Using Subtree Crossover Distance to Investigate Genetic Programming Dynamics

Leonardo Vanneschi[1], Steven Gustafson[2], and Giancarlo Mauri[1]

[1] Dipartimento di Informatica, Sistemistica e Comunicazione (D.I.S.Co.)
University of Milano-Bicocca, 20126 Milan, Italy
[2] School of Computer Science & IT, University of Nottingham
Jubilee Campus, Wollaton Rd. Nottingham, NG81BB, United Kingdom

Abstract. To analyse various properties of the search process of genetic programming it is useful to quantify the distance between two individuals. Using operator-based distance measures can make this analysis more accurate and reliable than using distance measures which have no relationship with the genetic operators. This paper extends a recent definition of a distance measure based on subtree crossover for genetic programming. Empirical studies are presented that show the suitability of this measure to dynamically calculate the fitness distance correlation coefficient during the evolution, to construct a fitness sharing system for genetic programming and to measure genotypic diversity in the population. These experiments confirm the accuracy of the new measure and its consistency with the subtree crossover genetic operator.

1 Introduction

Tree-based genetic programming (GP) uses transformation operators on tree structures [1] to carry out search. These operators define a neighbourhood structure over the trees. To analyse various dynamics of the GP search process, it is useful to quantify the distance between two trees in this topological space. For example, the distance between trees is useful if we want to monitor population diversity (see for instance [2–7]) or if we want to calculate well-known measures of problem hardness such as fitness distance correlation (FDC) (see among others [8–11]). Operator-based distance measures can make calculating distance and the analysis of the search process more accurate [10, 11, 2–5]. The difficulty in defining operator-based distance measures was highlighted in [12]. Defining a distance measure, or a measure of similarity, that is, in some sense "bound" to (or "consistent" with) the genetic operators informally means that if two trees are *close* to each other, or similar, one can be transformed into the other in a few applications of the operator(s). Mutation-based distance measures for GP have been defined, the most common being some variations on the Levenshtein edit distance [3] and the structural distance [7]. In [12], Gustafson and Vanneschi first defined the notion of a subtree crossover based pseudo-distance measure. In this paper, we extend and generalise that definition and we experimentally show the usefulness of this new distance measure to analyse some properties of the search process.

2 Subtree Crossover Distance

Following the same notation as in [12], let P be a population of trees, T_1 be the tree we want to compute a distance from (or the parent tree), T_2 be the tree which we would like to transform T_1 into, and let "T_1/T_2" be the "difference" of these two trees. By definition, this difference operator produces a pair of subtrees (s_{T_1}, s_{T_2}), where subtree $s_{T_2} \in T_2$ must replace $s_{T_1} \in T_1$ to make $T_1 = T_2$. Supposing that $T_1 \in P$, the subtree crossover distance (SCD from now on) between T_1 and T_2 depends on the ability to select s_{T_2} from some tree in P. Thus, the SCD[3] between T_1 and T_2 also depends on the population P. One possibility to define the SCD is to consider it as being equal to 1 in case $s_{T_2} \in P$, since it is possible to transform T_1 into T_2 in just one crossover application. On the other hand, if $s_{T_2} \notin P$ then it will require more than one application of subtree crossover to make $T_1 = T_2$. In that case, calculating the distance would mean counting all these possible applications. This definition of the SCD clearly has some problems: we would need to consider all the necessary next populations or, at least, to approximate them. Creating all the necessary future populations for a particular application of the SCD is clearly computationally infeasible. We might create the future expected populations using calculations similar to the ones found in the schema theorems for GP [13]. However, finding the future expected populations is also costly, essentially requiring a similar amount of computation as actually running the GP algorithm. Furthermore, we have assumed that the distance between T_1 and T_2 is equal to 1 if one crossover application to T_1 can build T_2. However, when we actually execute our algorithm, it is not certain that this particular application will occur. Therefore it may be useful to know the likelihood of creating a particular tree T_2 in the next generation. To overcome the difficulty of defining a multiple operator distance, and to incorporate the stochastic properties of the algorithm, Gustafson and Vanneschi [12] introduced the possibility of considering operator-distance in terms of the probability of correctly applying the operator once. That is, if one tree is in the neighbourhood of another, how likely is it that this neighbour will be found. Since we know (or we can easily calculate) the values of parameters like the selection probability of trees and the frequency of all subtrees in the current population, we could assign a probability to the selection of all subtrees in the next population. If we know what subtree is required to make two trees equal, then we may approximate distance in terms of the probability of selecting this subtree. Thus, given the subtree crossover operator V, Gustafson and Vanneschi defined

[3] The subtree crossover distance that we consider in this paper is a probability and thus it is clearly not a metric. Furthermore it is not just a function of two trees, but also of the population they belong to and in general it does not respect the properties of metrics (like for instance the triangle inequality). Thus, the term "pseudo-distance" (in the sense that it indicates how "far apart" the two items are) would be more appropriate than the term distance. In some senses, we could say that our measure is more like a similarity/dissimilarity measure than a proper distance (Euclidean) metric: it conveys information about how likely it is to make two trees equal, which does largely depend on their similarity. Nevertheless, we use the term distance for the sake of brevity.

the distance function by the following pseudo-code:

$$
\begin{array}{l}
\text{func distance}(T_1, T_2, V, P)\{ \\
\quad (s_{T_1}, s_{T_2}) = T_1/T_2 \\
\quad ps1 = probSelecting(s_{T_1}, T_1) \\
\quad ps2 = probCreating(s_{T_2}, P) \\
\quad return\ (1 - ps1 * ps2) \\
\}
\end{array}
$$

Given the subtree s_{T_2} that needs to replace $s_{T_1} \in T_1$, the distance is defined in terms of the probability of selecting s_{T_1} in T_1 and the probability of creating (or selecting) s_{T_2} from P. Both functions, *probSelecting()* and *probCreating()*, require knowledge of the selection probabilities used in the algorithm. Finding s_{T_1} and s_{T_2} and determining the probability of selecting $s_{T_1} \in T_1$ can be done in linear time in the size of T_1 and T_2. The *probSelecting()* function can be defined for subtree crossover based on the node selection probability. Given uniform node selection, selecting the subtree $s_{T_1} \in T_1$ has the probability of $\frac{1}{|T_1|}$. The *probCreating()* function for subtree crossover can be defined to consider all the occurrences of the subtree s_{T_2} in the population and their probability of selection. That is, for a tree that contains s_{T_2}, we may want to know how likely that tree will be selected by a selection method. We will then want to know the probability of selecting s_{T_2}. Since evolutionary algorithms use fitness-based selection to implement solution competition, not all trees have the same likelihood of being selected. Gustafson and Vanneschi [12] used this fact to provide an effective way of reducing complexity of this operator distance while preserving the utility of the measure: they only considered those trees and their subtrees that are *likely* to be selected. However, the distance used in [12] presents one major limitation: in that definition, in case a tree T_2 cannot be obtained from a tree T_1 with one crossover, the distance between T_1 and T_2 was the probability of selecting the root of T_1 as crossover point and a subtree equal to T_2 from the population. The likelihood of this event was considered to be very small and thus it was approximated to zero (and thus the distance was set to one, i.e. the maximum possible distance). In other words, the distance between two trees T_1 and T_2 was equal to one if T_1 and T_2 differed in more than one subtree. In some cases, this approximation may be too coarse, thus compromising the accuracy of the measure (for instance when using the SCD for calculating the FDC). In this paper, we extend that definition admitting that the distance between two trees T_1 and T_2 differing in more than one subtree can have a smaller value than one and thus overcoming this limitation. The new subtree crossover distance can be defined as follows: we define a new operator *diff(T_1, T_2)* which returns the *set* $S = \{(s^1_{T_1}, s^1_{T_2}), (s^2_{T_1}, s^2_{T_2}), ..., (s^n_{T_1}, s^n_{T_2})\}$ such that $\forall i \in [1, n]$ if we replace $s^i_{T_1}$ with $s^i_{T_2}$ in T_2 we obtain T_1; *diff(T_1, T_2)* returns the empty set if T_1 and T_2 share no genetic material. Now, the new SCD can be defined by the following algorithm:

```
func SCD(T1, T2, P ){
    S = diff(T1, T2)
    res = 0
    for i = 1 to cardinality(S) do
        ps1 = probSelecting(s^i_T1, T1)
        ps2 = probCreating(s^i_T2, P)
        res = res + (ps1 * ps2)
    endfor
    return(1 − res)
}
```

The main difference between this new definition and the one in [12] is that the returned value is a sum of probabilities, each of which is the product between the probability of selecting one subtree $s^i_{T_1}$ and the probability of creating one subtree $s^i_{T_2}$. The paper [12] contains a detailed discussion on the computational complexity of the old definition of crossover distance. The complexity of this new definition is not much higher than the complexity of the old one: the most expensive step is storing in a hash table all the subtrees in the population at each generation (and it was done also in the old version). Once it has been done, the cost of building the S set differs from the cost of generating only two subtrees s_{T_1} and s_{T_2} (eventually selecting the root of T_1 and $s_{T_2} = T_2$, as in the old definition) of a linear factor. This new distance measure will be used in the next section for analysing some properties of the GP search process.

3 Experimental Results

The goal of this section is to show the suitability of the new definition of SCD for monitoring various properties of the GP search process. In particular, Section 3.1 shows how this distance can be used to calculate fitness distance correlation (FDC) inside the population during the search process, Section 3.2 shows results of fitness sharing using SCD and Section 3.3 shows how the SCD can be used to measure genotypic diversity of populations.

3.1 Fitness Distance Correlation

FDC was first proposed as a difficulty measure for GAs in [8]. It is defined as follows: given a sample $F = \{f_1, f_2, ..., f_n\}$ of n individual fitnesses and a corresponding sample $D = \{d_1, d_2, ..., d_n\}$ of the n distances to the nearest global optimum, $FDC = C_{FD}/(\sigma_F \sigma_D)$, 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. In [8], Jones proposed that GAs problems may be partitioned into three classes, depending on the value of the FDC coefficient: *misleading* (if the FDC is positive, and thus fitness increases as the distance to the global optimum increases), *straightforward* (if FDC is negative, and thus fitness increases as individuals approach the global optimum) and *difficult* (if the

there is no correlation between fitness and distance). In [9–11], Vanneschi *et al.* showed the suitability of FDC as a measure of problem hardness for tree based GP. In particular, they used systems with single-node altering transformation operators (like single-node mutation) and they employed the well known structural distance [7] (which they proved to be bound to this operator) to accurately calculate the FDC. They also showed that FDC calculated using structural distance is a reasonable indicator of problem hardness for GP systems using subtree crossover. Nevertheless, given that no bound was proven between subtree crossover and structural distance, they used large samples of individuals (and not just the individuals composing the population) to calculate FDC. On the other hand, the study of the trend of the FDC in the population during the evolution would be very interesting, since this study would be more *dynamic* than studying the FDC once for all on a single large sample of individuals. In fact, this investigation would allow us to study how the FDC gets modified during the evolution and this information could allow us to draw some conclusions on the dynamics of the GP search process. In particular, we could imagine that if the FDC value decreases during the evolution and it tends towards -1, the population is converging towards the global optimum (individuals are approaching the global optimum as fitness is improving). On the other hand, if the FDC value increases during the evolution, or it remains static at some initial positive level or at zero, this probably means that the population is converging towards a local optimum (fitness is improving, but the distance to the global optimum is not decreasing). The following experiments have been done to confirm this hypothesis and to test the suitability of SCD to calculate the FDC.

Syntactic Trees

In the syntactic trees problem, as used in [12], trees are represented using the set of functions $\mathcal{F} = \{N\}$, where N is a binary operator (N stands for "*N*on-terminal") and the set of terminal symbols $\mathcal{T} = \{L\}$ (L stands for "*L*eaf"). No "content" is associated with the nodes and fitness is simply equal to the edit distance (ED from now on) to a fixed global optimum. The global optimum of an instance is generated using a random tree growing algorithm described in [12]. The definition of ED used here is the same as in [3]. Figure 1(a) shows the tree chosen as optimum for the experiments in Figure 2, and Figure 1(b) shows the tree chosen as optimum for the experiments in Figure 3. These experiments have been performed using the following set of parameters: generational GP, population size of 30 individuals, standard subtree crossover as the only genetic operator, tournament selection of size 5, ramped half-and-half initialisation, maximum depth of individuals for the initialisation phase equal to 4, maximum depth of individuals for crossover equal to 8. All the runs have been stopped at generation 100. Figure 2 reports the average values (with their standard deviations in Figure 2(b)) of the best fitness, the average fitness and the FDC (calculated using SCD) in the population (against generations) over 50 independent GP runs in which the global optimum has been found before generation 100 (*successful runs*). Producing 50 successful runs has been easy, probably for the very simple shape of the tree that we have used as optimum (shown in Figure 1(a)). The method that we have used to collect 50 successful runs was simply to execute a sequence of GP runs until 50 successful ones were found. It has been sufficient to execute 52 runs to get 50 successful ones. Figure 3 reports the same

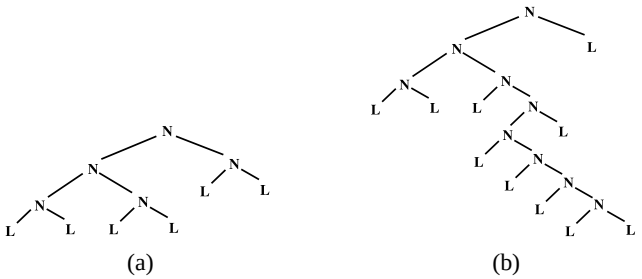


Fig. 1. (a) The tree used as optimum for the experiments in Figure 2. (b) The tree used as optimum for the experiments in Figure 3.

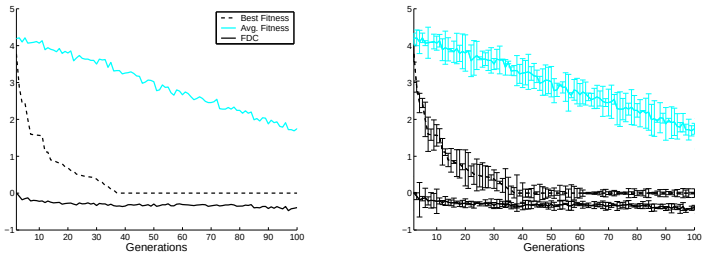


Fig. 2. Syntactic Trees Problem. Average values (a) and average values with their standard deviations (b) of average fitness, best fitness and FDC in the population against generations over 50 independent GP runs. In all these runs the optimum has been found before generation 100. The tree used as optimum in these experiments was the tree in Figure 1(a).

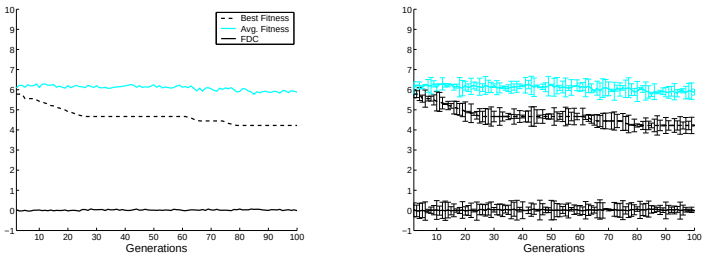


Fig. 3. Syntactic Trees Problem. Average values (a) and average values with their standard deviations (b) of average fitness, best fitness and FDC in the population against generations over 50 independent GP runs. In all these runs the optimum has *not* been found before generation 100. The tree used as optimum in these experiments was the tree in Figure 1(b).

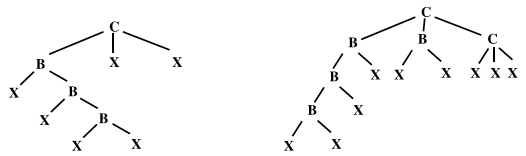
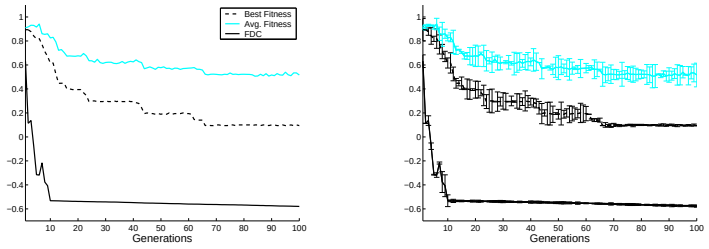


Fig. 4. (a) The tree used as optimum for the experiments in Figure 5. (b) The tree used as optimum for the experiments in Figure 6.

information, but this time for 50 *unsuccessful runs*. Collecting 50 unsuccessful runs has been easy, probably for the particular shape of the tree that has been used as optimum, shown in Figure 1(b) (over 61 runs, 50 were successful). These figures show that, in case of success, the FDC decreases until the global optimum is found and than remains negative until the end of the run. In case of unsuccessful runs, the FDC value always stays around zero, independently from the fact that fitness is slightly improving. Our interpretation is that, in this last case the evolutionary process in "leading" the population towards a local optimum, which probably has a rather large crossover distance from the global one. For successful runs the fact that FDC is negative indicates that evolution is "leading" the population towards the global optimum.

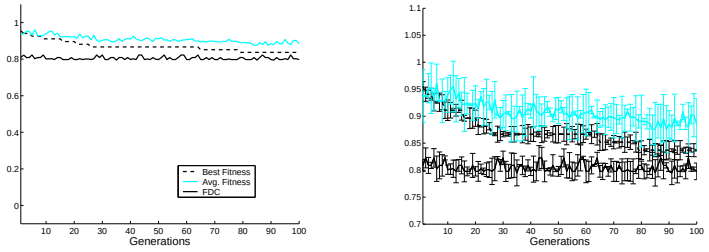
Trap Functions

We now define a problem where trees are represented using the same syntax as in [14], i.e. by means of the set of functions $\mathcal{F} = \{B, C\}$ (where B is a binary operator and C has arity $= 3$) and the set of terminal symbols $\mathcal{T} = \{X\}$. The fitness of each tree is a function of its structural distance (as defined in [7]) to a fixed global optimum and it is not defined here for lack of space (see for instance [11] for a formal definition). In this paper, it is sufficient to remember that the fitness definition for trap functions depends on two parameters (called b and r), which can be used to tune the difficulty of the problem (see [11] for a detailed discussion). Figure 4(a) shows the tree chosen as optimum for the experiments in Figure 5 and Figure 4(b) shows the tree chosen as optimum for the experiments in Figure 6. Parameters used in these experiments are as follows: generational GP, population size of 100 individuals, standard subtree crossover used as the sole genetic operator, tournament selection of size 10, ramped half-and-half initialisation, maximum depth of individuals for the initialisation phase equal to 6, maximum depth of individuals for crossover equal to 10. Here, we have used larger trees than in the case of the syntactic tree problem discussed in the previous section (arity 3 nodes and deeper trees have been considered). The reason is that we wanted to test our hypotheses in different conditions. Figure 5 reports the average values (with their standard deviations in Figure 5(b)) of the best fitness, the average fitness and the FDC (calculated using SCD) in the population (against generations) over 50 independent successful GP runs. The method used to collect 50 successful runs was the same as the one discussed in the previous section. In these experiments, we have set the b and r trap functions parameters as follows: $b = 0.9$ and $r = 0.1$. In this way, the fitness landscape



(a) (b)

Fig. 5. Trap Functions. Average values (a) and average values with their standard deviations (b) of average fitness, best fitness and FDC in the population against generations over 50 independent GP runs. In all these runs the optimum has been found before generation 100. The tree used as optimum in these experiments was the tree in Figure 4(a).



(a) (b)

Fig. 6. Trap Functions. Average values (a) and average values with their standard deviations (b) of average fitness, best fitness and FDC in the population against generations over 50 independent GP runs. In all these runs the optimum has *not* been found before generation 100. The tree used as optimum in these experiments was the tree in Figure 4(b).

is easy to search for GP [11] and thus it is easy to have successful runs. Figure 6 reports the same information as Figure 5, but for 50 independent unsuccessful GP runs. In this case, the b parameter was set to 0.1 and the r parameter to 0.9 in order to make the fitness landscape difficult to search for GP [11]. The method used to collect 50 unsuccessful runs was the same as the one discussed in the previous section. In Figure 3(b), the scale on the ordinates axis has been restricted in order to enlarge the graph and to make it clearer and more readable. These figures show that for successful runs the FDC decreases until the global optimum is found and than remains negative until the end of the run, while in case of failure the FDC is always positive. Here the phenomenon is even more marked than in the case of syntactic trees. In fact, for successful runs FDC rapidly stabilises to approximately -0.6, while for unsuccessful runs FDC always remains approximately equal to 0.8. Once again, our conclusion is that the value of FDC in the population (calculated using the SCD) is a good indicator of the "direction" the search process is "leading" the population: negative values of the FDC mean that

the search is moving towards a global optimum, while positive values of the FDC mean that the search is moving towards local ones.

3.2 Fitness Sharing

In the previous section, we have shown that SCD can appropriately be used to dynamically calculate the population FDC during the evolution. However, as discussed in [11], FDC is not a predictive measure (i.e. the global optima must be known to be able to calculate it), which makes the FDC almost unusable in practice. Other than a measure for diversity, can the SCD be useful for practitioners? In this section, we discuss *fitness sharing*, calculated using the SCD. Fitness sharing is a mechanism, first introduced by Goldberg and coworkers [15] for GAs, for counteracting premature convergence of populations. With this scheme, the fitness function is modified to incorporate a sharing function s, defined to determine the degree of sharing of each individual in the population. When fitness sharing is used, the fitness of each individual i in the population P is calculated as $f_s(i) = f(i)/\sum_{j \in P \wedge j \neq i} s(d(i,j))$, where f is the problem fitness function and d is a distance measure between genotypes. In this section, we compare the performance of two different fitness sharing systems using the SCD and the ED as distance d with standard GP systems. For the s function, we have simply used $s(x) = 1 - x$, after normalizing ED values into the set $[0,1]$ (there is no need of normalising SCD values, since they are probabilities, and thus they are already included into $[0,1]$). Experiments have been done on a problem which is the same as the syntactic tree problem described in Section 3.1, except for the fitness of an individual i is equal to the sum of the differences of the number of nodes of i and the ones of a fixed global optimum for each level in the trees. This is the same fitness used in [12]. In this way, the global optimum is not unique, which is a case in which using fitness sharing may be appropriate [15]. GP parameters are the same as the ones used for syntactic trees in Section 3.1. Results are shown in Figure 7, where two experiments, with two different global optima, are considered. These two optima were two different randomly generated trees (whose structure is not shown here for lack of space). Curves in Figure 7(a) and 7(c) show the average values of the best fitness in the population against generations over 50 independent GP runs for the two cases where the two different randomly generated trees have been used as optima. Figures 7(b) and 7(d) show standard deviations of the curves in Figures 7(a) and 7(c) respectively. As these figures show, the fitness sharing system using the SCD finds, on average, better quality solutions than standard GP and than fitness sharing systems using the ED for both the cases studied, even though standard deviations bars may slightly overlap in some cases[4]. We hypothesise that the better results achieved by the SCD occur because SCD is bound to the genetic operator used by GP and thus more accurate than ED in directing the evolutionary search process.

[4] In particular, the bars of the fitness sharing system using the ED and the ones of the standard GP system always overlap. On the other hand, the bars of the fitness sharing system using SCD and the ones of the other two systems only slightly overlap in some cases.

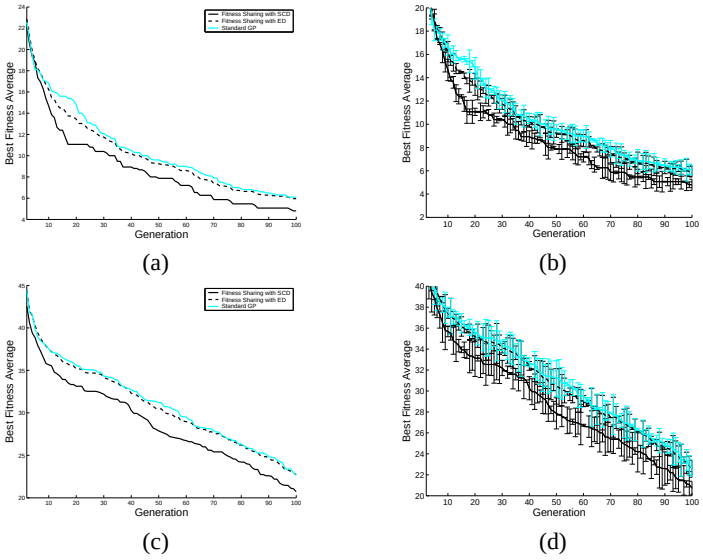
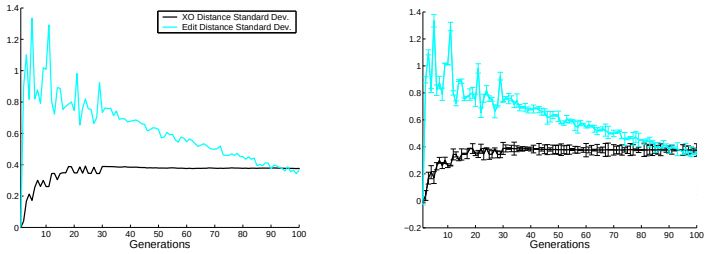


Fig. 7. Average best fitness values against generations (Figures (a) and (c)) and average best fitness values with their standard deviations (Figures (b) and (d)) over 50 independent GP runs using standard GP (gray curves) and fitness sharing (black curves). Figures (a) and (b) differ from Figures (c) and (d) for the particular tree chosen as optimum.

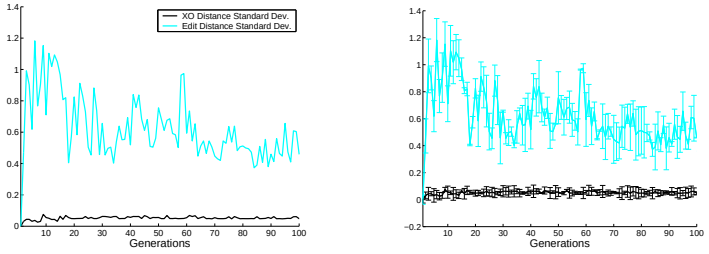
3.3 Diversity

In this section, we compare results of population genotypic diversity calculated by the SCD with the ones obtained using the ED. In both cases, diversity has been measured as the standard deviation of the distance of the individuals in the population to a fixed optimal tree. The problem and GP parameters used in this section are the same as the syntax trees problem described in Section 3.1, with the only difference that the tree chosen as optimum has been randomly generated at the beginning of each GP run (and thus it changes from one run to the other). Figure 8 shows average values (with their standard deviations, which are reported in Figure 8(b)) of diversity against generations over 50 independent successful GP runs The method used to collect 50 successful runs was the one presented in section 3.1. Figure 9 reports analogous results for 50 independent runs in which the optimum has *not* been found before generation 100. Also the method used to collect 50 unsuccessful runs was the one presented in section 3.1. First of all, we remark that the behavior of SCD diversity is rather different from the one of ED diversity: ED diversity is, in general, very “unstable”, in particular at the beginning of the runs. On the other hand, SCD diversity has much smaller variations during the evolution. Furthermore, while ED diversity curves are always decreasing and tend towards zero, the SCD diversity tends to stabilise on a certain value and to constantly maintain this value until the end of the runs. Finally, we observe that in the unsuccessful runs, SCD diversity tends to stabilise towards zero. This happens because in those cases, after a certain number of generations, all the individuals in the population tend to have a SCD



(a) (b)

Fig. 8. Average values (a) and average values with their standard deviations (b) of population's diversity against generations over 50 independent runs. In all these runs the optimum has been found before generation 100.



(a) (b)

Fig. 9. Average values (a) and average values with their standard deviations (b) of population's diversity against generations over 50 independent runs. In all these runs the optimum has been found before generation 100.

to the optimum equal to zero: it becomes more and more difficult to find trees that can be transformed into the global optimum by simply swapping some of its subtrees with some other in the population. On the other hand, in the successful runs, SCD diversity values remain approximately equal to a value that, although not very large, is always larger than zero. In those cases, producing the optimum by means of crossover from trees in the population is possible.

4 Conclusions

This paper has two main goals: extending the definition of subtree crossover distance (SCD) given in [12] and empirically showing that this new measure is useful to investigate some properties of the GP search process. Results that have been presented show that (1) the SCD is appropriate to study the trend of fitness distance correlation (FDC) of populations during the evolution; (2) the SCD is also appropriate for fitness sharing, producing better results than standard GP and than fitness sharing systems using edit

distance (ED); (3) if we use the SCD standard deviation to quantify population diversity, we obtain different results and we capture different properties than if we use ED standard deviation. We hypothesise that the reason for these results is that the SCD appropriately models subtree crossover. Future work includes exploring other definitions of operator-based measures and the tradeoffs involved with reducing their complexity.

References

1. J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.
2. S. Gustafson, A. Ekárt, E.K. Burke, and G. Kendall. Problem difficulty and code growth in genetic programming. *Genetic Programming and Evolvable Hardware*, 5(3):271–290, 2004.
3. S. Gustafson. *An Analysis of Diversity in Genetic Programming*. PhD thesis, School of Computer Science and Information Technology, University of Nottingham, Nottingham, England, February 2004.
4. E.K. Burke, S. Gustafson, and G. Kendall. Diversity in genetic programming: An analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, 8(1):47–62, 2004.
5. N.F. McPhee and N.J. Hopper. Analysis of genetic diversity through population history. In W. Banzhaf et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1112–1120, FL, USA, 1999. Morgan Kaufmann.
6. M. Tomassini, L. Vanneschi, F. Fernández, and G. Galeano. A study of diversity in multipopulation genetic programming. In *6th International Conference on Evolutionary Computation EA'03*, pages 69–81, 2003.
7. A. Ekárt and S. Z. Németh. Maintaining the diversity of genetic programs. In J. A. Foster *et al.*, editor, *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 162–171. Springer-Verlag, 2002.
8. T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192, San Francisco, CA, 1995. Morgan Kaufmann.
9. 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, 2005.
10. L. Vanneschi, M. Tomassini, P. Collard, and M. Clergue. Fitness distance correlation in structural mutation genetic programming. In C. Ryan et al., editors, *Genetic Programming, Proceedings of the European Conference*, volume 2610 of *LNCS*, pages 459–468, Essex, 14-16 April 2003. Springer-Verlag.
11. L. Vanneschi. *Theory and Practice for Efficient Genetic Programming*. Ph.D. thesis, University of Lausanne, Switzerland, 2004.
12. S. Gustafson and L. Vanneschi. Operator-based distance for genetic programming: Subtree crossover distance. In Keijzer, M., *et al.*, editor, *Genetic Programming, 8th European Conference, EuroGP2005*, Lecture Notes in Computer Science, LNCS 3447, pages 178–189. Springer, Berlin, Heidelberg, New York, 2005.
13. R. Poli and N.F. McPhee. General schema theory for genetic programming with subtree-swapping crossover: Part i. *Evolutionary Computation*, 11(1):53–66, 2003.
14. 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.
15. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.