Will Selection for Mutational Robustness Significantly Retard Evolutionary Innovation on Neutral Networks?

Seth Bullock

Informatics Research Institute, School of Computing, University of Leeds, Leeds, LS2 9JT, UK

seth@comp.leeds.ac.uk

Abstract

As a population evolves, its members are under selection both for rate of reproduction (fitness) and mutational robustness. For those using evolutionary algorithms as optimisation techniques, this second selection pressure can sometimes be beneficial, but it can also bias evolution in unwelcome and unexpected ways. Here, the role of selection for mutational robustness in driving adaptation on neutral networks is explored. The behaviour of a standard genetic algorithm is compared with that of a search algorithm designed to be immune to selection for mutational robustness. Performance on an RNA folding landscape suggests that selection for mutational robustness, at least sometimes, will not unduly retard the rate of evolutionary innovation enjoyed by a genetic algorithm. Two classes of random landscape are used to explore the reasons for this result.

Introduction

It is well known that evolution will select for solutions with both high fitness and high robustness. That is, evolution favours *volumes* of search space associated with fit phenotypes, rather than single points of high fitness. The robustness of a solution is here defined in terms of its insensitivity to the action of the genetic operators at work during its evolution. By definition, then, perturbing a robust solution's genotype through mutation (or perhaps crossover) will not tend to perturb its fitness.

It is easy to see why this type of robustness (hereafter termed mutational robustness) is implicated in evolutionary adaptation. An individual's biological fitness is often equated with the number of progeny it leaves, but it might be more accurate to use the propensity to leave *viable* offspring as an indicator of fitness (Mills & Beatty 1994). This move encourages us to view fitness as the property of a *lineage* rather than a single genotype. Two genotypes that give rise to equivalent phenotypes (and thus have an equivalent propensity to leave offspring) may nevertheless differ in fitness under this interpretation. If the mutant offspring of one tend to be non-viable whereas those of the other tend to be viable, lineages stemming from the latter will be more successful than those stemming from the former (see Figure 1). For the remainder of the paper, I will reserve the term fitness to refer to the proximal propensity to leave offspring in the next generation and *effective fitness* to refer to the tendency for lineages stemming from a genotype to be successful over generational time. A more detailed treatment of this type of issue is presented in association with the concept of the quasi-species (Eigen & Schuster 1979; Eigen, McCaskill, & Schuster 1988; 1989; Nowak 1992).

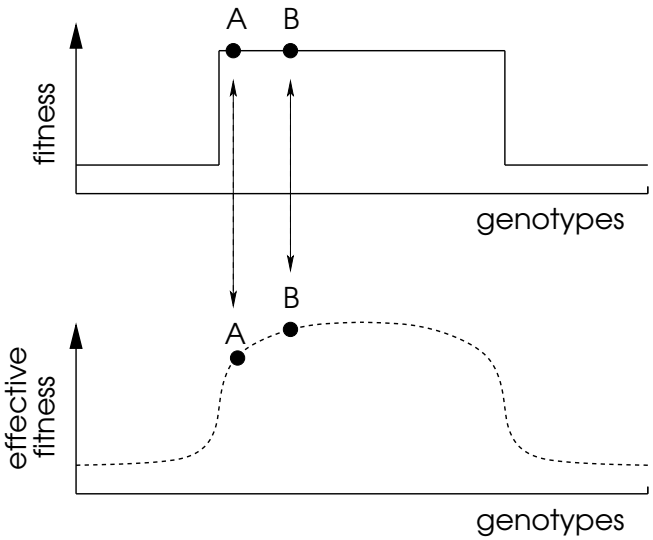

Figure 1: Two genotypes, A and B, achieve equivalent fitness scores (top). However, A's relative proximity to a fitness cliff ensures that it is less mutationally robust than B, and that, as a result, its effective fitness is somewhat reduced, while B's is increased (bottom).

For certain optimisation problems, the tendency for a genetic algorithm (GA) to favour mutationally robust solutions can be very useful. Consider a parametric design problem in which we assess the performance of each solution in simulation, but can only realise a solution in the real world subject to certain manufacturing tolerances. If the genetic operators at work during evolution mirror the nature of the real-world manufacturing errors, it is possible that a standard GA will discover not only

a good solution, but one that is robust to manufacturing errors. By exploiting the genetic algorithm's natural tendency to discover a fit volume of the search space, we can automatically steer the search towards realisable solutions.

Of course, if one wishes to discover the best *point* in a search space, a GA's tendency to be distracted by genotypes that are not as fit but are more mutationally robust can be frustrating (Schuster & Swetina 1998). In contrast, the behaviour of a hill-climbing algorithm (where a single solution is repeatedly mutated until an improvement is discovered at which point the improved mutant is repeatedly mutated, and so on) will not be affected by mutational robustness. Such a search process is able to scale ridges of increasing fitness that would be difficult or impossible for a genetic algorithm to climb due to their low mutational robustness. However, rather than infer that one class of algorithm is better or worse than another on the basis of these considerations, these different algorithmic biases should be understood to ensure that different algorithms are suited to different optimisation problems (Mitchell, Holland, & Forrest 1994; Wolpert & Macready 1997). Characterising which algorithms suit which problem types is the challenge that arises from this perspective.

Furthermore, comparing the behaviour of search algorithms as they traverse *flat* fitness landscapes may also reveal the presence of inherent biases. For example, the manner in which a search algorithm's mutation operators deal with illegal mutants may influence its search behaviour (Bullock 1999; 2001). Biases such as these can also be interpreted in terms of selection for mutational robustness. For instance, consider a real-valued, n-dimensional genotype where the value at each of the loci must lie in the range $[0, 1]$, and a mutation operator that perturbs the value at each of the loci by a small value drawn from a random distribution. Occasionally, the mutation operator will generate a genotype featuring one or more illegal values. Dealing with these illegal mutants may introduce a mutation bias.

For example, replacing any illegal mutant offspring with the nearest legal genotype increases the mutational robustness of solutions at the edges of the search space by lowering their effective mutation rate. In contrast, ignoring an illegal mutant offspring and replacing it with the offspring of a newly chosen parent will tend to reduce the mutational robustness of solutions close to the edges of the search space. The former approach will tend to bias evolutionary search in favour of extreme-valued genotypes, the latter away from these genotypes. Although the landscape appears to be flat, it is effectively warped by the mutation operator, which imposes a gradient in effective fitness where none was intended. The arbitrariness of these edge effects may retard evolutionary optimisation or introduce artefacts into an evolutionary simulation model (Bullock 1999; 2001).

Here this approach is extended to explore the impact of selection for mutational robustness on the behaviour of search algorithms trapped on neutral networks. First, the notion of neutrality and its relation to mutational robustness will be introduced.

Neutrality and Mutational Robustness

In the context of a search space, neutrality is the property of adjacent genotypes enjoying equivalent fitness scores. A neutral network consists of a set of genotypes with equal fitness, where each member of the set neighbours at least one other member. Again, these notions of adjacency and neighbourhood must be cashed out in terms of an algorithm's genetic operators.

For specific real-world search spaces, such as the RNA folding map, it has been demonstrated that the neutrality present is of a useful kind (Huynen 1996; Huynen, Stadler, & Fontana 1996; Fontana & Schuster 1998a; 1998b). Neutral networks *percolate* the search space, neighbouring a large proportion of possible alternative phenotypes. In addition, these neutral networks enjoy a property of *constant innovation* in that, over many generations, a neutral walk across these neutral networks tends to encounter novel phenotypes at a constant rate comparable to that which would be achieved by a random walk in the search space.

As such, a landscape exhibiting the right kind of neutrality will be much easier to search than the canonical rugged landscapes typically associated with complex search problems (Kauffman 1993). Rather than conceive a population to be hill-climbing in a rugged landscape, the picture painted by recent studies of neutrality is one of a population enduring periods of neutral drift punctuated by brief transitions to higher-fitness neutral networks (Barnett 1998; 2001).

Within the field of evolutionary computation, researchers have discovered that some difficult evolutionary optimisation problems already exhibit potentially useful neutral networks (Harvey & Thompson 1996). Others have attempted to encourage neutral networks where there are none by introducing redundancy into the genetic encoding, in the hope that this will improve the ability of evolutionary algorithms to find optimal solutions in general (Shipman *et al.* 2000). This research has prompted analysis of both naturally occurring neutral networks (Smith *et al.* 2002) and artificially crafted ones (Bullock 2001).

Implicit in the description so far has been the assumption that as genotypes on a neutral network code for equivalent phenotypes, they are selectively neutral with respect to one another. However, if a search algorithm has a tendency to favour mutationally robust genotypes this assumption does not hold. Rather than

drift at random across a neutral network, a genetic algorithm will tend to favour those parts of the network that have a higher degree of neutrality, since these parts are more mutationally robust (van Nimwegen, Crutchfield, & Huynen 1999; Wilke *et al.* 2001; Wilke 2001a; 2001b).

From the perspective of evolutionary optimisation, this tendency raises a concern. If we are interested in maintaining a high rate of constant innovation, we might not want our drifting population to concentrate on the most mutationally robust areas of the network, since these are precisely the areas in which the rate of innovation is at its lowest.

Here we will explore this issue by comparing the performance of a GA, in terms of the rate at which it discovers novel network neighbours, with that of a similar algorithm, termed a plateau crawler (PC), that has been designed to be immune to selection for mutational robustness.

A Simple Search Problem

In order to demonstrate the effects of selection for mutational robustness on evolutionary dynamics, and to specify the two search algorithms we will concentrate on in this paper, a very simple toy search problem will be introduced. The *thresholded counting ones problem* is intended to be a trivial example of a problem featuring neutrality and neutral networks with which to introduce the effects of selection for mutational robustness. **It is by no means intended to reflect the character of neutrality as it occurs in typical real-world problems.**

Each genotype is a binary string of length L with an associated phenotype dependent on the number of genotypic bits that are set to one. Genotypes with more than t bits set to one enjoy maximum fitness. All other genotypes have zero fitness. (All problems explored in this paper share the property that genotypes are assigned a fitness of zero if they lie off the neutral network being considered, and maximal fitness otherwise.)

As a result, the search space can be divided into two neutral sets of genotypes. One sub-threshold set with fitness zero, and one super-threshold set with maximum fitness. We will consider the case in which $L = 100$ and $t = 50$. For this problem, the search space comprises a single neutral network associated with fitness zero and a second neutral network associated with maximum fitness.

We can expect a search algorithm that commences searching somewhere on the upper network to drift across this network in some manner. Periodically, the algorithm may generate sub-threshold mutants that lie off the upper network. Given that we are interested in the rate at which algorithms discover novel genotypes that neighbour a neutral network, i.e., the rate at which they generate potentially useful innovations, we will record the rate at which novel sub-threshold genotypes are generated as a measure of performance.[1]

The Algorithms

We will contrast the behaviour of two simple search algorithms, a standard genetic algorithm and a plateau crawler. The motivation for the design of the latter was to create an algorithm that is identical to the genetic algorithm in all respects save that it is provably immune from selection for mutational robustness. The resulting algorithm shares similarities with both the *random mutation hill-climber* (Forrest & Mitchell 1993) and the Barnett's (2001) *net-crawler*.

Genetic Algorithm

1. Initialise a population of S clones of a random chosen genotype, G, with the phenotype P
2. Pick parents at random from the current generation until one is found with phenotype P
3. With probability $1 - M$: Copy the chosen parent unchanged to the next generation

   With probability M:

   (a) Mutate the parental genotype, forming a mutant genotype, G_m, by replacing the value at a randomly chosen loci with a randomly chosen legal alternative

   (b) Determine the phenotype, P_m, of the resulting mutant

   (c) If $P_m \not\equiv P$ and G_m has not been encountered before, increment C, a cumulative tally of the number of novel off-network genotypes discovered by the algorithm so far

   (d) Add the resulting offspring to the next generation
4. Repeat steps 2 and 3 until a complete new generation has been produced
5. Record the value of C for this generation, C_n
6. Repeat steps 2 thru 5 until N generations of search have been completed

Plateau Crawler

In order to implement a plateau crawler (PC), we need only add the following step between 3(c) and 3(d):

- If $P_m \not\equiv P$ replace the mutant with an exact copy of its parent.

[1] In reality, we are interested in the discovery of fit phenotypes, but in the absence of any information concerning how phenotypes are distributed in the space, the discovery of novel genotypes is a good proxy.

Notice that, for each algorithm, population size is fixed, parents are chosen in a standard roulette-wheel fashion, and reproduction involves only single point mutations and no crossover. Mutation rate, M, was set to 0.5 for all runs reported in this paper - i.e., 50% of offspring were identical to their parents, while 50% differed by exactly one allele.

This mutation rate is relatively low compared to rates typically used in evolutionary computation. To the extent that error-free reproduction can occur, pressure for mutational robustness will decrease. If every offspring were a perfect copy of its parent there would be no pressure to avoid genotypes that neighboured non-viable mutants. However, such a situation would lack the heritable variation that drives mutation – there would be no selection at all. As mutation rate is increased, we would expect selection for mutational robustness to become a more significant driver of evolutionary change. In the case where every offspring is a mutant, selection for mutational robustness will be critical. Fit points in genotype space will not be represented in a population unless they are surrounded by similarly fit mutant genotypes.

Although the plateau crawler algorithm is practically identical to the genetic algorithm, in operation it more closely resembles a population of S hill-climbers. It differs from the genetic algorithm in that every parent chosen to reproduce is guaranteed to either leave a copy of itself or a mutated offspring that shares its phenotype and fitness. As such the plateau crawler ensures that all offspring, and grandchildren, etc., have an equal chance of being chosen as parents themselves. This implies that the plateau crawler should be unaffected by mutational robustness, spending on average an equal amount of time at each point on a neutral network (Hughes 1996).

By contrast, the fact that the genetic algorithm allows parents to leave mutant offspring that are not viable ensures that lineages featuring genotypes with many non-neutral neighbours will not leave as many descendants as lineages featuring genotypes that are mutationally robust. This implies that, on average, the genetic algorithm should spend more time in the parts of the neutral network that enjoy a high degree of neutrality (van Nimwegen, Crutchfield, & Huynen 1999; Wilke 2001a).

Notice also that the plateau crawler *differs* slightly from a population of S independent hill-climbers in that rather than ensuring that each of the S members of the population generate exactly one offspring, the algorithm (like a GA) samples S parents at random from the population with replacement. As a result of the sampling error that this method entails, we can expect that a plateau crawler population, like a GA population, will tend to remain clustered rather than simply diffuse across the network as a population of independent hill-climbers would (Derrida & Peliti 1991; Barnett 2001).

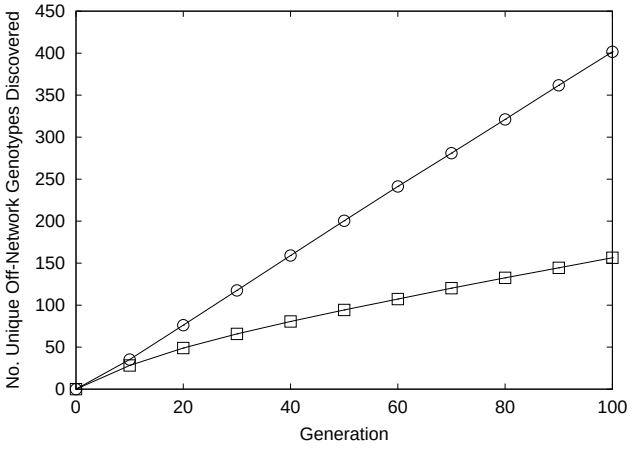

Figure 2: Mean (n=500) discovery rate of a plateau crawler (circles) and a genetic algorithm (squares) solving the thresholded counting-ones problem with genotype length 100 and threshold 50 for 100 generations of search.

Results

For our purposes, an algorithm should be rewarded for discovering novel non-neutral genotypes quickly and often. As such, each algorithm's performance is equated with the cumulative frequency which it discovers unique off-network genotypes (hereafter termed an algorithm's *discovery rate*). This value can be calculated as the area under a cumulative plot of unique off-network genotypes discovered over generational time,

$$D_{\text{alg}} = \sum_{n=1}^{N} C_n$$

The discovery rate of each algorithm for the thresholded counting ones game with $L = 100$ and $t = 50$, averaged over 500 runs of 100 generations each, is depicted in Figure 2. As might be expected given the above discussion, the genetic algorithm is outperformed by the plateau crawler. We can see why when we consider the manner in which the 500 populations were distributed over the phenotype space at the end of the runs (Figure 3). The plateau crawler has, on average, distributed the evolving population across the entire neutral network, closely approximating a random sampling of the high-fitness plateau. By contrast, the genetic algorithm has tended to avoid genotypes associated with phenotypes close to 50. For the genetic algorithm, this volume of the search space suffers from poor mutational robustness, and as a result is not successful.

These results, although drawn from a trivial search problem, raise some interesting possibilities. Could the

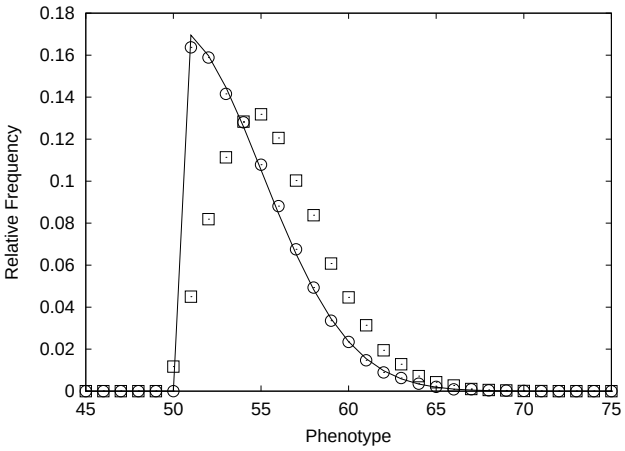

Figure 3: Three phenotypic frequency distributions for the counting-ones problem with genotype length 100 and threshold 50. The solid line represents the frequency distribution that would result from a population of randomly drawn super-threshold genotypes. Circles denote the aggregate phenotypic frequency distribution, after 100 generations of search, achieved by 500 runs of a plateau crawler. Squares denote the equivalent aggregate distribution achieved by 500 runs of a genetic algorithm. See text for details.

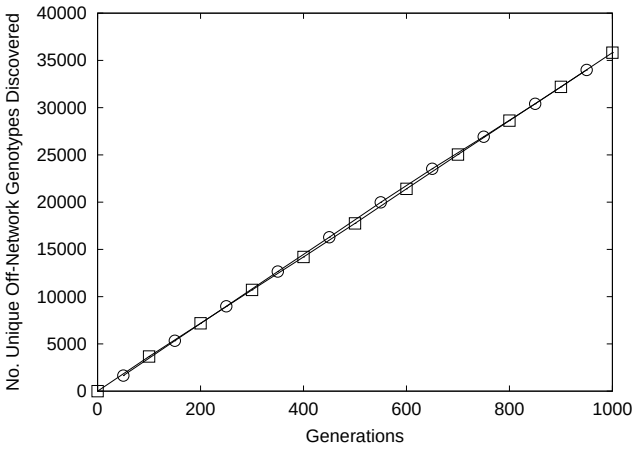

Figure 4: Discovery rate for a plateau crawler (circles) and a genetic algorithm (squares) searching an RNA folding landscape. Each algorithm was initialised with a population of 100 clones of the same randomly chosen RNA sequence of length 100.

difference between the performance of the two algorithm's somehow be used as an on-line metric with which to dynamically measure the strength of selection for mutational robustness? With this type of information at our disposal, could we dynamically alter a search algorithm to control for this search bias? Perhaps increasing or decreasing the mutation rate might allow us to counter an algorithm's tendency to concentrate on unproductive volumes of search space (Barnett 1998)? Alternatively, perhaps this information might be used by a hybrid algorithm to appropriately swapped from GA-style search to PC-style search. Before we can assess these possibilities, let us explore the performance of the same algorithms on a more realistic search problem.

RNA Folding Landscape

The manner in which different RNA sequences fold into their associated secondary structures has been a useful test case for assessing the role of neutrality and neutral networks in adaptive evolution (Schuster *et al.* 1994; Huynen, Stadler, & Fontana 1996; Schuster & Fontana 1999; Wilke 2001a). These studies have revealed that many RNA sequences fold into equivalent secondary structures and that the frequency distribution over these secondary structures exhibits a Zipf-like power law, with only a few very frequent secondary structures and very many rare secondary structures. In addition, it is common that a phenotype is easily accessible from an arbitrary point in sequence space, with randomly chosen phenotypes being separated by only a small number of point mutations. Recent studies have begun to explore the role of mutational robustness in influencing adaptation on these landscapes (van Nimwegen, Crutchfield, & Huynen 1999; Wilke 2001b; Wilke *et al.* 2001), concluding that selection for mutational robustness is an important driver at high mutation rates.

Here we compare the performance of the GA and PC algorithms on neutral networks chosen at random from a space of RNA sequences of length 100. Secondary structures were computed using version 1.4 of the Vienna RNA Package (Hofacker *et al.* 1994) which uses the free energies described in (Mathews *et al.* 1999) and is freely available from `http://www.tbi.univie.ac.at/~ivo/RNA/`. Folding was calculated at 30°C, and secondary structures were considered equivalent if the edit distance separating their tree representations was zero.

Since we expect that, in practice, search algorithms traversing a neutral network will not need to explore a large proportion of the network before encountering a transition to a higher-fitness phenotype, we will focus on the relatively short- or medium-term behaviour of search algorithms on neutral networks. In particular, we will not be particularly interested in the behaviour of these algorithms in the limit of infinite time.

Figure 4 shows the performance of the two algorithms run from the same initial RNA sequence for 1000 generations of search. In marked contrast to the results from the thresholded counting ones problem, both algorithms perform comparably. Does this result hold in general for neutral networks drawn at random from this search

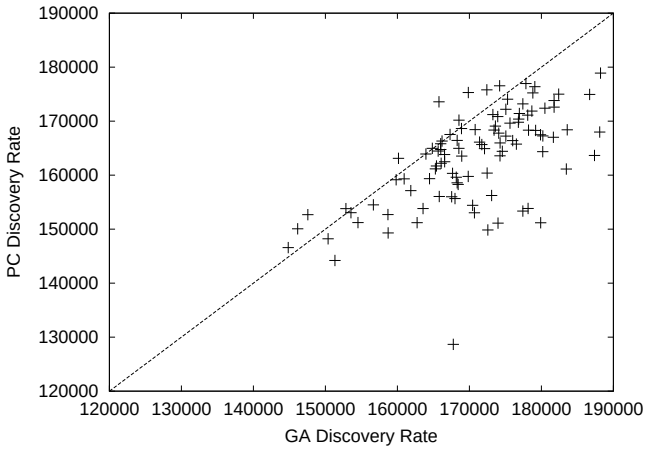

Figure 5: Scatterplot comparing the discovery rate of a genetic algorithm with that of a plateau crawler. Each of the 100 points represents a pair of 100-generation runs from a randomly chosen ancestral RNA sequence of length 100. The line $y = x$ separates the graph into a lower area in which the genetic algorithm outperforms the plateau crawler and an upper area in which the converse is true. On average, the genetic algorithm outperforms the plateau crawler across the sample.

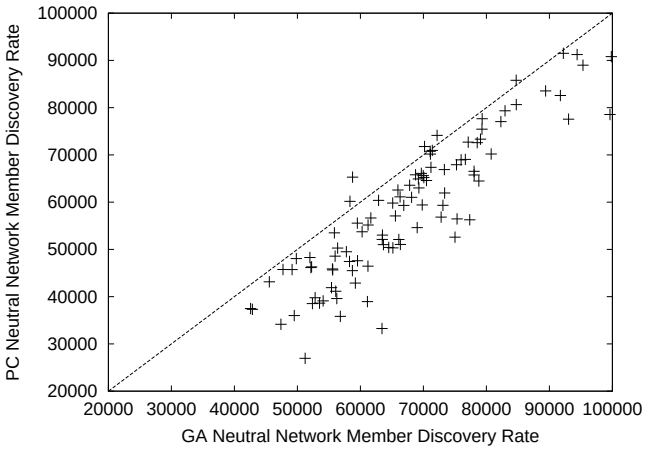

Figure 6: Scatterplot comparing the rate at which a genetic algorithm discovers novel members of a neutral network with the rate achieved by a plateau crawler. Points represent the same runs depicted in Figure 5.

space? Figure 5 shows the results obtained by running each algorithm for 100 generations from the same 100 randomly chosen RNA sequences. The two algorithms continue to perform comparably, with, on average, the GA slightly outperforming the PC, discovering novel network neighbours at a slightly higher rate.

Given that we expect the PC algorithm to sample the neutral network evenly, spending the same amount of time at each point on the network, whereas the GA is expected to favour the less productive parts of the network, how can we account for this result?

Figure 6 demonstrates that during the same runs depicted in Figure 5 the GA consistently visited a greater number of unique genotypes on the neutral network. The plateau crawler may, in the limit, visit the entire network, but the genetic algorithm, in the short-term is doing a better job of sampling the network. In addition, Figure 7 shows that the during the same runs, the plateau crawler revisited already explored network neighbours more frequently than the genetic algorithm. While the plateau crawler is not confined to mutationally robust volumes of the search space, it has a tendency to repeatedly sample the same neighbours, and in doing so wastes time that the genetic algorithm spends in novel parts of the neutral network.

How generic are the results we have obtained for first the thresholded counting ones problem and now the RNA landscape? What are the properties of each that are responsible for the performance of the two algorithms that we have tested? Clearly, there exist landscapes for which genetic algorithms are significantly retarded by selection for mutational robustness. Equally clearly there exist different landscapes for which this is not the case. Before addressing these questions, a class of random landscapes with which to explore these phenomena further will be introduced.

Random Landscapes

A useful class of fitness landscape has been proposed by Barnett (1998) in order to study the role of neutrality in adaptive evolution. Dubbed *NKp landscapes* they extend Kauffman's (1993) NK landscape formalism to include a tunable degree of neutrality. Future work will explore replicating the results presented here on NKp landscapes, but for the moment we will focus on a simpler, and perhaps as a result less interesting, scheme.

Search commences at an arbitrary genotype which is assigned a fitness of 1. Whenever, through mutation, a search algorithm generates a novel genotype, we determine its fitness by reference to the landscape's degree of neutrality, p. With probability p, the mutant enjoys fitness 1, otherwise it is assigned fitness zero. In this manner, the fitness landscape is generated at random as the population traverses it. In this simple random landscape, each point has, on average, an equivalent degree of neutrality (although, of course, each offspring genotype may be adjacent to genotypes that have already been encountered and thus have already had their fitness specified). By varying p, we can specify landscapes in which neutrality is rare or commonplace.

Figure 8 depicts how the two search algorithms perform across a range of landscapes, as we vary p. At extreme values of p both algorithms perform compara-

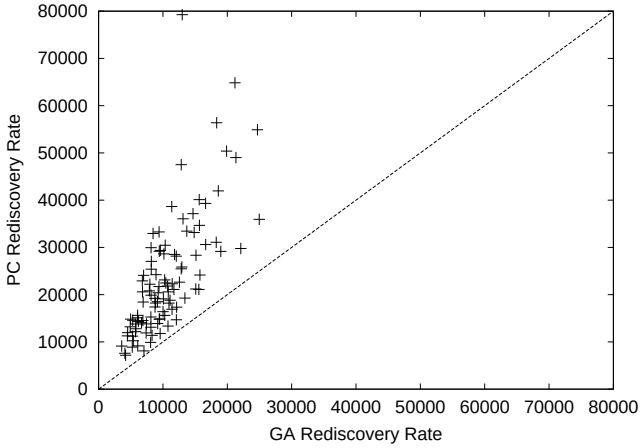

Figure 7: Scatterplot comparing the rate at which a genetic algorithm *rediscovers* genotypes that neighbour a neutral network with the rate achieved by a plateau crawler. Points represent the same runs depicted in Figures 5 and 6.

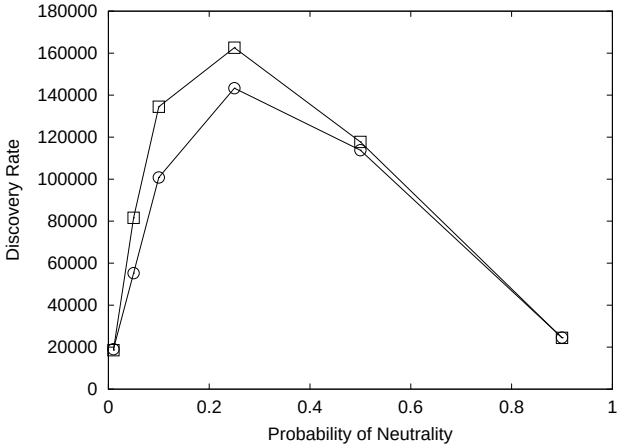

Figure 8: Variation in mean (n=50) discovery rate of a genetic algorithm (squares) and a plateau crawler (circles) after 100 generations of search on random graphs as the degree of neutrality, p, is varied from low ($p \approx 0$) to high ($p \approx 1$).

bly since there are either very few off-network genotypes to discover ($p \approx 1$) or very few neutral neighbours to discover ($p \approx 0$). At intermediate values of p, both algorithms improve, with the GA outperforming the PC over the entire range. The genetic algorithm appears more able to take advantage of the tendency for intermediate degrees of neutrality to increase the number of network neighbours available for discovery.

Figure 9 suggests that again an explanation for this disparity is to be found in the PCs tendency to resample network neighbours at a higher rate than the GA. Even though, on average, no point on the neutral network is more mutationally robust than any other, and hence no more productive, there still appears to be a significant difference between the ability of each algorithm to traverse the network.

In fact, the PC suffers a tendency to linger at each point in the neutral network that is inversely proportional to the degree of neutrality at that point. This tendency is necessary if the algorithm is to sample the entire network equally. Since, by definition, there are fewer neutral transitions to genotypes with a low degree of neutrality, they are hard to find. As a result, an algorithm that must spend equal amounts of time at each point on a neutral network must linger at these points when they are discovered. Unfortunately, during these periods there is a tendency to repeatedly generate the same mutant offspring.

In contrast, the GA has a tendency to desert points in the network that suffer a low degree of neutrality, exposing it to the complementary risk of repeatedly generating the same *neutral* mutants in an area of mutational robustness. However, it appears that these complementary risks do not negate one another in these landscapes—eschewing the most peripheral parts of a neutral network does not prohibit the GA encountering novel network neighbours at a higher rate than the PC.

Introducing Local Structure

By slightly altering the manner in which a random landscape is generated, we can introduce some of the local structure that is absent from the random landscapes explored above, but is presumably present in real search spaces such as that of the RNA folding problem. In this way, rather than tending to distribute neutrality evenly across the landscape, we can ensure that different parts of the network exhibit different degrees of neutrality, and can alter the degree to which genotypes with a high degree of neutrality tend to cluster together.

When a novel genotype is generated by a search algorithm, we assign it a fitness value as before, but assign that point in the search space a unique p value as follows:

$$p_{\text{offspring}} = \begin{cases} \Phi^{\kappa} & (p_{\text{parent}} < 0.5) \\ 1 - (\Phi^{\kappa}) & (p_{\text{parent}} \geq 0.5) \end{cases}$$

Where Φ is a uniformly distributed random variate in the range $[0, 1]$, and κ is a clustering parameter used to control the extent to which genotypes with a high degree of neutrality tend to neighbour genotypes with a similar high degree of neutrality. For $\kappa = 1$ the degree of neutrality associated with each point in the space is chosen at random from the range $[0, 1]$, and as such is independent of its neighbours. As κ increases, a genotype's degree of neutrality becomes increasingly determined by that of its "parent".

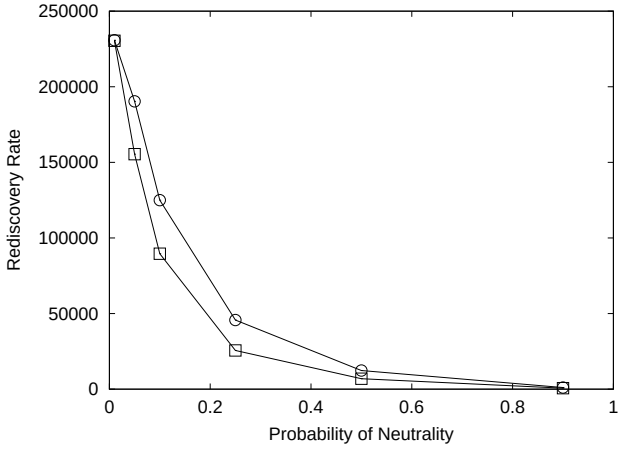

Figure 9: Variation in the mean (n=50) rate at which previously discovered off-network genotypes are revisited by a genetic algorithm (squares) and a plateau crawler (circles) after 100 generations of search on random graphs as the degree of neutrality, p, is varied from low ($p \approx 0$) to high ($p \approx 1$).

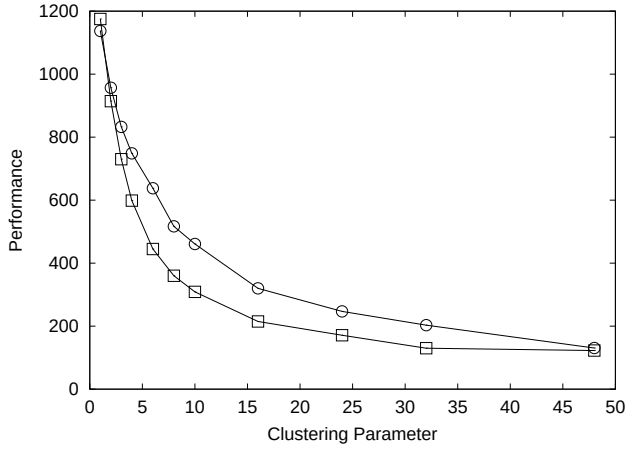

Figure 10: Variation in mean (n=50) discovery rate of a genetic algorithm (squares) and a plateau crawler (circles) after 100 generations of search on random graphs as a clustering parameter, κ, is varied from no clustering ($\kappa = 1$) to highly clustered ($\kappa \approx 50$).

Figure 10 depicts the manner in which the performance of each algorithm varies with the clustering parameter, κ. For $\kappa = 1$ the GA outperforms the PC. As κ increases, the performance of each algorithm falls to an equivalent asymptotic level. However, PC performance declines at a slower rate than GA performance, ensuring that the PC enjoys an advantage for the majority of the range explored. Here we begin to see the deleterious effects of selection for mutational robustness that were expected given the results from the thresholded counting ones problem.

Where there is a degree of local structure to the search space, the tendency for a genetic algorithm to concentrate on mutationally robust genotypes tends to retard the rate at which it discovers off-network novelty, at least by comparison with a plateau crawler.

It is likely that the locally structured random landscapes characterised by intermediate values of κ resemble that of the thresholded counting ones problem with which this study opened. Such landscapes will feature densely interconnected neutral networks separated by relatively sharp boundaries.

Discussion

The results presented here are preliminary in the sense that they should be regarded as the precursor to analytical work. Currently it is unclear what definitive answers can be given to the question raised in the title of this paper. Nevertheless, these simulations serve to demonstrate that different search algorithms cope with different types of neutrality in different ways. As such they suggest the potential for algorithms to be tailored to suit particular classes of neutrality, or to dynamically reconfigure themselves to cope with changes in the character of the landscape they find themselves searching.

Moreover, these results show that although selection for mutational robustness will impede evolutionary innovation on a neutral network, that this is a significant problem for evolutionary optimisation is by no means a foregone conclusion. It may well be that significant numbers of real-world search problem share the structural properties of the RNA landscape such that standard evolutionary algorithms will remain largely untroubled by selection for mutational robustness.

The extent to which the topology of a neutral network is a critical determiner of the dynamics of evolution adaptation is beginning to be appreciated (van Nimwegen, Crutchfield, & Huynen 1999). Here, we identify the character of a network's local structure as critical in determining the impact of selection for mutational robustness on the performance of a search algorithm.

We have also highlighted the difference between an algorithm's search properties in the limit and its short-term behaviour on a neutral network. While a plateau crawler is guaranteed to be immune to selection for mutational robustness and to sample an entire neutral network in an unbiased manner, this guarantee does not prevent its short-term behaviour from being inefficient. For certain landscapes, including the RNA folding map, this inefficiency is such that a search algorithm impeded by selection for mutational robustness is nevertheless able to outperform it.

Conclusion

Selection for mutational robustness (alongside selection for rate of reproduction) drives the evolutionary adaptation of natural and artificial populations. While this selection pressure may sometimes appear serendipitous, it is capable of frustrating evolutionary progress towards fit solutions (Schuster & Swetina 1998) or biasing simulation results (Bullock 1999). Here, numerical simulations were used to demonstrate the role of selection for mutational robustness in the dynamics of two classes of population-based search algorithm. Rather than suffer from selection for mutational robustness, the genetic algorithm achieved a level of performance comparable to or exceeding that of a plateau crawler designed to be immune to selection for mutational robustness.

A formal analysis of the manner in which selection for mutational robustness influences the dynamics of real search algorithms on neutral networks is still required, but the current study suggests that in comparison to an unbiased search algorithm and for real-world problems with fitness landscapes that resemble that of the RNA folding problem, the ability of a standard genetic algorithm to generate potentially useful innovations will not be significantly retarded by selection for mutational robustness.

Acknowledgements

The paper benefited from reviewers comments, discussion with Lionel Barnett, Matt Glickman, Inman Harvey, and Tom Smith and conversation with the University of Leeds biosytems reading group.

References

Barnett, L. 1998. Ruggedness and neutrality—the NKp family of fitness landscapes. In Adami, C.; Belew, R.; Kitano, H.; and Taylor, C., eds., *Artificial Life VI*, 18–27. MIT Press, Cambridge, MA.

Barnett, L. 2001. Netcrawling: Optimal evolutionary search with neutral networks. In *Proceedings of the Congress on Evolutionary Computation*, 3037. IEEE Press.

Bullock, S. 1999. Are artificial mutation biases unnatural? In Floreano, D.; Nicoud, J.-D.; and Mondada, F., eds., *Fifth European Conference on Artificial Life*, 64–73. Springer, Berlin.

Bullock, S. 2001. Smooth operator? Understanding and visualising mutation bias. In Kelemen, J., and Sosik, P., eds., *Sixth European Conference on Artificial Life*, 602–612. Springer, Heidelberg.

Derrida, B., and Peliti, L. 1991. Evolution in a flat fitness landscape. *Bulletin of Mathematical Biology* 53:355–382.

Eigen, M., and Schuster, P. 1979. *The Hypercycle—A Principle of Natural Self-Organization.* Berlin: Springer-Verlag.

Eigen, M.; McCaskill, J.; and Schuster, P. 1988. Molecular quasi-species. *Journal of Physical Chemistry* 92:6881–6891.

Eigen, M.; McCaskill, J.; and Schuster, P. 1989. The molecular quasi-species. *Advances in Chemical Physics* 75:149–263.

Fontana, W., and Schuster, P. 1998a. Continuity in evolution: On the nature of transitions. *Science* 280:1451–1455.

Fontana, W., and Schuster, P. 1998b. Shaping space: The possible and the attainable in rna genotype-phenotype mapping. *Journal of Theoretical Biology* 194:491–515.

Forrest, S., and Mitchell, M. 1993. Relative building block fitness and the building block hypothesis. In Whitely, D., ed., *Foundations of Genetic Algorithms 2.* Morgan Kaufmann, San Mateo, CA.

Harvey, I., and Thompson, A. 1996. Through the labyrinth evolution finds a way: A silicon ridge. In Higuchi, T.; Iwata, M.; and Weixin, L., eds., *First Internatonal Conference on Evolvable Systems*, 406–422. Springer-Verlag.

Hofacker, I. L.; Fontana, W.; Stadler, P. F.; Bonhoeffer, L. S.; Tacker, M.; and Schuster, P. 1994. Fast folding and comparison of RNA secondary structures. *Monatshefte Für Chemie* 125(2):167–188.

Hughes, B. D. 1996. *Random Walks and Random Environments*, volume 2. Oxford: Clarendon.

Huynen, M.; Stadler, P.; and Fontana, W. 1996. Smoothness within ruggedness: The role of neutrality in adaption. *Proceedings of the National Academy of Science USA* 93(1):397–401.

Huynen, M. 1996. Exploring phenotype space through neutral evolution. *Journal of Molecular Evolution* 43:165–169.

Kauffman, S. A. 1993. *The Origins of Order: Self-organisation and Selection in Evolution.* New York: Oxford University Press.

Mathews, D. H.; Sabina, J.; Zuker, M.; and Turner, D. H. 1999. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of Molecular Biology* 288(5):911–940.

Mills, S. K., and Beatty, J. H. 1994. The propensity interpretation of fitness. In Sober, E., ed., *Conceptual Issues in Evolutionary Biology.* Cambridge, MA: MIT Press. 3–23.

Mitchell, M.; Holland, J. H.; and Forrest, S. 1994. When will a genetic algorithm outperform hill climbing? In Cowan, J. D.; Tesauro, G.; and Alspector, J., eds., *Advances in Neural Information Processing Systems*, volume 6, 51–58. Morgan Kaufmann Publishers, Inc.

Nowak, M. 1992. What is a quasispecies? *Trends in Ecology and Evolution* 7:118–121.

Schuster, P., and Fontana, W. 1999. Chance and ne-

cessity in evolution: Lessons from RNA. *Physica D* 133:427–452.

Schuster, P., and Swetina, J. 1998. Stationary mutation distributions and evolutionary optimization. *Bulletin of Mathematical Biology* 50:635–660.

Schuster, P.; Fontana, W.; Stadler, P. F.; and Hofacker, I. L. 1994. From sequences to shapes and back: A case study in RNA secondary structures. *Proceedings of the Royal Society of London, Series B* 255:279–284.

Shipman, R.; Shackleton, M.; Ebner, M.; and Watson, R. 2000. Neutral search spaces for artificial evolution: A lesson from life. In Bedau, M. A.; McCaskill, J. S.; Packard, N. H.; and Rasmussen, S., eds., *Artificial Life VII*, 162–169. MIT Press, Cambridge, MA.

Smith, T. M. C.; Husbands, P.; Layzell, P.; and O'Shea, M. 2002. Fitness landscapes and evolvability. *Evolutionary Computation* 10(1):1–34.

van Nimwegen, E.; Crutchfield, J. P.; and Huynen, M. 1999. Neutral evolution of mutational robustness. *Proceedings of the National Academy of Science USA* 96(17):9716–9720.

Wilke, C. O.; Wang, J. L.; Ofria, C.; Lenski, R. E.; and Adami, C. 2001. Evolution of digital organisms at high mutation rates leads to survival of the flattest. *Nature* 412:331–3.

Wilke, C. O. 2001a. Adaptive evolution on neutral networks. *Bulletin of Mathematical Biology* 63(4):715–730.

Wilke, C. O. 2001b. Selection for fitness versus selection for robustness in RNA secondary structure folding. *Evolution* 55(12):2412–2420.

Wolpert, D. H., and Macready, W. G. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1(1):67–82.