Replacement Strategies in Steady State Genetic Algorithms: Dynamic Environments

J.E Smith and F. Vavak
Intelligent Computer System Centre
University of the West of England
Bristol England BS16 1QY
jim@ics.uwe.ac.uk

16th Feb. 1999

Abstract

Recent years have seen increasing numbers of applications of Evolutionary Algorithms to non-stationary environments such as on-line process control. Studies have indicated that Genetic Algorithms using "Steady State" models demonstrate a greater ability to track moving optima than those using "Generational" models, however implementing the former requires an additional choice of which members of the current population should be replaced by new offspring.

In this paper a number of selection and replacement strategies are compared for use in Steady State Genetic Algorithms working as function optimisers in dynamic environments. In addition to an algorithm with fixed mutation rates, the strategies are also compared in algorithms employing Cobb's Hypermutation method for tracking environmental changes. On-line and off-line metrics are used for comparison, which correspond to different types of real-world applications.

In both cases it is shown that algorithms employing some kind of elitism outperform those that do not, which is related to previous studies on stationary environments. An investigation is made of various methods

of implementing elitism, including an implicit method, "conservative" selection. It is shown that the latter, in addition to being computationally simpler, produces significantly better results on the problems used, and reasons are given for this behaviour.

Keywords: Selection, Replacement, Dynamic Environments, Genetic Algorithms

1 Introduction

The majority of the Genetic Algorithm (GA) work to date has been in problem domains in which the fitness landscape is time-invariant. However, it is not unusual for a real-world system to exist in an environment that changes over the course of time. The optimisation algorithm then has to be designed so that it can compensate for the changing environment by monitoring its performance and altering, accordingly, some aspects of the optimisation mechanism to achieve optimal or near-optimal performance. An objective of the resulting adaptation is not to find a single optimum, but rather to select a sequence of values over time that minimise or maximise the time-average of environmental evaluations. In this sense the optimisation algorithm "tracks" the environmental optimum as it changes with time.

The distributed nature of the genetic search provides a natural source of power for exploring in changing environments. As long as sufficient diversity remains in the population, the GA can respond to a changing search landscape by reallocating future trials. However, the tendency of GAs to converge rapidly results in the GA population becoming homogenous, which reduces the ability of the GA to identify regions of the search space that might become more attractive as the environment changes. It such cases it is necessary to complement the

standard GA with a mechanism for maintaining a healthy exploration of the search space.

The results in this paper concentrate on the case where the rate of change of the environment is sufficiently slow for the algorithms to converge between changes. Algorithms are compared by running them for a fixed period and calculating two time-averaged metrics, which correspond to different types of real-world applications.

The first of these is the "on-line" measure [DeJong 1975], and is simply the average of all calls to the evaluation function during the run of the algorithm. This measure relates to applications where it is desirable to maintain consistently good solutions e.g. on-line process control.

The second metric considered is the "off-line" performance, and is the time-averaged value of the best performing member of the current population. This was resampled after a number of calls to the evaluation function equal to the size of the population. In practice each member of the population was re-evaluated in order to measure this, but the fitnesses were not used to overwrite those stored in the individuals, since this would affect some of the strategies tested.

Unlike the on-line metric, off-line performance is unaffected by the occasional generation of individuals with very low fitness, and so is more suitable for problems where the testing of such individuals is not penalised e.g. parameter optimisation using a changing design model.

1.1 Previous Work

There are two basic modification strategies for the GA which enable it to evolve continually an optimal solution to a problem while the environment changes with time.

The first strategy expands the memory of the GA in order to build up a repertoire of ready responses for various environmental conditions. The main examples of this approach are the GA with Diploid Representation [Goldberg & Smith 1987] and the Structured GA [Dasgupta & McGregor 1992]. Goldberg and Smith examine the use of diploid representation and dominance operators to improve performance of the genetic algorithm in an oscillating environment, while Dasgupta and McGregor present a modified genetic algorithm with a multi-layered structure of the chromosome which constitutes a "long term distributed memory".

The second modification strategy effectively increases diversity in the population directly (i.e. without extending the GA memory) in order to compensate for changes encountered in the environment. Examples of this strategy involve the GA with the Hypermutation Operator [Cobb 1990] [Cobb & Grefenstette 1993], the Random Immigrants GA [Grefenstette 1992], the GA with Variable Local Search (VLS) Operator [Vavak et. al. 1996, 1997] and the Thermodynamic GA [Mori et. al. 1996]. The Hypermutation operator temporarily increases the mutation rate to a high value, called the Hypermutation rate, during periods when the time-averaged best performance of the GA worsens. The Random Immigrants mechanism replaces a fraction of a standard GA's population by randomly generated individuals in each generation in order to maintain a continuous level of exploration of the search space. The VLS operator uses a similar triggering mechanism to Hypermutation, and it enables local search around the location of the population members in the search space before the environmental change. The range of the search is variable and can be gradually extended to match degree of the environmental change. The thermo-

dynamic GA can maintain a given level of diversity in population by evaluating the entropy and free energy of the GA's population. The free energy function is effectively used to control selection pressure during the process of creating a new population.

Two different models of the GA can be used for an optimisation task. The Generational Genetic Algorithm (GGA) creates new offspring from the members of the current population, using the genetic operators (recombination and mutation) and places these individuals in a new population. When the same number of individuals have been created as there are members of the current population, the new population replaces the current, and the cycle begins again. [Goldberg 1989], [De Jong 1992]. The Steady State Genetic Algorithm (SSGA) [Whitley & Kauth 1988] differs from the generational model in that there is typically one single new member inserted into the new population at any one time. A replacement/deletion strategy then defines which member of the population will be replaced by the new offspring (e.g. the worst, oldest or random individual).

In [Vavak & Fogarty 1996a, 1996b] the suitability of GGAs and SSGAs was studied for use in dynamic environments. Results showed that the SSGA with a "delete-oldest" replacement strategy can adapt to environmental changes with reduced degradation of off-line and particularly on-line performance. The better performance of the SSGA can be explained by the fact that an offspring is immediately used as a part of the mating pool, making a shift towards the optimal solution possible in a relatively early phase of the optimization process. Selection of the steady state model for use in nonstationary environments is therefore advantageous, particularly for on-line applications.

1.2 Replacement Schemes in Steady State GA's

Selection is a vital force in any evolutionary algorithm, and an understanding of the nature of its effects is necessary if effective algorithms are to be developed. For GGAs selection has been well studied, and methods developed which reduce much of the noise inherent in the stochastic algorithm e.g. SUS [Baker 1987]. Unfortunately the very nature of SSGAs precludes the use of such methods and those available are inherently more noisy.

In [Smith & Vavak 1998] several selection strategies were studied using a Markov Chain analysis of the takeover probability vs. time, in order to understand the different sources of the noise.

For delete-oldest and delete-random strategies, the variations in performance arise in part from losing the only copy of the current best in the population, which happened approximately 50% of the time for delete random, and 10% of the time for delete-oldest. Performance comparisons on static landscapes demonstrated that the extent to which this affects the quality of the solutions obtained depends on the ability of the reproductive operators to rediscover the lost points.In [Chakraborty 1995] other strategies e.g deletion by exponential ranking were also shown to lose the optimum.

A common way of avoiding this problem is to incorporate elitism - often in the form of a delete-worst strategy. As has been documented by other authors, e.g. [Chakraborty 1995, Chakraborty et. al. 1996], the latter was shown to exhibit increased selection pressure, leading to premature convergence and poor performance on higher dimensional problems.

Two other ways of achieving elitism are also considered in this paper. The first is the common method of explicitly checking whether the member about to

be deleted is [one of] the current best in the population, and if so not replacing it. In this case the member can either be preserved with its original fitness value, or be re-evaluated and the new fitness value saved. The second, "conservative selection" is an implicit mechanism introduced in [Vavak 1997]. Here each parent is selected by a binary tournament between a randomly selected member of the population, and the member about to be replaced. If the latter is the current best, then it will win both tournaments, so recombination will have no effect, and (apart from the effects of mutation), elitism is attained. In [Smith & Vavak 1998] this was shown to guarantee takeover by the optimal class, but at a much slower rate than delete-worst or elitist delete-oldest.

In total ten selection strategies will be evaluated in this paper. Deletion of the oldest, worst and random members is done in conjunction with both standard (both members randomly selected) and conservative (member to be replaced plus one random member) tournaments. Additionally a delete oldest policy is tested with four variants of elitism. These are:

1. Oldest is not replaced if it is one of the current best, but is re-evaluated.

2. Oldest is not replaced if it is the sole copy of the current best, and is re-evaluated.

3. As 1. but without re-evaluation (original fitness value kept).

4. As 2. but without re-evaluation (original fitness value kept).

2 The First Test Problem: Cobb's Landscape

The two dimensional landscape used to compare the selection methods which are being discussed in this paper is formed by 14 sinusoidally shaped hills [the 2D landscape was kindly provided by Dr. Helen G. Cobb - Navy Center for

Applied Research in AI, Naval Research Laboratory, Washington]. Each of two dimensions is represented by 16 bits (i.e. total search space of 256 points) and ranges from -32.767 to 32.768 (Figure 1). The maximum fitness has a constant value 60 and the next highest peak has a value of 30.

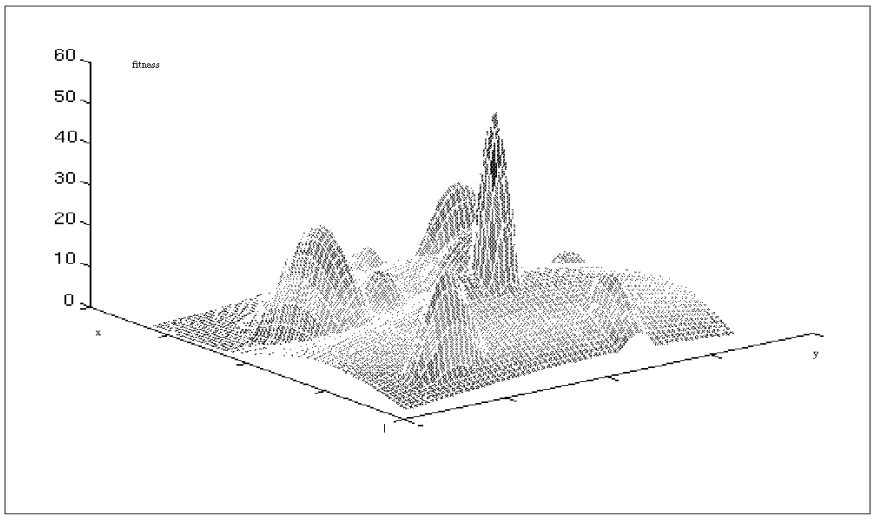

Figure 1: The 2-D Fitness Landscape

Environmental changes are simulated by moving the maximum peak in a random direction from its initial position. The starting position of the maximum peak is also generated randomly. To obtain the required degree of environmental change, one of the coordinates (selected at random) of the maximum peak initial position is shifted in positive or negative direction (selected at random) by a required value of the environmental change "EC". A random angle 0-45 degrees is then generated and the other coordinate is evaluated using the tangent function, the direction of change being obtained by selecting one of the four quadrants at random.

For the purposes of this investigation the algorithms were started with ran-

domly initialised populations, and allowed to run for 20,000 evaluations, with a single environmental change occurring halfway through this period. For this paper, the various selection methods and replacement strategies were tested across the range of possible magnitudes of environment change, EC = 1 to 20. For each degree of environmental change, 10 random starting positions of the maximum peak were generated and the GA was then initiated 10 times with different random generator seeds for each starting position, so the results presented are the mean of 100 runs.This was done for algorithms with and without Hypermutation.

2.1 Experimental Details for Cobb's function

The Steady State GA which was used for the tests implemented uniform crossover and tournament selection. The tournament size is 2 and the better chromosome always wins the tournament. After initial experimentation to determine suitable values, the following parameter settings were used: population size 120, bit mutation probability 0.001, crossover probability 1.0. The Gray coded chromosomes are 32 bits long (16 bits per dimension) and each chromosome represents two integer numbers 0-65535. To transform these integer values into rational variables, i.e. search space coordinates ranging from –32.767 to 32.768, the following mapping is used: $rational\_variable = (32.768 - integer\_variable/1000)$.

The Hypermutation operator was implemented as it is currently the most commonly used method for tracking. It is triggered if the running average of the best performing members of the population over an equivalent of three generations of the generational GA (i.e. over a number of evaluations equal to three times the population size) drops by an amount which exceeds a predefined threshold. In this case a value of threshold TH = 3 was used. The best

performing member of the population is evaluated after an equivalent of one generation of the generational GA. Once it has been triggered, the Hypermutation rate (0.2) is switched back to the "baseline" mutation rate (0.001) as soon as the best performing member of the population reaches 80% of its value before the environmental change occurred. The setting of the parameters (80% and Hypermutation rate 0.2) was found to provide good results for the given problem. A prolonged period of high mutation for values higher than 80% has a negative effect on-line performance due to diversity being introduced into the population despite the correct region of the search space having already been identified. Similarly to the choice of the threshold level described previously, the values of both parameters were selected empirically.

2.2 Results with Fixed mutation Rates

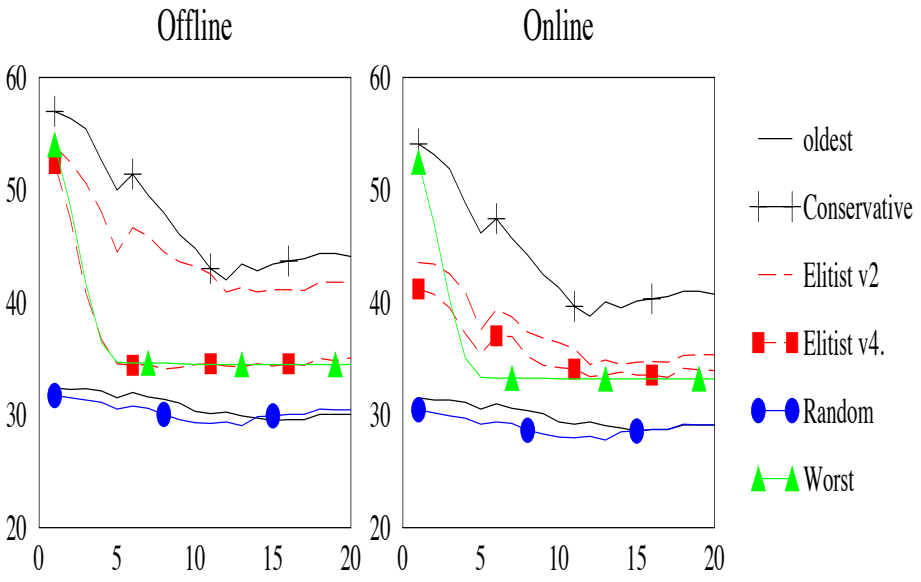

Figure 2: Performance with Fixed Mutation Rates

The mean off-line and on-line performance vs. size of Environmental Change (EC) are shown in Figure 2. Some curves are not shown for reasons of clar-

ity where they are not statistically significantly different from another (using Student's t-test at the 5% level). These are:

Conservative Selection - delete-random performs worse, but not significantly so than delete oldest for both metrics, so the latter only is shown.

Delete-Worst - using the conservative policy makes no significant difference and is not shown.

The elitist policies - whether re-evaluating or not, it makes no difference whether the algorithm checks for duplicates before keeping an individual which has fitness equal to the best. The versions where only the sole copy is preserved are shown.

As can be seen the performance drops for both metrics as the size of the environmental change is increased, but the extent to which this occurs is highly dependant on the replacement strategy. For all but delete-oldest or random (which show poor performance throughout the range) there is a steady decline in performance as EC increases. For the more successful strategies there is a dip in performance at EC =5, which is an artifact of the coding. Analysis of the results using Analysis of Variance (ANOVA) confirmed the statistical significance of the patterns visible from the graphs, namely:

1. Replacement strategies which can lose the current best value (delete oldest and random) perform poorly according to both metrics. This result on non-stationary landscapes fits in with the theoretical results in [Smith & Vavak, 1998]. The rediscovery of "lost" points of high fitness is liable to be harder on non-stationary problems since with a population converged around the previous optimum, mutation is more likely to be the cause of discovery of new good individuals than crossover. Even for small environmental changes, discovering

points close to the new optimum may require several simultaneous mutations, which will only happen with very low probability, hence the importance of not losing them demonstrated by these results

2. Replacement policies which keep the current best individuals but do not re-evaluate them (delete worst, the 3rd and 4th variants of elitism) perform well for EC = 1, but the performance drops off almost linearly as the size of the change is increased, up to the radius of the optimum (EC 5). Above this the performance curves are almost flat, at a value just above the fitness of the second highest peak. This shows that the algorithms are able to track the problem occasionally but not reliably.

The reason for this is that once the optimum has moved, several members of the population will have artificially high fitness values attached to them, and so will be selected to be parents a disproportionate number of times, without being replaced. Recombination involving these individuals will tend to create offspring which lie within the area of the original peak position. The probability that these offspring will lie on the slopes of the new optimum depends on the degree of overlap between the two regions, which is very low for $EC > 5$ and zero for $EC = 10$. If there is significant overlap the population can move to the new position by a process of simple hill climbing. However if there is little or no overlap, ($EC > 5$) the new position of the optimum can only be discovered by mutation, hence the constant low rates of discovery and correspondingly poor performance.

3. The strategies which combine (at least implicit) elitism with re-evaluation of the fitness of individuals are far more successful than the others at tracking the optimum. The conservative strategy (with deletion of either the oldest or

a random member of the population) performs significantly better than the elitism with re-evaluation according to both metrics, especially for the on-line performance.

Detailed examination reveals this is partly because in the first search phase, the elitist algorithms with standard tournaments converged to sub-optima more frequently than the conservative strategies. In [Smith & Vavak 1998] it was shown that the conservative policy exhibits less selection pressure and slower takeover times than a standard tournament. This translates to a lesser tendency for premature convergence, as is demonstrated by the improved off-line results.

There is an additional benefit arising from the reduced selection pressure. The longer retention of diversity gives more scope for recombination to act as a search operator once the optimum has moved. By comparison the tightly converged populations resulting from the standard tournament have to wait for mutation to create diversity, hence the exaggerated differences in on-line results.

The performance falls fairly smoothly for EC values up 12, beyond which it remains roughly constant. Inspection of the off-line performance shows that the lowest values for the conservative strategy are around 45, which is what would be expected if the population initially found the optimum, then after the change converged onto the highest of the neighbouring hills.

2.3 Results with Hypermutation

Figure 3 shows the results using Hypermutation for off-line and on-line performance. Again the delete-oldest and delete-random strategies performed very poorly and so these results are not shown. Also not shown for reasons of clarity are the delete-worst-conservative policies and elitism with multiple copies kept, as discussed above. However the random-conservative results are shown as they

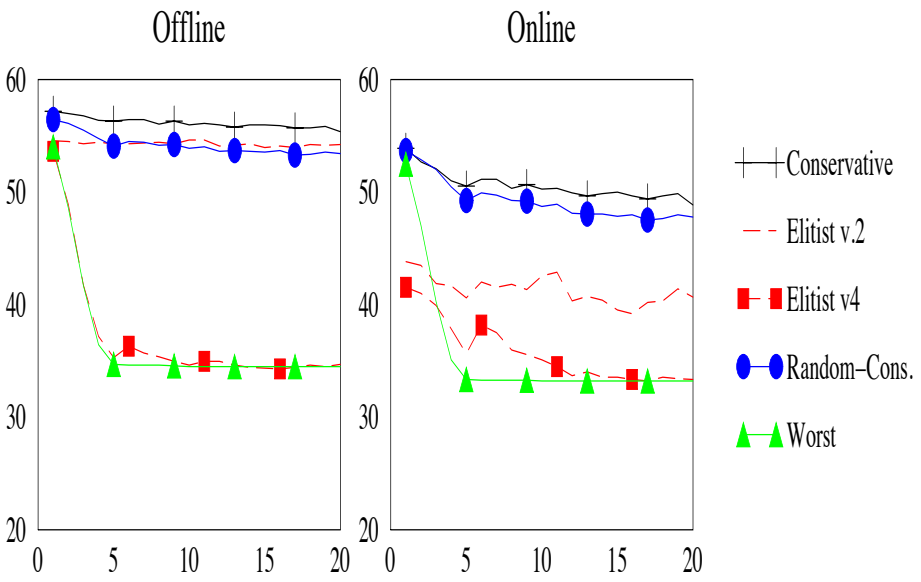

Figure 3: Performance with HyperMutation

are significantly different (worse) from oldest-conservative.

The patterns of relative performance are the same as with fixed mutation rates, for the same reasons, and most (but not all) strategies benefit from the introduction of Hyper Mutation.

The performance curves for delete-worst and elitism without re-evaluation are very similar to the previous set. This is because individuals occupying the previous peak never get re-evaluated, and so Hyper Mutation is never triggered.

For the two conservative variants, and elitism with re-evaluation, the algorithms are able to track large changes much better than with the fixed rates. This is because the once the higher rate of mutation is triggered in Hyper-Mutation, there is a greater probability of generating points at large Hamming distances from the previous optimum position. Again the conservative strategies perform better than elitism according to both metrics.

There is now a significant difference (at the 1% confidence level) between

replacing the oldest or random members. For both strategies the expected lifetime of a single individual is exactly *popsize* evaluations. However whilst this is exact for delete-oldest, for random deletion there is a significant probability of an individual remaining in the population for much longer. With the random-conservative strategy, a member has two opportunities to be selected to be a parent for every one to be replaced, and if it has an artificially high fitness, it is likely to win tournaments. Thus out of date information about the fitness of the previous optimal position will be propagated for longer with random replacement than with systematic replacement of the oldest. Not only does this mean that offspring may be created around the previous optimum position for longer, but there was also a longer delay observed before Hyper Mutation was triggered with random deletion.

3 Time Varying Knapsack Problem

This problem is a variant of that described in [Mori et. al. 1996]. A number of items each have a value (c_i) and a weight (a_i) associated with them, and the problem is to select a subset that maximises the sum of the elements' values while meeting a total weight constraint. Given a fixed number of elements N, each subset is represented by a binary string of length N, where a 1 in a position i, indicates that the corresponding item is included in the subset. For such a string x the fitness is given by:

$$F(x(t)) = \begin{cases} \sum_{i=1}^{N} c_i(t)x_i(t) & \sum_{i=1}^{N} a_i(t)x_i(t) \leq b(t) \\ \left(b_{sum} - \left(\sum_{i=1}^{N} a_i(t)x_i(t) - b(t)\right)\right) \times 0.01 & \sum_{i=1}^{N} a_i(t)x_i(t) > b(t) \end{cases} \quad (1)$$

where the term b_{sum} in the penalty clause for solutions which transgress the

constraint is the sum of all the items' weight.

3.1 Experimental Details- Alternating Knapsack Problem

In the particular case investigated here, a 32 bit problem was considered ($N = 32$), the values a_i, and weights c_i attached to the items remained constant, and were generated using a uniform random distribution over the interval [0,100.0]. The weight constraint b alternated between 50%, 30% and 80% of b_{sum}, changing once every 20000 evaluations.

After considerable experimentation to establish sets of parameter values, the experiments were run with a population of 100 using binary deterministic tournament selection, and uniform crossover at a rate of 100%. Fixed mutation rates of 0.005, 0.001, 0.01 and 0.05 were used. Each algorithm was tested for 200 runs of 60,000 evaluations each.

3.2 Results - Alternating Knapsack Problem

The on-line and off-line results obtained are presented in Tables 1 and 2 . Using the values for each run performed, an Analysis of Variance test was done, which confirmed that there is a statistically significant difference (at the 0.1% level) between the results for the different strategies. For each mutation rate, the method of Least Significant Difference was used to perform post-hoc comparisons between strategies to see if there was a significant difference between results . The subgroups identified by these tests are indicated in the tables below by stars, e.g. in the first line of Table 1 the subgroups are, in ascending order of fitness: {worst},{random*},{oldest**},{elite v.2, elite v4***},{conservative****}.

For all but the lowest mutation rate, the on-line performance of the different strategies is in the order worst < random/elite v4 < oldest < elite v.2 <

Mutation Rate	Replacement Strategy					
	Oldest	Random	Cons.	Worst	Elite v.2	Elite v.4
0.001	1010.19 **	982.27 *	1065.93 ****	631.81	1018.50 ***	1018.29 ***
0.005	1054.89 **	1036.57 *	1070.30 ****	620.25	1060.46 ***	1040.36 *
0.01	1011.17 **	989.74 *	1014.26 **	599.79	1011.62 **	989.74 *
0.05	719.66 **	715.79 *	728.74 ****	487.87	726.38 ***	709.21 *

Table 1: Online Performance: Alternating Knapsack Problem

Mutation Rate	Replacement Strategy					
	Oldest	Random	Cons.	Worst	Elite v.2	Elite v.4
0.001	1029.20 **	1012.73 *	1039.05 *****	639.68	1039.48 ***	1062.34 ****
0.005	1128.44 **	1118.55 *	1153.05 *****	648.15	1136.61 ***	1144.24 ****
0.01	1152.72 **	1143.07 *	1162.15 ***	649.85	1156.33 **	1164.40 ***
0.05	1072.96 *	1070.88 *	1082.26 **	651.38	1104.07 ***	1118.54 ****

Table 2: Offline Performance: Alternating Knapsack Problem

conservative.

The poor performance of the delete-worst strategy results from its being totally unable to track changes of environment in this problem. Because the populations converge around the first optimum, new individuals created after the environmental change appear to be less fit, and are quickly deleted. Thus unless an individual is created sufficiently close to the new optimum to be retained, the population is doomed to remain in its previous location (see discussion of Figure 4 later).

As noted above, both delete-oldest and delete-random policies suffer from losing the current best in the population, and this is more noticeable for delete-random.

Again the elitist policy with re-evaluation outperforms the one without, as

solutions around the previous optimum are not kept in the population.

Table 1 shows the mean on-line values for 200 runs, the standard deviations for a rate of 0.005 were (in ascending order): worst (15.34), elite v2 (26.86), conservative (29.06), random (29.72), oldest (30.59), elite v4. (32.14). The very low variance for delete-worst confirms that this strategy gets trapped in one position. As can be seen the strategies which combine some form of elitism with re-evaluation (elite v.2 and conservative) display the best mean values and smaller variances than those strategies which can lose the current best value (random and oldest).

The elitist strategy without re-evaluation showed poor mean performance and the highest variance. Such a strategy will preserve in the population at least one individual from the pre-change peak position, with an artificially high fitness. When such an individual is selected for crossover with a parent from the vicinity of the new optimum, the resultant offspring are likely to lie between the two positions.The fitness of such offspring is hard to predict, and as seen can display a high variance, depending as it does on such things as the random choice of bits to cross-over, as well as the underlying shape of the fitness landscape.

When the off-line performance is considered,(Table 2) the patterns of mean performance is similar, except that the Elite v.4 policy now shows the second best performance. The patterns for the variance in performance are, however, exactly the same as for on-line. Since the values quoted here are the "true" off-line figures, rather than those based on the remembered fitness of individuals, these results demonstrate that the elitist strategy without re-evaluation is able to track the optimum with at least some of the population, but the high variance suggests that the preservation of individuals with artificially high fitness makes

this unpredictable.

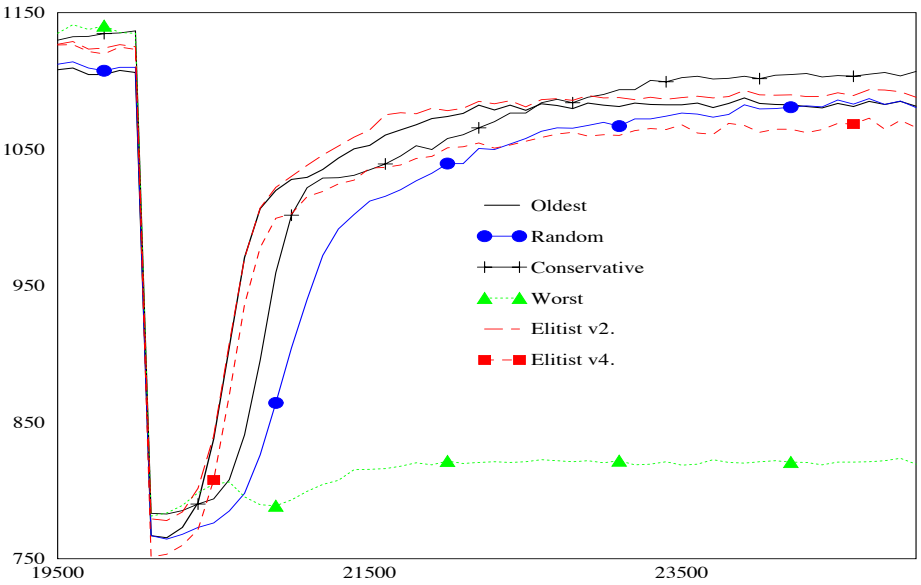

Figure 4: Transition Behaviour: Alternating Knapsack Problem

Figure 4 plots the running mean of the last 100 evaluations vs. time around the first environmental change, averaged over 200 runs for a mutation rate of 0.005. As can be seen this demonstrates some of the points made above:

The delete-worst strategy reaches the highest mean values initially, but is completely unable to track the change of the optima. Viewed over a longer time scale, the running mean performance deteriorates for the first two changes then remains constant at a very low level.

The two elitist strategies behave almost identically before the change, but the version without re-evaluation consistently produces less fit individuals after the change, as a result of the population being "dragged back" to the previous peak position.

Delete-random strategies react slower than delete-oldest, but reach similar fitness values. The values reached are less than both elitist versions before the

change, and than the re-evaluating elitist strategy after, as a result of the optima being lost and not rediscovered in some runs.

The conservative strategy exhibits slower takeover of the new optimum than delete-oldest, but reaches higher values, both before and after the change.

4 Conclusions

The ability to locate and track a moving optimum in a dynamic environment with a genetic algorithm requires two features, namely the ability to maintain (or create) diversity within the population, and the ability to exploit individuals of high fitness once discovered. As documented here, much work has been done on preserving or creating diversity, and lately it has been shown that the steady state model is more suited to dynamic environments than the generational one. In this paper attention was focused on the role of the selection and replacement policies. Several replacement strategies were tested in the context of a steady state algorithm using binary tournament selection

The results obtained clearly confirmed that for some algorithms an extra method for creating diversity (in this case Hypermutation) can improve tracking performance, although not all of the strategies tested were able to take advantage of this. However two factors are immediately apparent from these results which hold with or without Hypermutation.

Exploitation - strategies such as delete-oldest or delete-random, which can lose the sole copy of the current population best, performed poorly. This matched results on static landscapes noted above. Analysis of the takeover times for a single newly created best individual showed that these policies will actually lose the individual 50% of the time. Therefore some form of elitism is

desirable.

Re-evaluation - in potentially dynamic environments it is essential that the fitness of points on the landscape is continuously and systematically re-evaluated. Failure to do so leads to the population forever being "dragged back" to the original peak position. Although this was obvious for the 3rd and 4th variants of elitism tested, it also applies to the much more common delete-worst policy, since if the population had converged close to the optimum prior to the change, the "worst" members which get deleted may be the only ones with a true fitness value attached. The importance of systematic re-evaluation was clear from the difference between conservative-delete-oldest and conservative-delete-random. The former always produced better performance than the latter, very significantly so when Hyper mutation was present.

Of all the policies tested here, the conservative-delete-oldest is the best suited to the points noted above, and produced the best performance. The improvement over the elitist policy with re-evaluation is believed to result not merely from the reduced selection pressure, but from the fact that the exploitation of good individuals is not limited to preserving the very best, but will also apply (with decreasing probability) to the second best member and so on. Since the implicit elitism still allows changes via mutation, there is a higher probability of local search around individuals of high fitness, whilst worse members are less likely to win tournaments and so they are replaced with offspring created by recombination. The result is that even without Hypermutation the algorithm was able to track environmental changes of modest size.

5 References

J.E. BAKER Reducing Bias and Inefficiency in the Selection Algorithm. In *Proceedings of the 2nd International Conference on Genetic Algorithms* , (J.J. GREFENSTETTE ed.) (1987), pp14-21. Lawrence Elbraum Associates.

U.K. CHAKRABORTY An analysis of selection in generational and steady state genetic algorithms. In *Proceedings of the National Conference on Molecular Electronics* (1995), Nerist A.P. (India)

U.K. CHAKRABORTY, K. DEB, M. CHAKRABORTY Analysis of Selection Algorithms: A Markov Chain Approach. *Evolutionary Computation*, 4(2)(1996):133-167.

H. COBB An Investigation into the Use of Hypermutation as an Adaptive Operator in a Genetic Algorithm Having Continuous, Time-Dependent Nonstationary Environments. Naval Research Laboratory Memorandum Report 6760, 1990.

H. COBB, J. GREFENSTETTE Genetic Algorithms for Tracking Changing Environments. In *Proceedings of the 5th International Conference on Genetic Algorithms* (S. FORREST ed.) (1993), pp523-530. Morgan Kaufmann.

D. DASGUPTA, D. MCGREGOR sGA: A Structured Genetic Algorithm. Technical Report IKBS-8-92, University of Strathclyde, 1992.

K. A. DE JONG An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD Thesis, University of Michigan, 1975.

K.A. DE JONG Are Genetic Algorithms Function Optimizers? In *Parallel Problem Solving From Nature II* (R. MAENNER, B. MANDERICK eds.),(1992) pp. 3-13. Elsevier Science.

D.E. GOLDBERG, R.E. SMITH Nonstationary Function Optimisation Us-

ing Genetic Algorithm with Dominance and Diploidy. In *Proceedings of the 2nd International Conference on Genetic Algorithms*, (J. GREFENSTETTE ed),(1987) pp 59-68. Lawrence Elbraum Associates.

D.E. GOLDBERG *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company Inc., 1989

J.J. GREFENSTETTE Genetic Algorithms for Changing Environments. In *Parallel Problem Solving From Nature II* (R. MAENNER and B. MANDERICK eds.), (1992) pp. 137-144, Elsevier Science.

N. MORI, H. KITA, Y. NISHIKAWA Adaptation to a Changing Environment by Means of the Thermodynamical Genetic Algorithm. In *Parallel Problem Solving From Nature IV* (H. VOIGT, W. EBELING, I. RECHENBERG, H-P. SCHWEFEL eds.)(1996) pp513-522. Springer Verlag.

J.E. SMITH, F. VAVAK Replacement Strategies in Steady State Genetic Algorithms: Static Environments. In *FOGA V: Proceedings of the Foundation of Genetic Algorithms Workshop*(BANZHAF, REEVES eds), 1998. Morgan Kauffman

F. VAVAK. Genetic Algorithm Based Self-Adaptive Techniques for Direct Load Balancing in Nonstationary Environments. PhD thesis, University of the West of England, 1998.

F. VAVAK, T.C. FOGARTY. Comparison of Steady State and Generational Genetic Algorithms for Use in Nonstationary Environments. In *Proceedings of the 3rd IEEE International Conference on Evolutionary Computation*, (1996) pp 192-195. IEEE Publishing,.

F. VAVAK, T.C. FOGARTY A Comparative Study of Steady State and Generational Genetic Algorithms for Use in Nonstationary Environments. In

Evolutionary Computing, (T.C.FOGARTY ed.) (1996), pp. 297-304. Springer Verlag

F. VAVAK, T.C FOGARTY, K. JUKES A Genetic Algorithm with Variable Range of Local Search for Tracking Changing Environments. In*Parallel Problem Solving From Nature IV* (H. VOIGT, W. EBELING, I. RECHENBERG, H-P. SCHWEFEL eds.), (1996) pp. 376-385. Springer Verlag.

F. VAVAK, K.JUKES, T.C. FOGARTY Adaptive Combustion Balancing in Multiple Burner Boiler Using a Genetic Algorithm with Variable Range of Local Search. In *Proceedings of the 7th International Conference on Genetic Algorithms*, (T. BAECK ed.),(1997), pp. 719-726., Morgan Kaufmann.

D. WHITLEY, J. KAUTH GENITOR: A different Genetic Algorithm. Presented at *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, (1988) pp 118-130.