Fitness Distance Correlation and Problem Difficulty for Genetic Programming

Manuel Clergue
I3S Laboratory,
University of Nice,
Sophia Antipolis, France

Philippe Collard
I3S Laboratory,
University of Nice,
Sophia Antipolis, France

Marco Tomassini
Computer Science Inst.,
University of Lausanne,
Lausanne, Switzerland

Leonardo Vanneschi
Computer Science Inst.,
University of Lausanne,
Lausanne, Switzerland

Abstract

This work is a first step in the attempt to verify whether (and in which cases) fitness distance correlation can be a good tool for classifying problems on the basis of their difficulty for genetic programming. By analogy with the studies that have already been done on genetic algorithms, we define some notions of distance between genotypes. Then we choose one of these distances to calculate the fitness distance correlation coefficient and we use it to study the difficulty of some problems. First, we do this for a syntactically limited language. Then we extend the study to standard genetic programming. For the functions used here i.e., traps and royal trees, the results confirm that fitness distance correlation is a good predictor of genetic programming difficulty.

1 INTRODUCTION

Previous studies of problem difficulty in Genetic Algorithms (GAs) [6] have shown the importance of the notion of *distance* between genotypes as a tool to measure the difficulty of problems. The typical distance definition for GAs is the Hamming distance, while other distances like alternation [3] have also been used. Here we present for the first time a study for GP difficulty in analogy with GA research. For this reason, in the following, we offer some definitions of distance between individuals for GP. It should be noted that fitness landscape structure depends on the operators used to traverse it [6]. In the present work, the standard GP crossover [10] is used as the sole genetic operator.

The usual approach to problem difficulty in GP has been the use of a more or less agreed upon set of test problems that have their origin in Koza's work [10] such as the even-n parity problem, the multiplexer, symbolic regression and artificial ant. However, this point of view, while useful for practical benchmarking purposes, lacks generality since results are problem-dependent and it is difficult to infer more general issues relating to intrinsic GP difficulty by just looking at statistics derived from these problems.

There have been few attempts to date to characterize GP difficulty by means of a single measure. One early approach was proposed by Koza [10] and consists in calculating, for a given problem, the number of individuals that must be processed in order for a solution to be found with a given probability p (usually $p = 99\%$). This gives a number characterizing the required computational effort but it cannot be relied upon for distinguishing easy from hard in GP.

Another potentially more fruitful approach, has been to transfer to GP some of the considerations that have proven useful for studying GA difficulty. Thus, *synthetic* or *constructive* problems inspired from GA work have been devised in order to probe GP difficulty. One early example is the work of Kinnear [9] in which GP difficulty was related to the shape of the fitness landscape and analysed through the use of *correlation length*, a measure first proposed by Weinberger [16] and later used in genetic algorithms work by Manderick *et al.* [11]. Kinnear's results for GP, however, were difficult to interpret: essentially no simple relationship was found between correlation length values and GP hardness. In the same vein, Punch and coworkers [13, 14] proposed a new synthetic benchmark problem of tunable difficulty, the *Royal Tree problem*, which was inspired by the well-known *Royal Road problem* used in GA theory [12]. However, Royal Trees were used to test the effectiveness of multipopulation as compared to standard single population GP and not, to our knowledge, to gauge intrinsic GP difficulty. Even though we know that there are counterexamples to the use of fitness distance correlation (*fdc*) as a tool for problem difficulty in GA [1], [15], we are also stimulated by the success of *fdc* on a large number of GA functions (see for example [3]) and by the lack of *fdc* studies in GP.

We should also mention a more recent attempt to quantify GP problem difficulty by Daida *et al.* [4]. Their approach is based on the exhaustive study of the dynamics of a single problem of the symbolic regression type, the binomial-3 problem. The binomial-3 problem is tunable thanks to the existence of a range of ephemeral random constants. This approach is interesting but it is different from ours. Whereas they try to fully understand the whole range of behaviors of this particular problem depending on the parameter interval chosen, our goal is to characterize the GP difficulty of whole classes of functions. In our opinion, the two approaches are obviously compatible and both should be pursued.

This paper is structured as follows. In section 2 we define a language that will be used at first for constructing restricted genotypes, in section 3 we define different distances between these genotypes, in section 4 we compare these distances for a particular set of functions (the trap functions) and in section 5 we use one of these distances to evaluate the problem difficulty. In section 6, we generalize the results to genotypes without syntactical restrictions. In section 7 we extend the study to Royal Trees and in section 8 we give our conclusions.

2 A RESTRICTED LANGUAGE

We have decided to study the dynamics of GP by defining a syntax for the individuals and by artificially assigning a conventional fitness value to each point in the resulting fitness landscape. For simplicity, our first step is based on an extremely simple and constrained GP model. However, we will remove this assumption in section 6 where unrestricted GP trees are used. We decided to use function and terminal sets inspired by those proposed by Punch *et al.* [13]. Thus, we will consider a set of functions A, B, C, etc. with increasing arity (an A function has arity 1, a B function has arity 2, and so on) and a single terminal X as follows: $\mathcal{F} = \{A, B, C, D, ...\}$, $\mathcal{T} = \{X\}$.

Moreover, we observe that for GAs genomes have a finite and fixed size, so the number of possible individuals in the search space is finite and numerable. For GP, this is not the case, which makes the dynamics of GP more difficult to study. In order to avoid this kind of problem, we have decided, as a first step, to limit the structures of the possible individuals of our search space. In particular, we have forbidden individuals to have as a child in the tree structure a node representing a function with an arity greater or equal to the arity of the function represented by the father. See figure 1 for some examples of legal and illegal trees.

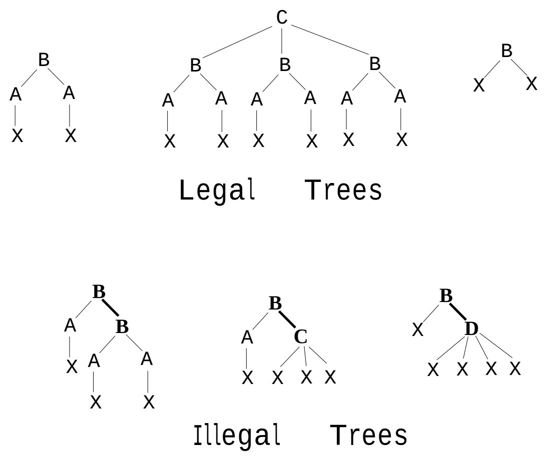

Figure 1: The three trees in the upper part are legal because every node has as son a node with a smaller arity. The other trees are illegal.

This syntactic restriction allows us to automatically limit the depth of the trees. On the other hand, this limitation obliges us to define a crossover operator such that no illegal tree is generated. This definition obviously limits what GP can do, but it is useful as a first approximation. From now on, we will refer to GP with the restrictions mentioned above as *limited GP* and to GP without the syntactical constraints as *standard GP*.

In our problems, we will consider only one individual in the search space to be the global optimum. This tree will be the perfectly balanced tree having as root the node with the maximum arity between the allowed nodes and having the maximum depth. The first tree in the upper left part of figure 1 shows the optimal tree for the set of functions $\{A, B\}$ while the second tree is the optimal one if we also insert node C in the function set. Note that this tree has C as root and three optimum trees of root B as subtrees. By the same argument, we can deduce the form of the optimum trees of any other root.

3 DISTANCE DEFINITIONS

Defining a distance between genotypes in GP is much more difficult than for GAs, given the tree structure of genotypes. For such a definition of distance for tree structures, see for instance [8]. In the following, we will give some new definitions of distance for our restricted language, starting from a preliminary *pseudo-distance*. In the next sections, we will compare these distances and we will choose one with the aim of measuring the difficulty of problems for GP.

3.1 THE FIRST STEP: d_1

Let T_1 and T_2 be two trees. Then, we define

$$d_1(T_1, T_2) = |weight(T_1) - weight(T_2)|$$

where:

$\forall T \; weight(T) = 1 \cdot n_X(T) + 2 \cdot n_A(T) +$

$3 \cdot n_B(T) + 4 \cdot n_C(T) + ...$

and: $n_X(T)$ is the number of symbols X in the tree T, $n_A(T)$ is the number of symbols A in the tree T, and so on.

3.1.1 Problems of d_1

Figure 2 shows a set of trees with their weights, defined as in 3.1.

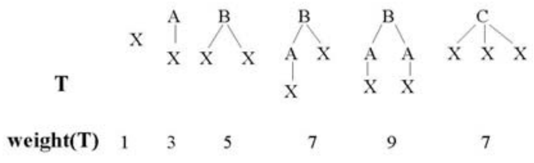

Figure 2: Some examples of legal trees with their weights, according to the definition of paragraph 3.1.

From figure 2, we can see that two trees can have a distance = 0 between each other, without being the same individual, and also having quite different structures, as is the case for trees number four and six, counting from the left in figure 2. For this reason, we called d_1 a *pseudo-distance*. The next definition, in a sense, extends d_1.

3.2 d_2: SYMBOLS WITH ROOT PRIZES

To overcome the problem presented in 3.1, we define a new concept of distance as follows:

- Each tree with root i must have a greater weight than all the trees with root j, if and only if $j < i$, where:
  - $i, j \in \{X, A, B, C, D, ...\}$
  - we consider an order such that: $X < A < B < C < D < ...$

- Since the simpler trees have a difference = 2 in the weights between each other, we give a prize to each root, such that the lighter tree of root $succ(i)$ has the weight of the heaviest tree of root i plus 2, where we impose that: $..., C = succ(B), B = succ(A), A = succ(X)$

Example

Let T_1 be the fifth tree of figure 2 and T_2 be the sixth. T_1, which is the heaviest tree of root B has a weight = 9. If we don't give any prize to the root, T_2 has root = 7. Since we want T_2 to have a weight = 11, we have to give a prize of 4 to all the trees with root C. Iterating this reasoning, we get that the prize for trees with root D must be = 28, the prize for trees with root E must be = 148, the prize for trees with root F must be = 788, the prize for trees with root G must be = 4688, and so on. With this new definition of distance, all the different trees have different weights, and so they have positive distances between each other.

Thus, the formal definition of d_2 can be given as follows: Let T_1 and T_2 be two trees. Then, we define

$$d_2(T_1, T_2) = |weight(T_1) - weight(T_2)|$$

where:

$\forall T \; weight(T) = 1 \cdot n_X(T) + 2 \cdot n_A(T) +$

$3 \cdot n_B(T) + 4 \cdot n_C(T) + ... + prize(T)$

Even though d_2 is more similar than d_1 to our idea of distance, it has a problem too: two trees that are symmetrical about the vertical have the same weights and thus the distance between them is 0, even though they are not the same tree. This problem is overcome by d_3 defined in the next section.

3.3 d_3: RECURSIVE DISTANCE

This distance has been defined with the idea of extending it to the case of general trees (i.e. trees without the limitations exposed in section 2). According to this definition, the distance of a tree T_1 from a tree T_2 can be calculated by the following formula:

$$d_3(T_1,T_2,k) = \begin{cases} k + |td(T_1) - td(T_2)| \\ \quad \text{if } root(T_1) \neq root(T_2) \\ \\ 0 \\ \quad \text{if } td(T_1) = td(T_2) = 0 \\ \\ \displaystyle\sum_{i=1}^{n(T_1)} \frac{d_3(s_i(T_1), s_i(T_2), k-1)}{n(T_1)} \\ \quad \text{otherwise} \end{cases}$$

where:

- $root(T)$ is a function that returns the root of the tree T.
- $td(T)$ is a function that returns the depth of the tree T, where the depth of the tree containing only the node X is considered to be $= 0$.
- $n(T)$ is a function that returns the number of sons of the root of the tree T (i.e. if the root of T is A it returns 1, if it is B it returns 2 and so on).
- $s_i(T)$ is the i^{th} subtree of the root of tree T.
- k is a constant that we use in order for two trees with different roots but the same depth to have a positive distance. It is useful to have smaller values of k for the nested recursive cases, so we insert k between the parameters of function d, and we pass $k-1$ to the recursive calls. The choice of the value of k is arbitrary, with the only restriction that it has to never become negative with the recursive calls, so it must be at least as large as the maximum depth allowed for the trees.

In the above formula, the upper limit of the sum is equal to the number of sons of the root of the tree T_1. We note that this quantity can be substituted by the number of sons of the root of T_2 without changing the result, since roots T_1 and T_2 have the same number of son nodes. In fact, recursion is applied only when T_1 and T_2 have the same root.

4 COMPARISON OF DISTANCES WITH TRAP FUNCTIONS

A good way to test the distances is using the *trap functions* [5]. Trap functions allow to define the fitness of the individuals as a function of their distance from the optimum, and the difficulty of trap functions can be changed simply by modifying two parameters. A function $f : distance \rightarrow fitness$ is a trap function if it is defined in the following way:

$$f(d) = \begin{cases} 1 - \dfrac{d}{B} & \text{if } d \leq B \\ \\ \dfrac{R \cdot (d - B)}{1 - B} & \text{elsewhere} \end{cases}$$

where d is the distance of the current individual from the global optimum, and B and R are constants $\in [0, 1]$. B allows to set the width of the attractive basin for each optimum and R sets their relative importance. Figure 3 depicts a trap function with $B = 0.2$ and $R = 0.8$, where we also see that there is a second local optimum at a maximum distance from the global one. The difficulty of trap functions decreases as the value

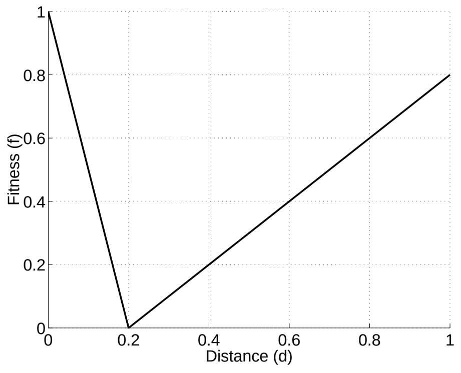

Figure 3: The graphic of a trap function with $B = 0.2$ and $R = 0.8$. Note that distances and fitnesses are normalized into the interval $[0, 1]$.

of B increases, while it increases as the value of R decreases. By keeping R constant and changing B, we are able to define a set S of trap functions of different difficulties. What we expect from a "good" distance is that GP is able to find the global optimum for the so-defined easy trap functions and not for the difficult ones. To measure the success rate to the global optimum for a function, we define a measure called *performance* (p), defined as the number of executions for which the global optimum has been found in less than 200 generations, divided by the total number of executions (100 in our experiments). Then, we perform a series of experiments on the set S and we choose the distance for which we get the best performance for the trap-easy functions to be used in the remaining of this work. All our experiments have been done with the following parameters: population size = 200, tournament selection with size = 10, crossover probability = 95%, mutation probability = 0%, maximum node allowed = F, and two different curves have been drawn for each series of experiments, respectively with $R = 0.8$ and $R = 0.2$. Results are shown in figures 4 and 5. In figure 4 we can see that for d_2 we have always a performance $= 0$ except for the function with $R = 0.2$ and $B = 0.9$, for the function with $R = 0.2$ and $B = 1$, and for the function with $R = 0.8$ and $B = 1$. Of these three functions, the two with $B = 1$ reach a performance of 1. Figure 5 shows the same kind of results for d_3, for which a larger number of functions have a positive performance. Moreover, even if this result is not shown here, we observe that solutions, when using d_3 as distance, are found in a lower number of generations than with d_2. As a further confirmation of our results, in table 1, we report some other performance results calculated on another set of trap functions. From the table and from the figures it appears that d_3 is the one that shows the best performances for the highest number of easy trap functions, and therefore it will be used it in the following.

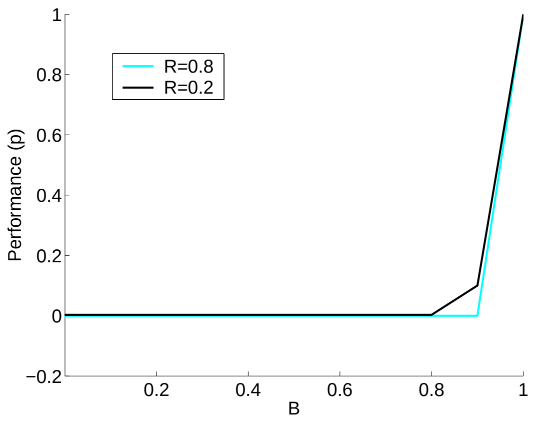

Figure 4: Curve of the performance as a function of the constant B, with $R = 0.2$ and $R = 0.8$, for distance 2.

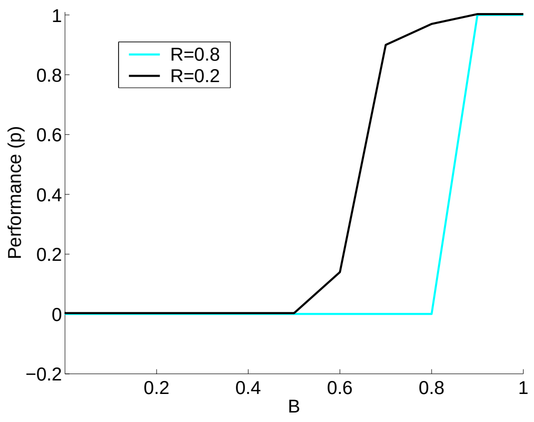

Figure 5: Curve of the performance as a function of the constant B, with $R = 0.2$ and $R = 0.8$, for distance 3.

p	**dist 1**	**dist 2**	**dist 3**
B = 0.2, R =0.8	0	0	0
B = 0.3,R = 0.7	0	0	0
B = 0.4, R = 0.6	0	0	0
B = 0.5, R = 0.5	0	0	0
B = 0.6, R = 0.4	0	0	0
B = 0.7, R = 0.3	0	0	0.64
B = 0.8, R = 0.2	0	0	0.97
B = 0.9, R = 0.1	0	0.9	1
B = 1.0, R = 0.0	1	1	1

Table 1: Some values of the performance (p) for some trap functions calculated with the distances defined in the text.

5 FITNESS DISTANCE CORRELATION

We now describe a heuristic that should allow us to measure the difficulty of problems to be solved with GP. An approach that has been proposed [7] states that what makes a problem hard for GAs is the relationship between fitness and the distance of the genotypes from the optimum. The easiest way to measure the extent to which the fitness function values are correlated with distance to a global optimum is to examine a problem with known optima, take *a sample* of individuals and compute the *fitness distance correlation* (*fdc*), given the set of (fitness, distance) pairs. By the way, correlation is only one of the possible ways of studying the relationship between fitness and distance. In some cases, *fdc* reveals itself to be a poor summary statistic. In such situations, examining the scatter plots of fitness versus distance to the optimum is more useful. Some works on GAs confirm previous results about the importance of a good *fdc*, and exemplify the existence of non-artificial problem, such as TSP, exhibiting this property. According to Altenberg [1], the fact that the *fdc* is only a statistical and static measure, based on a distance which is apparently only bound to mutation (like Hamming distance in GAs), implies two assumptions: either Hamming distance is connected to the way genetic algorithms work, or this relation exists in a fortuitous way among the test set chosen by Jones [7, 6]. In fact, counterexamples have been found for which this relation does not hold, and which, therefore, deceive the *fdc*. Since there seems to be no relation between re-combination operators and Hamming distance, and that mutation is supposed to play a marginal role in GAs, Altenberg claims that it is possible to construct a counterexample. The counterexample he constructs is GA-easy, but the correlation between distance and fitness to optimum is null by construction. Furthermore, the observation of the scatter plot gives no more information. This counterexample deceives Jones' conjecture which claims that if the *fdc* is close to 0 and if the scatter plot exhibits no particular structure, then the problem is GA-difficult. Moreover, Quick *et al.* [15] construct a class of problems, called ridge functions, which are GA-easy with a high positive correlation. While the Altenberg's counterexample is prone to discussion, in particular on the definition of the GA-easiness, the counterexample of Quick *et al.* is clear: there are functions that the *fdc* predicts misleading and which are in fact easy. Nevertheless, these two counterexamples exploit known weaknesses of the *fdc*: its nullity for the symmetrical functions and the low contribution of a particular path in the global calculation. Besides, Quick *et al.* recognize that the *fdc* calculated with the points actually sampled by the GA gives better results. Nevertheless, the success of the *fdc* on a large number of GA functions remains an unsolved question. Collard *et al.* [2] bring some elements of response, exhibiting a correlation between Hamming distance and instability implied by crossover.

Formally, given a set $F = \{f_1, f_2, ..., f_n\}$ of n individual fitnesses and a corresponding set $D = \{d_1, d_2, ..., d_n\}$ of the n distances to the nearest global optimum, *fdc* is defined as: $fdc = \frac{C_{FD}}{\sigma_F \sigma_D}$, where

$$C_{FD} = \frac{1}{n}\sum_{i=1}^{n}(f_i - \overline{f})(d_i - \overline{d})$$

is the covariance of F and D and σ_F, σ_D, $\overline{f}$ and $\overline{d}$ are the standard deviations and means of F and D. As we hope that fitness increases as distance to a global optimum decreases, we expect that with an ideal fitness function *fdc* will assume the value of -1. According to Jones [7], GA problems can be classified in three classes , depending on the value of the coefficient *fdc*:

- **Misleading** ($fdc \geq 0.15$), in which fitness increases with distance.
- **Difficult** ($-0.15 < fdc < 0.15$) in which there is virtually no correlation between fitness and distance.
- **Straightforward** ($fdc \leq -0.15$) in which fitness increases as the global optimum approaches.

The second class corresponds to problems for which the difficulty can't be estimated, because the coefficient *fdc* doesn't bring any information.

Our goal is to check if the same properties are also valid for GP on trap functions using distance d_3. For this reason, we have calculated the performance p and the coefficient *fdc* for various trap functions changing the values of the constants B and R. In these experiments, *fdc* has been computed via a sample of 4000 randomly chosen individuals. Figures 6 and 7 show the results of these experiments in a tridimensional space.

These results show that for GP and with distance d_3, we obtain approximately the same ranges as for GAs with unitation. In particular, from this figures, we can see that the performance is around 1 when the correlation is around -1, the performance is around 0 when the correlation is around 1 and the performance is comprised between 0 and 1 when the correlation is comprised between -1 and 1. These are exactly the results we were expecting.

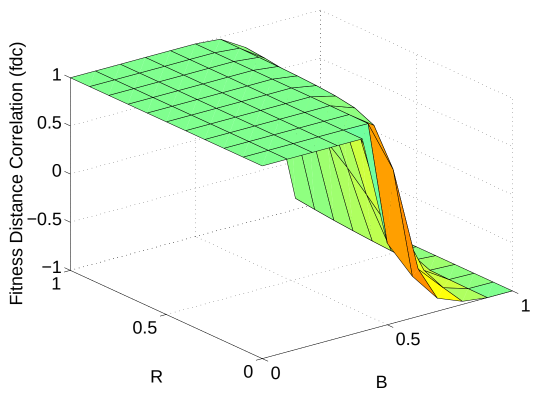

Figure 6: The values of the correlation *fdc* for the trap functions.

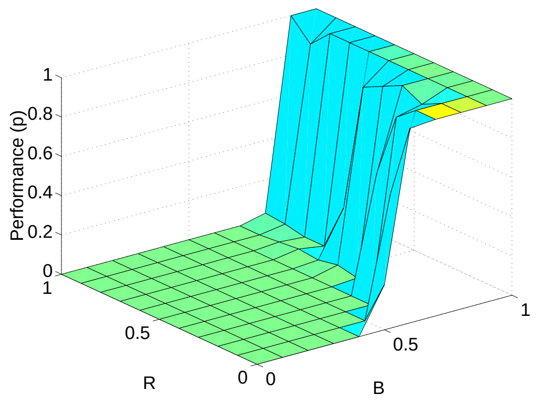

Figure 7: The values of the performance p for the trap functions.

6 EXTENSION TO STANDARD GP

The constraints we have imposed until now allow the creation and the survival of a restricted set of individuals (see section 2). Now we want to release this restriction, with the only obvious limitation of a maximum depth constraint. Thus, we consider the same sets of functions $\mathcal{F} = \{A, B, C, D, ...\}$ and terminals $\mathcal{T} = \{X\}$ as before but we allow the creation of any buildable tree with these symbols and with a maximum depth of 17. As before, we impose only one tree to be the optimum and, as a first step, we consider the same tree that was used for limited GP (see 6.1).

As a measure of the distance between trees, we have decided to use distance d_3 for two main reasons: it works with standard GP, and it is the distance that gives the highest performances with limited GP. Some experiments not reported here have shown that, if we consider GP without syntactical limitations, and we consider F as the function with the maximum arity, the convergence for trap functions is rather slow. In fact, no global optimum for a trap function was ever found before generation 600. This behavior is no doubt due to the huge increase in the search space when going from limited to standard GP and is also confirmed in [13]. For this reason, we have decided to perform our studies eliminating F from the function set, only after having observed that this doesn't change the main results.

6.1 FDC RESULTS

We have calculated p and fdc (see section 5) for various trap functions for the same optimum considered in limited GP. Figures 8 and 9 show the results of these experiments. From these figures we can see that the

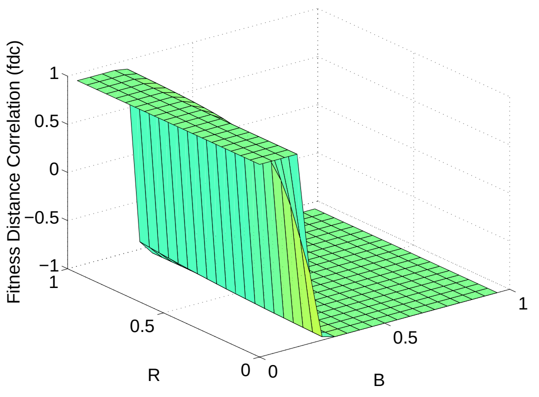

Figure 8: The values of the correlation fdc in the case of standard GP for some trap functions obtained by changing the values of the constants B and R.

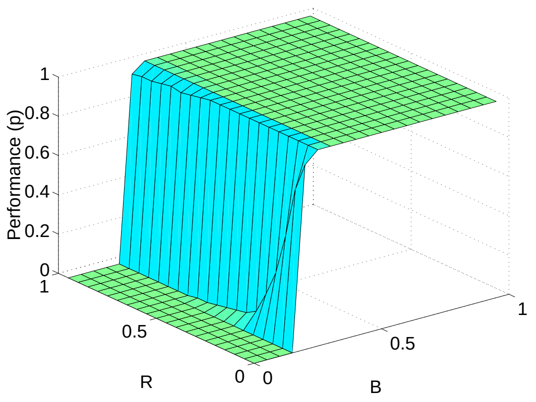

Figure 9: The values of the performance p in the case of standard GP for some trap functions obtained by changing the values of the constants B and R.

same fdc ranges also hold for standard GP with distance d_3 (see section 5) . The results are essentially the same as in the case of limited GP, with the only difference that in the case of standard GP, the number of trap functions for which $-0.15 < fdc < 0.15$ is considerably smaller. Moreover, in the case of standard GP, the number of trap functions for which $fdc = -1$ is larger than in the case of limited GP. This is partly due to the fact that we have considered F as the maximum arity function for limited GP and E for standard GP. But some results not reported here show that this would also be the case if we considered F for standard GP. Thus, for standard GP, there is a larger number of trap functions for which we can predict the difficulty, while the number of functions for which the difficulty is unpredictable is considerably smaller. In any case, the most important result is that, even for standard GP with distance d_3, the fitness distance correlation is a good measure for predicting the difficulty of problems for the class of trap functions.

6.2 RESULTS WITH A DIFFERENT OPTIMUM

In order to understand whether the results are dependent on the particular optimum chosen, we perform the same experiments with the tree shown in figure 10 as the optimum (see 6.2). This tree has an irregular structure and it is different from the optimum used until now. Values of p and fdc for various trap functions are shown in figures 11 and 12, where it appears that, even if the choice of the optimum has an influence on difficulty, the same fdc ranges previously found (see section 5) hold, and fdc with distance d_3 is confirmed to be a good measure for predicting problems difficulty. In the next section, we study the behaviour

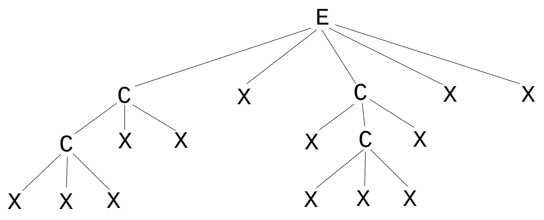

Figure 10: The tree used as optimum in the experiments of section 6.2.

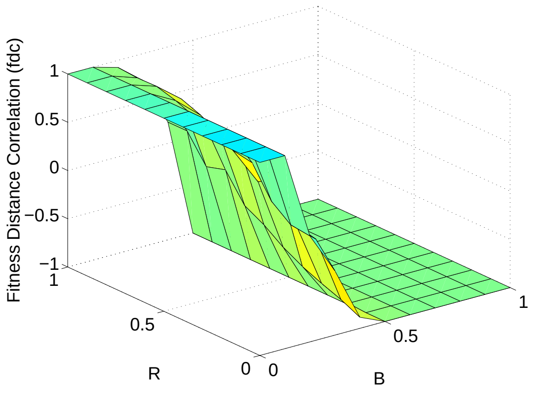

Figure 11: The values of the correlation *fdc* in the case of standard GP for some trap functions obtained by changing the values of the constants B and R. The optimum tree is the one shown in figure 10.

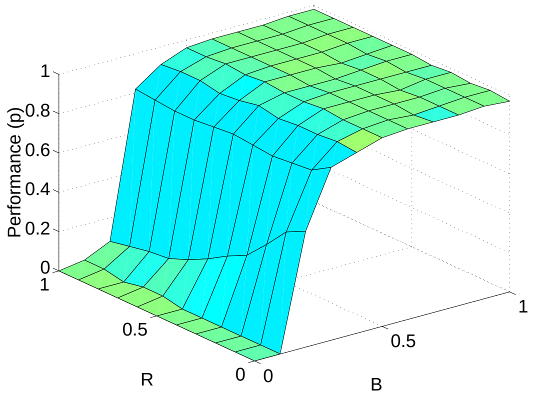

Figure 12: The values of the performance p in the case of standard GP for some trap functions obtained by changing the values of the constants B and R. The optimum tree is the one shown in figure 10.

of *fdc* for another class of functions.

7 ROYAL TREES

The last functions we take into account in this paper are the Royal Trees proposed by Punch and coworkers [13, 14]. These functions are based on the same language that was used in section 6, but the fitness is not calculated on the basis of the distance from the optimum (as it was the case for the trap functions); instead, the following algorithm is used: the raw fitness of a tree (or any subtree) is the score of its root. Each function calculates its score by summing the weighted scores of its direct children. If the child is a perfect tree of the appropriate level (for instance, a complete level-C tree beneath a D node), then the score of that subtree, times a *FullBonus* weight, is added to the score of the root. If the child has a correct root but is not a perfect tree, then the weight is *PartialBonus*. If the child's root is incorrect, then the weight is *Penalty*. After scoring the root, if the function is itself the root of a perfect tree, the final sum is multiplied by *CompleteBonus*. Values used here are: *FullBonus* = 2, *PartialBonus* = 1, *Penalty* = 0.0001, *CompleteBonus* = 2. Results on the study of *fdc* are shown in table 2. In this table, p (respectively p_1, p_2, p_3) indicates the number of executions for which the global optimum has been found in less than 200 (respectively 300, 400, 500) generations, divided by the total number of executions (100 in our experiments). From the table we can see that *fdc* correctly predicts the difficulty of level-A, level-B, level-C, and level-D functions. Level-E function is predicted by the *fdc* to be "straightforward" and it actually is, if we consider that the global optimum is found with a rate of 79% before generation 500. Level-F function is predicted to be "difficult" (where difficult means that the *fdc* doesn't give information on function hardness), and the global optimum is never found before generation 500. Finally, the level-G tree is predicted to be "misleading" (in accord with Punch [13, 14]). In conclusion, it appears that Royal Trees are a synthetic GP problem that effectively spans the classes of difficulty as described by the *fdc*.

8 CONCLUSIONS AND FUTURE WORK

In this work, we have shown that, at least for Trap Functions and Royal Trees, fitness distance correlation is a reasonable way of quantifying GP difficulty. In view of some counterexamples that have been mentioned in the text, it remains to be seen whether this measure extends to other cases such as typical GP benchmarks. This work is only a first step towards the characterization of GP difficulty from a fitness landscape point of view. In the future, we plan to extend our research to other classes of functions, to other distances, for instance the one defined in [8], and to fitness landscapes induced by operators other than standard

Root	*fdc*	*fdc* prevision	p	p_1	p_2	p_3
B	-0.45	straightf.	1	1	1	1
C	-0.33	straightf.	1	1	1	1
D	-0.26	straightf.	0.77	0.81	0.81	0.81
E	-0.22	straightf.	0.42	0.62	0.74	0.79
F	0.035	difficult	0	0	0	0
G	0.26	misleading	0	0	0	0

Table 2: Results of *fdc* for the Royal Trees

crossover, especially various kinds of tree mutations, as mutation seems to play an important role when using *fdc*. We also plan to look for a better measure than performance to identify the success rate of functions, possibly independent from the maximum number of generations chosen.

References

[1] L. Altenberg. Fitness distance correlation analysis: an instructive counterexemple. In T. Back, editor, *Seventh International Conference on Genetic Algorithms*, pages 57–64. Morgan Kaufmann, 1997.

[2] M. Clergue and P. Collard. Genetic heuristic for search space exploration. In *International Joint Conference on Artificial Intelligence (iJCAI'99)*, pages 1218–1224. Ed. Morgan Kaufmann, 1999.

[3] P. Collard, M. Clergue, and F. Bonnin. Misleading functions designed from alternation. In *Congress on Evolutionary Computation (CEC'2000)*, pages 1056–1063. IEEE Press, Piscataway, NJ, 2000.

[4] J. M. Daida, R. Bertram, S. Stanhope, J. Khoo, S. Chaudhary, and O. Chaudhary. What makes a problem GP-hard? analysis of a tunably difficult problem in genetic programming. *Genetic Programming and Evolvable Machines*, 2:165–191, 2001.

[5] K. Deb and D. E. Goldberg. Analyzing deception in trap functions. In D. Whitley, editor, *Foundations of Genetic Algorithms, 2*, pages 93–108. Morgan Kaufmann, 1993.

[6] T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search.* PhD thesis, University of New Mexico, Albuquerque, 1995.

[7] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann, 1995.

[8] R. Keller and W. Banzhaf. Explicit maintenance of genotypic diversity on genospaces. Unpublished, 1994. http://ls11-www.informatik.uni-dortmund.de/people/banzhaf/gp.html.

[9] K. E. Kinnear. Fitness landscapes and difficulty in genetic programming. In *Proceedings of the First IEEEConference on Evolutionary Computing*, pages 142–147. IEEE Press, Piscataway, NY, 1994.

[10] J. R. Koza. *Genetic Programming.* The MIT Press, Cambridge, Massachusetts, 1992.

[11] B. Manderick, M. de Weger, and P. Spiessens. The genetic algorithm and the structure of the fitness landscape. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 143–150. Morgan Kaufmann, 1991.

[12] M. Mitchell, S. Forrest, and J. Holland. The royal road for genetic algorithms: fitness landscapes and ga performance. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems, Proceedings of the First European Conference on Artificial Life*, pages 245–254. The MIT Press, 1992.

[13] B. Punch, D. Zongker, and E. Goodman. The royal tree problem, a benchmark for single and multiple population genetic programming. In P. Angeline and K. Kinnear, editors, *Advances in Genetic Programming 2*, pages 299–316, Cambridge, MA, 1996. The MIT Press.

[14] W. Punch. How effective are multiple populations in genetic programming. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, D. Goldberg, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 308–313, San Francisco, CA, 1998. Morgan Kaufmann.

[15] R.J. Quick, V.J. Rayward-Smith, and G.D. Smith. Fitness distance correlation and ridge functions. In *Fifth Conference on Parallel Problems Solving from Nature (PPSN'98)*, pages 77–86. Springer-Verlag, Heidelberg, 1998.

[16] E. D. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biol. Cybern.*, 63:325–336, 1990.