Fitness distributions and GA hardness

Yossi Borenstein and Riccardo Poli

Department of Computer Science
University of Essex

Abstract. Considerable research effort has been spent in trying to formulate a good definition of GA-Hardness. Given an instance of a problem, the objective is to estimate the performance of a GA. Despite partial successes current definitions are still unsatisfactory. In this paper we make some steps towards a new, more powerful way of assessing problem difficulty based on the properties of a problem's fitness distribution. We present experimental results that strongly support this idea.

1. Introduction

For over a decade GA researchers have attempted to predict the behavior of a GA in different domains. The goal is to be able to classify problems as hard or easy according to the performance a GA would be expected to have on such problems, accurately and *without actually running the GA*.

The Building Block (BB) hypothesis (Goldberg 1989) states that a GA tries to combine low, highly fit schemata. Following the BB hypothesis the notion of deception (Goldberg 1989, Forrest and Mitchell 1992), isolation (Goldberg 1993) and multimodality (Rana 1998) have been defined. These were able to explain a variety of phenomena. Unfortunately, they didn't succeed in giving a reliable measure of GA-hardness (Grefenstette 1993, Jansen 1999).

Given the connection between GAs and theoretical genetics, some attempts to explain the behavior of GAs were inspired by biology. For example, epistasis variance (Davidor 1991) and epistasis correlation (Naudts 1998) have been defined in order to estimate the hardness of a given real world problem. NK landscapes (Kauffman 1993, Altenberg 1997b) use the same idea (epistasis) in order to create an artificial, arbitrary, landscape with a tunable degree of difficulty. These attempts, too, didn't succeed in giving a full explanation of the behavior of a GA (Naudts and Kallel 1999, Jansen 1999, Guo and Hsu 2003).

Finally, fitness distance correlation (Jones and Forrest 1995) tries to measure the intrinsic hardness of a landscape, independently of the search algorithm. Despite good success, fitness distance correlation is not able to predict performance in some scenarios (Altenberg 1997a).

The partial success of these approaches is not surprising. Several difficulties present themselves when developing a general theory that explains the behavior of a GA and is able to predict how it will perform on different problems. These include:

- A GA is actually a *family* of different algorithms. Given a problem the GA designer first decides which representation (e.g. binary, multiary, permutation, real numbers) to use, then how to map the solution space into the search space, and finally which operator(s) (mutation, crossover) to use. Moreover, there are limited concrete guidelines on how to choose a representation and a genotype-phenotype mapping. Indeed this is a very difficult task. Different genotype-phenotype representations can completely change the difficulty of a problem (Rothlauf 2002). There have been attempts to evolve the right representation (Altenberg 1994) and there are some general design guidelines (Moraglio and Poli 2004, Radcliffe 1991, Rothlauf 2002). However, the reality is that the responsibility of coming up with good ingredients for a GA is still entirely on the GA designer.
- According to (Huttel, 2001) it is impossible to predict the performance of an algorithm based only on the on the *description* of the problem instance and the algorithm without actually running the algorithm.

In the absence of a good, predictive theory of GA performance, unavoidably we are only left with an experimental approach. The idea is to divide the space of real-world problems into GA hard and GA easy by finding (by experimentation and experience) a good GA (with its representation, mapping, and operators) for every specific instance (or class of instances) of a problem.

The No Free Lunch (NFL) theorem (Wolpert and Macready 1997) states that, on average, over all possible problems, the performance of each search algorithms is equal. The basis for this claim is the observation that any prediction based on sampling a sub-space may not be valid. By studying some limitations that general knowledge of a problem might put on this observation, we introduce a new way to classify problems for GAs. In particular we explore assessing problem difficulty based on the properties of a problem's fitness distribution.

In the next sections we introduce our notation, we explain our motivation for the new classification idea, define it and give some empirical results that support it. We conclude with a discussion about possible implications and future work.

2. Definition

2.1 Notation

Following (Liepins and Vose 1990) we use the following notation. A **problem** is a fitness function $f : X \to Y$ that we would like to maximize (where X is the search space and Y is the ordered set of fitness values). This assigns a fitness value to each point in the search space. A **representation** is the way that we choose to represent the search space (i.e. real numbers, binary strings). We define a point in the search space as a **phenotype,** the chosen representation for it as a **genotype,** and the function that transforms the genotype into the phenotype as a **genotype-phenotype mapping**.

If we sampled the search space randomly and we recorded the fitness of the solutions sampled, different fitness values (i.e. different elements of Y) would have potentially different probabilities of occurrence. We define the **fitness distribution** $p(y)$ as the frequency of each particular $y \in Y$ for the problem of interest f. For example, the fitness distribution for a one-max problem is (assuming $X=\{0,1\}^N$):

$$p(y) = \binom{N}{y}.2^{-N}$$

2.2 Problem classification via fitness distribution

We suggest classifying problems according to the properties of their fitness distribution.

Traditionally, a class of problems is defined as the set of all the instances of a problem that share the same description. For example, TSP is a class of problems which share the following description: "Given a set of cities and the distances between them, find the shortest path starting from a particular city, passing through all the others and returning to the first city". Each configuration of cities and distances is an instance of a TSP.

Classifying problems based on descriptions is very useful for humans, for obvious reasons. However, this doesn't work well in a black-box scenario. The key feature of a black-box algorithm is the absence of any explicit information about the problem being solved. The information is embedded *somehow* into the landscape - it is not clear how and to what extent the landscape captures the essential features of the problem. From this point of view, the notion of a problem and a fitness distribution is closely related in the black-box paradigm. So, classifying problems according to their fitness distribution is a very natural thing to do.

2.3 Fitness distribution and search performance

The performance of a search algorithm depends crucially on the fitness landscape and on the way the algorithm searches the landscape. The landscape, in turn, is a combination of a representation (genotype), a syntactic neighborhood (defined over the representation) and a genotype-phenotype mapping. The number and type of different fitness landscapes for a particular problem is dependent on the fitness distribution: for some distributions there may be only very few possible landscapes (e.g. if only one fitness value is possible then there is only one possible landscape), for others there may be many. In any case, *the fitness distribution constrains what is possible*, and so there is a strong connection between the fitness distribution and the expected performance of a search algorithm.

Let us consider the case of the simplest randomized algorithm (or black-box search algorithm): *random search*. Random search doesn't use any knowledge about the landscape, hence doesn't use any information about a problem. The fitness

distribution is the only information needed in order to predict the performance of random search.

For a GA the situation is more complicated. Yet, the fitness distribution may still help predict and explain the performance of a GA. For example, let us consider two well-studied problems: the one-max, which is known to be an easy problem, and the needle-in-the-haystack (NIAH), which is a hard one. The fitness distribution of the NIAH restricts the landscape to be flat everywhere except in the needle. Therefore, irrespective of the genotype-phenotype mapping, the GA doesn't have any information to use. It is expected, therefore, to perform as random search (or worse, due to the negative effects of resampling). On the other hand, for the one-max fitness distribution, there is at least one genotype-phenotype mapping (the standard representation for the problem) that allows the GA to do better than random search by exploiting the information present in the landscape.

3. Empirical Results

At this stage we focus on a specific instance of a GA using only binary representations. We postpone the study of different representations and different types of GA to a later stage of this research.

The NFL doesn't impose any restriction on the possible problems. Using our notion of classification, we make one assumption: we restrict our attention to problems with specific fitness distributions. We make a Monte-Carlo sampling over all the possible genotype-phenotype mappings (representations). This enables us to study common characteristics of all the problems that belong to a specific class.[1]

For our study we will use the fitness distribution given by the one-max and NIAH problems.[2] In all the experiments we use a generational selecto-recombinative GA, with one point crossover. The selection mechanism is a tournament of size 2. We use genotypes of length N=12. The population size was 50, the crossover rate was 100%, and no mutation was used. Runs were stopped when either a solution had been found or the population had fully converged.

We performed 500 experiments. In each experiment we chose, randomly with a uniform probability, a genotype-phenotype mapping. For each mapping we made 1000 runs (500,000 runs in total).

Random search is used as a reference to compare the performance of the GA. It is run for exactly the same number of fitness evaluations as the GA.

3.1 NIAH is GA-easier than one-max

We tested the performance of the GA and random search as both optimization algorithms (by looking at their ability to find the fittest individual in the search space)

[1] Alternatively and equivalently one can view this as a method to study the difficulty of a problem under all possible representations.

[2] It is important to emphasize that the one-max (NIAH) problem and the one-max (NIAH) fitness distribution are two related, but distinct things (see Sect. 2).

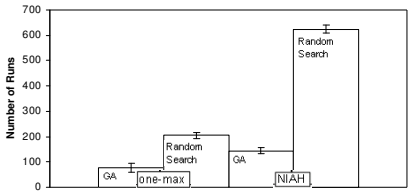

Fig.1 The average number of runs in which the GA and random search found a solution with the one-max and NIAH fitness distributions.

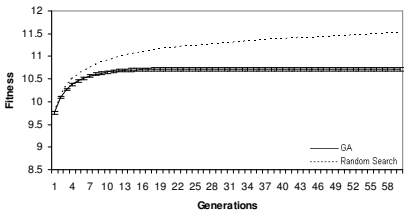

Fig. 2 Average fitness of the fittest individual discovered by each generation

and approximation ones (by looking at the fittest individuals found in each run *by* any given generation).

In order to assess the performance of the GA and random search as optimization algorithms we counted, for each mapping of one-max and NIAH, the numbers of runs in which the algorithms succeeded in finding the fittest solution in the search space. Figure 1 shows the average number of runs in which the GA and random search found the optimum solution. The average is taken over all the random mappings sampled. In this and the following figures the error bars represent the standard deviation. The difference between the two plots for random search is due to the fact that random search was run for exactly the same number of fitness evaluations as the GA and that the GA had different average convergence times with the two fitness distributions.

In order to measure the performance of the GA and random search as approximation algorithms we plotted the fittest individual discovered so far in each run against the generation number.[3] This is shown in Figure 2, which does not include the results for the NIAH fitness distribution because the notions of approximation and optimization coincide in this case.

These results are clear: firstly, random search outperforms the GA, and, secondly, rather surprisingly, the GA performs better on the NIAH fitness distribution than on the one-max fitness distribution.

The reason for the superiority of random search is that the GA resamples the same points in the search space more often than random search. This is illustrated in Figure 3, which shows the fitness distribution of the *distinct* solutions that were sampled by both algorithms for the one-max fitness distribution. The area beneath each curve represents the total number of distinct points in the search space that were sampled. Clearly, the GA has sampled a much smaller area. Moreover, the quality of the solutions that were sampled by the GA is no better than the quality of the solutions sampled by random search.

[3] For random search a generation is taken to mean as many samples as the GA's population size. This definition allows a direct comparison with the GA.

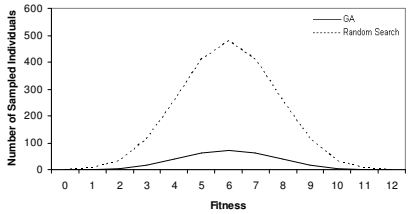

Fig.3 The histogram of the distinct points in the search space that were sampled on average during a run for the one-max fitness distribution.

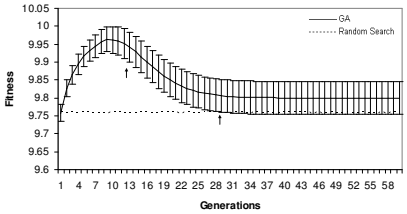

Fig. 4 The average (over all representations) fitness of the best individual found in each generation for the one-max fitness distribution.

3.2 Three stages in the GA search

In order to understand better the dynamics exhibited by the GA, in Figure 4, we plot for each generation the fitness of the best individual found *in that generation* for the one-max fitness distribution. The results are averaged over all the representations.

In contrast to what we observed in the previous section the GA *appears* to outperform random search. This is not a contradiction. The flat line representing the random search algorithm in Figure 4 is the result of averaging. With random search the best solutions found in a run can be discovered at any generation, unlike a GA where typically the best solutions emerge towards the end of a run. Averaging fitness of the best individual found in each generation over multiple runs hides this effect. The GA, on the other hand, as already noted, is more consistent as to when the best solutions emerge. So, it doesn't suffer from any hiding effect related to averaging.

The performance of the GA rapidly improves in the first few generations and then gradually worsens until it reaches a plateau (where the GA is still superior to random search). Effectively for the one-max fitness distribution *the search of the GA can be divided into three stages*: in the first one (before generation 12) the GA keeps finding new fit individuals, in the second (from generations 13 to 30) the performance drops, while in the third the population has converged. In the following sections we analyze these stages more carefully.

3.3 Stage I: implicit elitism

In order to check whether the GA indeed initially finds good regions of the search space, we plot the best individual found in each generation that *has not been sampled in the same run before*. This is shown on Figure 5. Like Figure 4, this plot shows three stages (with end/start points highlighted by the arrows). However, it is now clear that in the first stage the GA doesn't discover new fit individuals: the GA, to some extent, resamples the same fit individuals from previous generations. So, the GA seems to have an *implicit form of elitism*. It is important to emphasize that the negative slope shown in the plot of Figure 5 doesn't mean that the fittest individuals

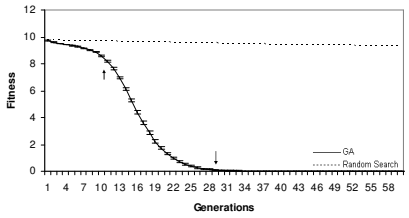

Fig.5 The average fitness of the best individuals found in each generation that haven't been sampled before for the one-max fitness distribution

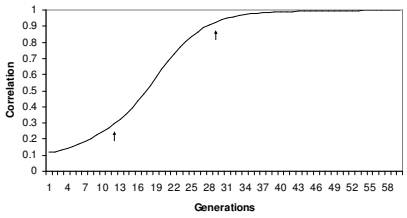

Fig.6 The correlation between the fitness of the parents and the fitness of the offspring for the one-max fitness distribution

are found in the first generation. This is another averaging effect: in each generation the probability to resample the same individuals increases. Thus, the number of new discovered individuals decreases. Since we sample, each time, a smaller portion of the search space, *on average*, as the run continues, the fitness of the best solution found gets lower. This, in turn, explains the negative slope. Random search is less prone to resampling, which explains the difference between random search and the GA.

4.3 Stage II: Evolvability

In order to explain the second phase of the GA's search, we consider the evolvability of the population. Plotting the parents-offspring fitness correlation (Fig. 6) shows that although the fitness of the best individual found in each generation decreases (see Figure 4, generations 13-30) the overall evolvability as measured by the parent/offspring fitness correlation of the population increases. So, *the GA is trading fitness for evolvability*. This makes sense: when it becomes hard to increase the population fitness by discovering enough highly fit individuals, the best option is to go for areas of the landscape where the fitness of the offspring is as close as possible to the fitness of their (above average) parents.

3.4 Stage III: Convergence

Since we use no mutation in our experiments, selection and drift eventually lead the GA to converge to a population containing only copies of the same individual. This corresponds to the third phase of the GA's search shown in the previous figures.

4. Discussion

In the experiments we evaluated the performance of a GA when the representation chosen was randomly picked from the set of all possible representations. At first this procedure may seem odd, but, as we will argue below, it is not.

Firstly, one might object, in everyday practice representations are never random: they are carefully chosen by the user/designer of a GA so as to guarantee good

performance. By and large this is true. However, in the presence of a new problem, for the fitness function of which no information is available, except for the information one can gather by evaluating the fitness of specific points in the search space, the designer has very little to work with to ensure the representation chosen will be a good one. In this case, one should assume that the quality of the representation chosen will be average. Finding out what exactly being "average" means for a representation is the objective of the experiments we reported above, which should clarify the value of our approach.

Secondly, one might object, if the representation is picked randomly, surely the resulting fitness landscape will include no information on the problem it originated from, and so, following NFL, one should expect equal performance in all conditions — the fact that we did not get equal performance would then suggest something was wrong in our experiments. The fallacy in this argument is assuming that no information is present in the landscape: the fitness distribution associated to a problem may induce very strong regularities in the fitness landscapes induced by the representation, even for random representations. So, it is entirely possible that on the *typical* (random) landscape induced by a certain fitness distribution a particular GA will do better than on the typical landscape induced by another fitness distribution (which is exactly what happened in our experiments with one-max and NIAH). It is therefore very important to understand what the regularities imposed by fitness distributions are and whether certain types of regularities lead to good performance (GA easiness) while others lead to bad performance (GA hardness).

Our hypothesis is that fitness distributions capture important features that make a problem GA easy or GA hard, and that grouping problems by their fitness distribution is, therefore, a very meaningful and useful way to create a *high-level* GA-hardness problem taxonomy.[4]

Let us look back at our data to try and identify the fitness-distribution features that were responsible for the (unexpected) difficulties the GA hit when solving one-max-type problems. Based on our results, we can see that the GA search initially focuses on highly fit individuals. However, in the longer run, regions that contain fit, evolvable individuals are the real targets of the GA. This in turn can explain the difference in performance just mentioned.

The one-max fitness distribution spontaneously creates many evolvable, relatively fit regions. This is because, although there is only one string with maximum fitness (N), there are many strings of high fitness (e.g. there are N strings with fitness $N-1$, $N(N-1)/2$ strings with fitness $N-2$, and so on). Since the representation is random, in most cases a sub-optimal string of fitness $N-1$ will actually be an isolated local optimum. Similarly strings of fitness $N-2$ can generate many other local optima. The landscape around these local optima is also random, and so we should not expect to see any substantial regularity. However, since there are so many local optima, some of these will have more regular neighborhoods (basins of attraction) than the others. These represent highly attractive areas for the GA since individuals in such areas are both fit and evolvable. So, very likely the GA will zoom onto one such area. Once the GA zooms into a particular region of the search space it searches for new individuals only within that region. Since the landscape is random the probability that the highest

[4] A complete taxonomy should consider the neighborhood structure (representation) as well.

fitness solution is indeed within that region is quite low. So, in the one-max case, the relatively large number of high fitness regions causes the GA to converge prematurely to one of them, rather then continue searching for the optimum, effectively making (typical) problems from the one-max fitness distribution deceptive and GA hard.

In the case of the NIAH fitness distribution, the fitness landscape is flat everywhere, except at the optimum. So, there cannot be any deceptive attractors for the GA irrespective of the representation chosen. Indeed, in our experiments the GA did not converge prematurely to such an attractor. It sampled more points in the search space and therefore performed better on the NIAH fitness distribution.

Random search is insensitive to the fitness distribution (although, naturally, it is sensitive to the frequency of the highest fitness solutions). Also, random search was observed to be less prone to resampling, and so, within the same number of fitness evaluations, it covered a larger portion of the search space. This explains why it performed better than the GA.

5. Conclusions

Different attempts to come up with a good classification for problems in relation to the performance of search algorithms, particularly GAs, have been reported in the literature. Roughly speaking, most of them firstly postulate what feature makes a problem hard for a GA, then check whether a specific problem has this feature, and finally infer the degree of GA hardness of the problem from the extent to which such a feature is present. To the best of our knowledge, this approach has so far only achieved partial success in creating meaningful and reliable problem taxonomy.

In this paper we have proposed a new way of classifying problems: grouping them on the basis of the features of their fitness distribution. As we have argued in Section 4 and empirically shown in Section 3, there are features of certain fitness distributions that induce clear regularities in the fitness landscapes of typical problems with those distributions. For example, the availability of intermediate fitness values (like in the one-max fitness distribution) can lead to landscapes (problems) with a huge number of deceptive basins of attraction, which are therefore, on average, GA hard.

Naturally, we are fully aware that more often than not the fitness distribution for a problem is not available. In these cases, the fitness distribution can be estimated. Rose, Ebeling and Asselmeyer (1996) suggest the Boltzmann ensemble method as an efficient way to obtain the fitness distribution. Naudts and Landrieu (2000) present a normalization technique which enables the grouping of distributions with different fitness range. The features of the estimated distribution could then be used to infer whether the problem is expected to be GA easy or hard, without actually having to run the GA. The distribution itself (Naudts and Landrieu 2000) might not be sufficient to predict the difficulty of the problem. Still, grouping problems with similar fitness distributions is meaningful. This approach could be used to check whether a particular choice of representation is reasonable. If it is not, it could suggest how to refine it (using knowledge extracted from other problems of the same group).

Our hope is that, over time, we will be able to identify which classes of representations generally work well with which types of fitness-distribution features.

If this could be achieved, then the design of competent GAs would be much facilitated: one would only need to check in which fitness-distribution class a problem falls to immediately identify which representation would put the GA in good operation conditions.

References

L.Altenberg. Fitness Distance Correlation analysis: An instructive counter-example. ICGA, pp 57-64, 1997a.

L.Altenberg. NK fitness landscapes. Handbook of Evolutionary Computation 1997b.

L.Altenberg. Evolving better representations through selective genome growth. Proceedings of the 1st IEEE Conference on Evolutionary Computation. Part 1, 1994.

Y.Davidor. Epistasis variance: A viewpoint on GA-hardness. In G.J.E Rawlines, editor FOGA-1,pp. 23-55. Morgan Kaufman 1991.

S.Forrest and M.Mitchell. Relative Building Block Fitness and the Building Block Hypothesis. In D. Whitley (ed.) Foundations of Genetic Algorithms 2. Morgan Kaufmann, San Mateo, CA, 1992.

D. E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Morgan Kaufmann, 1989.

D. E. Goldberg. Making genetic algorithm fly: a lesson from the Wright brothers. Advanced Technology For Developers, 2 pp.1-8, February 1993.

J.J.Grefenstette. Deception considered harmful. In Foundations of Genetic Algorithms 2, D. Whitley (Ed.), San Mateo, CA: Morgan Kaufmann, 1993.

H. Guo and W.H.Hsu. GA-hardness Revisited. GECCO 2003 Poster paper.

H. Huttel. On Rice's Theorem. Aalborg University. 2001.

T. Jansen. On Classifications of Fitness Functions. Reihe CI 76/99, SFB 531, Universität Dortmund, 1999.

T.Jones and S.Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In Larry Eshelman, editor, Proceedings of the Sixth International Conference on Genetic Algorithms, pages 184-192, San Francisco, CA, 1995.

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

G. E. Liepins and M. D. Vose. Representational issues in genetic optimization. Journal of Experimental and Theoretical Artificial Intelligence, 2:101-115, 1990

A.Moraglio and R.Poli. Topological interpretation of crossover. GECCO 2004.

B.Naudts..Measuring GA-hardness. Ph.d thesis. UV. Of Antwerp. Belgium, 1998.

B.Naudts and L.Kallel, Comparison of summery statistics of fitness landscapes, IEEE Trans.Evol.Comp. 2000.

B.Naudts and I.Landrieu, Comparing population mean curves. FOGA 2000

S. Rana. Examining the Role of Local Optima and Schema Processing in Genetic Search. PhD thesis, Colorado State University, 1998.

N.J.Radcliffe. Equivalence class analysis of genetic algorithms. In Foundations of Genetic Algorithms 3. Morgan Kaufmann, 1994.

H.Rose,W.Ebeling and T.Asselmeyer. The Density of States – a Measure of the Difficulty of Optimization Problems. PPSN 1996.

F.Rothlauf. Representations for Genetic and Evolutionary Algorithms. Studies in Fuzziness and Soft Computing, Volume 104. Heidelberg: Springer, 1st edition 2002.

D.H..Wolpert and W.G. Macready. No free lunch theorems for optimization.IEEE Transactions on Evolutionary Computation, 4:67-82, 1997.