An Empirical Investigation of How Degree Neutrality Affects GP Search

Edgar Galván-López[1] and Riccardo Poli[2]

[1] Natural Computing Research & Applications Group, Complex and Adaptive Systems Lab
University College Dublin
edgar.galvan@ucd.ie
[2] University of Essex, School of Computer Science and Electonic Engineering,
Wivenhoe Park, Colchester, CO4 3SQ, UK
rpoli@essex.ac.uk

Abstract. Over the last years, neutrality has inspired many researchers in the area of Evolutionary Computation (EC) systems in the hope that it can aid evolution. However, there are contradictory results on the effects of neutrality in evolutionary search. The aim of this paper is to understand how neutrality - named in this paper degree neutrality - affects GP search. For analysis purposes, we use a well-defined measure of hardness (i.e., fitness distance correlation) as an indicator of difficulty in the absence and in the presence of neutrality, we propose a novel approach to normalise distances between a pair of trees and finally, we use a problem with deceptive features where GP is well-known to have poor performance and see the effects of neutrality in GP search.

1 Introduction

Despite the proven effectiveness of Evolutionary Computation (EC) systems, there are limitations in such systems and researchers have been interested in making them more powerful by using different elements. One of these elements is *neutrality* (*the neutral theory of molecular evolution* [8]) which the EC community has incorporated in their systems in the hope that it can aid evolution. Briefly, neutrality considers a mutation from one gene to another as neutral if this modification does not affect the fitness of an individual.

EC researchers have tried to incorporate neutrality in their systems in the hope that it can aid evolution. Despite the vast number of publications in this field, there are no general conclusions on the effects of neutrality and in fact, quite often, there is a misconception with regard to what neutrality is. There are also many contradictory results reported by EC researchers on neutrality.

For instance, in *"Finding Needles in Haystacks is not Hard with Neutrality"* [20], Yu and Miller performed runs using the well-known Cartesian GP (CPG) representation [9, 10] and also used the even-n-parity Boolean functions with different degrees of difficulty ($n = \{5, 8, 10, 12\}$). They compared performance when neutrality was present and in its absence and reported that the performance of their system was better when neutrality was present.

A few years later, Collins claimed the opposite and presented the paper entitled *"Finding Needles in Haystacks is Harder with Neutrality"* [2]. He further explored the idea presented by Yu and Miller and explained that the choice of this type of problem is unusual and in fact not suitable for analysing neutrality using CGP. This is because both the landscape and the form of the representation used have a high degree of neutrality and these make the drawing of general conclusions on the effects of neutrality difficult.

These works (both nominated as best papers in their conference tracks!) are just two examples of many publications available in the specialised literature which show controversial results on neutrality.

The aim of this paper is to understand the effects of neutrality in GP search. For this purpose a new form of neutrality, called *degree neutrality*, will be proposed and studied in detail and a problem with deceptive features will be used to see how GP behaves in the absence and in the presence of neutrality.

This paper is structured as follows. In the next section, a new form of neutrality will be introduced. In Section 3, a well-known measure of difficulty, called fitness distance correlation will be described. A novel method to normalise the distance between a pair of trees will be explained in Section 4. Section 5 provides details on the experimental setup used. Analysis and conclusions of the results found using our approach are presented in Section 6.

2 Degree Neutrality

Degree neutrality is a form of neutrality induced by adding 'dummy' arguments (i.e., terminals) to internal nodes (i.e., functions). More specifically, this type of neutrality works as follows:

- Once an individual has been created, the internal nodes that have an arity lower than a given arity are marked (i.e., if an internal node is *A* and needs to be marked, it is relabelled as *NA*).
- Dummy terminals are added to the internal nodes (i.e., functions) that have been marked. The number of dummy terminals that will be added is determined by a given arity (i.e., an arity specified by the user). For instance, suppose that the given arity is 5 and a tree has functions of arity 1 then those nodes will be marked and 4 dummy terminals will be added to the marked nodes. These are dummy in the sense that their value is totally ignored when computing the output of the node.

Suppose that one is using the language proposed in [14]. That is, the function set is formed by letters (i.e., $F_{set} = \{A, B, C, \cdots\}$) and the arities of each function are as follows: 1 for function *A*, 2 for function *B*, 3 for function *C* and so forth. The terminal set is defined by a single element $T_{set} = \{X\}$. Now, let us define a function set $F_{set} = \{A, B, C, D, E\}$. This function set has a maximum arity of 5. Let us assume that this is the arity that will be used to add degree neutrality. A typical GP individual using the function set F_{set} is shown at the top of Figure 1 and the same individual with degree neutrality is shown at the bottom of the figure. Notice how all internal nodes of the resulting individual (Figure 1 bottom) have now the same arity (i.e., 5).

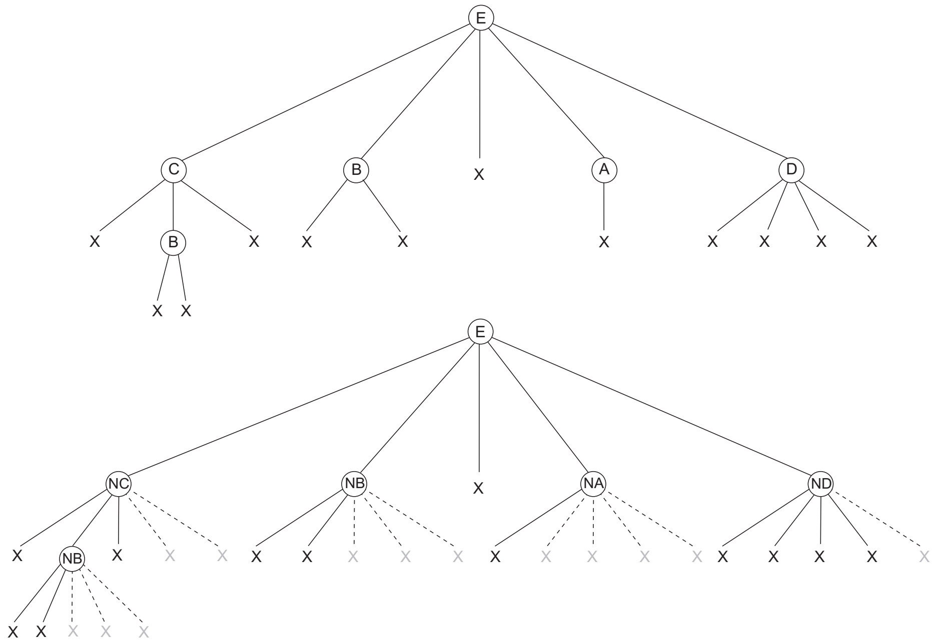

Fig. 1. A typical GP individual created using the language proposed in [14] (top) and the same individual with degree neutrality using a maximum arity of 5 (bottom).

This can easily be extended if one would like to specify, for instance, that the functions defined in the function set are of arities 4 and 5. Then all the function with arities lower than 4 could be marked and extended with dummy arguments. As a result of this, all the functions of the function set would be of arities 4 and 5. The same technique could be applied if the user wanted all the functions defined in the function set to have arities 3, 4 and 5 (i.e., all the internal nodes whose arities are lower than 3 should be "filled" by dummy arguments).

To analyse how degree neutrality will affect the sampling of individuals performed by GP, this form of neutrality will be examined in conjunction with constant neutrality. This form of neutrality was first studied using a binary GA [5, 3] and then analysed using GP [4]. Briefly, the idea is that in this approach, neutrality is "plugged" into the traditional GP representation by adding a flag to the representation: when the flag is set, the individual is on the neutral network and its fitness has a pre-fixed value (denoted in this work by f_n). When the flag is not set, the fitness of the individual is determined as usual. See [4] for a full description of its implementation. We decided to use constant neutrality to study the effects of degree neutrality because with many primitive sets, GP has the ability to create a rich and complex set of neutral networks. This may be a useful feature, but it is a feature that is hard to control and analyse. However, using these types of neutrality we are in total control so, we are in position of study its effects in detail.

In the following section, a well-defined measure of hardness will be introduced and this will help us to better understand the effects of neutrality in evolutionary search.

3 Fitness Distance Correlation

In [7], Jones proposed an heuristic called *fitness distance correlation* (fdc) using the typical GA representation (i.e., the bitstring representation) and successfully tested it on several problems.

fdc is an algebraic measure to express the degree to which the fitness function conveys information about distance to the searcher.

The idea of using *fdc* as an heuristic method, as stated in [7], was to create an algebraic metric that can give enough information to determine the difficulty (for a GA) of a given problem when the global optimum is known in advance. To achieve this, Jones explained that it is necessary to consider two main elements:

1. To determine the distance between a potential solution and the global optimum (when using a bitstring representation, this is accomplished using the Hamming distance) and
2. To calculate the fitness of the potential solution.

With these elements in hand, one can easily compute the *fdc* coefficient using Jones' calculation [7] thereby, in principle, being able to determine in advance the hardness of a problem.

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 of such individuals from the nearest optimum, *fdc* is given by the following correlation coefficient:

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

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, respectively. The n individuals used to compute *fdc* are obtained via some form of random sampling.

According to [7] a problem can be classified in one of three classes, depending on the value of fdc:

- *misleading* ($fdc \geq 0.15$), in which fitness tends to increase with the distance from the global optimum,
- *difficult* ($-0.15 < fdc < 0.15$), for which there is no correlation between fitness and distance; and
- *easy* ($fdc \leq -0.15$), in which fitness increases as the global optimum approaches.

There are some known weaknesses with *fdc* as a measure of problem hardness [1, 15]. However, it is fair to say that this method has been generally very successful [3–5, 7, 12].

Motivated by the good results found by *fdc*, this measure of hardness has been further explored using tree-like structures. There are some initial works that have attempted calculating the distance between a pair of trees [11, 16]. However, these works

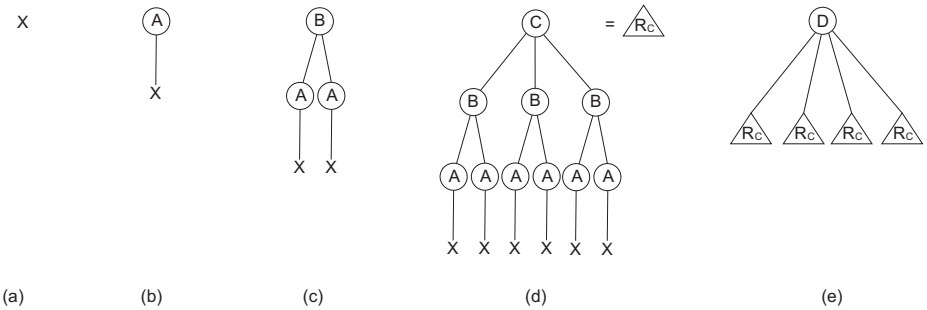

Fig. 2. Distances calculated between different trees. The language that has been used is the one proposed in [14]. $k = 1$ and c has been defined as the arity that each function has (i.e., $c(A) = 1$, $c(B) = 2$, $c(C) = 3$ and so forth). The distance between trees (a) and (b) is denoted by $distance(a,b)$. So, $distance(a,b) = 1.5$, $distance(b,c) = 4.0$, $distance(c,d) = 11.75$ and $distance(d,e) = 36.75$.

were limited in the sense that they did not offer a reliable distance. In, [18, 19, 17] the authors overcame these limitations and computed and defined a distance[3] between a pair of tress.

There are three steps to calculate the distance between tree T_1 and T_2:

- T_1 and T_2 must be aligned to the most-left subtrees,
- For each pair of nodes at matching positions, the difference of their codes c (typically c is the index of an instruction within the primitive set) is calculated, and
- The differences calculated in the previous step are combined into a weighted sum (nodes that are closer to the root have greater weights than nodes that are at lower levels).

Formally, the distance between trees T_1 and T_2 with roots R_1 and R_2, respectively, is defined 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: $d(R_1,R_2) = (|c(R_1) - c(R_2)|)^z$ and; $child_i(Y)$ is the i^{th} of the m possible children of a node Y, if $i < m$, or the empty tree otherwise. Note that c evaluated on the root of an empty tree is 0 by convention. The parameter k is used to give different weights to nodes belonging to different levels in the tree. In Figure 2 using Equation 1, various distances have been calculated using different trees. The distance produced successful results on a wide variety of problems [3, 4, 17–19].

Once a distance has been computed between two trees, it is necessary to normalise it in the range $[0,1]$. In [18], Vanneschi proposed five different methods to normalise the distance. In this work, however, we will introduce a new method that carries out this task more efficiently. This will be presented in the following section.

[3] This is the distance used in this work. We will use the code provided in [3, Appendix D] to calculate the distance between a pair of trees.

4 Normalisation by Maximum Distance Using a Fair Sampling

Inspired by the methods proposed in [18], we propose a new method called "Normalisation by Maximum Distance Using a Fair Sampling" which is used to conduct the empirical experiments show in this work. This method works as follows.

1. A sample of n_s individuals is created using the ramped half-and-half method using a maximum depth greater than the maximum depth defined to control bloat,
2. The distance is calculated between each individual belonging to the sample and the global optimum,
3. Once all the distances have been calculated using n_s individuals that belong to the sample, the maximum distance K_s found in the sampling is stored,
4. n_p individuals that belong to the population are created at random,
5. The distance is calculated between each individual belonging to the population and the global optimum,
6. Once all the distances have been calculated using n_p individuals that belong to the population, the maximum distance K_p found in the population is stored,
7. The global maximum distance, K, is the largest distance between K_s and K_p.

At the end of this process, the global maximum distance K is found. Given that the depth d_s used to create a sample of n_s individuals (for our experiments n_s is typically 10 times bigger than the population size) is greater than the depth d_p defined to control bloat (for our experiments $d_s = d_p + 2$), then throughout the evolutionary process, it is highly unlikely we will ever find a higher value for the global maximum distance. In fact, as we mentioned previously, this normalisation method was used to conduct the experiments reported in this work and in none of them was a higher distance found.

The global maximum distance is found after the sampling of individuals and the creation of the population. Clearly, the main advantage of this process is that during the evolution of individuals, the complexity of the process to normalise distances can be reduced substantially compared to Vanneschi's methods (e.g., "constant-normalisation by iterated search").

5 Experimental Setup

5.1 Trap Function

The problem used to analyse the proposed form of neutrality is a Trap function [6]. The fitness of a given individual is calculated taking into account the distance of this individual from the global optimum. Formally, a trap function is defined as:

$$f(\ell) = \begin{cases} 1 - \frac{d(\ell)}{d_{min}} & \text{if } d(\ell) \leq d_{min}, \\ \\ \frac{r(d(\ell) - d_{min})}{1 - d_{min}} & \text{otherwise,} \end{cases}$$

where $d(\ell)$ is the normalised distance between a given individual and the global optimum solution. $d(\ell)$, d_{min} and r are values in the range $[0,1]$. d_{min} is the slope-change

Table 1. Parameters used for the problems used to conduct extensive empirical experiments using degree neutrality.

Parameter	*Value*
Population Size	400
Generations	300
Neutral Mutation Probability (P_{nm})	0.05
Crossover Rate	90%
Tournament group size	10
Independent Runs	100

location for the optima and r sets their relative importance. For this problem, there is only one global optimum and by varying the parameters d_{min} and r, the problem can be made either easier or harder.

For this particular problem, the function and terminal sets are defined using the language proposed in [14] (see Section 2).

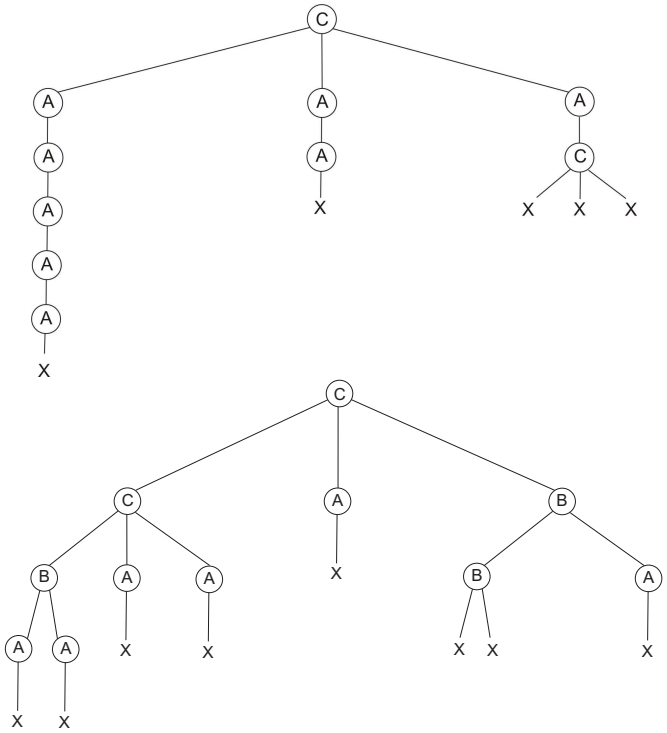

Fig. 3. A global optimum used in our first experiment (top) and a global optimum used in our second experiment (bottom).

Table 2. Performance of GP using swap-crossover. Constant neutrality (i.e., $\{1,2,3,4,5\}$) and degree neutrality with functions of different arities (i.e.,$\{2,3,4,5\},\{3,4,5\},\{4,5\}$) were used. f_n stands for the different fixed fitness values used. The global optimum is shown at the top of Figure 3 and setting $B = 0.01$ and $R = 0.8$ (i.e., the problem is considered to be very difficult).

f_n value	$\{1,2,3,4,5\}$		$\{2,3,4,5\}$		$\{3,4,5\}$		$\{4,5\}$	
	Avr. Gen	% Suc.	Avr. Gen	% Suc.	Avr. Gen	% Suc.	Avr. Gen	% Suc.
-	4.50	2%	4.00	1%	4.00	2%	4	1%
0.10	N/A	0%	N/A	0%	N/A	0%	N/A	0%
0.20	6.00	1%	N/A	0%	6.00	1%	N/A	0%
0.30	11.16	6%	15.50	2%	29.66	3%	18.00	9%
0.40	43.75	12%	37.00	9%	65.83	6%	86.57	7%
0.50	76.80	10%	61.30	13%	67.00	10%	80.69	13%
0.60	111.75	8%	97.00	6%	107.85	7%	104.87	8%
0.70	157.25	8%	118.00	3%	124.60	5%	83.00	1%
0.80	266.50	2%	N/A	0%	N/A	0%	N/A	0%
0.90	N/A	0%	N/A	0%	N/A	0%	N/A	0%

Standard crossover was used to conduct our experiments. Tournament selection was used to conduct our experiments. Furthermore, runs were stopped when the maximum number of generations was reached. The parameters used are given in Table 1.

To avoid creating the global optimum during the initialisation of the population for the Trap function, the full initialisation method has been used. Figure 3 shows the global optima used in our experiments.

6 Analysis of Results and Conclusions

The results of *fdc*, the average number of generations required to find these global optima and the percentage of successes are shown in Tables 4 and 5 when constant neutrality and degree neutrality with a maximum arity of 5 are used (i.e., all the functions are of the same arity).

In Tables 2 and 3, we show the results when using constant neutrality (indicated by $\{1,2,3,4,5\}$ which means that there are functions of those arities) and degree neutrality. As we mentioned previously, the performance of the GP system increases when either form of neutrality is added, specifically in the range of $[0.30-0.65]$. As can be seen from these results, degree neutrality has almost the same effect as constant neutrality when the functions declared in the function set are of mixed arities.

This, however, is not the case when neutral degree is added and the functions declared in the function set are of the same arity. Under these circumstances, the performance of the GP system is increased in almost all cases, as shown in Tables 4 and 5. As can be seen, the predictions done by *fdc* are roughly correct. That is, in the presence of

Table 3. Performance of GP using swap-crossover. Constant neutrality (i.e., arities of functions = $\{1,2,3,4,5\}$) and degree neutrality with functions of different arities (i.e.,$\{2,3,4,5\},\{3,4,5\},\{4,5\}$) were used. f_n stands for the different fixed fitness values used. The global optimum is shown at the bottom of Figure 3 and setting $B = 0.01$ and $R = 0.8$ (i.e., the problem is considered to be very difficult).

f_n value	$\{1,2,3,4,5\}$		$\{2,3,4,5\}$		$\{3,4,5\}$		$\{4,5\}$	
	Avr. Gen	% Suc.	Avr. Gen	% Suc.	Avr. Gen	% Suc.	Avr. Gen	% Suc.
-	N/A	0%	N/A	0%	N/A	0%	N/A	0%
0.10	N/A	0%	N/A	0%	N/A	0%	N/A	0%
0.20	N/A	0%	N/A	0%	N/A	0%	N/A	0%
0.30	52.20	10%	66.55	20%	62.07	13%	58.50	18%
0.40	79.57	14%	113.69	13%	77.66	9%	86.09	11%
0.50	110.30	10%	109.30	13%	75.07	13%	70.66	3%
0.60	204.00	3%	129.50	6%	82.71	7%	145.00	2%
0.70	128.00	2%	101.00	1%	10.66	3%	161.00	1%
0.80	N/A	0%	N/A	0%	N/A	0%	N/A	0%
0.90	N/A	0%	N/A	0%	N/A	0%	N/A	0%

Table 4. Performance of GP using swap-crossover. Constant neutrality (i.e., arities of functions = $\{1,2,3,4,5\}$) and degree neutrality using the maximum arity (i.e., all function have the same arity) were used. f_n stands for the different fixed fitness values used. The global optimum is shown at the top of Figure 3 and setting $B = 0.01$ and $R = 0.8$ (i.e., the problem is considered to be very difficult).

f_n value	*fdc*	$\{1,2,3,4,5\}$		$\{5\}$	
		Avr. Gen	% Suc.	Avr. Gen	% Suc.
-	0.9987	4.50	2%	N/A	0%
0.10	0.9623	N/A	0%	N/A	0%
0.20	0.8523	6.00	1%	N/A	0%
0.30	0.7012	11.16	6%	35.60	10%
0.40	0.5472	43.75	12%	49.38	13%
0.50	0.4682	76.80	10%	84.11	18%
0.60	0.3912	111.75	8%	120.40	15%
0.70	0.3298	157.25	8%	100.25	8%
0.80	0.2892	266.50	2%	104.33	3%
0.90	0.2498	N/A	0%	56.00	2%

neutrality, the percentage of successes tends to increase, particularly when the fitness of the constant value lies in the range of $[0.30-0.75]$. There is, however, a variation in the percentage of successes when there is a mixture of arities (i.e.$\{1,2,3,4,5\}$) and when all the functions are of the same arity (i.e.,$arity = 5$). So, it is clear that while *fdc* computes some of the characteristics of a problem in relation to its difficulty, it does not capture all.

Table 5. Performance of GP using swap-crossover. Constant neutrality (i.e., arities of functions = $\{1,2,3,4,5\}$) and degree neutrality using the maximum arity (i.e., all function have the same arity) were used. f_n stands for the different fixed fitness values used. The global optimum is shown at the bottom of Figure 3 and setting $B = 0.01$ and $R = 0.8$ (i.e., the problem is considered to be very difficult).

f_n value	*fdc*	$\{1,2,3,4,5\}$		$\{5\}$	
		Avr. Gen	% Suc.	Avr. Gen	% Suc.
-	0.9991	N/A	0%	N/A	0%
0.10	0.9714	N/A	0%	N/A	0%
0.20	0.8693	N/A	0%	N/A	0%
0.30	0.7023	52.20	10%	82.76	17%
0.40	0.5802	79.57	14%	78.00	14%
0.50	0.4674	110.30	10%	81.40	15%
0.60	0.3879	204.00	3%	147.12	6%
0.70	0.3342	128.00	2%	246.23	2%
0.80	0.2787	N/A	0%	N/A	0%
0.90	0.2467	N/A	0%	N/A	0%

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 improvements can be seen over the cases where neutrality is absent and when crossover is used. Here there is a rough agreement between *fdc* and actual performance although as f_n is increased beyond a certain level, *fdc* continues to decrease (suggesting an easier and easier problem) while in fact the success rate reaches a maximum and then starts decreasing again.

To explain why GP with crossover is able to sample the global optimum in the presence of neutrality (i.e., see for instance Tables 4 and 5 when $f_n = \{0.30, 0.40, 0.50\}$), we need to consider the following elements. Firstly, the flatter the landscape becomes the more GP crossover will be able to approach a Lagrange distribution of the second kind [13]. This distribution samples heavily the short programs. Secondly, since the global optima used in our experiments (i.e., see Figure 3) are all relatively small, this natural bias might be useful.

On the other hand, as can be seen in Tables 4, and 5, when the constant value on the neutral layer is high (i.e., $f_n \geq 0.70$) the perfomance of the GP system tends to decrease. This is easy to explain given that the higher the value of f_n, the flatter the landscape. Thus, flattening completely a landscape (i.e., $f_n \geq 0.80$) makes the search totally undirected, i.e., random. So, there is no guidance towards the optima. The flattening of the landscape also reduces or completely removes bloat. This is a good feature to have in this type of problem and for the chosen global optima because bloat moves the search towards the very large programs, but we know that none of them can be a solution. So, if bloat were to take place during evolution, it would hinder the search. Effectively, we can say that by changing the values of f_n, the user can vary the balance between two countering forces: the sampling of short programs and the guidance coming from fitness.

References

1. L. Altenberg. Fitness Distance Correlation Analysis: An Instructive Counterexample. In T. Back, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 57–64, San Francisco, CA, USA, 1997. Morgan Kaufmann.
2. 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.
3. E. Galván-López. *An Analysis of the Effects of Neutrality on Problem Hardness for Evolutionary Algorithms*. PhD thesis, School of Computer Science and Electronic Engineering, University of Essex, United Kingdom, 2009.
4. E. Galván-López, S. Dignum, and R. Poli. The Effects of Constant Neutrality on Performance and Problem Hardness in GP. In M. ONeill, L. Vanneschi, S. Gustafson, A. I. E. Alcazar, I. D. Falco, A. D. Cioppa, and E. Tarantino, editors, *EuroGP 2008 - 11th European Conference on Genetic Programming*, volume 4971 of *LNCS*, pages 312–324, Napoli, Italy, 26–28 Mar. 2008. Springer.
5. 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 (PPSN IX). 9th International Conference*, volume 4193 of *LNCS*, pages 778–787, Reykjavik, Iceland, 9-13 Sept. 2006. Springer-Verlag.
6. D. E. Goldberg, K. Deb, and J. Horn. Massive Multimodality, Deception, and Genetic Algorithms. In R. Männer and B. Manderick, editors, *PPSN II: Proceedings of the 2nd International Conference on Parallel Problem Solving from Nature*, pages 37–48, Amsterdam, 1992. Elsevier Science Publishers, B. V.
7. T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, Albuquerque, 1995.
8. M. Kimura. *The Neutral Theory of Molecular Evolution*. Cambridge University Press, Cambridge, UK, 1983.
9. J. F. Miller. An Empirical Study of the Efficiency of Learning Boolean Functions Using a Cartesian Genetic Approach. In W. Banzhaf, J. M. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. J. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'99*, volume 2, pages 1135–1142, Orlando, Florida, 13-17 July 1999. Morgan Kaufmann.
10. J. F. Miller and P. Thomson. Cartesian genetic programming. In R. Poli, W. Banzhaf, W. Langdon, J. Miller, P. Nordin, and T. Fogarty, editors, *Third European Conference on Genetic Programming EuroGP 2000*, volume 1802 of *LNCS*, pages 121–132, Edinburgh, 15-16 Apr. 2000. Springer-Verlag.
11. U.-M. O'Reilly. Using a Distance Metric on Genetic Programs to Understand Genetic Operators. In *IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation*, volume 5, pages 4092–4097, Orlando, Florida, USA, 12-15 Oct. 1997. IEEE Press.
12. 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. Stadler, editors, *Foundations of Genetic Algorithms IX*, Lecture Notes in Computer Science, pages 138–164, Mexico city, Mexico, 8-11 Jan. 2007. Springer-Verlag.

13. R. Poli, W. B. Langdon, and S. Dignum. On the limiting distribution of program sizes in tree-based genetic programming. In M. Ebner, M. O'Neill, A. Ekárt, L. Vanneschi, and A. I. Esparcia-Alcázar, editors, *Proceedings of the 10th European Conference on Genetic Programming*, volume 4445 of *Lecture Notes in Computer Science*, pages 193–204, Valencia, Spain, 11 - 13 Apr. 2007. Springer.
14. B. Punch, D. Zongker, and E. Godman. The Royal Tree Problem, A Benchmark for Single and Multi-population Genetic Programming. In P. Angeline and K. Kinnear, editors, *Advances in Genetic Programming 2*, pages 299–316, Cambridge, MA, 1996. The MIT Press.
15. 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.
16. 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.
17. 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.
18. L. Vanneschi. *Theory and Practice for Efficient Genetic Programming*. PhD thesis, Faculty of Science, University of Lausanne, Switzerland, 2004.
19. L. Vanneschi, M. Tomassini, P. Collard, and M. Clergue. Fitness Distance Correlation in Structural Mutation Genetic Programming. In C. Ryan, T. Soule, M. Keijzer, E. P. K. Tsang, R. Poli, and E. Costa, editors, *Proceedings of the sixth European Conference on Genetic Programming, EuroGP 2003*, volume 2610 of *LNCS*, pages 455–464, Essex, 14-16 Apr. 2003. Springer-Verlag.
20. T. Yu and J. F. Miller. Finding Needles in Haystacks is not Hard with Neutrality. In J. A. Foster, E. Lutton, J. F. Miller, C. Ryan, and A. 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.