A Brief Overview of Population Diversity Measures in Genetic Programming

Nguyen Thi Hien[1], Nguyen Xuan Hoai[2]

School of IT, VietNamese Military Technical Academy
100 Hoang Quoc Viet Str, Ha Noi, VietNam

[1] hiennt@vnu.edu.vn
[2] nxhoai@gmail.com

Abstract. In the field of Genetic Programming (GP), there has been a growing interest in the effects of loss of genetic diversity, which causes the whole population prematurely converge to local optima. Improving diversity of the population is always an implicit goal of almost any basic genetic programming system. Most research in this area suggests a diversity measurement and controls this quantitative metric to maintain genetically diverse populations. In this paper, we brief overview of the measures used in Genetic Programming for diversity maintenance and promotion.

1 Introduction

Diversity is a crucial factor in the theory of natural selection, and when being used in genetic programming, it indicates the difference in structures or behaviours of individual in a population. In Genetic Programming (GP), population diversity has been long considered as an important research issue [4]. In general, genetic search will be more robust if the population contains more various individuals. The reason for this is that population diversity will encourage the exploration phase of the search and prevent the population from converging prematurely to local optima.

Unfortunately, the loss of diversity of the population at an early stage of the evolution is usually observed in GP [4]. Therefore, an implicit attempt to improve or maintain the population diversity is in almost every genetic programming systems [5], [7], [14],.. Those authors have proposed a numbers of measures for quantifying the GP population diversity according to some properties.

The focus of this paper is to give a brief review on the metrics (measures) which have been used in GP literature for measuring and controlling population diversity. Therefore, the organization of the paper is as follows. Section 2 first gives a classification and details of some diversity measures that have been used in Genetic Programming with discussion on the advantages and disadvantages of each measure. Finally, in section 3, we state my conclusions and some possible future research problems and directions in this area.

2 An Overview of Population Diversity Measures in GP

In order to maintain genetic diversity during the evolution, it is a common way to measure diversity by a metric and to control this value if necessary. Diversity metric studied in genetic programming is usually accomplished by defining a measure over some population features such as the individual fitness values, structures, or even the combination of the two [4]. Banzhaf et al. [33] logically classified all of these measures into two classes: genotypic diversity, which measures the structural differences between individual genotypes, and phenotypic diversity, which measures the differences in individual phenotypes. This section gives an overview of the up-to-date genotypic and phenotypic measures used in GP. We also give some personal comments and thoughts on the advantages and disadvantages of those measures. For the ease of comprehension, Figure 1 summarizes our further classification (developed from the some what rather simple and out-of-date classification of Banzhaf [33]).

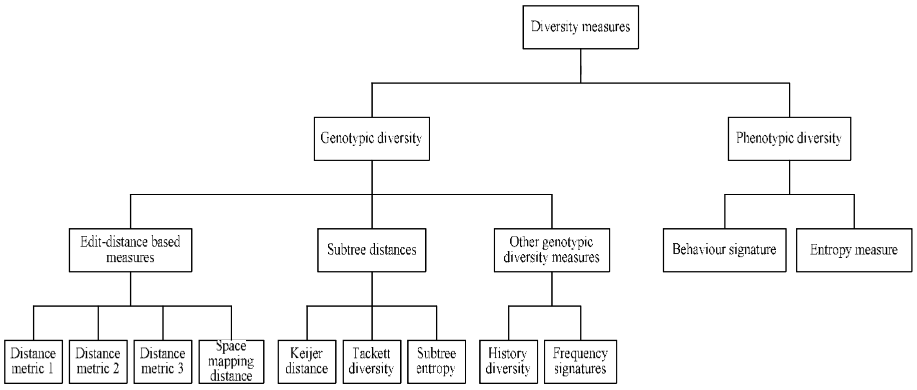

Fig. 1. A GP diversity classification scheme.

2.1 Genotypic diversity

This approach measures the population diversity as the quantification of the variety among the actual structures in the GP population – the tree, the graphs, or the linear genomes. It was perhaps originated from the term *variety* that Koza [10] used to indicate "is the percentage of individuals for which no exact duplicate exists elsewhere in the population". Although Koza's concept of genotypic diversity [10] is easy to understand and relatively exact for GP population diversity, it is probably too difficult to implement in practice due to time complexity of the proposed algorithm. Instead, in the genetic programming community, a number of 'approximate' genotypic diversity measures have been proposed. In this paper, we classify those measures in to three classes: edit-distance based measures, subtree based measures, and other genotypic measures. They are subsequently reviewed and discussed in this subsection.

2.1.1 Edit-distance based measures

The idea of using genotypic edit-distance in GP was originated from Genetic Algorithms (GAs), which is claimed to be the successor of GP. An example of such edit-distance in binary-coded GAs is the Hamming distance, in which the distance between two genotypes is measured by the number their different bits [33]. In other word, the Hamming distance between two GA binary chromosomes determines how many 'edit' operations (by bit flipping) needed to transform one chromosome to the other. However, it is notable that while GAs usually use linear and fixed-length chromosomes (genotypes), most GP systems employ non-linear structured (e.g tree or graph) and length-variant chromosomes. Consequently, it is expected that the edit-distance based measures in GP tend to be more various and complicated than in GAs.

In GP literature, there have been a number of measures based on edit-distance for genotypic diversity, and they are listed and discussed in the following:

Distance metric 1

O'Reilly [30] used Levenshtein distance to give an insight in to the genetic operators of GP. In this paper, we denoted this edit distance as distance metric 1. In distance metric 1, the amount of syntactic differences between two genetic programming trees is calculated as the shortest cost sequence of primitive operations used to transform from one tree to the other. These transformations include:

- Substitution: change label of one node in the tree to an other
- Insertion: add one node to the tree
- Deletion: remove a node from the tree

Figure 2 shows an example of distance metric 1.

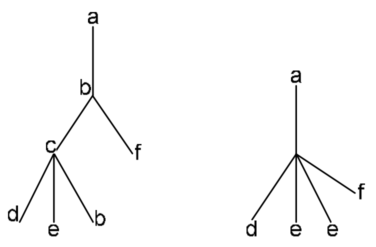

Fig. 2. A number of primitive operations to transform from left tree to right tree is 3 – (delete(b), delete(c), substitute(b, e))

Although Levenshtein has been proven to be a genuine metric on discrete tree structures [30], the implementation of this distance is complicated and its computing algorithm is time consuming. Therefore, it has not been used broadly in subsequent work on GP population diversity.

Distance metric 2

Alternative to Levenshtein edit distance, in [7], genotypic distance between two program trees is defined as the sum of the distance of the corresponding nodes. The corresponding nodes in the two compared trees are the nodes in the 'common area' of the two trees when they are aligned. The distance value between the two trees is then normalized by dividing by the size of the smaller tree of the two. Content distance between the two corresponding nodes is computed as follows. If the corresponding nodes are identical the distance is zero and otherwise is one. Figure 3 depicts an example of distance metric 2.

In [7], the individual diversity value is calculated as the average squared distance to the other members of the population. This individual diversity value is then used as a second objective to GP. In other word, the original single objective problem is then changed to multi-objective problem where GP is required to find the solutions to the original problem and maintain the population diversity (based on distance metric 2) at the same time.

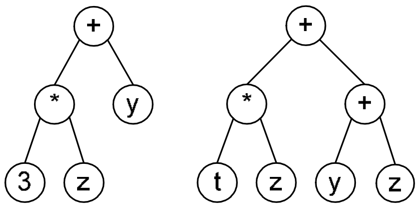

Fig. 3. The distance between these trees is 5/5.

The algorithm in [7] for computing distance between the two trees is simple and low time complexity (O(|T1|+|T2|)). However, one possible weak point of distance metric 2 compared to distance metric 1 could be figured out - the structural difference is calculated on the corresponding nodes only. While it makes the computation much easier and quicker (compared to distance metric 1), it also makes the distance as a less exact measure of differences since the overlapping is heavily dependent on the arities of the functions used in the function sets. It the arities of the function set is uniform (e.g binary), it is expected that this distance metric is a good and true measure of differences. On the other hand, if there are different arities in the function set, this distance tends to overestimate the distances between the two trees, compared to distance metric 1. Up to date, this distance has only been used in the frame work of multi-objective optimization.

Distance metric 3

Ekárt and Németh [1] investigated a distance of two genetic program trees (assuming that they are all binary trees), which is calculated in three following steps:

1. Making two genetic programs to be compared to the same tree-structure (adding NULL nodes if needed)
2. Counting the distance between any two symbols located at the same position in the two trees.
3. Combining the distances which computed in the previous step in a weighted sum to form the distance of the two trees.

Figure 4 presents an example of this distance metric.

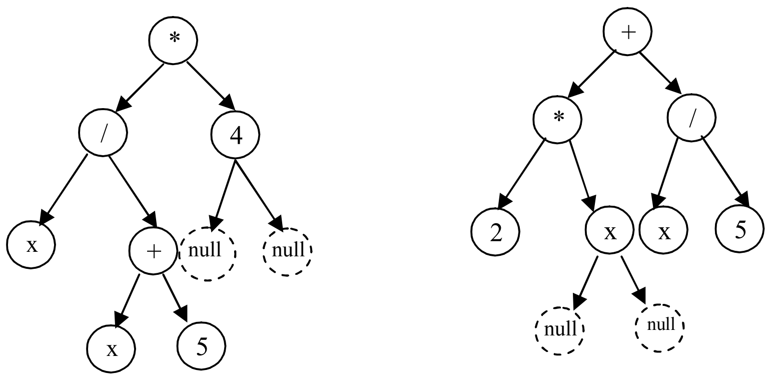

Fig. 4. The trees are completed by NULLs to have the same layout.

The first step of the above algorithm is to ensure that the two program trees are in the same in shape (totally overlapped). In the second step, to compute the distance between the corresponding tree nodes, they proposed that the program symbols are grouped in to n classes A_0, A_1,…, A_n , which are problem dependent. An example of this classification is A_0 being the set including NULL. The distance between the nodes is computed based on their class membership (Eq. 2). The distance between tree T_1 and T_2 is calculated in step 3 as:

$$dist(T_1,T_2)=\begin{cases} d(p,q)\, if\ neither\ T_1\ nor\ T_2\ have\ any\ children \\ d(p,q)+\frac{1}{K}*\sum_{l=1}^{m} dist(s_l,t_l)\ otherwise \end{cases} \quad \textbf{(1)}$$

where $K \geq 1$ is a constant and $T_1 = p(s_1, s_2, \ldots, s_m)$ has its root p and subtrees s_i, i=1,..,m, $T_2 = q(t_1, t_2, \ldots, t_n)$ has root q and subtrees t_i, i=1,..,n
d(x, y) – the distance between two node symbols $x \in A_i$ and $y \in A_j$ is calculated as:

$$d(x,y)=\begin{cases} 0 & if\ x=y \\ C*\frac{|x-y|}{\underset{v,w\in A_1}{Max}|v-w|} & if\ i=j=1 \\ \delta(i,j) & otherwise \end{cases} \quad (2)$$

where C is a constant, δ(i, j) is a function defined as follows:
Let δ: {0, 1, …, n} x {0, 1, …, n} → (0, +∞) be a function satisfies properties:

$$\begin{aligned} &\delta(i,j) \leq \delta(i,k)+\delta(k,j), \\ &\delta(i,j)-\delta(j,i) \\ &\delta(i,i) < \delta(j,k),\ j \neq k, \\ &for\ all\ i,j,k \in \{0,1,\ldots,n\} \end{aligned} \quad (3)$$

The time complexity of the algorithm for computing distance between two trees is small [1] (O(|T1|+|T2|)). The metric was general in the way that the user could use his/her problem dependent knowledge to define the quantitative semantic component in the differences between two trees. However, the metric defined in [1] does not take into account some aspect of the function nodes, such as commutativity. In the case that the operator presented by a function node is commutative, distance metric 3 could produce different values (where as they should not be counted as different). Example, if the two tree fragments X+ s_1 and s_1+X are compared they will add to the weighted sum of the distances a value more than zero, even though that these two fragments are actually the same.

Space mapping distance

In [21], a complex algorithm was proposed to measure the population diversity. Firstly, each of genotypes in the population is mapped to a point (x, y) in the coordinated two-dimensional space D_G. The origin of coordinate is the image of the most primitive tree – O_G - built from the function and terminal set. For each genotype *h*, its corresponding point (x_h, y_h) is calculated as follows. x_h is the number of positive node operations and y_h is the number of positive label operations to transform from Og into h. The operations are similar to the transformations in [30], but they are classified into positive and negative mode.

i) *Node operation*: Append an unlabeled node to certain node (positive mode); Delete an unlabeled terminal node (negative mode).

ii) *Label operation*: Increment the label of a node – the label with lower coding is replaced the label with higher coding (positive mode); Decrement the label of a node – the label with higher coding is replaced the label with lower coding (negative mode).

Next, the population diversity measure at generation *g* is computed as:

$$\Delta(g) = 1 - \frac{area(LERg)}{area(\Pi_{D_G})} \tag{4}$$

where D_G denotes the two-dimensional distance space for genospace G; LERg (largest-area rectangle) is the largest of all rectangle R that its opposite corners of R are the two image points in D_G and that there is no image point of any genotype in G contained in R, Π_{D_G} is approximate of D_G, which is the smallest rectangle containing all image points D_G.

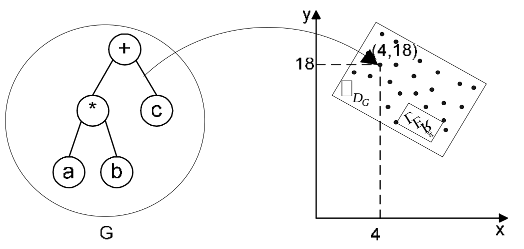

Fig. 5. The genospace-distance-space mapping.

The time complexity of the algorithm for computing the image points of D_G is no less than the time complexity of Levenshtein distance algorithm. In addition, it is also time-consuming to compute LER. Moreover, it is an open question as whether the two-dimensional space expresses completely the structural space.

2.1.2 Subtree distances

Alternative to edit-distance based measures, a number of authors have proposed some ways to compute genotypic tree distance for GP, based on subtrees. It is noted that almost all edit-distance based measures describes in subsection 2.1.1 are node-based. It means that, in order to compute how long a tree could be transformed to the other, those algorithms calculate and accumulate the differences on node-to-node basis. On the contrary, subtree distances, which are reviewed in this subsection, were subtree-based. It helps them to measure the structural differences between genotypic trees more directly than edit-distance based measures.

Keijzer distance

Keijer [12] investigated the distance $\delta_{dag}(X,Y)$ of two trees X and Y as:

$$\delta_{dag}(X,Y) = |D(X) \cup D(Y)| - |D(X) \cap D(Y)| \tag{5}$$

where D is defined as the set of all subtrees in a program tree; |D(X)| is the cardinality of the set D(X); $\cap$ and $\cup$ are respectively the intersection and the union of two sets.

The population diversity is measured by adding the number of subtrees that are removed in the previous generation and the number of subtrees that have just been created in the current generation.

Tackett diversity

Similar to [12], Tackett also used a subtree-based measure for qualitifying population diversity. He calculated the population diversity by dividing the number of newly created subtrees (schemata) in the current generation compared to the previous generation by the population size. To keep track the newly created subtrees at from generation to generation, [31] used a list to maintain all generated subtree, which makes the algorithm is memory consuming.

Subtree entropy

In [13], the authors used a subtree-based measure for computing the population genotypic diversity. However, the thing that distinguishes this work from the above is that, an information content measure (entropy – described in the next subsection) was employed.

2.1.3 Other genotypic diversity measures

History diversity

Another approach to measuring genotypic diversity was presented [14]. The methodology is to keep track each individual nodes from the initial population to the end. It is noted that as a result of GP crossover between two trees, a newly created tree is formed by two parts. One of these parts is inherited from the root parent – the parent that contributes the root to the offspring. This part is called the root part. The other part is inherited from the not-root parent and called the not-root part. To implement this, every node in the tree is assigned a label as ID:memID, where ID, memID are two integer numbers denoting its root parent ID and the ID of the newly created subtree through crossover. It means that when a child is produced by crossover, only nodes along the path from the root parent to the crossover point are tagged with the same ID as before crossover but their memID are newly created. Figure 6 depicts an example of this tagging scheme.

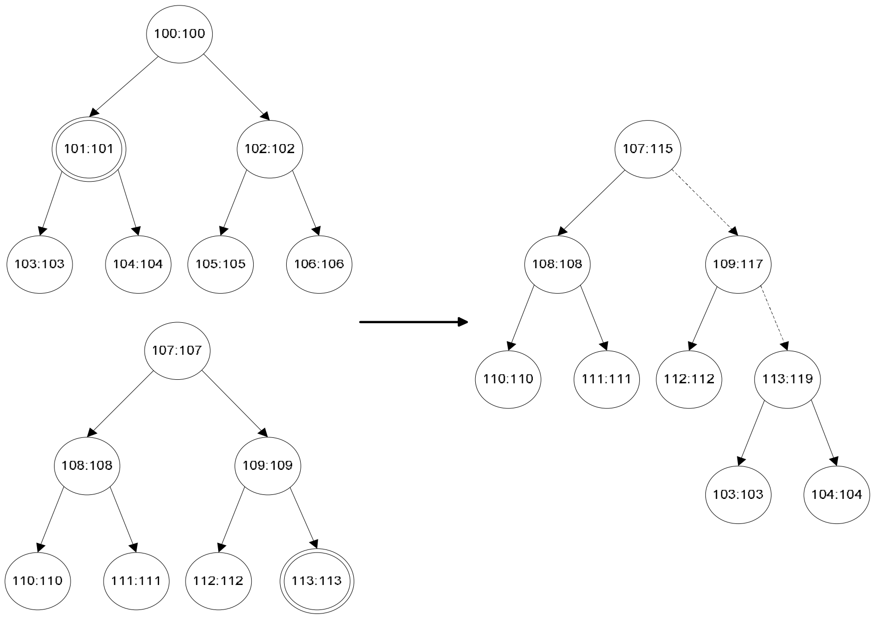

Fig. 6. The root parent and the non-root parent and the child resulting from crossover of two trees for an example.

Population genetic diversity is then calculated by counting a number of distinct IDs (either ID or memID) values appear in the population.

Frequency signature

While addressing the simulation of robot tanks, Patrik [18] et al explored another way to investigate population genotypic diversity. In his experiments, each of the terminal and function sets has six elements. he then defined the frequency signature as the frequency of occurrences for each of the twelve elements. The average of frequency signatures was then used to determine the population genotypic diversity.

2.2 Phenotypic diversity

The genotypic and structural approach to diversity presented in 2.1 could be problematic if introns are abundant in the population. Introns [33] are code (tree) fragments that do not affect the fitness of individual program trees. Consequently, two semantically identical program trees could be treated as two very different trees by genotypic measures described in 2.1, even though one program tree is just the other adding with introns. Therefore, there have been a number of attempts from GP researchers to measure the population diversity by the calculating variance in the performance of the phenotype. In other word, instead of measuring how differently the two (or more) programs look like, they measure how differently those program behave. In addition, one of the main advantages of genotypic diversity approach over the genotypic diversity one is the ease of computation. The algorithms for computing phenotypic diversity usually use the fitness values of the individuals only. Therefore the computation is often much more straight forward and much less time-consuming compared to the genotypic diversity approaches. In this subsection, we briefly overviewed some of such phenotypic diversity measure proposed in GP literature.

2.2.1 Behaviour signature

In robot tanks problem [18] mentioned above, the authors also investigated the population phenotypic diversity based on behaviour of each individual when trying it in combat against each of thirteen seed tanks. Comparing this value for every two individuals in the population determines the phenotypic diversity.

2.2.2 Entropy measures

The concept of entropy, as a degree of thermodynamic equilibrium, was used firstly by Clausius in 1865 in order to interpret the irreversibility in some kinds of transformations [24]. In 20th century, entropy was also used as a measure for information (started with work from Shanon [28]). In the context of G, entropy represents the amount of disorder of the GP population.

In [11], Rosca treated population phenotypic diversity as:

$$E(P) = -\sum_{k} p_k \, . \log p_k \tag{6}$$

where the population are partitioned according to fitness value, and p_k is the proportion of the population that have the fitness value in the fitness partition k^{th}.

3. Conclusion and future work

Similar to biological counterpart, population diversity also plays a very important role in the success of evolutionary algorithms in general, and GP in particular. The desire to adaptively control this diversity has attracted a number of GP researchers to study different quantitative measures for it.

In this paper, we tried to review and reclassify work in GP literature on different quantitative population diversity measures. Furthermore, the paper also showed that there are many measures used to define GP population diversity and that each of them has advantage and disadvantage sides, which GP-users should be aware of when using for different problem domains.

Our future work will include some comprehensive experimental comparison between those different population diversity measures and a study of the combinations of some of them.

Acknowledgements

The authors would like to thanks all members of the Natural Compuation Group at the Vietnamese Military Technical Academy (VMTA) for useful comments and suggestions related to this work. This work is partly funded by the Vietnamese Ministry of Science and Technology through a research grant for fundamental sciences.

References

1. A. Ekárt, S. Z. Nemeth, A metric for genetic programs and fitness sharing, in R. Poli et al., editors, Genetic Programming, Proceedings of the 3rd European Conference, volume 1802 of LNCS, pages 259–270, Edinburgh, 2000. Springer-Verlag.
2. —, Maintaining the diversity of genetic programs, in J. Foster et al., Eds, Proceedings of the 5th European Genetic Programming Conference, volume 2278 of LNCS, pages 162–171, Kinsale, Ireland, 3-5 April 2002. Springer-Verlag.
3. E. K. Burke, S. Gustafson, G. Kendall, A survey and analysis of diversity measures in genetic programming, in W. B. Langdon et al., editors, Proceedings of the Genetic and Evolutionary Computation Conference, pages 716–723, New York, 9-13 July 2002. Morgan Kaufmann Publishers.
4. —, Diversity in genetic programming: An analysis of measures and correlation with fitness, in IEEE Transactions on Evolutionary Computation, Vol. 8, No. 1, Feb 2004.
5. E. K. Burke, S. Gustafson, G. Kendall and N. Krasnogor, Advanced population diversity measures in genetic programming, in Proc. 7th Int. Conf. Parallel Problem Solving From Nature, vol. 2439, LNCS, J. J. M. Guervós et al., Eds, Granada, Spain, Sept. 2002, pp. 341-350.
6. —, Is increased diversity in genetic programming beneficial an analysis of lineage selection, Proceedings of the 2003 Congress on Evolutionary Computation CEC, Canberra, 2003, pp. 1398-1405.
7. E. D. de Jong, R. A. Watson, J. B. Pollack, Reducing bloat and promoting diversity using multi-objective methods, in Proc. Genetic Evolutionary Computation Conf., L.Spector et al., Eds., San Francisco, CA, July 7-11, 2001, pp. 11-18.
8. J. M. Daida, M. E. Samples, B. T. Hart, J. Halim, and A. Kumar, Demonstrating constraints to diversity with a tunably difficult problem for genetic programming (2004)
9. J.M. Daida et al (1999), Analysis of Single-Node (Building) Blocks in Genetic Programming, in L. Spector, W.B. Langdon et al, editors, Advances in Genetic Programming III, The MIT Press, 217-241, 1999.

10. J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MA: MIT Press, Cambridge, 1992.
11. J. P. Rosca, Entropy-driven adaptive representation, in Proc. Workshop Genetic Programming: From Theory to Real-World Applications, Justinian P. R., Ed., Tahoe City, CA, July 9, 1995, pp. 23-32.
12. M. Keijzer, Efficiently representing populations in genetic programming, in Advances in Genetic Programing II, Peter J. A. and Kenneth E. K., Jr., Eds. Cambridge, MA: MIT Press, 1996, ch. 13, pp 259-278.
13. N. Mori, A novel diversity measure of genetic grogramming, in Randomness and Computation Joint Workshop "New horizon in Computing" and "Statistical Mechanical Approach to Probabilistic Information Processing", 18-21 July, 2005, Sendai, Japan.
14. 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, Florida, USA, 1999. Morgan Kaufmann.
15. N. Geard, J. Wiles, Diversity maintenance on neutral landscapes: An argument for recombination, in Proc. IEEE Congress Evolutionary Computation, 2002, pp. 211-213.
16. N. I. Nikolaev, H. Iba, and V. Slavov, Inductive genetic programming with immune network dynamics, in Advances in Genetic Programing II, Lee S., William. B. L., Una-May. O. and Peter. J. A., Eds. Cambridge, MA: MIT Press, 1999, ch. 15, pp 355-376.
17. P. D'haeseleer, Context Preserving Crossover in genetic programming, in Proceedings of the 1994 IEEE World Congress on Computational Intelligence. IEEE Press, pp. 256-261.
18. P. D'haeseleer, J. Bluming, Effects of locality in individual and and population evolution, in Advances in Genetic Programming, Kenneth E. K., Jr., Ed. Cambridge, MA: MIT Press, 1994, ch. 8, pp.177-198.
19. R I (Bob) McKay, Fitness sharing in genetic programming, in Proc. Genetic Evolutionary Computation Conf., D. Whitley et al., Eds., Las Vegas, NV, July 10-12, 2000, pp. 435-442.
20. R I (Bob) McKay and H. A. Abbass, Anti-correlation: A diversity promoting mechanism in ensemble learning, The Australian J.Intell.Inform.Processing Syst., no.3/4, pp. 139-149, 2001.
21. R. E. Keller and W. Banzhaf, Explicit maintenance of genetic diversity on genospace, Internal Report, University of Dortmund, 1995.
22. R. E. Keller, W. Banzhaf, J. Mehnen, K. Weinert, CAD surface reconstruction from digitized 3D point data with a genetic programming/evolution strategy hybrid, in L. Spector, W.B. Langdon et al, editors, Advances in Genetic Programming III, The MIT Press, 217-241, 1999.
23. Sh.-H. N. Cheng, Distance between Herbrand interpretations: a measure for approximations to a target concept, in Pro. 7th Int. Workshop Inductive Logic Programming, N.Lavrac and S. Dzeroski, Eds., 1997, pp.213-226.
24. Sh-M. A. Yang, C-T. Sun, Ch-H. Hsu, Energy, matter, and entropy in evolutionary computation, in the Proc. Of 1996 International Conference on Evolutionary Computation 1996.
25. S. Gustafson, A. Ekárt, E. K. Burke, G. Kendall, Problem difficulty and code growth in genetic programming, in Genetic Programming and Evolvable Machines, 5, 271-290, Kluwer Academic Publishers.
26. S. Gustafson, L. Vanneschi, Operator-based distance for genetic programming: subtree crossover distance, Proceedings of the 8th European Conference on Genetic Programming, Springer Publishers, vol. 3447, 2005, pp. 178-189.
27. Terence S., James A. F., Effects of code growth and parsimony pressure on populations in genetic programming, Evol. Comput., vol. 1, no. 2, pp. 293-309, 1998.
28. T. M. Cover, J. A. Thomas, Elements of Information, A Wiley-Interscience Publication John Wiley & Sons, INC, 1991.

29. T. F. Bersano-Begey, Controlling exploration, diversity and escaping local optima in genetic programming, in J.R. Koza, editor, Late Breaking Papers at the Genetic Programming Conference, pages 7–10, Stanford University, CA, July 1997.
30. 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, Florida, USA, 1997.
31. W. A. Tackett, Recombination, selection, and the genetic construction of computer programs, Ph.D. dissertation, Dept. Elec.Eng.Syst., Univ. Southern California, Los Angeles, 1994.
32. W. B.Langdon, R. Poli, Foundations of Genetic Programming, Springer-Verlag, 2002 .
33. W. Banzhaf, P. Nordin, R. E. Keller, F. D. Francone, Genetic programming ~ An introduction, Morgan Kaufmann Publishers, 1998.