefits of the shifting balance theory are indeed operating. It also shows that the SBGA thrives in a dynamic environment.

5.3 DIVERSITY

Figure 3 shows the diversity of the population for every generation (we have only recorded the diversity of the core population for the SBGA). Like figure 2, the 111 diversity values for a given generation have their median taken because we found that the results were not normally distributed. Consequently, the confidence bands are computed in the same manner as the median fitness results.

As expected, the GA rapidly loses almost all of its diversity. Quite surprisingly the GA does recover very quickly. After about 50 generations it is very close to the diversity level which it will maintain for the rest of the run.

Once again, the SBGA's results are fairly surprising. At first the SBGA behaves as one would predict: the diversity stays very high. However, as soon as the environment begins to move, the diversity rapidly declines (albeit at a slower rate than the various diversity changes that the GA experiences). As with the median fitness behavior, this fact is also due to the effect of the shifting balance. As fit colony members, from colonies that are well positioned in the search space, migrate to the core, they experience a selective advantage with respect to the existant core members. Consequently they start to outperform those original core members and begin to 'take over' the core. As a result, the core becomes populated with individuals of similar genetic makeup, and the diversity drops as the core shifts its position in gene space.

6 CONCLUSION

The experimental results suggests that the SBGA does indeed improve the behavior of the GA in dynamic environments. This is especially noticeable if one compares the small advantage that the SBGA has over the GA in the stationary enviroment versus the vast relative improvements obtained when the optimum started to move.

This was achieved because extra diversity in the SBGA is not generated blindly as with most extensions of the GA which attempt to increase its adaptiveness. The SBGA uses small subpopulations that are forced to explore in other areas than the main population. Members from these new regions are brought back to the main population to enhance its diversity. However the subpopulations are not blindly entering the novel territory but are doing so by following the fitness landscape.

Acknowledgments

We would like to thank Steffen Christensen for his help in interpreting the data.

References

Cohoon, J., Hegde, U., Martin, W., and Richards, D. (1987). Selection in Massively Parallel Genetic Algorithms. In J.J. Grefenstette (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 148-154. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Hollander, M. and Wolfe, D. A. (1973). Nonparametric Statistical Methods. New York: John Wiley & Sons.

Neter, J., Kutner, M. H., Nachtsheim, C. J., Wasserman, W. (1996). Applied Linear Statistical Models, Fourth Edition. Chicago: McGraw-Hill.

Oppacher, F., and Wineberg, M. (1999). The Shifting Balance Genetic Algorithm: Improving the GA in a Dynamic Environment. In W. Banzhaf et. al. (Eds.) *The Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO-99), pp. 504-510. San Francisco: Morgan Kaufmann.

Pettey, Leuze, & Grefenstette, (1987). A Parallel Genetic Algorithm. In J.J. Grefenstette (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 148-154. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Tanese, R. (1989). Distributed Genetic Algorithms. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 434-439. San Mateo, California: Morgan Kaufmann.

Whitley, D., and Starkweather, T. (1990). Genitor II: A Distributed Genetic Algorithm. In the *Journal of Experimental and Theoretical Artificial Intelligence*, 2, 189-214.

D. Whitley, K. Mathias, S. Rana and J. Dzubera (1996). Evaluating Evolutionary Algorithms. In *Artificial Intelligence* Volume 85, pp. 245-276, 1996.

M. Wineberg and F. Oppacher (1996). The Benefits of Computing with Introns. In J. R. Koza, et. al. (Eds.) *Genetic Programming 1996: Proceedings of the First Annual Conference*, pp. 410-415. Cambridge Massachusetts: MIT Press.