An Investigation of Fitness Sharing with Semantic and Syntactic Distance Metrics

Quang Uy Nguyen[1], Xuan Hoai Nguyen[2],
Michael O'Neill[3], and Alexandros Agapitos[3]

[1] Faculty of Information Technology, Military Technical Academy, Vietnam
[2] IT Research and Development Center, Hanoi University, Vietnam
[3] Natural Computing Research & Applications Group,
University College Dublin, Ireland
{m.oneill,alexandros.agapitos}@ucd.ie,
{quanguyhn,nxhoai}@gmail.com

Abstract. This paper investigates the efficiency of using semantic and syntactic distance metrics in fitness sharing with Genetic Programming (GP). We modify the implementation of fitness sharing to speed up its execution, and used two distance metrics in calculating the distance between individuals in fitness sharing: semantic distance and syntactic distance. We applied fitness sharing with these two distance metrics to a class of real-valued symbolic regression. Experimental results show that using semantic distance in fitness sharing helps to significantly improve the performance of GP more frequently, and results in faster execution times than with the syntactic distance. Moreover, we also analyse the impact of the fitness sharing parameters on GP performance helping to indicate appropriate values for fitness sharing using a semantic distance metric.

Keywords: Genetic programming, Fitness sharing, Semantic, Syntactic.

1 Introduction

Genetic Programming (GP) [1,2] is an evolutionary paradigm for automatically finding solutions for a problem. Since its introduction, GP has been applied to a wide range of fields [1], and routinely exhibits human-competitive performance [3]. In GP, one of the crucial properties that strongly affects its performance is the diversity and dispersion of the population [4,5,6,7]. The diversity and dispersion of a population represents its ability to cover different parts of the search space. Therefore, promoting dispersion and diversity is important for the efficiency of search. There have been a number of methods for enhancing diversity and dispersion [8,5,6,9,10,11], of these fitness sharing has been widely used in Genetic Algorithms (GA) and Genetic Programming.

In Genetic Algorithms, fitness sharing was introduced as a technique for maintaining population diversity [12,13]. The basic idea is to cluster the population into a number of groups, based on their similarity with respect to a distance metric. Members of the same group are penalized by having to share fitness, while isolated individuals retain the full reward. In GP, Langdon is perhaps the first person who used fitness sharing to preserve population diversity [14]. In Langdon's work, the distance metric is based on the fitness of individuals. Then, Ekart and Nemeth [15] proposed a metric for fitness sharing based on syntactic (structural) distance between two tree-based individuals. The method was applied to a symbolic repression problem with some success. Following this McKay [16] used implicit fitness sharing, in which the reward for each fitness case is shared by all individuals that give the same output. However, this method is only applied to Boolean problems and is not available to directly apply to continuous real-valued problems.

In this paper, we propose an approach to fitness sharing based on a semantic distance metric. We compared the performance of GP using fitness sharing with semantic and syntactic metrics. We also analyse the impact of the parameters of fitness sharing with semantic distance on the performance of GP. The remainder of the paper is organised as follows. In the next section, we briefly describe fitness sharing, the way we modify fitness sharing to speed up its execution and two distance metrics used for implementing fitness sharing. The experimental settings are detailed in Section 3. The results of the experiments are presented and discussed in section 4. Section 5 concludes the paper and highlights some potential future work.

2 Methods

This section briefly presents fitness sharing. The manner in which we modify fitness sharing is discussed, and following this two distance metrics for implementing fitness sharing are detailed.

2.1 Fitness Sharing

Fitness sharing treats fitness as a shared resource of the population, and thus requires that similar individuals share their fitness. It lowers each population element's fitness by an amount equal to the number of similar individuals in the population. Typically, the shared fitness f'_i of an individual with the raw fitness f_i is simply calculated as follows [17].

$$f'_i = \frac{f_i}{m_i} \tag{1}$$

where m_i is the niche count which measures the approximate number of similar individuals with whom the fitness f_i is shared. The niche count of individual i is calculated by summing a sharing function over all members of the population.

$$m_i = \sum_{j=1}^{N} sh(d_{ij}) \tag{2}$$

where N denotes the population size and d_{ij} represents the distance between the individual i and the individual j. Hence, the sharing function sh measures the similarity level between two population elements, it returns one if the elements are identical, zero if their distance d_{ij} is higher than a threshold of dissimilarity, and an intermediate value at intermediate level of dissimilarity. The most widely used sharing function is given as follows:

$$sh(d_{ij}) = \begin{cases} 1 - (d_{ij}/\sigma)^{\alpha} & \text{if } d_{ij}<\sigma \\ 0 & otherwise; \end{cases} \tag{3}$$

where σ denotes the threshold of dissimilarity (also the niche radius) and α is a constant parameter which regulates the shape of the sharing function. While α is commonly set to one with the resulting sharing function referred to as the triangular sharing function [13], in this paper, several values of σ will be tested to find a range of suitable values for symbolic regression problems.

The distance d_{ij} between two individuals i and j is characterized by a similarity metric based on either semantic or syntactic similarity. In this paper, we will compare the efficiency of fitness sharing with two similarity metrics: semantic versus syntactic similarity.

2.2 Modifying Fitness Sharing

In our experiments, we modify fitness sharing to speed up its execution. The first modification is the way to calculate the shared fitness. It can be seen that the shared fitness calculated in Equation 1 can only be used if the raw fitness favors bigger values, meaning that the bigger value is better. Since for symbolic regression, we use the raw fitness as the mean of the absolute error with the condition that smaller values are better, it can not directly use Equation 1 to calculate the shared fitness. Instead we use the following equation for quantifying the shared fitness.

$$f_i' = (f_i) * (m_i + 1) \tag{4}$$

The main drawback of fitness sharing is that the computation of the shared fitness for the entire population in each generation can be very time-consuming [17]. We alleviate this by calculating the niches for only a small subset of the population that is randomly sampled from the whole population. Let P be the number of individuals that are randomly sampled from the population for each individual niche that is being calculated. In our experiments, we will investigate different values of P to find the appropriate values for GP.

2.3 Syntactic Distance

To implement fitness sharing using a syntactic metric, a syntactic distance between any two trees is required. In this paper, we use an extended version of tree distance that has been use by Ekart and Nemeth [15]. In other words, the syntactic distance between two trees is calculated as follows:

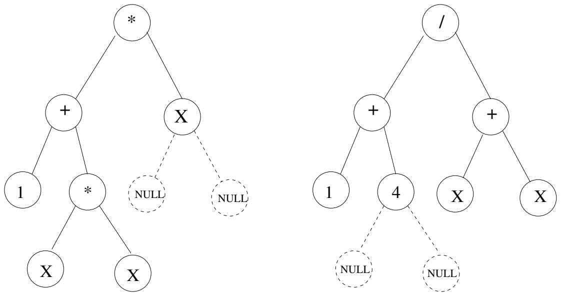

Fig. 1. Two trees are added the NULL nodes to have the same layout

1. Make the two trees to be compared to have the same tree-structure (adding NULL nodes if necessary). Figure 1 gives an example of two trees which are completed by adding NULL nodes so that they have the same structure.
2. Count the distance between any two nodes located at the same position in the two trees. If two nodes are labeled with the same symbol, the distance between them is 0, otherwise the distance is 1.
3. Sum the distances computed in the previous step to form the distance of the two trees.

2.4 Semantic Distance

To calculate the semantic distance between two individuals, a way to quantify their semantics must first be defined. In this paper, we use Sampling Semantics that has been used in previous work on semantic based crossovers [18,19]. Formally, sampling semantics between two trees (subtrees) is defined as follows:

Let F be a function expressed by a (sub)tree T on a domain D. Let P be a set of points sampled from domain D, $P = \{p_1, p_2, ..., p_N\}$. Then the *Sampling Semantics* of T on P on domain D is the set $S = \{s_1, s_2, ..., s_N\}$ where $s_i = F(p_i), i = 1, 2, ..., N$.

The value of N depends on the problem. If it is too small, the approximate semantics might be too coarse-grained and not sufficiently accurate. If N is too big, the approximate semantics might be more accurate, but more time consuming to measure. The choice of P is also important. If the members of P are too closely related to the GP function set (for example, π for trigonometric functions, or e for logarithmic functions), then the semantics might be misleading. For this reason, in this paper, the number of points for evaluating sampling semantics is set as the number of fitness cases of the problem (20 points), and we choose the set of fitness cases as the sample points for evaluating sampling semantics.

Based on *Sampling Semantics* (SS), we define a *Sampling Semantics Distance* between two trees. In the previous work [19], *Sampling Semantics Distance* (SSD) was defined as the sum of absolute difference of all values of SS. While the experiments show that this kind of SSD is acceptable, it has undoubted weakness that the value of SSD strongly depends of the number of SS points

(N) [19]. To soften this drawback, in this paper we use the mean of absolute distance as the SSD between trees. In other words, let $U = \{u_1, u_2, ..., u_N\}$ and $V = \{v_1, v_2, ..., v_N\}$ be the SS of $Tree_1(Tr_1)$ and $Tree_2(Tr_2)$ on the same set of evaluating values, then the SSD between Tr_1 and Tr_2 is defined as follows:

$$SSD(Tr_1, Tr_2) = \frac{|u_1 - v_1| + |u_2 - v_2| + ... + |u_N - v_N|}{N} \tag{5}$$

Since it could be expensive to compute SS, we reduce the cost by caching. The SS of each subtree is stored in the root node using attributes; the resulting GP system is known as *Attributes Genetic Programming* (AGP). In more detail, assume that the problem has N fitness cases; then N attributes are added to each node in the individual's tree. In figure 2 N is set to 3, so three attributes A_1, A_2, A_3 are added to every node, to cache the SS of the corresponding subtree.

Figure 2 also describes the process of evaluating attribute values in AGP. Initially (Figure 2a), the attributes are set to zero. Assume that the fitness cases include three values 0, 0.5, and 1, then, in the second step, the attributes of the leaves of the individual are assigned with these values (Figure 2b, attributes at the nodes labeled with a constant are assigned with the value of that constant). Next, the attributes at the level above the leaves are assigned with values. At this point, the semantics of the leaves is passed upward to their parents, and the operator at those nodes are applied to calculate the values for the attributes (Figure 2c) at these nodes. This process is then continued until the attributes at the root node are assigned with values (Figure 2d). It is noted that when this process of value propagation completes, the fitness of the individual can be ob-

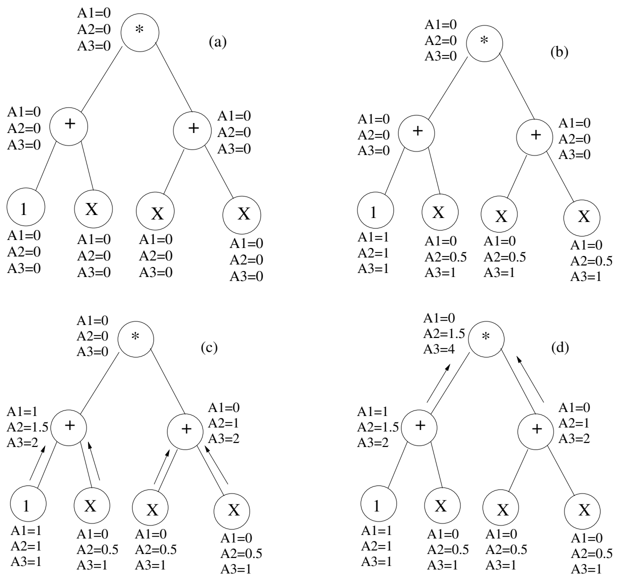

Fig. 2. An individual in AGP and the Process of Evaluating its Attributes

Table 1. Symbolic Regression Functions

Functions	Training Data
$F1 = x^3 + x^2 + x$	20 random points $\subseteq$ [-1,1]
$F2 = x^4 + x^3 + x^2 + x$	20 random points $\subseteq$ [-1,1]
$F3 = x^5 + x^4 + x^3 + x^2 + x$	20 random points $\subseteq$ [-1,1]
$F4 = x^6 + x^5 + x^4 + x^3 + x^2 + x$	20 random points $\subseteq$ [-1,1]
$F5 = (x + 1)^3$	20 random points $\subseteq$ [-1,1]
$F6 = x^3 - x^2 - x - 1$	20 random points $\subseteq$ [-1,1]
$F7 = 0.3sin(2\pi x)$	20 random points $\subseteq$ [-1,1]
$F8 = cos(3x)$	20 random points $\subseteq$ [-1,1]

Table 2. Run and Evolutionary Parameter Values

Parameter	Value
Population size	500
Generations	50
Selection	Tournament
Tournament size	3
Crossover probability	0.9
Mutation probability	0.05
Initial Max depth	6
Max depth	15
Max depth of mutation tree	5
Non-terminals	+, -, *, / (protected version), sin, cos, exp, log (protected version)
Terminals	X, 1
Raw fitness	mean absolute error on all fitness cases
Trials per treatment	100 independent runs for each value

tained by comparing the semantics of the root node with the values of the target function on the corresponding fitness cases and the semantic distance between two trees can be calculated by summing the attributes of their root nodes. This helps to speed up the calculation of semantic distance between individuals in fitness sharing.

3 Experimental Settings

To investigate the impact of using these distance metrics in fitness sharing on GP performance, we used eight real-valued symbolic regression problems. The problems and training data are shown in Table 1. These functions were taken from previous work on using semantics based operators in GP [20,21].

The GP parameters used for our experiments are shown in Table 2. It should be noted that the raw fitness is the mean of absolute error on all fitness cases.

Therefore, the smaller values are better. For each problem and each parameter setting, 100 runs were performed.

We divided our experiments into two sets. The first is to compare the performance of fitness sharing using a semantic metric with a syntactic metric and with standard GP, and the second set aims to investigate the impact of some parameters (the number of the sampled individuals and the niche radius, σ) on the performance of GP using fitness sharing with semantic metric. Hereafter, the fitness sharing using the semantic metric is called *Semantic Sharing* and the fitness sharing using the syntactic metric is called *Syntactic Sharing.*

4 Results and Discussion

This section first presents the comparison on the performance of GP using semantic sharing with syntactic sharing and standard GP. After that the impact of some parameters on the performance of GP using semantic sharing is discussed.

4.1 On the Performance

We tested fitness sharing using semantic and syntactic distance on the above eight problems. For semantic sharing, we selected the niche radius, σ at 0.1. This is the value determined to achieve the best performance with semantic sharing (the following subsection will investigate the impact of σ on the performance of GP using semantic sharing). Three values for the number of the individuals that are sampled to calculate niche were tested. They are 5, 10 and 15. Semantic sharing with these values will be shorthanded as SS5, SS10, and SS15.

For syntactic sharing, we fixed the niche radius, σ at 10. This was indicated from experiments as the best value for GP performance. Similarly, three values for the number of sampled individuals are 5, 10 and 15 were tested. Syntactic sharing with these three values are referred to as SyS5, SyS10, and SyS15.

To measure the performance of GP with these approaches we use a classical performance metric: mean of the best fitness. Table 3 shows the best fitness found, averaged over all 100 runs of each GP system. We tested the statistical significance of the results in Table 3 using a Wilcoxon signed-rank test with a confidence level of 95%. In Table 3, if a run of semantic sharing or syntactic sharing is significantly better than Standard GP (GP), its result is printed bold face.

It can be seen from Table 3 that syntactic sharing barely improves the performance of GP. Sometimes, syntactic sharing is even worse than standard GP. This can be observed in some cases on Function F2, F5, F7 and F8. In fact, syntactic sharing only significantly improves GP performance on three occasions, namely on F1 (SyS10 and SyS15) and F4 (SyS15). These results are not entirely surprising as Ekart and Nemeth [15] also showed that fitness sharing based on structural distance provides very little advantage for GP performance.

On the contrary, semantic sharing always helps to improve the performance of GP. It can be seen from Table 3 that the mean best fitness found by semantic sharing is consistently smaller than the value found by standard GP. For the

Table 3. Mean best fitness of three methods. Note that the values are scaled by 10^2

Methods	F1	F2	F3	F4	F5	F6	F7	F8
GP	1.05	1.52	2.14	2.78	2.65	3.17	4.40	1.50
SyS5	0.85	1.39	1.86	2.56	3.00	2.90	4.44	1.73
SyS10	**0.78**	1.73	1.94	2.57	2.52	3.01	4.58	1.37
SyS15	**0.69**	1.37	1.78	**2.06**	2.58	2.97	4.33	1.56
SS5	**0.69**	**1.13**	1.66	**2.09**	**2.22**	**2.56**	**3.87**	**1.12**
SS10	**0.55**	**1.12**	1.70	**2.03**	2.32	**2.48**	**3.62**	**1.11**
SS15	**0.75**	**1.19**	1.71	**2.14**	**2.15**	**2.59**	**3.91**	**0.97**

three values of the numbers of the individuals that are sampled for calculating the niche, it can be seen that the performance of semantic sharing is consistent. The table also shows that the majority of the improvement achieved by semantic sharing over standard GP is statistically significant.

Table 4. Average time of a run (in seconds) for the two fitness sharing strategies

Methods	F1	F2	F3	F4	F5	F6	F7	F8
GP	4.36	5.09	4.85	5.32	5.30	6.44	7.47	5.85
SyS5	14.2	15.5	15.8	16.8	16.8	18.7	22.7	17.9
SyS10	23.1	26.4	27.4	28.8	27.6	32,8	36.6	28.8
SyS15	33.2	36.2	38.6	39.7	36.4	41.2	45.9	39.7
SS5	5.21	5.42	5.26	5.37	5.31	6.34	7.11	5.52
SS10	5.12	5.69	5.30	5.61	5.52	6.59	7.34	5.94
SS15	4.94	5.62	5.38	5.54	5.59	6.48	7.38	5.90

As has been previously mentioned, one of the weaknesses of fitness sharing is that it takes time to calculate the semantic or syntactic distance between individuals in the population. To estimate the extra time of these methods, we measured their running time. The average time of a run of these methods compared to standard GP is shown in Table 4.

It can be seen from Table 4 that it is very time-consuming to implement syntactic sharing. The average time of a run of syntactic sharing is much greater than the value of standard GP. When the number of individuals that are sampled increases, the average running time also increases. Conversely, semantic sharing runs almost at the same speed as standard GP. The average run time of semantic sharing is mostly equal to that of standard GP, and in some cases these values are even smaller. This represents the effectiveness of using attributes to store semantics in the calculation of semantic distance.

Table 5. Mean best fitness of semantic sharing with different values of the niche radius. Note that the values are scaled by 10^2

Methods	F1	F2	F3	F4	F5	F6	F7	F8	*Mean*
GP	1.05	1.52	2.14	2.78	2.65	3.17	4.40	1.50	*2.40*
SSR005	0.61	1.25	1.75	2.19	2.43	2.70	3.71	1.20	*1.98*
SSR0075	0.52	1.11	1.66	2.23	2.31	2.56	3.61	1.14	*1.89*
SSR01	0.55	1.12	1.70	2.03	2.32	2.48	3.62	1.11	*1.87*
SSR0125	0.53	1.34	1.79	2.16	2.25	2.70	3.92	1.14	*1.98*
SSR015	0.61	1.35	1.74	2.07	2.45	2.82	3.92	1.13	*2.01*

4.2 Parameters Analysis

This section analyses the impact of some parameters on the performance of GP using semantic sharing [1]. There are two parameters that potentially impact the performance of GP with semantic sharing: the threshold of dissimilarity (also the niche radius), σ and the size of sampled individuals P. To investigate the sensitivity of the niche radius on GP performance, we fixed the size of sampled individuals at 10 and tested 5 values of σ. The five values tested are: 0.05, 0.075, 0.1, 0.125, 0.15. Five configurations of semantic sharing with these values are referred to as SSRX with X=0.05, 0.075, 0.1, 0.125, 0.15.

To estimate the effect of changing σ, we recorded the best fitness of a run. These values were averaged over 100 runs and are shown in Table 5. For the purpose of comparison, the mean best fitness of standard GP is also shown in the top row of this table.

It can be seen from Table 5 that the values of the niche radius around 0.1 are good values overall. The performance of semantic sharing with values 0.075 and 0.1 are the most consistent. When this value is too small (0.05) or too great (0.125 and 0.15), the performance is worse.

We now examine the impact of the second parameter, the size of sampled individuals, on the performance of GP using semantic sharing. To do this, we fixed σ at 0.1 and 6 values of the size of sampled individuals were tested. The six values are 5, 10, 15, 20, 40, and 80. The corresponding configurations of semantic sharing with these six values are shorthanded as SSSX with X=5, 10, 15, 20, 40, and 80, respectively.

To discover the effect of changing this parameter, we again recorded the best fitness of a run. These values were averaged over 100 runs and are shown in Table 6. For the purpose of comparison, the mean best fitness of standard GP is also shown in the top row of this table.

Table 6 shows that the size of sampled individuals needs only relatively small values. It can be seen that the performance of semantic sharing is best with the values from 5 to 20 (1 to 4% of the total population size). If this value is

[1] Since the performance of syntactic sharing is not as good as the performance of semantic sharing, it is not investigated further in this paper.

Table 6. Mean best fitness of semantic sharing with different values of the size of sampled individuals. Note that the values are scaled by 10^2.

Methods	F1	F2	F3	F4	F5	F6	F7	F8	*Mean*
GP	1.05	1.52	2.14	2.78	2.65	3.17	4.40	1.50	*2.40*
SSS5	0.69	1.13	1.66	2.09	2.22	2.56	3.87	1.12	*1.92*
SSS01	0.55	1.12	1.70	2.03	2.32	2.48	3.62	1.11	*1.87*
SSS15	0.75	1.19	1.71	2.14	2.15	2.59	3.91	0.97	*1.93*
SSS20	0.58	1.23	1.65	1.99	2.41	2.69	3.72	1.17	*1.93*
SSS40	0.85	1.61	2.17	2.48	2.32	3.04	4.06	1.24	*2.22*
SSS80	1.14	1.52	2.33	2.56	2.53	2.96	4.72	1.35	*2.39*

too great (40 and 80) the performance is worse. This can be explained by the fact that promoting too much diversity (when increasing the value of the size of sampled individuals) can hinder the convergence of GP to the global optimal.

5 Conclusions and Future Work

In this paper, we investigate the efficiency of fitness sharing using semantic and syntactic distance metrics. We propose a novel way to implement fitness sharing using a semantic distance metric based on sampling semantics. We also modify fitness sharing to speed up its execution. We compare the performance of Genetic Programming using fitness sharing with semantic and syntactic distance on a class of real-value symbolic regression problems. The experimental results show on the tested problems that while fitness sharing with the syntactic metric hardly improves the performance of GP, fitness sharing with the semantic metric often significantly improves GP performance. At the same time, fitness sharing that implements the semantic distance metric runs much faster than with the syntactic metric. Further analysis shows the impact of the two main parameters on the performance of fitness sharing with the semantic distance metric.

There are a number of areas for future work which arise from this paper. First, we want to measure the change of semantic diversity and syntactic diversity of fitness sharing implemented in this paper during the course of evolution to understand its impact on GP performance. Second, we would like to combine promoting semantic diversity with controlling semantic locality [20] to see if it provides additional improvement in performance. Last but not least, we aim to investigate the impact of this method on dynamic problems where maintenance of population dispersion/diversity is critical for adaptation to a changing environment.

Acknowledgements. This work was funded by The Vietnam National Foundation for Science and Technology Development (NAFOSTED), under grant number 102.01-2011.08. The third author acknowledges the support of Science Foundation Ireland under grant number 08/IN.1/I1868. A. Agapitos is supported by Science Foundation Ireland Grant No. 08/SRC/FM1389.

References

1. Poli, R., Langdon, W., McPhee, N.: A Field Guide to Genetic Programming (2008), `http://lulu.com`
2. Koza, J.: Genetic Programming: On the Programming of Computers by Natural Selection. MIT Press, MA (1992)
3. Koza, J.: Human-competitive results produced by genetic programming. Genetic Programming and Evolvable Machines 11(3-4), 251–284 (2010)
4. Gustafson, S., Burke, E.K., Kendall, G.: Sampling of Unique Structures and Behaviours in Genetic Programming. In: Keijzer, M., O'Reilly, U.-M., Lucas, S., Costa, E., Soule, T. (eds.) EuroGP 2004. LNCS, vol. 3003, pp. 279–288. Springer, Heidelberg (2004)
5. Burke, E.K., Gustafson, S., Kendall, G.: Diversity in genetic programming: An analysis of measures and correlation with fitness. IEEE Transactions on Evolutionary Computation 8(1), 47–62 (2004)
6. Looks, M.: On the behavioral diversity of random programs. In: GECCO 2007: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, July 7-11, vol. 2, pp. 1636–1642. ACM Press (2007)
7. O'Neill, M., Vanneschi, L., Gustafson, S., Banzhaf, W.: Open issues in genetic programming. Genetic Programming and Evolvable Machines 11(3-4), 339–363 (2010)
8. Gustafson, S.: An Analysis of Diversity in Genetic Programming. PhD thesis, School of Computer Science and Information Technology, University of Nottingham, Nottingham, England (February 2004)
9. Beadle, L., Johnson, C.G.: Semantic analysis of program initialisation in genetic programming. Genetic Programming and Evolvable Machines 10(3), 307–337 (2009)
10. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers (2011)
11. Morrison, R.: Designing Evolutionary Algorithms for Dynamic Environments. Springer, Heidelberg (2004)
12. Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor (1975)
13. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
14. Langdon, W.B.: Genetic Programming and Data Structures: Genetic Programming + Data Structure = Automatic Programming! Kluwer Academic, Boston (1998)
15. Ekárt, A., Németh, S.Z.: A Metric for Genetic Programs and Fitness Sharing. In: Poli, R., Banzhaf, W., Langdon, W.B., Miller, J., Nordin, P., Fogarty, T.C. (eds.) EuroGP 2000. LNCS, vol. 1802, pp. 259–270. Springer, Heidelberg (2000)
16. McKay, B.: An investigation of fitness sharing in genetic programming. The Australian Journal of Intelligent Information Processing Systems 7(1/2), 43–51 (2001)
17. Sareni, B., Kraehenbuehl, L.: Fitness sharing and niching methods revisited. IEEE-EC 2(3), 97 (1998)
18. Nguyen, Q.U., Nguyen, X.H., O'Neill, M.: Semantic Aware Crossover for Genetic Programming: The Case for Real-Valued Function Regression. In: Vanneschi, L., Gustafson, S., Moraglio, A., De Falco, I., Ebner, M. (eds.) EuroGP 2009. LNCS, vol. 5481, pp. 292–302. Springer, Heidelberg (2009)

19. Nguyen, Q.U., O'Neill, M., Nguyen, X.H., Mckay, B., Galván-López, E.: Semantic Similarity Based Crossover in GP: The Case for Real-Valued Function Regression. In: Collet, P., Monmarché, N., Legrand, P., Schoenauer, M., Lutton, E. (eds.) EA 2009. LNCS, vol. 5975, pp. 170–181. Springer, Heidelberg (2010)
20. Nguyen, Q.U., Nguyen, X.H., O'Neill, M., McKay, R.I., Galvan-Lopez, E.: Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genetic Programming and Evolvable Machines, 91–119 (2011)
21. Nguyen, Q.U., Nguyen, T.H., Nguyen, X.H., O'Neill, M.: Improving the Generalisation Ability of Genetic Programming with Semantic Similarity based Crossover. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (eds.) EuroGP 2010. LNCS, vol. 6021, pp. 184–195. Springer, Heidelberg (2010)