Continuing Beyond NFL: Dissecting real world problems

Oliver Sharpe
School of Cognitive and Computing Sciences, University of Sussex
Brighton BN1 9QH, UK. ++44 1273 748 253
`olivers@cogs.susx.ac.uk`

Abstract

In this paper it is argued that the classification of search spaces by the performance of different algorithms on them is an essential part of a principled methodology for tackling new search problems. It is further argued that the success of a given algorithm on a given problem can often be attributed to particular aspects of the generation or selection mechanisms used in the algorithm. This hypothesis is examined by removing different parts of the algorithm and observing the resulting changes in its performance. It is shown how dissecting the algorithm in this way can give rise to greater insight into how the algorithm is working on a given search space and at the same time help classify the nature of that space.

1 Introduction

This paper is a contribution to the on-going discussion about the nature of real problems for which search techniques like genetic algorithms are used. The paper has three parts to it. The first part gives the theoretical framework in which this work is based. The second demonstrates some of the ideas on the well known NK landscapes (Kauffman, 1993), and the third continues with the same themes onto a real world problem.

2 The need for classification

A lot of the current work in the field of GAs rests on various different opinions about the nature of real problems. Indeed, the idea that real world problems (RWPs from now on) are in some way different from general abstract problems, is the basis for most

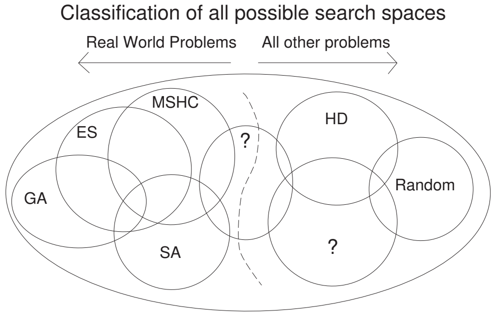

Figure 1: MSHC = Multi-start hill-climbing, ES = Evolutionary Strategies, GA = Genetic Algorithm, SA = Simulated Annealing, HD = Hill Descending

practitioners dismissal of the depressingly pessimistic No Free Lunch theorem (Wolpert & Macready, 1996) (NFL from now on), which states that when performance is averaged over all possible search problems, no search algorithm can perform better than random search. This is depressing because we know that random search is very bad at solving most interesting RWPs and therefore the NFL result suggests that on average all search algorithms will perform as badly. It is dismissed because anyone who uses stochastic search techniques knows through experience that this is simply not true. Take your average GA and it will easily outperform random search on most (if not all) interesting[1] RWPs. So what is going on?

The NFL result holds only when performance is averaged over all possible problems, and so that includes abstract problems which are of no interest to anyone - for example the problem where every point in the solution space represents a successful solution. Therefore the standard explanation of this discrepancy is

[1]We have to be careful not to let the word 'interesting' slip into meaning: precisely those problems that will prove my point !

the idea that RWPs form a class, or at least meta-class (figure 1), of problems which may represent only a small fraction of the set of all possible problems - hence even though the NFL result holds true for all problems, over the small class of RWPs it can be dismissed.

Although this argument is very persuasive, it cannot be more than a working hypothesis until the elusive notion of RWP is pinned down more precisely. At the moment RWP is in a dangerous position of meaning those problems for which we can do better than NFL would predict. However, a couple of characteristics of RWPs can be clearly stated:

First, and most importantly for random search's poor performance, interesting RWPs have very few good solutions in the search space and many bad solutions. The performance of random search is directly related to the relative distribution of good and bad points in the search space. RWPs have few good points: random search performs badly.

Secondly, a simple, first approximation of an answer as to why a GA does do well on RWPs (relative to random search) is because the designer of the search space has tried as hard as possible to make the paradigm of the search (the representation and the basic genetic operators on that representation) to be a fitness landscape as much like a Mt.Fuji as possible. To do otherwise would be self defeating on the part of the engineer (unless, of course there is a very specific and principled reason for setting up the paradigm in a different manner - e.g. when modelling a natural search process). The engineer attempts this by making a representation for which nearby genotypes will, in general, have nearby phenotypes and that nearby phenotypes will, in general, have similar fitness values. Nearby is defined by a mutation or other genetic operator that is thought to have usefully constructive properties. It is unlikely that a Mt.Fuji fitness landscape will be achieved (assuming that it is not a trivial problem), but at the very least the designer hopes that there will be a number of genetic operator based paths that lead from poor solutions incrementally towards good solutions. The existence of such paths is the only hope for a GA-like search strategy and the designer of the paradigm is deliberately trying to ensure that such paths will exist. Once again, to do otherwise would be self-defeating.

However, with all the best will in the world there is no recognised methodology that can ensure that the designer of a paradigm for a new search space will be able to ensure that a particular search algorithm will perform well on it. Such a methodology is urgently needed for the application of evolutionary algorithms

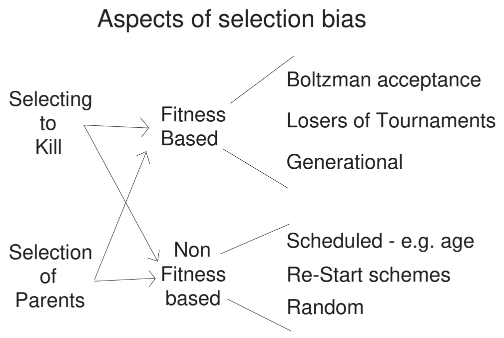

Figure 2: The selection 'dimension' of search bias

to turn into a science rather than the black art that it is today. However, such a methodology must rely on prior knowledge about the nature of the search space, such as by comparing its similarity to other well understood search spaces. To work otherwise would contradict the NFL result. NFL suggests that if we really have no knowledge at all about our search space then there can be no meaningful methodology for the engineer as, on average, any search technique, including the search technique of using a complex methodology, can do no better than random search. So in this situation we might as well use random search. In fact recent work (Langdon & Poli, 1998) has shown how there is a class of commonly used test problems on which random search is just as effective as genetic programming.

The conclusions sections of many papers in the EA literature are precisely a declaration that algorithm X is significantly better than algorithm Y at problem Z. So, even in the relative shelter of the RWPs, it still matters which algorithm is chosen for which tasks. An extreme example of this problem comes with the issue of when to re-start the search (Fukunaga, 1998). Many seasoned GA users will know the situation where a run is simply not going anywhere, and so the GA is re-started, this time leading to better results more quickly. The problem here is: when to re-start ? When have you waited long enough to know that the search is going nowhere? The two extreme ends of the spectrum in this problem would be the search space on which it is best to restart many times to find a short path to a good solution, versus the search space where what is needed for success is one very long path of search from any random start location. Without knowing on which class of search space you are on there is little you can do, except explore the search space, to find out which tactic will work. Without prior knowledge you need to evaluate the space to understand the space - there is still no free lunch.

Unlike in the field of machine learning, the no free

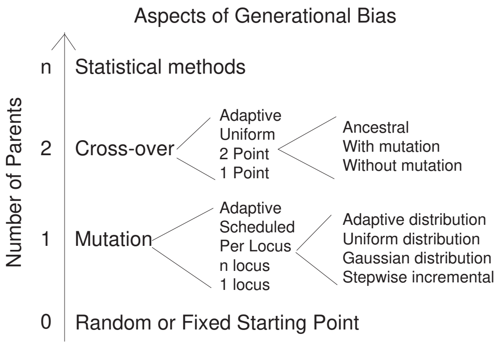

Figure 3: The generation 'dimension' of search bias

lunch problem faced by the search community is not an in principle problem as if we had enough time we could in theory evaluate every point of the search space and find the absolute optimum. The problem of search is a problem of time. As search spaces get larger (and more interesting) this problem gets progressively worse in comparison to the size of the whole search space. Each run of a search algorithm can only hope to scratch at a tiny percentage of the entire search space, and hopefully in so doing find a good solution. The use of purely on-line methods for improving the performance of search is undermined both by the NFL result and by the vanishingly small percentage of the space that will be seen. Hence the strong need to classify the natures of different types of RWP by the properties known about them prior to any evaluations on them.

2.1 The Classification Dimensions

What is needed is a corpus of well understood RWPs, each of them classified by which kinds of search algorithms perform well on them. This way the evolutionary algorithms community can gain confidence to induce the likely algorithm of choice for new instances of problems from a class of problems in the corpus. Central to the usefulness of the corpus would be the ability to assess the similarity of an new problem to one of the known problems in the corpus without doing many (or preferably any) evaluations. This will be very difficult to achieve, and first we need to get a much better understanding of why certain algorithms work on certain problems and others do not.

There are various different aspects of a search algorithm that can affect its behaviour. In a previous paper (Sharpe, 1998) I have argued that the different aspects of the algorithm each have their own type of bias and that the bias of the whole search algorithm may in some way be separable into the bias of the components of the algorithm. In particular there are the selection mechanism of the algorithm (that chooses which individuals to throw away and which to choose as parents for the next generation) and the generation mechanism (that makes the new individuals given the parents). To be successful, each search algorithm must have both a selection bias and generation bias that will be effective on the given problem. Hence, these two defining components of the algorithm can be seen as lying on two orthogonal 'dimensions' of the space of all possible search algorithms. Furthermore these dimensions, or aspects, of bias can be subdivided again (figures 2 and 3), for example the generation bias is constructed out of the biases of the genetic operators being used. Another possible dimension is the population size. I argue that often the success of a given algorithm on a given problem can be attributed to certain parts of the algorithm, rather than the whole.

From this viewpoint certain parts of the algorithm are 'doing the useful work' at any given time. If this is true then it should be possible to remove the parts of the algorithm that are not meant to be contributing and still observe successful performance. This paper tries to do precisely that with the generation aspect of bias, first on the NK landscapes, a well known set of abstract landscapes, and then on a very large real-world problem, namely optimising the neural network controller for an octopod robot. In both cases it will be argued that at certain times it is largely the mutation aspect of the generation bias that is doing the work of the GA. This is explored by removing the cross-over part of the generation mechanism of a GA and examining the performance of the resulting algorithm. Also examined was the behaviour of an algorithm with only cross-over, and the behaviour of random search. The differences between the behaviours of these algorithms also gives us some clear insights into the nature of the search spaces.

The aspects of bias that are seen to be most effective on a given search space are the ones that most clearly classify the nature of the search space. This paper will therefore see the natures of these search spaces from the perspectives of mutation, cross-over and random search.

3 Testing on NK landscapes

The NK Landscapes (Kauffman, 1993) are a well known and much used class of abstract problems. Although they have been criticised in a number of different ways they are still a useful tool for initial investigations into the behaviour of evolutionary algorithms. The main difference between the algorithms used here is in their generation bias and hence only the genera-

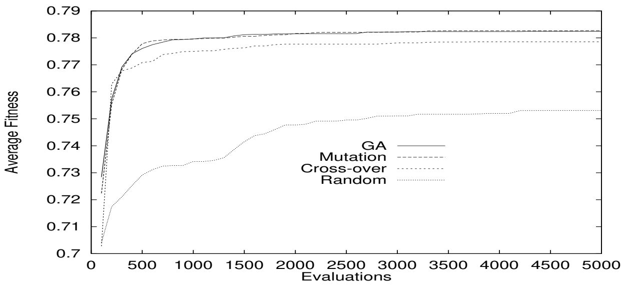

Figure 4: On NK landscape with N=20 K=2

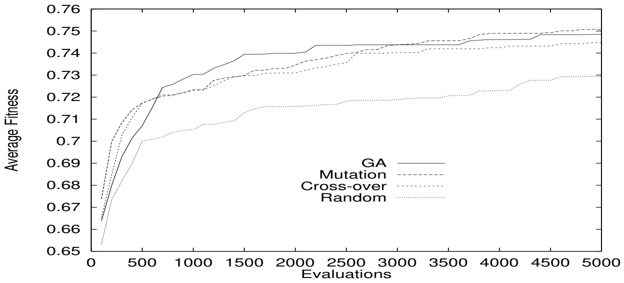

Figure 5: On NK landscape with N=20 K=9

tion aspect of classification can be investigated.

The landscapes looked at here all have $N = 20$ and have K values of K=2, K=9 and K=15. As the ruggedness of the landscape is increased, so the performance of the various algorithms changes. Performance can only be directly compared between algorithms on the same landscape as the exact values on NK landscapes are randomly generated for each new landscape. Therefore only the relative performance of the algorithms can be compared between the landscapes with different NK values. Each line represents the average performance of 30 runs of each algorithm. The standard deviations have not been shown for clarity of viewing, so the relative performance of the algorithms should only be seen as a rough trend rather than a concrete ordering. This is especially true when the performance of two algorithms is very similar.

The algorithms used are random search plus three algorithms based on a very simple GA. The population is sized 20, with a steady state replacement update rule. A tournament of three individuals is held and the best two are the parents for the generation of an offspring which replaces the worst of the three in the tournament. The mutation only algorithm simply has the cross-over part of the GA generation mechanism removed. The cross-over only algorithm has the mutation removed plus a small amount of new random variation is introduced in the following manner. If the

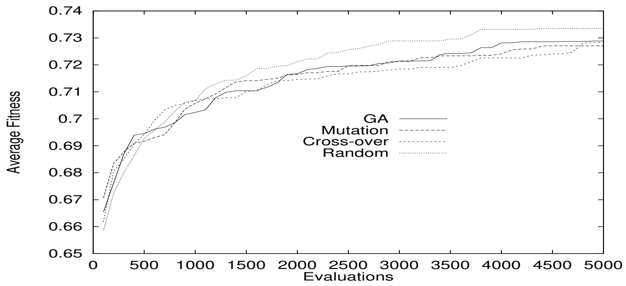

Figure 6: On NK landscape with N=20 K=15

two parents are identical then the offspring is random. This is the same mechanism as used in the previous work (Sharpe, 1998). This extra variation is added to ensures that the cross-over algorithm always has some new variation in the population to work with. By comparing the performance of this algorithm with the performance of purely random search we can see the added effects of cross-over, and thereby compare the effectiveness of cross-over to that of mutation in a fairer way. Without this extra variation the population converges to a single genotype so quickly that cross-over is unable to do any further useful work. Cross-over is implemented as uniform cross-over.

3.1 Results

On the N=20, K=2 landscape (figure 4), the mutation only algorithm and GA perform equally well. The cross-over only algorithm clearly performs better than random search, but also clearly worse than the other two. This strongly suggests that the good performance of the GA can largely be attributed to the good performance of the mutation aspect of its generation bias. Hence this landscape can be classified as mutation friendly.

On the more rugged landscape with K=9 (figure 5), the two algorithms mutation only and cross-over only are performing equally well and significantly better than random search. The GA is performing ever so slightly better still. In this situation the performance of the GA cannot be attributed mainly to one or other of its constituent generation biases, instead there appears to be a synergistic effect between the two biases that allows them to perform better together than apart. It is in this situation that the GA can truly be said to be the algorithm of choice for a given problem. Hence such a landscape can be classified as GA friendly.

The most rugged of the landscapes examined here has K=15 (figure 6). Here random search is performing

Figure 7: The mobile robot

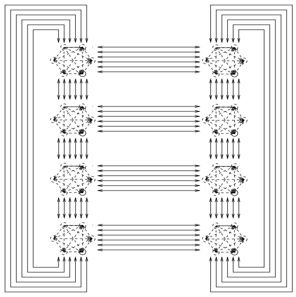

Figure 8: This diagram shows the architecture of the recurrent neural network used in the RWP and gives a rough impression of its complexity

slightly better than the other algorithms. Hence the performance of the other algorithms is hardly impressive and cannot be meaningfully attributed to their constituent generation biases. Such a search space is random search friendly.

4 Testing on a RWP

The RWP used here is actually a simulation of an octopod robot written by (Jacobi, 1998) as part of his thesis to demonstrate that minimal simulations can be used to successfully evolve robot controllers in simulation that work in the real world. The optimisation task presented to the GA in (Jacobi, 1997) was to evolve neural network control architectures that would allow a robot (figure 7) to wander around its environment avoiding objects using its infra-red sensors and backing away from objects that it hits with its bumpers. This is a hard behaviour to evolve when one considers that in order to achieve any sort of coherent movement the controller has to control not just one or two motors in a coordinated fashion but sixteen.

Competent control architectures evolved in the simulation have been shown by (Jacobi, 1997) to transfer successfully onto the real robot. The fact that it is a simulation makes the task of analysis both faster and more convenient than using the real robot, whilst the proven success of transfered evolved controllers gives us confidence to treat the simulation as an example real-world problem.

The nature of minimal simulations means that there is a large amount of noise within the evaluation of a given genotype. The sad reality is that each line on the graphs for the octopod robot takes at least 5 days of computer time.

4.1 Encoding scheme

As this work is intended to be part of the growing understanding about the nature of different types of RWPs, it is essential to give some details of the representation used.

The layout of the neural network architecture was fixed and predefined for every individual (figure 8), a simple direct encoding scheme was employed. Every parameter was encoded on the genotype by a real number in the range 0 to 99, and this was mapped onto the relevant range during decoding. In total there were 177 different parameters encoding for the weights, thresholds and time constants of the neurons in the neural network control architecture. For a more detailed description of the architecture see (Jacobi, 1997). The important point here is that the search space is that of a very large recurrent neural network control architecture for a real[2] mobile robot.

The fitness function is the sum of the performance of the robot in four different scenarios plus a penalty function based on how well the robot keeps its body off the floor. The four scenarios are 1) when no sensor is on move forward, 2) when the left infra-red sensor is on turn right, 3) when the right infra-red sensor is on turn left, and 4) when the front bumper is on move backwards. The fitness landscape thus generated is likely to be full of large neutral networks.

4.2 The Algorithms

The GA from the original work by (Jacobi, 1997) was used as the basis for the comparison of different generation biases as it is an algorithm that has already demonstrated that it can successfully find solutions to the problem. The original selection mechanism was used to compare three different generation mechanisms: the original GA-like generation mechanism with both mutation and cross-over, generation with mutation only and generation with cross-over only.

[2] We can say it is for a real robot as Jacobi has demonstrated that control architectures successfully transfer onto the real robot.

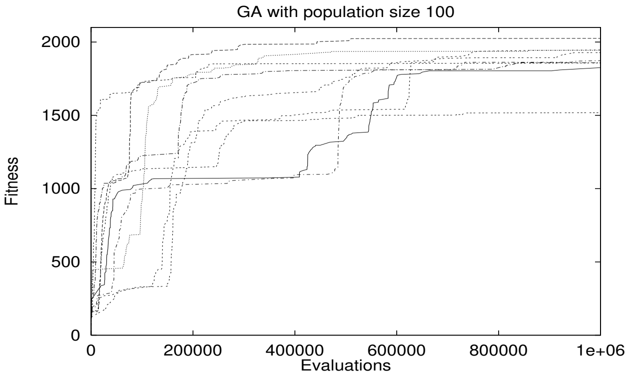

Figure 9: GA with pop size 100

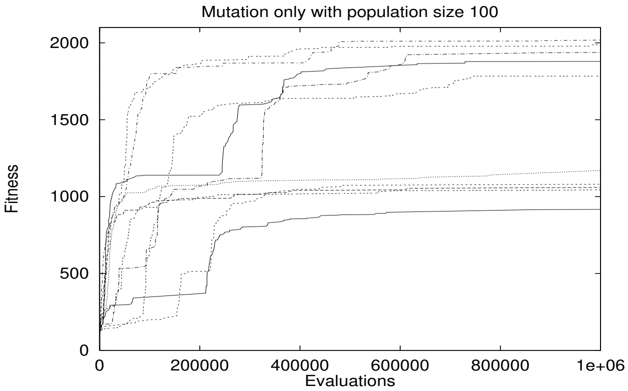

Figure 10: Same as GA but using mutation only

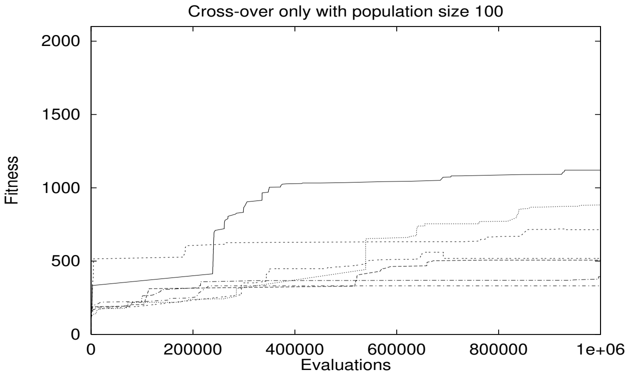

Figure 11: Same as GA but using cross-over only

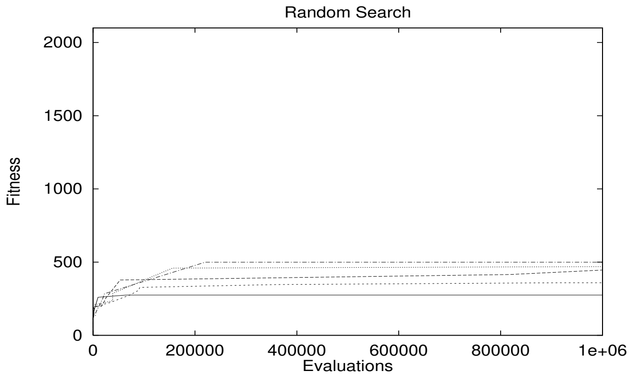

Figure 12: Random search

The comparison of these three should shed light on the relative importance and influence of the two genetic operators mutation and cross-over.

The selection mechanism of the algorithm uses a generational update of a population of 100 individuals. Each new generation is generated from the previous as follows. First, the fittest individual is copied into the new generation. For each of the remaining places in the population two rounds of two player tournaments are held to find the two parents, and then a new individual is generated from these two parents using the generation mechanism of the particular algorithm, i.e. either mutation, cross-over or a combination of both. In the three search algorithms used it is only the generation mechanism that is different. Each member of the new population is evaluated, and the old generation replaced with the new. This is repeated for a over 1,000,000 evaluations, which is 10,000 generations.

From the original GA style generation mechanism with both cross-over and mutation, the mutation generation mechanism is created simply by omitting the cross-over, and therefore only one of the parents is used (there is no extra selection as to which parent is used). To ensure that the cross-over based algorithm does not converge too fast, a degree of non-mutation based variation is added to the population by the following mechanism. Whenever the two parents have identical genotypes, the offspring generated is completely random. This is the same cross-over only generation mechanism as used on the NK landscapes and in the previous paper (Sharpe, 1998). Cross-over itself is a very simple one-point cross-over mechanism as used in the original work on the octopod RWP (Jacobi, 1997).

Finally random search is used as a baseline comparison. An algorithm that cannot do much better than random search on a problem is obviously not doing much useful work.

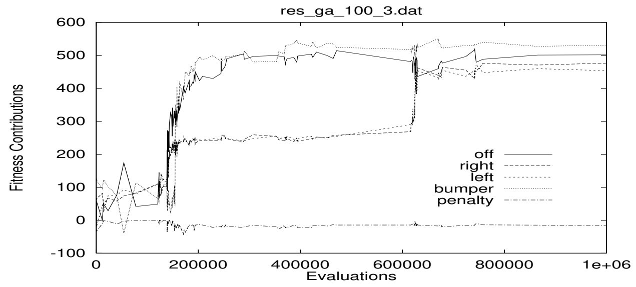

Figure 13: Fitness contributions in a typical GA run

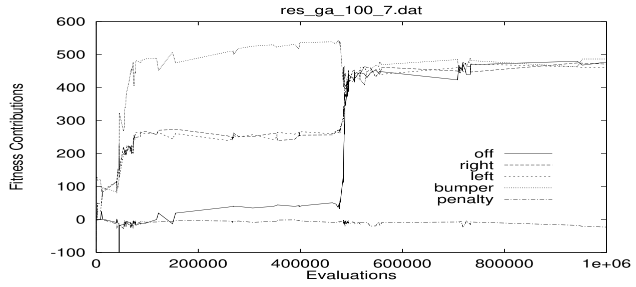

Figure 14: Fitness contributions in a typical GA run

4.3 Results

There are a number of striking features of the results. First of all, across all of the results, it is clear that the fitness landscape is divided into a series of quite distinguishable levels of performance. By splitting the fitness scores of some GA runs into its 5 summed components we can see more clearly where these fitness levels come from (figures 13 to 16). In particular it is clear that scenarios 1) and 4) are either achieved completely or not at all, whereas 2) and 3) not only have a mid level of performance but also their performance is usually the same as each other. The combined fitness level of 1000 is usually achieved when 2) and 3) are at their mid level and either 1) or 4) is completely successful.

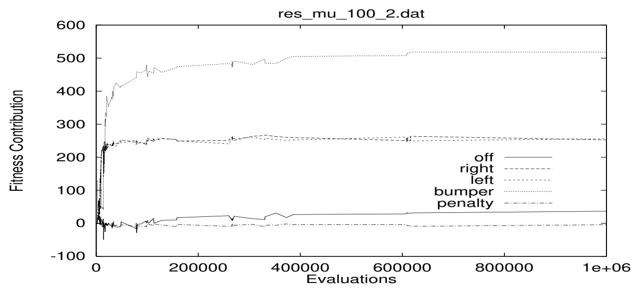

Figure 15: Fitness contributions in a typical mutation only run

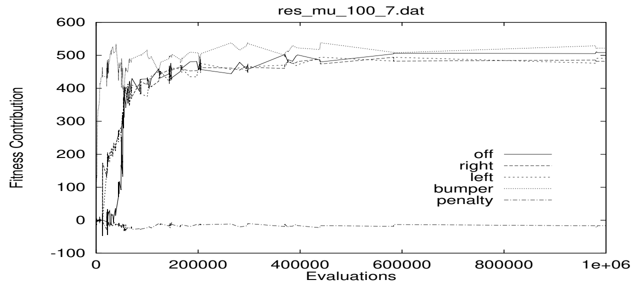

Figure 16: Fitness contributions in a typical mutation only run

It is clear from the graph of performance of the mutation algorithm (figure 10) that this level of fitness is quite easily achieved by the work of mutation alone. In contrast, cross-over alone appears to find this a relatively difficult level of fitness to get to by itself (figure 11). Random search never achieves this level of fitness (figure 12), and hence we can be confident that the cross-over algorithm is doing at least some useful work, as it sometimes reaches a fitness level of 1000 within 1000000 evaluations, but its effectiveness at this stage compared to that of mutation is negligible.

Hence, at this stage of the search (i.e. getting up to a fitness value of 1000), it appears quite clearly that the success of the GA can be attributed largely to the fact that its generation bias includes mutation. This part of the result therefore supports the idea that generation biases are in some way separable. The result also suggests that this kind of large search space for neural network architectures is more amenable to local search biases rather than the global search bias of a cross-over operator.

It is also clear that to get reliably to a higher fitness level than 1000 requires the generation bias of the GA (figure 9). The mutation algorithm achieves the higher levels of fitness on half of its runs, and hence we can know that the mutation bias is capable of finding a path up to the successful solutions, however it appears as if the added help of cross-over becomes useful at this stage of the search. One possible explanation comes from the fact that the assumptions that support the use of mutation also support the use of *local* cross-over (see figure 17). Another view on this comes from looking again at the graphs of the components of fitness functions of different runs. It appears as if the mutation only algorithm is more likely to get stuck at a given performance level of the four scenarios (figure 15), whereas the GA is better able to find a way to combine good performance on all four scenarios (figures 13 and 14).

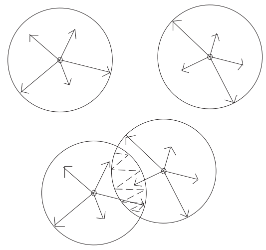

Figure 17: a) The assumption that nearby to good points there will be some better points underpins the success of mutation. b) The same local assumption also suggests that between two close good points there should be a fertile area for better points. Hence local cross-over should also be successful.

This ability to combine different types of behaviours is precisely what people have always believed that cross-over operators would be good at doing. The results here may be good evidence that indeed it can help in this way, thereby allowing the GA to perform better than the mutation only algorithm.

It is worth noting again though that the ability of cross-over to help combine different behaviours was only shown to be useful in this space once the population had already reached a relatively high level of fitness and was fairly converged[3]. Before this the actions of mutation alone can readily explain the performance of the GA. Also, it is not clear whether or not it would be possible for cross-over alone to reach the highest levels of fitness from the mid level. Given that mutation alone can reach the higher levels of fitness, it would appear more likely that cross-over is being useful in a synergistic relationship with mutation at this stage of the search process.

In summary, in terms only of the generation bias, this search space is mutation friendly at lower levels of fitness and then mutation plus one point cross-over friendly at the higher levels of fitness. The use of cross-over at higher levels of fitness is not essential, but useful. The success of the combination of both operators at the higher level suggests that the space may also be friendly to local cross-over on its own at the higher fitness levels, but this is not fully investigated here. Which kinds of selection biases that are effective on this search space, apart from the original one, and indeed which other kinds of generation biases or population sizes, have also not been looked at here.

[3]The population usually converges from its random start point to its variation-selection equilibrium point within the first 100 generations

5 Conclusions

This paper has suggested a way of looking at the success of a given algorithm on a given problem as being derived from the algorithm's constituent parts. The usefulness of this approach has been demonstrated by showing how it gives us insight into why a search algorithm is working at different stages of the search process. This approach also sheds light on the nature of the search space at hand by showing which types of bias do effective work on the space.

The analysis showed how important the mutation component of the GA was for explaining the GAs performance on both the NK landscapes and the RWP. It was also possible to show at which times cross-over was playing an important role in the GAs performance.

It is hoped that in time this style of dissecting apart the useful aspects of search bias on various different problems will enable us to build a corpus of classified problems which, together with some kind of similarity measure, could help us induce an effective search algorithm for each new problem.

Acknowledgements

I would like to thank Rudi Lutz for his helpful comments on this paper and Nick Jacobi for letting me use his original work on the octopod simulation.

Reference

Fukunaga, A. (1998). Restart scheduling for genetic algorithms. In *Proceedings of the Seventh International Conference on Genetic Algorithms 1998*.

Jacobi, N. (1997). Half-baked, ad-hoc and noisy: Minimal simulations for evolutionary robotics. In *Proceedings of the Fourth European Conference on Artificial Life*.

Jacobi, N. (1998). *Minimal Simulations for Evolutionary Robotics*. Ph.D. thesis, School of Cognitive and Computing Sciences, Sussex University.

Kauffman, S. (1993). *Origins of Order*. Oxford University Press, New York.

Langdon, W., & Poli, R. (1998). Why ants are hard. In Koza, J. R. (Ed.), *Genetic Programming 1998*.

Sharpe, O. (1998). Beyond NFL. In Koza, J. R. (Ed.), *Genetic Programming 1998*.

Wolpert, D., & Macready, W. (1996). No free lunch theorems for search. Tech. rep. SFI-TR-95-02-010, The Santa Fe Institute.