The Effects of Constant Neutrality on Performance and Problem Hardness in GP

Edgar Galván-López, Stephen Dignum, and Riccardo Poli

Department of Computing and Electronic Systems, University of Essex, Colchester, UK
egalva, sandig, rpoli@essex.ac.uk

Abstract. The neutral theory of molecular evolution and the associated notion of neutrality have interested many researchers in Evolutionary Computation. The hope is that the presence of neutrality can aid evolution. However, despite the vast number of publications on neutrality, there is still a big controversy on its effects. The aim of this paper is to clarify under what circumstances neutrality could aid Genetic Programming using the traditional representation (i.e. tree-like structures) . For this purpose, we use fitness distance correlation as a measure of hardness. In addition we have conducted extensive empirical experimentation to corroborate the fitness distance correlation predictions. This has been done using two test problems with very different landscape features that represent two extreme cases where the different effects of neutrality can be emphasised. Finally, we study the distances between individuals and global optimum to understand how neutrality affects evolution (at least with the one proposed in this paper).

1 Introduction

Evolutionary Computation (EC) systems are inspired by the theory of natural evolution. The theory argues that through the process of selection, organisms become adapted to their environments and this is the result of accumulative beneficial mutations. However, in the late 1960s, Kimura [11] put forward the theory that the majority of evolutionary changes at molecular level are the result of random fixation of selectively neutral mutations. Kimura's theory, called the *neutral theory of molecular evolution* or, more frequently, the *neutral theory*, considers a mutation from one gene to another as neutral if this modification does not affect the phenotype.

Neutral theory has inspired researchers from the EC community to incorporate neutrality in their systems in the hope that it can aid evolution. Despite the vast work carried out towards understanding the effects of neutrality in evolutionary search, as will be seen in the following section, there are no general conclusions on its effects.

In this paper we make an effort to understand how neutrality works and identify under what circumstances it could aid Genetic Programming (GP) [13].

The paper is organised as follows. In the next section, previous work on neutrality in EC is summarised. In Section 3, the approach used to carry out our research is described. In Section 4 we review the notion of fitness distance correlation (*fdc*) and present the distance used to calculated *fdc* in tree-like structures. In Section 5 we introduce the problems used to analyse the effects of neutrality. In Sections 6 and 7 we present and discuss the results of experiments with unimodal and deceptive landscape problems and draw some conclusions.

2 Previous Work on Neutrality

As mentioned previously, there has been a considerable amount of work on neutrality. However, there is a lack of final conclusions on its effects. For instance, Fonseca and Correia [7] developed two redundant representations using different approaches based on mathematical tools. The authors focused their attention on the properties highlighted by Rothlauf and Goldberg [17] and pointed out some potential fallacies in such work. In [17], Rothlauf and Goldberg stated that when using synonymously redundant representations, the landscape connectivity is not increased. Fonseca and Correia, however, stated that this is not necessarily true. They reported that in their proposed representations the connectivity tends to increase accordingly to the number of redundant bits.

In [5], an effort has been made to analyse some aspects of the search space in GP. Ebner focused his attention on the fact that finding a given behaviour on such search spaces is not as difficult as finding a given individual. As the author pointed out, this is due to in GP using tree-like structures, many individuals map exactly to the same phenotype. Correspondingly, the same situation is observed in nature (i.e., highly redundancy). With these elements in hand, Ebner suggested that search spaces that present similar characteristics as the ones found in nature may be beneficial in evolving potential solutions towards finding a global solution to a specific problem.

This line of thought was further explored by Ebner *et al.* [6]. The authors pointed out that the presence of neutrality could aid evolution under certain circumstances. To illustrate this point,they used two types of mappings: random Boolean networks (RBNs) and cellular automaton (see [6] for a full description of both mappings). In their studies, Ebner *et al.* pointed out that neutrality could have a positive impact in evolutionary search, if a given population is spread out in a neutral network. To exemplify this point, Ebner *et al.* used a dynamic fitness landscape and shown how the presence of neutrality gives the opportunity to a population to ”start” over again and eventually (if this is the case) being able to escape from local optima (which is not the case on a non-redundant mapping). In [12], Knowles and Watson distinguished advantages and disantages of what they called random genetic redundancy. In particular the authors used RBNs [6] to conduct their experiments. In their work, Knowles and Watson mentioned that in particular RBNs are useful because help to mantain diversity. On the other hand, the authors also mentioned that the performance, in some cases, was better when neutrality introduced by RBNs was not present and so, one should be carefull when adding artificial neutrality.

Yu and Miller [22] argued that in the traditional GP representation, implicit neutrality is difficult to identify and control during evolution and so, they used Cartesian GP (CGP) to add what they called *explicit* neutrality. To analyse the effects of neutrality they tested their approach on the even parity problems and used an Evolutionary Strategy. CGP uses a genotype-phenotype mapping that allows programs to have inactive code (i.e., this is how neutrality is artificially added) at genotype level. The genotype uses an integer string coding. This type of encoding allowed the authors to use Hamming distance to measure the amount of neutrality present in the evolutionary search. In their studies, they found that the larger the amount of neutrality present during evolution, the higher the percentage of success in finding the global optimum, regardless the mutation rate. They concluded that neutrality is fundamental to improve evolvability.

Collins [4] claimed that Yu and Miller's conclusions in [22] were flawed. Collins started his analysis by highlighting that the use of a Boolean parity problem is a strange choice given that the problem in itself is neutral (i.e., the fitness value of individuals is the same except for the one that finds the global optimum) and, so, the effects of neutrality are harder to analyse using this type of problem. Moreover, Collins focused his attention on the results found for the even-12-parity Boolean problem and pointed out that the CGP representation used in [22] favours shorter sequences than those yielding solutions for this problem. He also showed that the good results reported in [22] (i.e., 55% of success in finding the solutions) are not surprising and that random search has a better performance. He also concluded that the effects of neutrality are more complex than previously thought.

Theoretical work has also been developed in an effort to shed some light on neutrality. In [8] we studied perhaps the simplest possible form of neutrality using GAs: a neutral network of constant fitness, identically distributed in the whole search space. For this form of neutrality, we analysed both problem-solving performance and population flows. We used the fitness distance correlation, calculating it in such a way to make the dependency between problem difficulty and neutrality of the encoding explicit.

In [2], Beaudoin *et al.* proposed a family of fitness landscapes called the *ND* landscapes, where one can vary the length of the genome N and the neutral degree distribution D. This presents some advantages over other types of landscapes (i.e., *NKp*, *NKq* and *Technological*) previously proposed in the literature to analyse neutrality, which, however, do not consider the distribution of neutrality. This, according to the authors, is instead a key feature in evolution.

Recently, we [14] proposed and studied three different types of genotype-phenotype encodings that add neutrality in the evolutionary search. To analyse in detail the effects of these kinds of neutrality on three different types of landscape, we used the *fdc* and the newly introduced notion of *phenotypic mutation rates*. We also developed a mathematical framework that helped explain some of our empirical findings.

As it can be seen from the previous summaries, the results reported on the effects of neutrality in evolutionary search are very mixed (except perhaps the theoretical works previously summarised).

3 Constant Neutrality

With many primitive sets, GP has the ability to create a rich and complex network of natural networks. This may be a useful feature, but it is a feature that is hard to control and analyse. For this reason, in this paper we propose to artificially create a further neutral network within the search space, which is simple and entirely under our control, thereby making it possible to evaluate the effects of neutrality on GP behaviour and performance. In particular, we propose to add a neutral network of constant fitness. More specifically:

– In our approach, called *constant neutrality*, neutrality is "plugged" into the traditional GP representation (i.e., tree-like structures) by adding a flag to the representation: when the flag is set, the individual is on the neutral network and, as indicated

previously, its fitness has a pre-fixed value. When the flag is off, the fitness of the individual is determined as usual.
- We use fitness distance correlation (*fdc*) as a measure of hardness when neutrality is present in the evolutionary search and in its absence. We also perform extensive empirical experiments to corroborate the results found by *fdc*.
- We use two benchmark problems with significantly different landscape features: a unimodal landscape where we expect neutrality to be detrimental and a multimodal deceptive landscape where neutrality helps evolution by escaping from local optima.

As mentioned previously, to allow the presence of constant neutrality in the GP process, we added a flag to the representation. This is in charge of indicating if a given individual is or is not on the neutral layer. To allow the migration to and from the neutral layer we use a special mutation, which is applied with probability P_{nm}. The process to set or unset the neutral flag works as follows. Firstly, we initialise the population by creating random individuals in the usual way, and, with probability P_{nm}, we activate the flag of the resulting individuals. Secondly, during the evolutionary process, every time an offspring is created, it inherits the flag of its parent. However, before the individual is inserted in the population, with probability P_{nm}, we flip its flag. For example, if the parent of the individual was already on the neutral layer (i.e., flag activated) and the flag is flipped, the offspring is off the neutral layer and its fitness is calculated as usual. If, instead, the flag was not flipped, the individual's flag remains activated and its fitness is constant. The situation is symmetric if the original parent was not on the neutral layer.

In the proposed approach, we used traditional crossover (i.e., swapping subtrees) and structural mutation [20] and so, to add neutrality using any of these operators, we set the value of $P_{nm} > 0$ (see Section 5). The main reason of using structural mutation in our experiments is mainly because it is more close to the defition of distances between tree-based structures widely discussed in [19, 20]. Structural mutation involves two types of mutation: inflate and deflate mutation. Inflate mutation takes a random internal node whose arity a is lower than the maximum arity defined in the function set, and it replaces it with a random function of arity $a+1$. A terminal is inserted as the $(a+1)$ argument of the new function. Deflate mutation takes any internal node with an arity a greater than the minimum arity defined in the function set and where at least one argument is a terminal, and it replaces the node with a function of arity $a-1$, deleting one of the terminals rooted on the original node.

In the form of neutrality explained previously, we can easily see how the size of the search space has increased dramatically. However, we still are in the presence of a single global optimum. So, the addition of neutrality comes at a cost: after all we are expanding the search space without correspondingly expanding the solution space. Thus, we should expect that the presence of neutrality will aid evolution only if it modifies the bias of the search algorithm in such a way to make the sampling of the global optimum much more likely.

With these elements in hand, it is very difficult to imagine how adding neutrality to a unimodal landscape can aid evolution. The addition of the form of neutrality explained previously – *constant neutrality* – changes the unimodal landscape into a landscape with plateaus where the search becomes totally random, which could lead to not finding the

optimum solution. If the global solution is found, we should expect that it will take longer for the evolutionary process to find it because of the presence of plateaus.

At this point some question arise. What will the effects of neutrality be on multimodal landscapes? Specifically, will the problem be easier in the presence of neutrality? Will neutrality provide a path to cross optima solutions and be able to find global solutions? Will the presence of neutrality provide advantages on tree-like structures, like for example, control bloat? To answer these questions, we will use fitness distance correlation (fdc) as a measure of hardness. Moreover we will conduct extensive empirical experiments to compare the performance of our approach and the findings of fdc.

4 Fitness Distance Correlation

Jones [10] proposed *fitness distance correlation* (fdc) to measure the difficulty of a problem by studying the relationship between fitness and distance. The idea behind *fdc* was to consider fitness functions as heuristics functions and to interpret their results as indicators of the distance to the nearest global optimum in the search space.

The definition of *fdc* is quite simple: given a set $F = \{f_1, f_2, ..., f_n\}$ of fitness values of n individuals and the corresponding set $D = \{d_1, d_2, ..., d_n\}$ of distances to the nearest global optimum, we compute the correlation coefficient r, as:

$$r = \frac{C_{FD}}{\sigma_F \sigma_D},$$

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, respectively. The n individuals used to compute *fdc* can be chosen in different ways. For reasonably small search spaces or in theoretical calculations it is often possible to sample the whole search space. However, in most other cases, fdc is estimated by constructing the sets F and D via some form of random sampling.

According to [10] a problem can be classified in one of three classes: (1) *misleading* ($r \geq 0.15$), in which fitness tends to increase with the distance from the global optimum, (2) *difficult* ($-0.15 < r < 0.15$), for which there is no correlation between fitness and distance, and (3) *easy* ($r \leq -0.15$), in which fitness increases as the global optimum approaches. There are some known weaknesses with *fdc* as a measure of problem hardness [1, 16]. However, it is fair to say that the method has been generally very successful [3, 10, 14, 18–21].

Several papers have proposed the use of *fdc*. Slavov and Nikolaev were among the first to use *fdc* in GP [18]. In their experiments, they calculated *fdc* using fitness-distance pairs which were recorded during runs. The authors defined the distance between a given individual (DT) in the form of tree-like structure and the global optimum (O) as follows

$$d(DT, O) = \begin{cases} 1 + \sum_{i \in DT,O} d(child(DT_i), child(O_i)) & \text{if root } DT_i \neq \text{root } O_i, \\ \sum_{i \in DT,O} d(child(DT_i), child(O_i)) & \text{otherwise.} \end{cases}$$

Later, Clergue *et al.* [3] extended this idea. As a first step in their investigation, they used the same function and terminal sets defined by Punch *et al.* [15] where the function set was $F_{set} = \{A, B, C, \cdots\}$, where A has arity 1, B has arity 2 and so on. The terminal set included a single symbol, X. Moreover, they set a restriction and generated trees respecting one rule: for every node in a tree, if the arity of the node is n, the nodes below it must have an arity less than n. Initially, Clergue *et al.* defined the distance between trees T_1 and T_2 as follows $d_1(T_1, T_2) = |weight(T_1) - weight(T_2)|$ where $weight(T) = 1 \cdot n_X(T) + 2 \cdot n_A(T) + 3 \cdot n_B(T) + 4 \cdot n_C(T) + \cdots$, $n_X(T)$ 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. However, this definition of distance between a pair of trees had a major problem: two trees with very different structures can have distance of 0. In an effort to overcome this problem, Clergue *et al.* came up with following idea. Each tree with root i must have a greater weight than the trees with root j, if $j < i$, where: (a) $i, j \in \{X, A, B, C, \cdots\}$ and (b) there is an order such that $X < A < B < C \cdots$. Moreover, a prize is given to each root. This new definition improved the situation but there was still a problem: two individuals that have vertical axis of symmetry have a distance 0, despite their structures being different.

The same authors eventually overcame these limitations [3, 21] and computed and defined a distance (which is the distance used in this work) between two trees in three stages: (a) The trees are overlapped at the root node and this process is recursively applied starting from the leftmost subtrees, (b) For each pair of nodes at matching positions, the difference of their codes c (i.e., index of an instruction within the primitive set) is calculated and (c) The computed differences are combined in a weighted sum. That is, the distance between two trees T_1 and T_2 with roots R_1 and R_2 is calculated as follows:

$$dist(T_1, T_2, k) = d(R_1, R_2) + k \sum_{i=1}^{m} dist(child_i(R_1), child_i(R_2), \frac{k}{2}) \quad (1)$$

where $d(R_1, R_2) = (|c(R_1) - c(R_2)|)^z$. $child_i(Y)$ is the i^{th} of the m possible subtrees of a generic node Y, if $i \leq m$, or the empty tree otherwise, and c evaluated on the root of an empty tree is equal to 0. Finally, k is a constant used to give different weights to nodes belonging to different levels in the trees. This distance produced successful results on a wide variety of problems [19–21].

Calculating distances between trees is not as simple as it is when the distance is calculated for a pair of bitstrings. Once the distance has been computed between two trees, it is necessary to normalise it in the range $[0, 1]$. In [20], Vanneschi proposed five different methods to normalised the distance. Here, we propose another way to carry out more efficiently this task and that we have called "normalisation by maximum distance using a fair sampling". This works as follows: (a) A sample of n_s individuals is created using the ramped half and half method and using a global maximum depth greater than the maximum depth allowed during evolution, (b) The distance is calculated between each individual belonging to n_s and the global optimum, (c) Once all the distances have been calculated, the maximum distance m_s found in the sampling is stored.

At the end of this process, the global maximum distance m_s is used to normalise the distances throughout the evolutionary process. The global maximum depth[1] used to

[1] For our experiments the maximum distance is given by $maximum\_depth + 2$.

create a sample of individuals n_s[2] is greater than the maximum depth allowed through evolution. So, through the evolutionary process a higher value for the global maximum distance it is highly unlikely to be found. Moreover, to control bloat we allow a maximum length that is determined during the application of the sampling method.

In the following section, we calculate fdc using Equation (1) on the problems used to study the problem hardness in the absence and in the presence of neutrality in evolutionary search.

5 Experimental Setup

We have used two problems to analyse neutrality. The first one is the Max problem. The problem consists of finding a program, subject to size or depth (D), which produces the largest possible output. For this problem we have defined $F = \{+\}$, $T = \{0.5\}$ and maximum depth $D = 5$ (for all our examples the root node is at depth 0). Naturally, using these sets, the problem has only one global optimum (a full tree with depth 5) and the landscape is unimodal. For this example, we have used the grow method [13] to create our population.

The second problem is a trap function [9]. For this example, we have used the function:

$$f(X) = \begin{cases} 1 - \frac{d}{B} & \text{if } d \leq B, \\ \frac{R(d-B)}{1-B} & \text{otherwise} \end{cases}$$

where d is the normalised distance between a given individual and the global optimum solution. d, B and R are values in the range of $[0,1]$. B is the slope-change location for the optima and R sets their relative importance. For our problem, there is only one global optimum and by varying the parameters B and R, we make the problem easier or harder.

Figure 1 depicts the global optimum solution used in the trap problem where $B = 0.01$ and $R = 0.8$ (i.e., the problem is considered to be very difficult). The language that has been used to code individuals in the trap function is the one proposed by Punch *et al.* [15]. Their idea was to use functions with one increasing arity. That is, for a function set $F = \{A, B, C, \cdots\}$ their corresponding arities are 1, 2, 3, $\cdots$ and the terminal set is defined by $T = \{x\}$ which its arity is 0. Moreover, we have initialised our individuals with the full method [13] using $D = 5$. The maximum allowed depth for programs was 7. Notice that we have used two different methods to initialise the populations for each of the two examples. This has been done to avoid sampling the global solutions for both problems.

The experiments were conducted using a GP with tournament selection size 10. We used standard crossover and mutation (inflate and deflate) independently (i.e., when crossover was used, mutation did not take place during evolution and *vice versa*). To obtain statistically meaningful results, we performed 100 independent runs for each of the values of fitness of the neutral layer. Runs were stopped when the maximum number of generations was reached. The parameters we have used for both problems

[2] For our experiments n_s is typically 10 times larger than the population size.

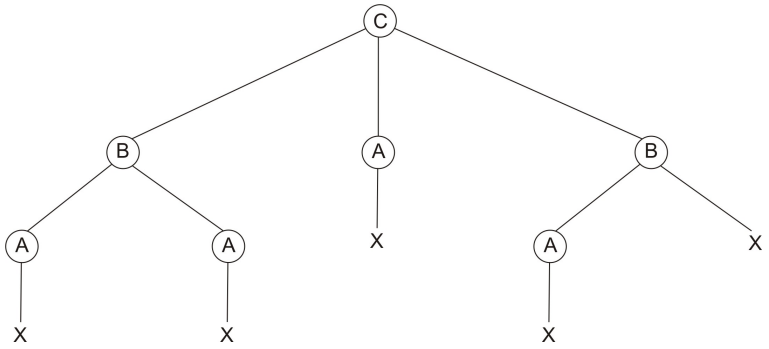

Fig. 1. A tree used as global optimum in the trap function setting $B = 0.01$ and $R = 0.8$ meaning that the problem is considered to be very difficult.

Table 1. Summary of Parameters.

Parameter	*Value*
Population Size	400
Generations	300
Neutral Mutation Probability (P_{nm})	0.05
Mutation Rate	90%
Crossover Rate	90%

are summarised in Table 1. In Tables 2 and 3 we show the constant value (f_n) assigned to the neutral layer for each of the problems.

6 Results and Analysis

6.1 Performance comparison

Let us focus our attention on the Max problem (see Table 2). As discussed previously, it is very hard to imagine how neutrality could aid evolution in a unimodal landscape. When neutrality is not present, GP is able to find the global solution without difficulties both when using crossover and when using structural mutation, the percentage of success being 100% regardless the operator used. This situation, however, changes radically when neutrality is added in the evolutionary search. That is, when neutrality is added the performance of GP decreases. As shown in Table 2, the percentage of success goes from 100% when neutrality is not present to 0% when the fitness of constant neutrality is set to 15 (remember that for this problem the global optimum has fitness 16). This is easy to explain because as discussed previously, individuals which fitness is below the fitness of the neutral layer will tend to move there and once they are in the neutral layer, the search will behave like random search. Note that fdc correctly predicts these performance variations.

Now, let us consider the second problem – the trap function. In Table 3 we show the results found on this problem when calculating fdc. Again we have complemented this

Table 2. Statistical information on the Max problem using $F = \{+\}$, $T = \{0.5\}$ and $D = 5$. The fitness of the global optimum is 16. Avr. Gen. refers to the average number of generations required to find the global optimum.

f_n value	fdc	*Crossover*		*Structural Mutation*	
		Avr. Gen	% Suc.	Avr. Gen	% Suc.
No neutrality	-0.9999	29.14	100%	14.22	100%
5	-0.1994	65.69	95%	17.08	100%
10	0.0661	350.29	17%	28.94	100%
15	0.1380	NA	0%	42.08	100%

Table 3. Statistical information on the Trap function using as global optimum the program shown in Figure 1.

f_n value	fdc	*Crossover*		*Structural Mutation*	
		Avr. Gen	% Suc.	Avr. Gen	% Suc.
No neutrality	0.9971	5.00	1%	3.60	5%
0.10	0.9627	6.00	1%	3.25	3%
0.20	0.8638	6.00	1%	5.33	3%
0.30	0.7070	64.41	12%	107.75	4%
0.40	0.5677	66.83	12%	4.00	2%
0.50	0.4616	94.87	8%	NA	0%
0.60	0.3828	202.20	5%	NA	0%
0.70	0.3234	202.80	5%	NA	0%
0.80	0.2778	470.00	1%	NA	0%
0.90	0.2419	NA	0%	NA	0%

by comparing the performance of GP in the presence and in the absence of neutrality using standard crossover and structural mutation.

When neutrality is not present in the evolutionary search, we can see how fdc classify the problem as very difficult (i.e., $fdc = 0.9999$), which is actually the case. When neutrality is added, there are some circumstances where its presence is more helpful than others. For instance, when the constant fitness in the neutral layer is 0.30 and 0.40, the performance of GP increases dramatically when using standard crossover (i.e., when neutrality is not present, the percentage of success is only 1% compared with 12% when neutrality is added). By how much neutrality will help the search strongly depends on the constant fitness assigned in the neutral layer, f_n. However, for almost all values of f_n we observe improvements over the case where neutrality is absent when crossover is used. Here there is a rough agreement between fdc and actual performance.

Surprisingly, however, when structural mutation is used, in virtually all cases the addition of neutrality hinders performance, and there appears to be a general trend indicating that the higher f_n the worse the results. This goes exactly in the opposite direction of the predictions of fdc. This is perhaps the result of the distance in Equation (1) not being well-suited to capture the offspring-parent differences produced by the actions of the mutation operator.

6.2 Distances between Individuals and Global Optimum

As shown previously, neutrality aids evolution in deceptive landscapes when using crossover. This situation, however, varies depending on the constant value assigned to the neutral layer. Since GP with crossover is effectively the standard, we want to analyse in more detail how neutrality affects evolution. To do so, we study how the distance between the individuals in the population and the global optimum (shown in Figure 1) varies generation after generation for different values of f_n.

In the top left-hand side of Figure 2, notice how the normalised distance between individuals and the global optimum for the cases $f_n = \{0.10, 0.20\}$ effectively varies in the same ways as when neutrality is not present in evolution. Indeed, as confirmed by results shown in Table 3 the percentage of success are almost the same (i.e., in the range of 0% and 1%).

This situation, however, changes radically when using fitter neutral layers, i.e., $f_n = \{0.30, 0.40\}$, as shown at the top right-hand side of Figure 2. Notice how the average distance between individuals and the global optimum tends to drop dramatically compared to when $f_n \leq 0.20$. Individuals are now on average much closer to the global optimum and, so, it is easier for the GP system to eventually sample it and solve the problem.

A reason why GP with crossover is able to sample the global optimum more often in the presence of neutrality with $f_n = \{0.30, 0.40\}$ is that GP tends to produce shorter encodings. This can be observed in the bottom right-hand side of Figure 2. However, this does not mean that GP produces in general smaller individuals regardless the constant fitness set in the neutral layer.

7 Conclusions

The effects of neutrality are unclear. The goal of this paper is to clarify under what circumstances neutrality could aid GP evolution.

In this paper we considered perhaps the simplest possible form of neutrality in GP. This is introduced simply by adding a flag to each individual which indicates whether or not the individual is on the neutral layer. We used the distance, shown in Equation (1), proposed and studied in [19–21] to calculate fitness distance correlations (fdc) in GP landscapes. We used it as a measure of hardness and compared its findings with extensive empirical experimentation using the Max problem with unimodal landscape features and a Trap function with deceptive landscape features.

We found that fdc roughly predicts how problem difficulty is affected by the presence of neutrality for GP with subtree crossover. The prediction of difficulty for GP with structural mutation is, instead, more problematic.

Based on these observations and on empirical results, it is clear that the form of neutrality studied in this paper (constant neutrality) can only aid evolution when the landscape is complex and multimodal. This is interesting, since, in fact, most realistic GP landscapes present such features. However, we have also found that it is important how to set the fitness of the neutral layer (f_n) carefully if for the potential benefits of neutrality to materialise. Much less we can say about the problems where GP with structural mutation could benefit from the use of constant neutrality.

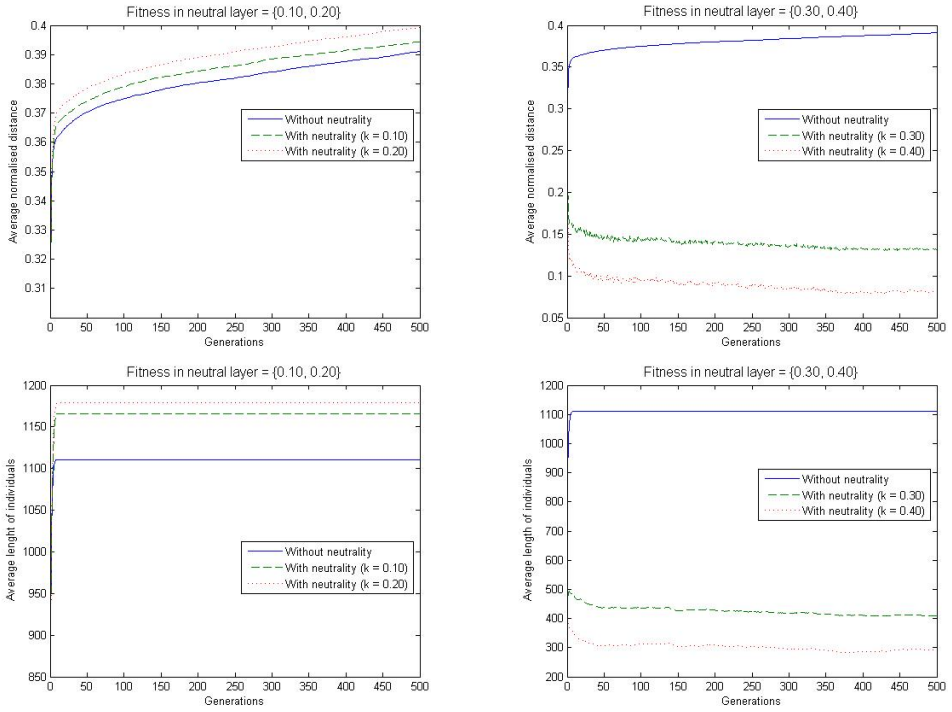

Fig. 2. Average normalised distance (top) and average length of individuals (bottom) using crossover on a difficult trap function. Figure 1 shows the global optimum.

References

1. L. Altenberg. Fitness Distance Correlation Analysis: An Instructive Counterexample. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 57–64, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.
2. W. Beaudoin, S. Verel, and C. Escazut. Deceptivennes and Neutrality The ND Family of Fitness Landscapes. In M. Keijzer, M. Cattolico, D. Arnold, V. Babovic, C. Blum, P. Bosman, M. V. Butz, C. Coello Coello, D. Dasgupta, S. G. Ficici, J. Foster, A. Hernandez-Aguirre, G. Hornby, H. Lipson, P. McMinn, J. Moore, G. Raidl, F. Rothlauf, C. Ryan, and D. Thierens, editors, *Proceedings of the 2006 Conference on Genetic and Evolutionary Computation*, pages 505–514, Seattle, WA, USA, 8-12 July 2006. ACM Press.
3. M. Clergue, P. Collard, M. Tomassini, and L. Vanneschi. Fitness Distance Correlation and Problem Difficulty for Genetic Programming. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 724–732, New York, 9-13 July 2002. Morgan Kaufmann Publishers.
4. M. Collins. Finding Needles in Haystacks is Harder with Neutrality. In H.-G. Beyer, U.-M. O'Reilly, D. V. Arnold, W. Banzhaf, C. Blum, E. W. Bonabeau, E. Cantu-Paz, D. Dasgupta, K. Deb, J. A. Foster, E. D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G. R. Raidl, T. Soule, A. M. Tyrrell, J.-P. Watson, and E. Zitzler, editors, *GECCO 2005: Proceedings of the 2005 Conference on Genetic and evolutionary computation*, volume 2, pages 1613–1618, Washington DC, USA, 25-29 June 2005. ACM Press.

5. M. Ebner. On the Search Space of Genetic Programming and its Relation to Nature's Search Space. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999)*, volume 2, pages 1357–1361, Washington, D.C., USA, 6-9July 1999. IEEE Press.
6. M. Ebner, M. Shackleton, and R. Shipman. How Neutral Networks Influence Evolvability. *Complexity*, 7(2):19–33, 2001.
7. C. Fonseca and M. Correia. Developing Redudant Binary Representations for Genetic Search. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, pages 372–379, Edinburgh, 2-4 Sept. 2005. IEEE.
8. E. Galván-López and R. Poli. Some Steps Towards Understanding How Neutrality Affects Evolutionary Search. In T. P. Runarsson, H.-G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley, and X. Yao, editors, *Parallel Problem Solving from Nature IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 778–787, Reykjavik, Iceland, 9-13 Sept. 2006. Springer-Verlag.
9. D. E. Goldberg, K. Deb, and J. Horn. Massive Multimodality, Deception, and Genetic Algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 37–48, Amsterdam, 1992. Elsevier Science Publishers, B. V.
10. T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, Albuquerque, 1995.
11. M. Kimura. Evolutionary Rate at the Molecular Level. In *Nature*, volume 217, pages 624–626, 1968.
12. J. D. Knowles and R. A. Watson. On the Utility of Redundant Encodings in Mutation-Based Evolutionary Search. In J. J. M. Guervós, P. Adamidis, H.-G. Beyer, J. L. F.-V. Martín, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII: 7th International Conference*, pages 88–98, Granada, Spain, 2002. Springer Verlag.
13. J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, Massachusetts, 1992.
14. R. Poli and E. Galván-López. On the Effects of Bit-Wise Neutrality on Fitness Distance Correlation, Phenotypic Mutation Rates and Problem Hardness. In C. R. Stephens, M. Toussaint, D. Whitley, and P. F. Stadler, editors, *Foundations of Genetic Algorithms IX*, Lecture Notes in Computer Science, Mexico city, Mexico, 8-11 Jan. 2007. Springer-Verlag.
15. 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 Prorgamming 2*, pages 299–316. MIT Press, 1996.
16. R. J. Quick, V. J. Rayward-Smith, and G. D. Smith. Fitness Distance Correlation and Ridge Functions. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 77–86, London, UK, 1998. Springer-Verlag.
17. F. Rothlauf and D. Goldberg. Redundant Representations in Evolutionary Algorithms. *Evolutionary Computation*, 11(4):381–415, 2003.
18. V. Slavov and N. I. Nikolaev. Fitness Landscapes and Inductive Genetic Programming. In G. D. Smith, N. C. Steele, and R. F. Albrecht, editors, *Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference, ICANNGA97*, University of East Anglia, Norwich, UK, 1997. Springer-Verlag.
19. M. Tomassini, L. Vanneschi, P. Collard, and M. Clergue. A Study of Fitness Distance Correlation as a Difficulty Measure in Genetic Programming. *Evolutionary Computation*, 13(2):213–239, Summer 2005.
20. L. Vanneschi. *Theory and Practice for Efficient Genetic Programming*. PhD thesis, University of Lausanne, Switzerland, 2004.
21. L. Vanneschi, M. Tomassini, M. Clergue, and P. Collard. Difficulty of Unimodal and Multimodal Landscapes in Genetic Programming. In E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, editors, *Genetic and Evolutionary Computation – GECCO-2003*, volume 2724 of *LNCS*, pages 1788–1799, Chicago, 12-16 July 2003. Springer-Verlag.
22. T. Yu and J. F. Miller. Needles in haystacks are not hard to find with neutrality. In J. A. Foster, E. Lutton, J. Miller, C. Ryan, and A. G. B. Tettamanzi, editors, *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 13–25, Kinsale, Ireland, 3-5 Apr. 2002. Springer-Verlag.