Non-stationary Subtasks Can Improve Diversity in Stationary Tasks

Christopher Willis-Ford and Terence Soule

University of Idaho, Department of Computer Science, Moscow, ID 83844, USA
`cwillisf@acm.org`, `tsoule@cs.uidaho.edu`

Abstract. Low diversity in a genetic algorithm (GA) can cause the search to become stagnant upon reaching a local optimum. To some extent, non-stationary tasks avoid this problem, which would be a desirable feature of GA for stationary tasks as well. With this in mind, we show that several methods of introducing artificial non-stationary elements help to promote diversity in a GA while working on an inherently stationary task. By analyzing online and offline diversity and fitness measures, we show that it is possible to improve overall performance through this technique, and that some measures intuitively related to performance are misleading.

1 Introduction

Considerable research has been done in the field of non-stationary tasks in genetic algorithms (GA). Search techniques such as random restart, adaptive mutation [1], and the island model [2] can be quite effective in helping an evolutionary system track a dynamic problem. One effect of this tracking is that the GA is not allowed to converge on a particular local optimum – a common pitfall when solving stationary problems using GA. Relatively little exploration has been done to investigate whether this benefit can be applied to stationary tasks as well.

One problem with the GA approach is the tendency for the search to become ‘stuck’ on local optima. If one individual within the population happens to find a very good solution relative to the other individuals, it is likely to have considerable influence over future generations. This effect compounds over several generations and frequently results in a near total loss of diversity in the population: premature convergence. Such low diversity results in redundancy: two individuals representing the same solution store less information than two representing different solutions. This in turn means that fewer solutions are explored, and the search is less complete.

In this paper we propose that lessons can be learned from non-stationary tasks and applied to stationary ones. Specifically, we show that diversity is improved when a stationary problem is made non-stationary, and that the benefit of doing so outweighs the cost.

2 Terminology

One may expect that GA will perform better when diversity is high, and when convergence is avoided. In order to test this intuition, it must be made precise. This section will define the terms that will be used for the remainder of this discussion. When necessary, we present a concrete definition for a term with respect to both a binary-valued and a real-valued problem.

The binary-valued problem we will address is target matching: the goal is to evolve a solution identical to a predetermined target. The fitness function for an individual $\boldsymbol{a}$ is equal to the Hamming distance between that individual and the target $\boldsymbol{t}$, and is to be minimized.

Our real-valued problem is a function optimization, which is similar to target matching. For ease of reference, we will discuss a two-dimensional problem, in which the fitness function $f(\boldsymbol{a}) = F(a_x, a_y)$ is to be maximized. This leads naturally to discussion of the fitness landscape as a three-dimensional space, in which peaks represent good solutions, and valleys represent bad ones.

Performance of the GA will be measured in terms of its offline performance: the fitness value of the single most fit individual encountered at any point during the run. Online performance – the average fitness of individuals per generation – will be noted, but is not the primary measure of performance. In fact, an increase in diversity will almost guarantee a drop in online performance, as fewer individuals will cluster at peaks in the fitness landscape.

Diversity in a GA run is a critical measure of the behavior of the search. Unfortunately, it is also a very complex and ambiguous term. We define two types of diversity: online and offline. *Online diversity* refers to the 'spread' of individuals across the search space in a given generation, measured in, for instance, standard deviation. *Offline diversity* is that of the entire run viewed as a whole – an approximation of search space coverage.

The base diversity function for a given population $D(P_i)$ will be defined differently for each problem, but will commonly be based on standard deviation or average Hamming distance between individuals in the population.

3 Background

Many low-cost methods for increasing diversity have been researched. Perhaps the simplest method is to increase the crossover or mutation rate inherent in the GA. Another low-cost method of promoting diversity is the random restart method, in which the entire population is re-seeded with random values. This restart may happen at a fixed interval, or when diversity becomes low, etc. These low-cost methods share one downside: they cause little improvement in the performance of the search [3].

On the other hand, fitness sharing [4] is extremely effective in increasing diversity, and has a marked positive effect on performance as well. In fitness sharing, the fitness score associated with a particular location in the search space is seen as a resource; individuals at that location must share that resource. As a result, if many individuals share one particular high-fitness location, the fitness of each individual might be lower than that of a single individual situated in a region of lower fitness. The effect of fitness sharing is that individuals are forced to spread out over a larger amount of the search space, helping to avoid the loss of diversity associated with population convergence and encouraging the GA to seek better solutions. Many diversity-promoting techniques that are effective in improving fitness also suffer fitness sharing's major drawback: a prohibitive computational cost [4].

4 Improving Diversity

One might expect that diversity levels in non-stationary tasks would be higher than those in stationary tasks – if nothing else, a dynamic fitness landscape could be expected to discourage convergence on one particular location in the search space. It would seem then that transforming a stationary task into a non-stationary one would yield higher levels of diversity. A stationary problem is made non-stationary by changing the environment within the GA during the run – effectively making a static fitness landscape dynamic. One immediate benefit of this method is that the computational cost is directly controllable: in fact, all of the methods presented here add a constant-time operation to a fitness evaluation, and no more.

This type of change is best expressed in terms of an altered fitness function with an added dependence on time. In general, if the original fitness function is $f_{\boldsymbol{a}} = f(\boldsymbol{a})$, and t represents the current time in the context of the GA run, then the new fitness function will be of the form

$$f'_{\boldsymbol{a}} = f'(t, \boldsymbol{a}, f_{\boldsymbol{a}}) \quad . \tag{1}$$

This investigation focuses on two optimization problems of two real-valued variables. The first function (Problem 1) being optimized was

$$f_1(\boldsymbol{a}) = \frac{1 - (|\boldsymbol{a}_x| + |\boldsymbol{a}_y|)}{2} \cdot \frac{\cos(\boldsymbol{a}_x \cdot \pi \cdot peaks) + \cos(\boldsymbol{a}_y \cdot \pi \cdot peaks) + 2}{4} \quad , \tag{2}$$

where $peaks = 8$. The fitness landscape is depicted graphically in Fig. 1. This landscape is somewhat difficult, but primarily serves an illustrative purpose. The second function (Problem 2) was the Griewangk function, frequently used as a benchmark for genetic algorithms. The Griewangk function is scalable to any number of parameters, but becomes simpler as it scales up [5]. The two-parameter Griewangk landscape, the one used here, is shown in Fig. 2. The function was expanded and simplified to

$$f_2(\boldsymbol{a}) = 1 + \frac{\boldsymbol{a}_x^2 + \boldsymbol{a}_y^2}{4000} - \cos x \cdot \cos \frac{y}{\sqrt{2}} \quad . \tag{3}$$

Both functions were normalized to output in the range of 0 to 1, and the Griewangk function was inverted in order to make it a maximization problem.

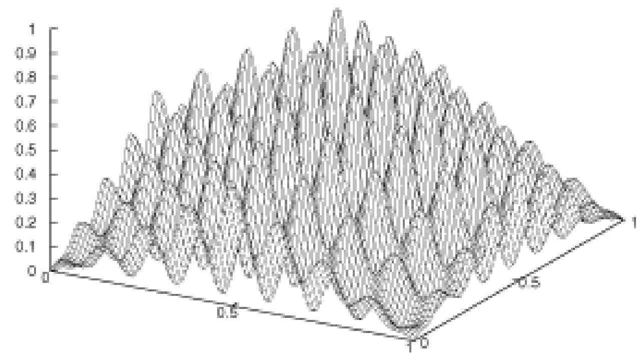

Fig. 1. Fitness landscape for Problem 1. The global optimum is at the center, coordinates (0.5, 0.5). Every depression has a point of zero fitness at its center, including those surrounding the global optimum

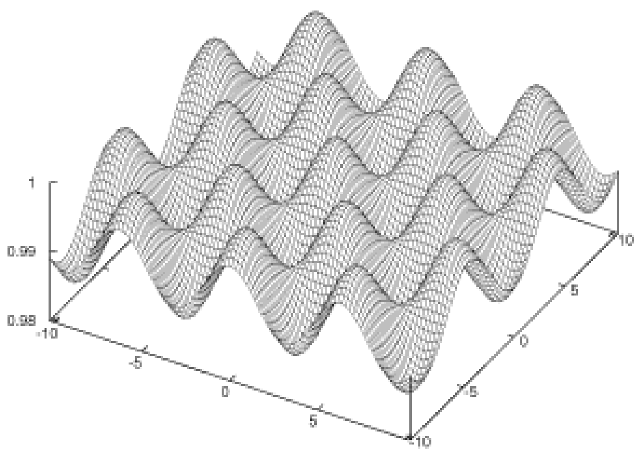

Fig. 2. Fitness landscape for Problem 2, the Griewangk function. The global optimum is again at the center, at coordinates (0, 0). This is a detail view of the center at 50x zoom. Note that pits do not reach zero fitness

The domain of each parameter in the first landscape was -1 to 1; in the second, -512 to 512. These were normalized to take inputs in the range from 0 to 1..

All GA runs were made to run for exactly 300 generations. GA parameters are depicted in Table 1, and were the same for both problems. Solutions were represented as ordered pairs of floating point numbers: $(\boldsymbol{a}_x, \boldsymbol{a}_y)$. As such, mutation of a coordinate was done by adding a random number (Gaussian distributed about 0). Crossover of two parents was implemented to yield two new coordinates that lay along the segment between the parents, 1% of the distance from either end. Mathematically, if $\boldsymbol{a}'$ was the first child, and $\boldsymbol{a}$ and $\boldsymbol{b}$ were the parents, then

$$\boldsymbol{a}' = \left(\frac{\boldsymbol{a}_x * 99 + \boldsymbol{b}_x}{100}, \frac{\boldsymbol{a}_y * 99 + \boldsymbol{b}_y}{100} \right) , \tag{4}$$

and similarly for $\boldsymbol{b}'$ with weights reversed. Elitism was based on the raw fitness function $f(\boldsymbol{a})$ rather than $f'(\boldsymbol{a})$.

Table 1. Parameters for each GA run

Population size:	500
Selection:	Tournament of 3
Chance of Crossover:	100%
Chance of Mutation:	0.1% per coordinate ($\boldsymbol{a}_x$ and $\boldsymbol{a}_y$)
Elitism:	2 individuals

4.1 Baseline: Random Noise

Perhaps the simplest method of making the landscape dynamic is to randomly modify each individual's score during fitness evaluation. This randomness ensures that individuals that are on or near a peak in the unmodified fitness landscape are not necessarily the individuals that receive the highest modified score – effectively, peaks are constantly shifting at a rate faster than once per generation. This could cause individuals near a good solution to not always be favored, and thus help maintain diversity. Of course, there is also a clear downside: in some cases, this practice could cause the GA to completely miss a good solution.

Depending on the size of the random change, relative to the unmodified fitness, the effects of the random noise approach vary wildly. In extreme cases, the search either exhibits behavior indistinguishable from the original GA if noise is low, or effectively becomes random search if noise is high. It is worth noticing that in random search, convergence is avoided and so diversity is very high. Thus, it is clear that diversity by itself does not improve GA performance, as random search performs poorly. In moderation, the 'noisy fitness' tends to have slight effects toward both improving diversity and delaying convergence, but does not have a significant effect on performance [6].

Our goal in this research is to discover whether it is possible to use non-stationary modifications to a stationary GA in such a way as to improve both diversity and fitness, with low computational cost. To this end, we explored three methods detailed in the following sections.

4.2 The Moving Target Approach

The fitness function representing the "moving target" approach (MT) is

$$f'(t, \boldsymbol{a}, f_{\boldsymbol{a}}) = w f_{\boldsymbol{a}} + (1 - w)\, dist\,(x, T(t)) \ , \tag{5}$$

where w is a weighting factor, $dist\,(x, y)$ calculates the distance between two individuals, and the function $T\,(t)$ constructs an individual that represents the target at time t. The $dist\,(x, y)$ function measures the average Hamming distance in the population for bit string problems, or the sum of the standard deviation across all values in real-valued problems.

The implementation of MT discussed here regards a secondary target with an additive effect on the raw fitness function, with linear falloff. The target begins at a random location within the search space, and moves to new random locations throughout the run. Moves occur at most every 20 generations, but only when online diversity falls below 0.1% of the search space.

The desired effect of the secondary target is to temporarily encourage the population to explore in a direction that it would not otherwise. The secondary target 'bends' the landscape around a new peak, such that population travel toward that peak earns higher fitness than travel away. For instance, if the secondary target positions itself in the middle of a low-fitness region, that region is then easier for the population to enter or cross. A potentially beneficial side effect is that if the target then leaves that region, the population should then leave th low-fitness region for higher-fitness regions of the search space – which have not necessarily already been explored.

One potential drawback of the MT approach is that the secondary target may "distract" the GA from finding the global optimum. An extreme example is that of a deceptive fitness function: if the secondary target is placed such that the population is encouraged to explore the deceptive portion of the landscape, it will then be difficult to lead exploration back toward the true optimum, even if the secondary target is moved to encourage such behavior. In fact, in order to reliably induce such action, the weight of the secondary target must dominate that of the raw fitness – and in that situation, the GA cannot be expected to find good solutions to anything but the secondary target's fitness function. Clearly this is an extreme example, but it is not difficult to see this situation arising in non-deceptive landscapes as well.

4.3 The Peak Removal and Peak Erosion Approaches

The peak removal (PR) approach attempts to guide the population away from good solutions after they have been found. The fitness function that describes this is

$$f'\,(t, \boldsymbol{a}, f_{\boldsymbol{a}}) = f_{\boldsymbol{a}} \cdot S\,(t, \boldsymbol{a}) \quad , \tag{6}$$

where the function S calculates the 'squash factor' associated with that particular location in the search space at that particular time. This method is similar in concept to Tabu search, described in [7]. Here, we focus exclusively on the idea of simplistically modifying the fitness landscape.

The squash factor is determined by keeping a list of local optima that have been found recently, and negating fitness values within that range. That is,

$$S(t,\boldsymbol{a}) = \begin{cases} -1 \text{ if } \mathit{dist}(\boldsymbol{a},\boldsymbol{p}_i) < \delta \text{ for any of the past } n \text{ local optima } \boldsymbol{p}_i \\ 1 \quad \text{otherwise.} \end{cases} \tag{7}$$

Whenever diversity falls below a certain level, the current best position in the area is recorded and added to the set of previously found peaks, $\boldsymbol{p}$. If the list is longer than n, the oldest element is discarded. From then on, any individual within a certain distance δ of that known peak is penalized, encouraging exploration of other regions of the search space. It is important to note that, though selection is based on the modified fitness, elitism is based on the raw fitness. This prevents the GA from completely losing a good solution when it is squashed.

One critical factor in using PR is choosing the right δ. If it is chosen too large, peaks may be missed; if it is too small, then some peaks (with more rounded tops) will not be eliminated. It is conceivable that some sort of adaptive method might determine a good value for δ every generation (or even for every peak), but developing such a method is beyond the scope of this discussion.

Peak Erosion (PE) is a relaxed form of PR. Rather than completely inverting a peak if it is on the peaks list, the fitness is decreased according to how many times that region of the search space is found on the list. Specifically, the squash factor for PE is defined as

$$S(t,\boldsymbol{a}) = 0.9^m \ , \tag{8}$$

where m is the number of the past n local optima $\boldsymbol{p}_i$ satisfying $\mathit{dist}(\boldsymbol{a},\boldsymbol{p}_i) < \delta$.

Peak Erosion is intended to avoid the possibility of prematurely forcing the population to abandon a particular peak. Rather than immediately removing a peak, PE causes the population to gradually become 'disinterested' in that region of the search space. This allows a few generations of further exploration, perhaps yielding a slightly better solution, before the population moves on to the next area of interest.

For this investigation, δ was chosen to be equal to $\frac{1}{16}$ of the range of each parameter for Problem 1. That is, a squashed peak implied a circular "pit" in the fitness landscape, with radius equal to $\frac{1}{16}$ of the horizontal or vertical span of the square space. For Problem 2, the squash was much smaller: only approximately 0.5% of the range of each parameter. This is because peaks in Problem 2 are proportionally smaller than those in Problem 1. The list of recent peaks was 20 peaks long. Settings for both PR and PE were the same; only the $S(t,\boldsymbol{a})$ function was changed.

5 Results

Results were collected for 100 runs of each version of the GA: normal, MT, PR, and PE. The diversity, maximum fitness, and average fitness were calculated for each generation of each run. For Problem 1, Fig. 3 shows the average diversity

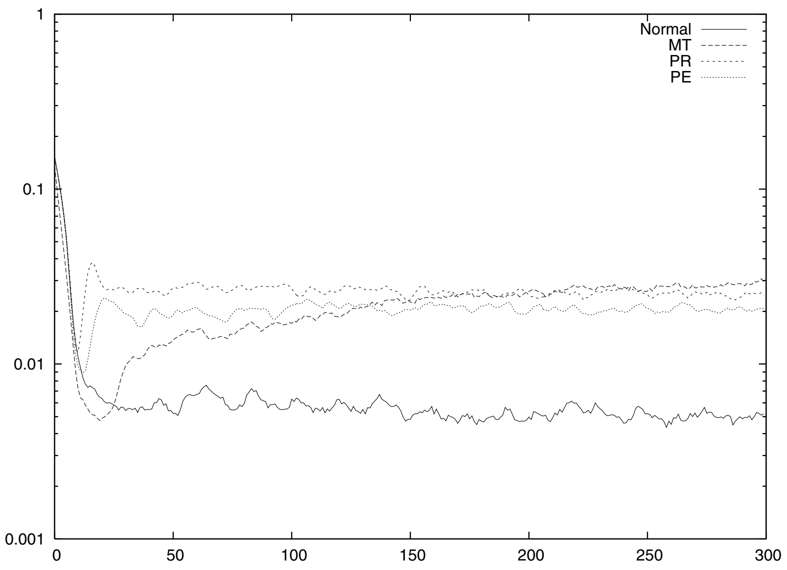

Fig. 3. Online population diversity with logarithmic scale. Diversity levels are comparable for all three modified GAs; lower diversity is seen in the unmodified GA

Table 2. Summary of averaged results. Each method was run 100 times; the values here represent the average of each measurement across those 100 runs

Problem 1:

Method	Offline Fitness	Offline Diversity
Normal	0.927	0.0492
Moving Target	0.814	0.0963
Peak Removal	0.957	0.1115
Peak Erosion	0.964	0.1009

Problem 2:

Method	Offline Fitness	Offline Diversity
Normal	0.982	0.0495
Moving Target	0.917	0.0755
Peak Removal	0.995	0.0590
Peak Erosion	0.997	0.0568

at each generation for each version of the GA, Fig. 4 shows the average fitness, and Fig. 5, the maximum fitness at each generation[1]. The maximum fitness and offline diversity for each method, and both problems, is shown in Table 2.

Several interesting trends can be noted in these results. First, it is clear that all modifications had a positive effect on offline diversity for both problems (Table 2) – Student's two-tailed t-test verifies that fact with $P < 1\%$. Looking

[1] Graphs for Problem 2 are similar, omitted due to space constraints

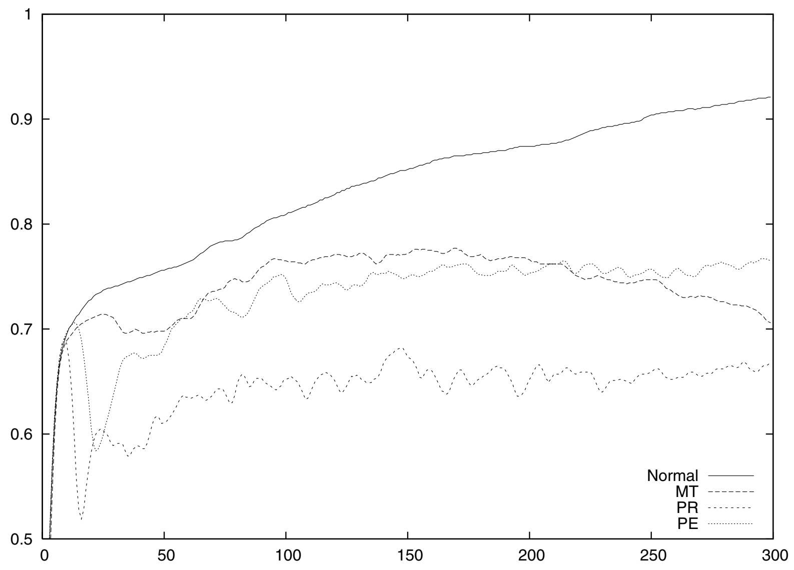

Fig. 4. Online fitness with lower limit of 0.5 to emphasize detail. The unmodified GA clearly has higher average fitness than the others, and Peak Removal is lowest. The Moving Target and Peak Erosion methods are similar in online fitness

at the Problem 1 graphs as a guide, initial diversity levels are high (as expected for an initially random population), then as poorly fit individuals are culled diversity drops dramatically. The stationary GA never recovers from this drop in diversity, though the others seem to exhibit a reactive bloom in diversity in just a few generations. Subsequently, PR and PE maintain a roughly static level of diversity, whereas with MT diversity maintains a slow rate of increase. Similar behavior was seen in Problem 2.

The fitness graphs are even more interesting. First, online fitness is significantly lower for MT, PR, and PE than in the stationary GA, with the lowest results in PR (Fig 4). On the other hand, offline fitness is significantly lower for MT, but higher in both PR and PE (Fig 5). All of these differences are statistically significant (Student's t-test $P < 1\%$). These results may not be intuitive, but do make sense: they imply that PR and PE are better at forcing the population to leave peaks that have already been found, thus forcing more individuals into low-fitness regions of the search space and decreasing average fitness. Again, results for Problem 2 are similar.

6 Conclusions

There are three conclusions that may be drawn from these results. First, diversity in static tasks can indeed be improved by adding non-stationary subtasks, which

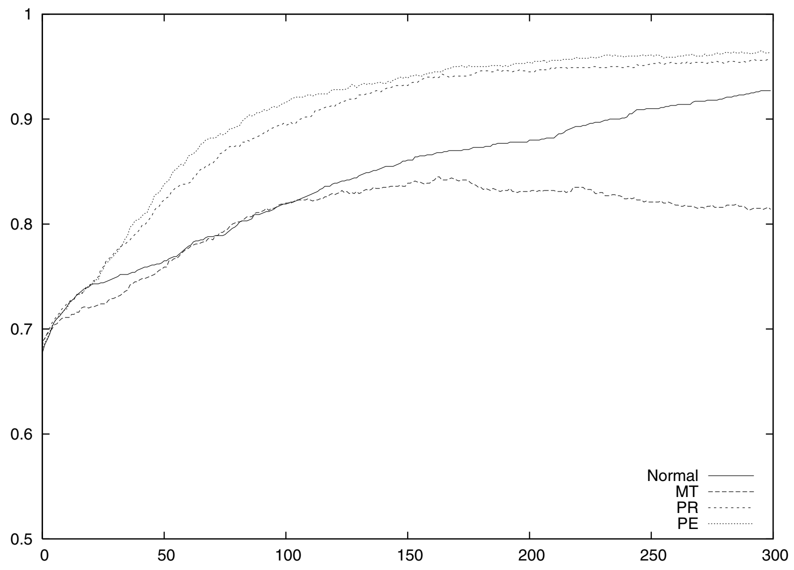

Fig. 5. Maximum fitness at each generation, with lower limit of 0.5. This represents offline fitness “in progress.” The Moving Target method has a negative impact on offline fitness, whereas the Peak Removal and Peak Erosion methods have a positive influence, especially in early generations

can also be seen as making the static landscape dynamic. Second, non-stationary subtasks can improve performance – but not necessarily in proportion to the increase in diversity. Finally, as an extension of the former points, online fitness and diversity are not necessarily accurate predictors of offline fitness.

All three modifications exhibited significantly increased diversity. Intuitively, it makes sense that a dynamic fitness landscape would lead to increased diversity: if the local optimum that the GA is working toward repeatedly moves or changes before the population converges, then the loss of diversity related to full convergence will never occur. In the case of the Moving Target, the population is attracted in varying directions in order to pull individuals away from discovered peaks in the fitness landscape. In Peak Removal or Erosion, peaks found are made less attractive – metaphorically pushing the population away. In a rank-based selection method, these pulling and pushing effects are basically equivalent, as the diversity results suggest (Fig 3).

Though intuitively in may seem that increased diversity should lead directly to increased performance, it is clear that there is at least some decoupling between the two. PR and PE perform better than a normal GA; MT performs significantly worse – even though the diversity levels in PR, PE, and MT are comparable. It would seem that taking more information about the current landscape and population distribution into account will generate more intelligent – and more effective – methods of modifying that landscape at runtime. Of course, in doing so one must consider the computational cost of the intelligence as compared to the resulting fitness benefit. We hope to test these ideas by exploring other potential dynamic subtasks in future work.

The relationship between online fitness and offline fitness is also indirect, largely due to the presence or absence of diversity. Specifically, a drop in online fitness does not necessarily correspond to a drop in offline fitness, as shown by PR and PE. In fact, it would seem that the ideal case would have high online and offline diversity, and low online fitness – though MT seems to fit this pattern and yields low offline fitness as well. More research needs to be done in order to discover the difference between the apparent 'good' and 'bad' sorts of diversity. Early results seem to indicate that diversity is only 'good' as long as some amount of convergence is allowed, and that convergence must be related to an optimum in the static landscape. Allowing total convergence is to be avoided, so perhaps an adaptive method which balances a diversification process against runtime-observed convergence would be effective. This is an area we hope to explore further.

References

1. D. Whitley, T. J. Starkweather, and C. Bogart, "Genetic algorithms and neural networks: Optimizing connections and connectivity," *Parallel Computing*, vol. 14, no. 3, pp. 347–361, 1990.
2. T. J. Starkweather, D. Whitley, and K. Mathias, "Optimization using distributed genetic algorithms," in *Parallel Problem Solving from Nature*, H.-P. Schwefel and R. Männer, Eds. Berlin: Springer Verlag, 1990, pp. 176–185.
3. C. R. Houck, J. A. Joines, and M. G. Kay, "Comparison of genetic algorithms, random restart and two-opt switching for solving large location-allocation problems," *Computers & Operations Research*, vol. 23, no. 6, pp. 587–596, 1996.
4. J. Horn, *GAs (with sharing) in search, optimization and machine learning.* San Mateo, CA: Morgan Kaufmann, 1997.
5. D. Whitley, S. B. Rana, J. Dzubera, and K. E. Mathias, "Evaluating evolutionary algorithms," *Artificial Intelligence*, vol. 85, no. 1-2, pp. 245–276, 1996.
6. J. M. Fitzpatrick and J. J. Grefenstette, "Genetic algorithms in noisy environments," *Machine Learning*, vol. 3, pp. 101–120, 1988.
7. F. Glover, "Tabu search: A tutorial," *Interfaces*, vol. 20, no. 4, pp. 74–94, 1990.