

Operator-Based Distance for Genetic Programming: Subtree Crossover Distance

Steven Gustafson¹ and Leonardo Vanneschi²

¹ School of Computer Science & IT, University of Nottingham
Jubilee Campus, Wollaton Rd. Nottingham, NG81BB, United Kingdom
`smg@cs.nott.ac.uk`

² Dipartimento di Informatica, Sistemistica e Comunicazione (DISCo)
University of Milano-Bicocca 20126 Milano, Italy
`vanneschi@disco.unimib.it`

Abstract. This paper explores distance measures based on genetic operators for genetic programming using tree structures. The consistency between genetic operators and distance measures is a crucial point for analytical measures of problem difficulty, such as fitness distance correlation, and for measures of population diversity, such as entropy or variance. The contribution of this paper is the exploration of possible definitions and approximations of operator-based edit distance measures. In particular, we focus on the subtree crossover operator. An empirical study is presented to illustrate the features of an operator-based distance. This paper makes progress toward improved algorithmic analysis by using appropriate measures of distance and similarity.

1 Introduction

In canonical tree-based genetic programming (GP), part of the search process is carried out using transformation operators on tree structures [1]. From a topological point of view, these operators can be thought of as defining the neighbourhood of these trees. To analyse various properties of the search process, it is often useful to know the distance between two trees. For example, if we wish to calculate a well-known measure of problem hardness, such as fitness distance correlation [2–5], we have to calculate the distance of a sample of trees from a particular global optimum. As trees become closer to the optimum, we would like the improvement in fitness to be positively correlated with the decrease of distance, making the search more predictable. Furthermore, when studying diversity, the distance between two trees is required in order to find the average pair-wise degree of similarity of the trees in a population. When the average pair-wise distance of the population approaches 0, the population is converging and we can expect the search to become stuck in a local optimum. The use of tree structures and multi-node altering transformation operators (e.g. subtree crossover), typical of GP, makes, among other things, defining operator-based distance measures complex. Thus, the study of fitness distance correlation has

largely progressed using systems with single-node altering transformation operators like single-node mutation [4, 5]. This allows appropriate conclusions to be drawn from empirical results for correlating the improvement of fitness with the change in distance to an optimum. Likewise, most research for diversity methods and measures in turn rely on using distance measures based on single-node differences between trees [6–9]. The most common are variations on the Levenshtein edit distance.

Defining a distance measure, or measure of similarity, that is, in some senses “bound” to (or “consistent” with) the genetic operators being used 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). The complexity involved in using operator-based distance measures is two-fold: firstly, distance needs to be re-defined for the specific operators being used. Thus, if we add a new mutation operator or a variation of subtree crossover, we will need to reconsider the distance definition. This results in a large design complexity. Secondly, actually computing an operator-based distance can be much more computationally expensive than the more straightforward edit distance. For example, complexity is increased for operator-based measures as typically the distance between two trees depends on the current population. The computational complexity involved in operator-based distance measures encourages the use of approximations such as edit distance.

The complexity of the edit distance between two trees is in $O(k)$, i.e. dependent upon the number of k nodes in the trees. Computing the pair-wise distance between every tree in the population has complexity $O(M^2 \times k)$, where M is the size of the population and k is the average size of the trees. If the edit distance measure defines a metric space, symmetry only requires $\frac{M(M-1)}{2}$ comparisons. For fixed-length bit-string genetic algorithms, Wineberg and Oppacher showed how this can be done in $O(M \times k)$ with preprocessing of the population[10]. This method would become more complex to design with a variable size and shape representation like GP trees.

The contribution of this paper is the exploration of defining operator-based distances and a discussion of the approximation of such distances. A distance measure based on subtree crossover is defined for a constructed problem, and an empirical study demonstrates its features. Section 2 introduces the concept of distance based on a genetic operator and its applications to a canonical GP system. Section 3 defines a new distance measure bound to standard subtree crossover. Since the calculation of this distance measure may require a lot of computing resources, some techniques to approximate this measure, thus reducing computational complexity without compromising its efficacy, are presented. Section 4 presents some experimental results showing the suitability of this distance measure, compared to a well known tree distance measure.

2 Defining Operator-Based Distance

Initially, we explore what operator distance means in a typical system. In this section, we consider a simple steady-state GP system using syntax trees to represent individuals, and subtree crossover for variation. Subtree crossover proceeds by selecting a subtree in a parent tree and a subtree in a donor tree, where any node in either tree can be selected. Next, subtree crossover replaces the parent’s subtree with the donor’s subtree. New trees replace the parent trees. Concerning distance, the question we want to answer is: what is the distance between two trees contained in the population according to a given operator, such as subtree crossover? In other words, we would like to know the *algorithmic* distance between two solutions according to a particular representation, operators, and fitness measure. First, we define some notation:

- P is the population, containing M trees,
- T_1 is the tree we want to compute a distance from, or the parent tree,
- T_2 is the tree which we would like to transform T_1 into,
- T_1/T_2 is the difference of the two trees. This operator produces a tuple (s_{T_1}, s_{T_2}) , i.e. a pair of subtrees, 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 crossover distance between T_1 and T_2 depends on the ability to select s_{T_2} from some tree in P . Thus, the crossover distance³ between T_1 and T_2 also depends on the population P : if $T_2 \in P$, then s_{T_2} will also be in P . In the case where $T_2 \in P$, we could state that the distance is equal to 1, 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 $T_1 = T_2$. Following this idea, calculating a distance value in the light of multiple applications of an operator would need to consider if any applications of subtree crossover to T_1 would result in a new tree T'_1 that required subtree s'_{T_2} to transform T'_1 into T_2 . If $s'_{T_2} \in P$, the distance is 2, since it is possible to transform T_1 into T_2 with 2 crossover applications. To find distances greater than 2, we need to continue this process. This definition of operator-distance is an accurate reflection of the subtree crossover operator for a steady-state model with offspring-parent replacement. However, there are obvious problems with calculating this distance. Let us assume the average size of the M trees in P is k , and that crossover can choose any node in the population as the root of the new subtree. The number of potential intermediate trees T'_1 to consider for distances greater than 1 is $M \times k$. The distance calculation needs to be carried out for each of these trees to decide if the distance is 2.

Now, if we consider a generational model, which seems to be largely used in the GP community, an operator defined in terms of the population makes defining distance using multiple operator applications even more difficult. For example, we may consider the new tree T'_1 after an operator application on T_1

³ The generic term “metric” would probably be more suitable than the term “distance” here; anyway, we go on using the term “distance” for simplicity.

using the population P . In a generational algorithm, we will then need to consider the next population P' when thinking about another operator application on T'_1 . We can either create all the possible next populations or we can approximate them. Creating the future populations presents computational limitations. Similarly, we might create the future expected populations using calculations similar to the ones found in the schema theorems for GP [11]. However, finding the future expected populations is also costly, essentially requiring a similar amount of computation as actually running the GP algorithm.

Furthermore, let us assume that we calculate the distance between T_1 and T_2 as 1, meaning that one application of our operator to T_1 can build T_2 . However, when we actually execute our algorithm, it is not certain that this particular application will occur. If we calculate the distance between T_1 and T_2 as 3, which depends on two intermediate trees, we know with less confidence if either T'_1 or T''_1 will actually be produced. The decreasing amount of confidence we will have in the accuracy of distance values based on future generations is likely to make this type of distance measure less useful. Therefore, in practical applications of operator-based distance measures, it may only be useful to know the likelihood of creating a particular tree T_2 in the next generation.

To overcome the difficulty in defining a multiple operator distance, and to incorporate the stochastic properties of the algorithm, we can consider 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. While this measure would only consider one operator application, it will do so with a higher confidence and at significantly less computational effort than considering the future expected populations. However, we could attempt to approximate the creation and selection of subtrees in future populations using a similar method. We now look at this new idea of distance measure more closely.

The new operator-based distance can now be formulated as: given an operator V , trees T_1 and T_2 , and a population of trees P , can V be applied such that $V(T_1, P) = T_2$? That is, can an operator V , that uses the genetic material in P , be applied once to T_1 to produce T_2 ? If the answer is 'yes', we would not say that distance is 1, but we would bind this distance value to the probability of generating T_2 from the application of V to T_1 . Instead of asking for the required number of edit operations to transform a tree T_1 into another tree T_2 , we ask *how probable* is it that an operator will transform T_1 into T_2 . In other words, if the required genetic material to transform T_1 into T_2 is present in the population, how probable is it that our operator V selects it? In case T_1 and T_2 share no common material, this will be the probability that subtree crossover selects the root of T_1 and a subtree equal to T_2 . Note that using

this type of distance measure reporting a probability may pose a problem doing fitness-distance correlation studies. We are currently looking at ways to overcome these problems by accurately approximating the likely construction of missing subtrees. We will now describe this probability-based operator distance measure for subtree crossover.

3 Subtree Crossover Distance

We will consider a subtree crossover distance between two trees in a population in context of the genetic material contained in the population. Given the subtree crossover operator V_{SC} , a distance function can be defined by the following pseudo-code:

```

func distance( $T_1, T_2, V_{SC}, P$ ) {
    ( $s_{T_1}, s_{T_2}$ ) =  $T_1/T_2$ 
     $ps1$  = probSelecting( $s_{T_1}, T_1$ )
     $ps2$  = probCreating( $s_{T_2}, P$ )
    return  $ps1 * ps2$ 
}

```

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 crux of the subtree crossover operator-based distance is finding the probability of generating the subtree s_{T_2} . As T_2 will be in the current population, the function will report a non-zero value (this is the case if selecting the root-node of T_2 is possible).

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} . Determining the number and selection probability of each occurrence of $s_{T_2} \in P$ is in $O(M \times k)$, where k refers to the average size of an individual in the population. To carry out this search for each pair-wise distance computation would have a complexity in $O(M^3 \times k^2)$. Preprocessing the population prior to carrying out the pair-wise distance calculation can reduce this complexity.

So far we have limited our distance measure to the single application case, which would seem appropriate for standard GP as only one application of an operator is typically used to generate a new individual. We have also considered the whole population as a source of potential subtrees. However, as we know,

evolutionary algorithms use fitness-based selection to implement solution competition. Therefore, not all trees have the same likelihood of being selected as donor trees. We can use this fact to provide an effective way of reducing complexity of this operator distance while preserving the utility of the measure.

A way to reduce the complexity of the above subtree crossover-based definition, which can also be applied for other operator-based definitions, is to only consider those trees and their subtrees that are *likely* to be selected. We can define threshold values α for tree selection and β for subtree selection. Then, we can produce the set of fit (or likely to be selected) trees F according to the following definition:

$$F = \{\forall i \in P \mid \text{Better_Than}(i, \alpha)\},$$

where the predicate $\text{Better_Than}(i, \alpha)$ is True if individual i has better fitness than more than α individuals in the population. The value of α should reflect the behaviour of the selection method being used, e.g. in terms of the tournament size in tournament selection. We can find the set F in $O(M)$ time and are only required to do so once for the whole population (if we are calculating the pairwise measure). Now, for subtrees s and w in an individual $i \in F$, we can define the set R of the likely-to-be selected subtrees according to:

$$R = \{(\forall s \in i) \wedge (i \in F) \mid \frac{|\{\forall w \in i \mid |s| = |w|\}|}{|i|} > \beta\},$$

That is, a subtree s in individual i (from F) will be in R if the number of subtrees in i , with the same size as s , divided by the the total size of i is greater than β . In effect, we are only considering those subtrees of a given size that are likely to be selected by a uniform node selection probability. For example, in a full tree with 7 nodes (depth 3), and a uniform node selection probability, selecting a specific subtree with 3 nodes has probability of $\frac{1}{7}$. Selecting any (of the two) subtrees of size 3 has a probability of $\frac{2}{7}$, and selecting a subtree of size 7 has a probability of $\frac{1}{7}$. However, selecting a subtree of size 1 has probability of $\frac{4}{7}$. Thus, while it may not be the exact probability of selecting a specific subtree with 3 nodes, we can focus our attention only on those subtrees that are more likely to be selected due to their size. So, if subtree crossover needs to select a very large subtree to transform one tree into another, we might assume the likelihood of doing this will be so small that it is effectively 0. Additionally, if the missing subtree is only in a tree that has a very low chance of being selected, due to its poor fitness, again it may be reasonable to report a probability of selection as 0.

In summary, two ways to reduce the complexity of a subtree crossover-based distance are: (1) only consider the trees in the population that are likely to be selected, and (2) in those trees, only consider the subtrees that are likely to be selected. Tuning the parameters α and β will allow us to reduce the complexity of the computing the population pair-wise distance. However, a potential disadvantage of this approximation, particularly in discrete fitness spaces, is the fact that the number of individuals with fitness better than α individuals in the population can vary. Thus, if the population has no duplicating fitness values, this

approximation scheme may reflect reality. However, if the population contains only one unique fitness value, then all the individuals have the same probability of being selected. Preprocessing the population prior to computing the pair-wise edit distances can also consider α and β to reduce the memory demands of the distance calculation. We now complement the above discussion of operator-based distance measures with an empirical study of one such operator-based measure.

4 Experimental Results

In this section, we investigate the practical side of implementing an operator-based edit distance measure for the subtree crossover operator. To reduce the complexity of this study and its presentation, we use a constructed problem similar to the problems that emphasise solution structure, or tree shape, like the Lid [12] problem.

4.1 The GP System and Problem

The GP system is the same as above, but a new tree produced by subtree crossover replaces the worst fit one in the population (instead of the parent tree). Again, only subtree crossover is used, where node selection for subtree crossover is uniform. Tournament selection is of size 3, and a population of 20 trees is used. The functions are two-argument nodes that have no meaning. Primitives are empty, null nodes. Thus, our trees are binary, where internal nodes always have two child nodes. We use a maximum depth of 7 during subtree crossover, and initialise the population with full trees of depth 3, where these tree shapes are all identical.

A random tree shape is generated to define the goal state, or instance. A tree shape is generated by randomly picking an odd number between 16, to ensure trees are not too small, and 31, the size of a full tree of depth (7-2). We then randomly, with uniform probability, assign two child nodes to each available leaf node, starting with a root node. All leaf nodes that have depth less than the maximum are deemed available. This tree shape defines the instance. This construction method is similar to one found in [12].

The tree shape, defining an instance, is abstracted by representing it by the number of nodes present at each depth. Fitness is the absolute difference between the number of nodes at each depth in a candidate tree and the target random tree shape. For example, Figure 1(a) shows a random tree that defines an instance, and Figure 1(b) shows a perfect solution with the same number of nodes at each depth. Note that the shape of the solution is not identical and allows flexibility, as well as possible deception, in the search process.

We generate 30 random instances, and collect 30 random runs of the system for each instance, with a maximum of 500 generations (subtree crossover applications). The GP algorithm found an optimum solution 614 out of the 900 runs, the earliest of these at generation 16 and the latest at generation 499. The average generation where an optimum was found was 291, with a standard deviation of 117 generations.

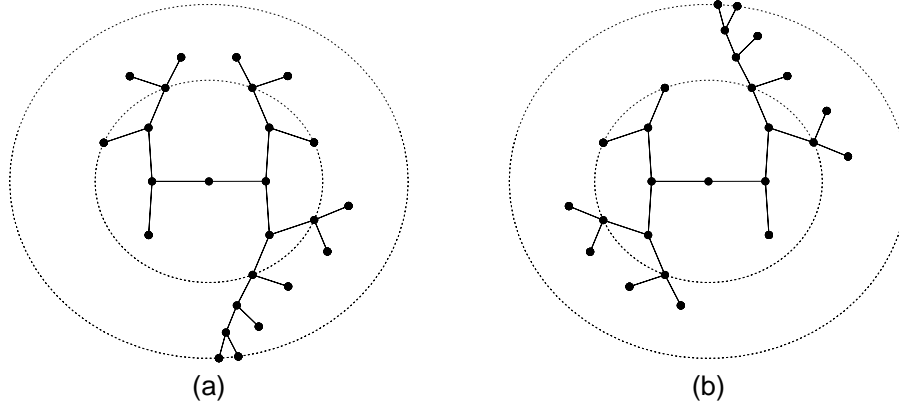


Fig. 1. An instance is defined by a randomly generated tree shape (a) and can be solved perfectly by a candidate tree shape that has the same number of nodes at each depth (b).

4.2 Operator-based Distance

In this study, we will use a simpler version of the operator distance defined in Section 3. Specifically, we will only consider the frequency of subtrees required to make two trees equal. We define subtree crossover distance D_{SC} between two trees T_1 and T_2 , given the population P , as:

$$D_{SC}(T_1, T_2, P) = \frac{\text{occur}(s_{T_2}, P)}{\#\text{subtrees}(P)}, \quad (1)$$

where $\text{subtrees}(P)$ returns the set of all subtrees in a population, and $\text{occur}(s, P)$ counts the number of occurrences of a subtree s in a population P . Earlier we defined the difference of two trees as $T_1/T_2 = (s_{T_1}, s_{T_2})$, where the resulting tuple defined the subtree in T_1 that needed to be replaced by s_{T_2} to make $T_1 = T_2$. We will define our operator-based distance for an average pair-wise distance measure in a population, thus $s_{T_2} \in P$. However, it is possible that s_{T_2} will only be represented by the tree T_2 itself, requiring that $s_{T_1} = T_1$.

4.3 Complexity

To calculate the average pair-wise distance using D_{SC} requires $M^2 - M$ distance calculations, where M is the number of trees in the population P . An edit distance pair-wise calculation would have an average complexity, assuming k is the average size of a tree in the population, of $O(k \times M^2)$. However, and of particular interest here, our operator-based distance D_{SC} , while having the same worst case bound, is likely to be less as the entire trees do not need to be explored. As soon as two subtrees that do not match are encountered, D_{SC} computes the

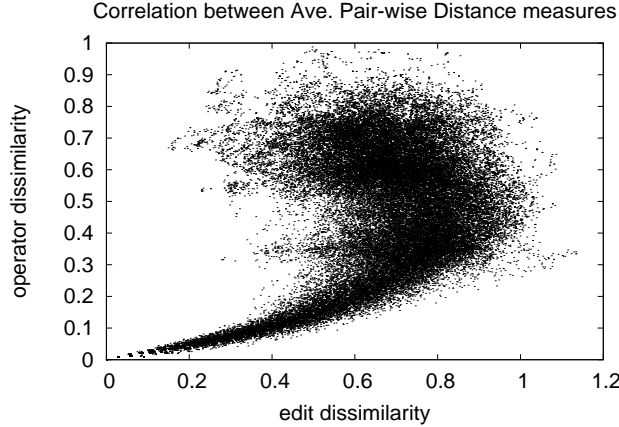


Fig. 2. A scatter plot showing the correlation between the edit distance and the operator-based distance, average pair-wise distance in each population.

frequency of the missing subtree and returns. That is, the complexity of D_{SC} is in $O(m \times M^2)$, where $m \leq k$. The function *subtrees* can incur a memory cost as it needs to store all unique subtrees in P and their frequency. However, this function only needs to be carried out once prior to each average pair-wise distance calculation, and requires linear time to visit all subtrees.

4.4 Pair-Wise Distance

Measures of distance or similarity can be useful for a variety of analysis. For example, the average pair-wise distance between all the trees in the population can indicate the amount of genetic material remaining in the population, as well as the likely behaviour of the operators. In both cases, it is important that the distance measure reflects the behaviour of the operator for meaningful results. We carried out two average pair-wise distance calculations using a standard edit distance and our subtree crossover operator-based distance. The edit distance measure counts the number of non-identical nodes between two overlapped trees. We normalise the average pair-wise distance of a population by dividing it by the average tree size (number of nodes) in that population. The operator-based distance measure divides the number of occurrences of the missing subtree by the total number of subtrees in the population, indicating the likelihood of selecting this missing subtree (but not with respect to fitness). We subtract this number from 1 to produce a measure of dissimilarity similar to the edit distance, where values close to 0 indicate high similarity, and values close to 1 indicate high dissimilarity.

Figure 2 shows the correlation between the two above measures. As the operator-distance is based on the frequency of missing subtrees in the populations, which tend to contain more and more subtrees in subsequent populations,

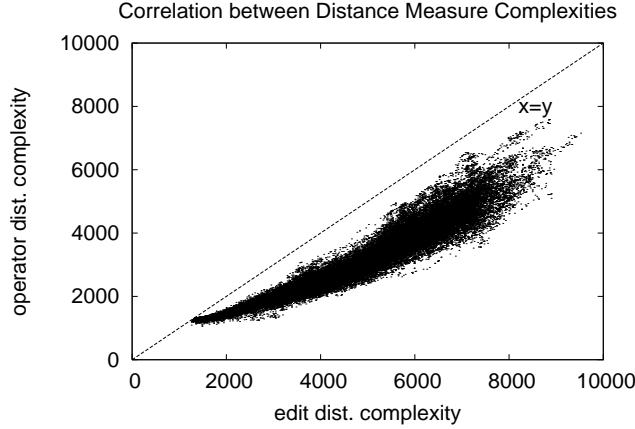


Fig. 3. A scatter plot showing the correlation between the edit distance and the operator-based distance complexities, or the average number of nodes visiting during the calculation of each pair-wise distance measure.

the range of these frequency values will also vary from population to population. Therefore, to visualise the frequencies with varying ranges more effectively, we multiply a population’s average pair-wise operator distance by the population’s average tree size. This value is then scaled for the $[0, 1]$ range. We can see that for low values of dissimilarity (~ 0 to 0.4), there is the expected positive correlation. However, as the edit distance dissimilarity increases, the operator-based dissimilarity takes on a wide range of values, and vice versa. Even in our simulations using a simplified problem of only tree shapes with no node contents and a fairly simple measure of operator distance, the disparity between the operator-based distance and the common edit distance is clear. That is, two trees that appear to be similar according to edit distance are not necessarily similar in terms of our operators.

4.5 Complexity Reductions

As mentioned earlier, there may be some cases where an operator-based distance measure can reduce the complexity of measuring distance. For example, using a basic string edit distance generally requires that all nodes of each tree need to be checked. However, if we are using only the subtree crossover operator, if the root of a subtree does not match with the root of another subtree in a second tree, we do not need to continue the comparison of these subtrees. That is, we know that the whole subtree will need to be replaced using subtree crossover to make the two trees equal.

Figure 3 shows the correlation between the complexities of each pair-wise distance measure. We can see that the edit distance complexity, approximated here as the average tree size in the population, grows at a faster rate than the

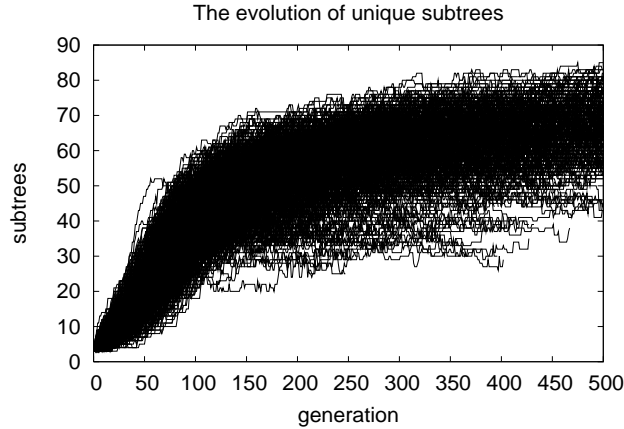


Fig. 4. The number of unique subtrees in each generation for all runs. Initially, with a population of full trees of depth 3, there are only three unique subtrees.

operator-based distance, where the former is a function of size and the latter a function of dissimilarity. However, this should be taken in the light of the higher cost of preprocessing the population prior to the operator-based distance calculation as well as the memory constraints involved in storing and looking up the frequencies of various subtrees. For example, Figure 4 shows the evolution of the number of unique subtrees in each population during the runs. For a population of size 20, with a depth limit of 7, there averages around 65 or so unique subtrees that need to be stored. While the representation used here is quite simple (binary trees with no node content), a memory requirement (number of unique subtrees) for computing operator distance that is within a constant multiple of the population size is promising.

5 Conclusions

This paper represents a first step in the study of the issues concerning operator-based distance measures for genetic programming. The variable shaped and sized solutions in the tree representation can make defining operator-based distance measures difficult. Therefore, it has become very common to use edit distance measures instead. Distance measures that do not capture the operator behaviour, however, are not always applicable for analytical studies like fitness distance correlation. Also, these measures need to be used carefully when studying population diversity. Thus, we aim to examine the practical difficulties in measuring distance using the canonical operator, subtree crossover. A series of possible definitions of operator-based distance were defined in this paper. Importantly, we discussed ways of reducing the complexity of such measures by means of various approximations. An empirical study showed how an edit distance measure and

an operator-based distance measure can fail to correlate. We showed in the simulations that operator-based distance can result in a reduction of complexity over an edit distance measure, where complexity is the number of nodes examined during a pair-wise distance calculation. Our future work is exploring other definitions of operator-based measures, for subtree crossover and other operators, 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. 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.
3. P. Collard M. Clergue M. Tomassini, L. Vanneschi. A study of fitness distance correlation as a difficulty measure in genetic programming. *Evolutionary Computation*, in press.
4. 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.
5. L. Vanneschi. *Theory and Practice for Efficient Genetic Programming*. Ph.D. thesis, University of Lausanne, Switzerland, 2004.
6. 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.
7. 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.
8. 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.
9. 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.
10. M. Wineberg and F. Oppacher. Distance between populations. In E. Cantú-Paz et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2724 of *LNCS*, pages 1481–1492, Chicago, USA, 2003. Springer-Verlag.
11. 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.
12. J.M. Daida, H. Li, R. Tang, and A.M. Hilss. What makes a problem GP-hard? validating a hypothesis of structural causes. In E. Cantú-Paz et al., editors, *Proceedings of the Genetic and Evolutionary Computation*, volume 2724 of *LNCS*, pages 1665–1677, Chicago, IL, USA, 12-16 July 2003. Springer-Verlag.