Semantic Aware Crossover for Genetic Programming: The Case for Real-Valued Function Regression

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

[1] Natural Computing Research & Applications Group, University College Dublin, Ireland
[2] School of Computer Science and Engineering, Seoul National University, Korea
quanguyhn@yahoo.com, nxhoai@gmail.com, m.oneill@ucd.ie

Abstract. In this paper, we apply the ideas from [2] to investigate the effect of some semantic based guidance to the crossover operator of GP. We conduct a series of experiments on a family of real-valued symbolic regression problems, examining four different semantic aware crossover operators. One operator considers the semantics of the exchanged subtrees, while the other compares the semantics of the child trees to their parents. Two control operators are adopted which reverse the logic of the semantic equivalence test. The results show that on the family of test problems examined, the (approximate) semantic aware crossover operators can provide performance advantages over the standard subtree crossover adopted in Genetic Programming.

Keywords: crossover, semantic, and genetic programming.

1 Introduction

Since genetic programming was born, it has been seen by some researchers in and out of the field that GP is a potentially powerful method for automated synthesis of computer programs by evolutionary means. The 'program' is usually presented in a language of syntactic formalism such as s-expression trees [7], a linear sequence of instructions, grammars, or graphs [1, 12]. The genetic operators in such GP systems are usually designed to ensure the syntactic closure property, i.e. to produce syntactically valid children from any syntactically valid parent(s). Using such purely syntactical genetic operators, GP evolutionary search is conducted on the syntactical space of programs with the only semantic guidance from the fitness of program measured by the difference of behavior of evolving programs and the target programs (usually on a finite input-output set called fitness cases). Although GP has been shown to be effective in evolving programs for solving different problems using such (finite) behavior-based semantic guidance and pure syntactical genetic operator, this practice is somewhat unusual from real programmers' perspective. Computer programs are not just constrained by syntax but also by semantics. As a normal practice, any change to a program should pay heavy attention to the change in semantics of the program and not just those changes that guarantee to maintain the program syntactical validity. To amend this deficiency in GP resulting from the lack of semantic guidance on genetic

L. Vanneschi et al. (Eds.): EuroGP 2009, LNCS 5481, pp. 292–302, 2009.

operators, recently, Beadle and Johnson have proposed a semantic-based crossover operator for genetic programming [2]. They showed that their semantically driven crossover operator could help GP in achieving better results and less code bloat on some standard Boolean test problems.

In this paper, we extend the ideas from [2] to investigate the effect of some semantic based guidance on the crossover operator in GP on a family of real-valued symbolic regression problems. We also propose a new form of semantic-aware crossover for Genetic Programming, which considers approximations of the semantics of the exchanged subtrees. In contrast to the approach of Beadle and Johnson [2], the new semantic-aware crossover can be applied to both Boolean and continuous problem domains.

The paper is organized as follows. In the next section, we give a review of related work on crossover operators and semantic based operations in GP. Section 3 contains the descriptions of some possible methods for using semantic information to guide the crossover operator in GP. These methods are used in experiments described in section 4 of the paper. The results of the experiments are then given and discussed in section 5. Section 6 concludes the paper and highlights some potential future extension of this work.

2 Related Work

In a series of work, Johnson has advocated for the use of semantic information in the evolutionary process of GP [3, 4, 5, 6]. He proposed a number of possible ways for incorporating program semantics extracted by static analysis techniques into the fitness function of GP. In [2], the authors elaborated on their suggestions by investigating the effect of using semantic information to guide the crossover operation in GP for Boolean domains, the resultant operator is called semantically driven crossover (SDC). Their main idea is to check the semantic equivalence between newly born children as the results of the crossover operator with their parents. The semantic equivalence checking of two Boolean expression trees is done by transforming the trees to reduced ordered binary decision diagrams (ROBDDs). They have the same semantic if and only if they are reduced to the same ROBDD. The semantic equivalence checking is then used to determine which of the individual participating in crossover operation will be copied to the next generation. If the children born as the result of crossover are semantically equivalent with their parents, they are not copied to the next generation, instead their parents are copied. By doing this, the author argued that it helps to increase the semantic diversity of evolving population of programs. Indeed the idea is rather similar to deterministic fitness crowding, one of the common mechanisms for promoting diversity in evolutionary algorithms [8, 9]. In the deterministic fitness crowding method, the children must compete directly with the parents to be copied to the next generation and they are copied only if they are different enough (usually measured by a distance function) from their parents. The important difference between work in [2] and deterministic fitness crowding is the use of semantic information (which seems to be attributed to GP only). Semantic driven crossover was reported useful in [2] in both increasing GP performance but

also reducing code bloat on some standard Boolean test problems (even though it seems that reducing bloat was not intentionally the motivation for the introduction of semantic driven crossover).

Prior to Beadle and Johnson [2], there were a number of works on analyzing the effect of crossover and attempts to improve them. However, the main focus of this work has been on code bloat, fitness-destructive effect of crossover [15, 16, 19] and length distributions, and how to improve it in this respect [17, 18].

The work on semantic driven crossover operators in this paper is different from [2] in two ways. Firstly, the domain for testing semantically driven crossovers is real-valued rather than Boolean. For real-valued domains, the idea of checking semantic equivalence by reducing to common ROBDDs is no longer possible. Secondly, the semantic guidance of the crossover operator is not just derived from the whole program tree behavior but also from subtrees. This is inspired by recent work in [10] for calculating subtree semantics. This method is used in [14] as a way for measuring the fitness in GP. However, the subtree semantic calculated in this paper is for real-valued domains but not Boolean domains as in [2, 10].

3 Methodologies

The aim of this study is to extend earlier work [2, 10] to real-valued domains. For such problems it is not easy to compute the semantics or semantic equivalence of two expression trees by reducing them to a common structure as for Boolean domain as in [2]. In fact, the problem of determining semantic equivalence between two real-valued expressions is known to be NP-hard [20]. Similarly, the problem of completely enumerating and comparing fitness of subtree expressions as in [10] is also impossible on real domains. We have to calculate the approximate semantics. In this paper, a simple method for measuring and comparing the semantics of two expressions is used both for individual trees and subtrees and the two methods are compared experimentally. To determine the semantic equivalence of two expressions, we measure them against a random set of points sampled from the domain. If the output of the two trees on the random sample set are close enough (subject to a parameter called semantic sensitivity) then they are designated as semantically equivalent. It can be written in pseudo-code as follows:

```
If Abs(Value_On_Random_Set(P1)-
   Value_On_Random_Set(P2))<∑ then
     Return P1 is semantically equivalent to P2.
```

Where `Abs` is the absolute function and $\sum$ is a predefined constant called the *semantic sensitivity*. This method is inspired by the simple technique for simplifying s-expression program trees proposed in [11] called equivalence decision simplification (EDS), where complicated subtrees could be replaced by much simpler and template subtrees if they are semantically equivalent. The method of checking semantic equivalence of EDS is similar to the method used here and EDS has shown to be an

efficient tool for removing redundant codes for tree-based genetic programming[1] [11]. To test the effect of using semantic information to guide the subtree crossover in genetic programming, we propose 4 possible scenarios based on [2, 10].

In scenario 1, we constrain the crossover in such a way that if two crossover points are chosen the two subtrees under the crossover points are checked for semantic equivalence. If they have equivalent semantics, the operator is forced to be executed on two new crossover points. The pseudo-code for this scenario can be summarized as follows:

```
1.1.Select two parents: P1, P2
1.2.Choose at random crossover points at Subtree1 in P1
    Choose at random crossover points at Subtree2 in P2
   if(Subtree1 is not equivalent with Subtree2{
              Execute crossover
              Add the children to the new population
              Return TRUE
      }
      else{
              Add P1 and P2 to new population
              Return TRUE
      }
```

The motivation for scenario 1 is to encourage GP individual trees to exchange subtrees that have different semantics, which is expected to encourage the change in semantics of the whole trees after each crossover.

In the second scenario, we reverse the semantic bias in the crossover operator compared to the first scenario in that the two subtrees are selected for crossover if and only if they are semantically equivalent. The only change in code is the condition of if statement in 1.2 as follows:

```
            if (Subtree1 is equivalent with Subtree2)
```

The third scenario is similar to the method used in [2]. Here the semantics of newly created children are checked against the semantics of their parents. If their semantics are found to be equivalent with their parents then they are discarded and the parents would be passed to the new generation instead. The objective of such an implementation of crossover, as stated in [2], is to enforce the semantic diversity (i.e., keep generating new individual programs with new behavior). The pseudo-code for this scenario is as follows:

[1] The JAVA code of EDS could be freely downloaded from: sc.snu.ac.kr

```
2.1. Select two parents: P1, P2
2.2. If (condition to make crossover is satisfied)
     {
      Make Crossover to have Children C1 and C2
      if (C1 is equivalent with P1 Or P2)
          Add P1 to the new population
      else
          Add C1 to the new population
      if (C2 is equivalent with P1 Or P2)
          Add P2 to the new population
      else
          Add C2 to the new population
      }
      else
          Add P1 and P2 to new population
```

The last and fourth scenario tested in this paper is the reverse of the third, where the children are accepted to go in to the new population if and only if they are semantically equivalent with their parent(s).

4 Experiments

To investigate the possible effects of semantic aware crossovers given in the previous section, they are tested on the symbolic regression problems with target functions in a family of polynomials of increasing degree given in [13] and they are:

$F_1 = X^3+X^2+X$.
F2= $X^4+X^3+X^2+X$.
F3= $X^5+X^4+X^3+X^2+X$
F4=$X^6+ X^5+X^4+X^3+X^2+X$

The parameters setting for GP in the experiment is as follows:

- Population size: 500.
- Number of generation: 50
- Tournament selection size: 3
- Crossover probability: 0.9
- Mutation probability: 0.1

- Max depth of program tree at the initial generation: 6
- Max depth of program tree at all time: 15
- Non-terminals: +, -, *, / (protected version), sin, cos, exp, log (protected)
- Terminals: X, 1
- Number of sample: 20 random points from [-1…1].
- Hit: when an individual has an absolute error < 0.01 on a fitness case.
- Termination: when a program score 20 hits or running out of generations.

The *semantic sensitivities* ($\sum$) used in the experiments are: 0.01, 0.02, 0.04, 0.05, 0.06, 0.08, 0.1, 0.5, and 1. For each crossover scenario, each target problem, and semantic sensitivity, 100 runs are performed, which makes the total number of runs 14800.

5 Results and Discussion

The number of successful runs (out of 100 runs) is given in Table 1. It can be seen that the effect of using semantic aware crossover depends on the manner in which it is applied. When semantics is calculated on subtrees (Scenario 1), it can improve the performance of GP in terms of the number of successful runs (e.g., 71 versus 62 on F1, and 36 versus 28 on F2). While semantics as used in a similar way to [2] by comparing the resulting children to their parents only give slight improvements in some cases (Scenario 3). When we reverse the logic of the semantic equivalence test in the Control Scenarios 2 and 4 no improvement in the number of successful runs is achieved when compared with standard GP.

However, the improvement in Scenario 1 also depends on the value *of semantic sensitivity* ($\sum$).The results suggest that the value of sensitivities from 0.01 to 0.1 are suitable for all four test functions, and the greater sensitivities seem to confer less improvement.

In Table 2 we show mean and standard deviation of best fitness for the 50^{th} generation of each run. The results shown in this table is consistent with the results in Table 1. The semantic aware crossover as used in scenario 1 where the semantics of subtrees to be exchanged are compared, can slightly improve the performance of GP in terms of the mean and standard deviation of best fitness. Whereas in Scenario 3, where the semantics of children are compared to their parents only results in an improvement in some settings of semantic sensitivity. Again, no improvements are observed for the control scenarios 2 and 4 where the logic of the equivalence test is reversed. From these results it would appear that the more difficult problems benefit more in terms of the mean and standard deviation of the best fitness when semantic aware crossover is adopted. This claim can be confirmed when we consider the result of performing t-tests of the results in Scenario 1. These results are shown in Table 3.

Table 1. The number of successful runs over 100 runs

Scenario	Sensitivity	F1	F2	F3	F4
GP		62	28	15	10
Scenario 1	0.01	68	33	22	10
	0.02	**70**	**33**	**22**	**14**
	0.04	**70**	**34**	**20**	**19**
	0.05	**71**	**33**	**19**	**14**
	0.06	**71**	**32**	**20**	**17**
	0.08	**70**	**35**	**20**	**17**
	0.1	**66**	**36**	**17**	**14**
	0.5	62	33	17	12
	1	65	27	12	8
Scenario 2	0.01	41	31	13	7
	0.02	40	18	13	10
	0.04	40	15	14	8
	0.05	44	26	12	7
	0.06	41	17	13	7
	0.08	41	19	15	7
	0.1	42	24	12	10
	0.5	45	18	10	10
	1	55	25	18	5
Scenario 3	0.01	52	24	18	12
	0.02	57	32	16	12
	0.04	52	32	23	8
	0.05	54	33	15	7
	0.06	52	29	16	10
	0.08	50	26	13	11
	0.1	44	26	13	8
	0.5	38	15	15	5
	1	39	17	14	3
Scenario 4	0.01	2	0	0	0
	0.02	3	0	0	0
	0.04	3	0	0	0
	0.05	2	0	0	0
	0.06	2	0	1	0
	0.08	3	1	0	0
	0.1	3	1	1	0
	0.5	7	0	0	1
	1	12	1	0	1

Table 2. Mean and standard deviation of the best fitness at 50 generations

	Sensitivity	F1 Mean (Stdev)	F2 Mean (Stdev)	F3 Mean (Stdev)	F4 Mean (Stdev)
GP		0.128 (0.170)	0.262 (0.241)	0.302 (0.252)	0.397 (0.355)
Scenario 1	0.01	0.135 (0.205)	0.230 (0.214)	0.274 (0.215)	0.335 (0.271)
	0.02	**0.133** **(0.209)**	**0.236** **(0.218)**	**0.275** **(0.216)**	**0.329** **(0.275)**
	0.04	**0.125** **(0.182)**	**0.231** **(0.214)**	**0.268** **(0.212)**	**0.328** **(0.278)**
	0.05	**0.126** **(0.185)**	**0.224** **(0.206)**	**0.266** **(0.213)**	**0.330** **(0.277)**
	0.06	**0.121** **(0.181)**	**0.227** **(0.207)**	**0.265** **(0.215)**	**0.319** **(0.270)**
	0.08	**0.137** **(0.215)**	**0.223** **(0.205)**	**0.279** **(0.218)**	**0.333** **(0.281)**
	0.1	**0.133** **(0.185)**	**0.225** **(0.208)**	**0.274** **(0.215)**	**0.334** **(0.268)**
	0.5	0.120 (0.157)	0.217 (0.203)	0.276 (0.213)	0.336 (0.266)
	1	0.123 (0.174)	0.214 (0.149)	0.287 (0.215)	0.397 (0.292)
Scenario 2	0.01	0.258 (0.357)	0.275 (0.222)	0.325 (0.235)	0.459 (0.421)
	0.02	0.221 (0.318)	0.307 (0.234)	0.330 (0.237)	0.458 (0.348)
	0.04	0.229 (0.321	0.310 (0.238)	0.332 (0.244)	0.435 (0.314)
	0.05	0.253 (0.356)	0.289 (0.216)	0.336 (0.256)	0.439 (0.395)
	0.06	0.241 (0.333)	0.302 (0.237)	0.327 (0.238)	0.443 (0.314)
	0.08	0.239 (0.325)	0.315 (0.256)	0.334 (0.245)	0.434 (0.305)
	0.1	0.250 (0.355)	0.279 (0.197)	0.337 (0.260)	0.436 (0.396)
	0.5	0.236 (0.395)	0.280 (0.237)	0.314 (0.195)	0.377 (0.293)
	1	0.168 (0.205)	0.328 (0.300)	0.294 (0.237)	0.398 (0.262)
Scenario 3	0.01	0.229 (0.373)	0.239 (0.215)	0.321 (0.321)	0.362 (0.305)
	0.02	0.208 (0.326)	0.218 (0.184)	0.229 (0.286)	0.348 (0.292)
	0.04	0.224 (0.334)	0.229 (0.211)	0.306 (0.311)	0.353 (0.241)
	0.05	0.212 (0.312)	0.216 (0.197)	0.303 (0.232)	0.364 (0.307)
	0.06	0.204 (0.333)	0.229 (0.230)	0.280 (0.241)	0.315 (0.218)

Table 2. *(Continued)*

	0.08	0.234 (0.350)	0.248 (0.225)	0.335 (0.302)	0.347 (0.251)
	0.1	0.224 (0.329)	0.252 (0.214)	0.359 (0.302)	0.418 (0.305)
	0.5	0.225 (0.332)	0.297 (0.211)	0.344 (0.310)	0.406 (0.275)
	1	0.216 (0.224)	0.343 (0.247)	0.378 (0.266)	0.460 (0.247)
Scenario 4	0.01	1.144 (0.911)	1.287 (0.959)	1.590 (1.219)	1.641 (1.215)
	0.02	1.153 (0.908)	1.304 (0.964)	1.569 (1.251)	1.662 (1.228)
	0.04	1.122 (0.911)	1.280 (1.002)	1.561 (1.247)	1.650 (1.214)
	0.05	1.204 (0.946)	1.259 (0.967)	1.513 (1.203)	1.658 (1.236)
	0.06	1.127 (0.914)	1.281 (0.997)	1.473 (1.236)	1.640 (1.223)
	0.08	1.121 (0.910)	1.205 (0.999)	1.495 (1.999)	1.604 (1.225)
	0.1	1.144 (0.938)	1.180 (0.980)	1.519 (1.191)	1.650 (1.216)
	0.5	0.994 (0.895)	1.038 (0.834)	1.249 (1.015)	1.575 (1.233)
	1	0.752 (0.787)	0.870 (0.645)	1.177 (1.107)	1.350 (1.277)

Table 3. Mean and standard deviation of best fitness and p-value of t-test

Scenario	Sensi-tivity	f1 Mean Stdev	P-value	f2 Mean Stdev	P-value	f3 Mean Stdev	P-value	f4 Mean Stdev	P-value
GP		0.128 0.170		0.262 0.241		0.302 0.252		0.397 0.355	
Scenario 1	0.01	0.135 0.205	0.647	0.230 0.214	0.351	0.274 0.215	0.405	0.335 0.271	0.165
	0.02	0.133 0.209	0.687	0.236 0.218	0.462	0.275 0.216	0.412	0.329 0.275	0.129
	0.04	0.125 0.182	0.932	0.231 0.214	0.367	0.268 0.212	0.304	0.328 0.278	0.128
	0.05	0.126 0.185	0.894	0.224 0.206	0.282	0.266 0.213	0.274	0.330 0.277	0.137
	0.06	0.121 0.181	0.982	0.227 0.207	0.225	0.265 0.215	0.264	0.319 0.270	**0.080**
	0.08	0.137 0.215	0.604	0.223 0.205	0.255	0.279 0.218	0.495	0.333 0.281	0.157
	0.1	0.133 0.185	0.678	0.225 0.208	0.161	0.274 0.215	0.395	0.334 0.268	0.162
	0.5	0.120 0.157	0.928	0.217 0.203	0.131	0.276 0.213	0.414	0.336 0.266	0.171
	1	0.123 0.174	0.986	0.214 0.149	0.090	0.287 0.215	0.543	0.397 0.292	0.997

6 Conclusions and Future Work

The effect of semantics with crossover in Genetic Programming is investigated in this paper. In this study we focus on the family of real-valued problem domains in form of polynomials (symbolic regression). The investigation is performed on four scenarios. The experimental results show that semantic aware crossover as adopted in Scenario 1, where the semantics of subtrees to be exchanged are analysed, can improve performance of Genetic Programming in both number of successful runs and the mean best fitness on the problems examined. In the alternative Scenario 3, where the semantics of the children resulting from crossover are compared to their parents performance gains can be observed in some cases. Comparing subtree semantic aware crossover to the individual-based form, there is an advantage for the subtree approach on the four symbolic regression instances examined. The two control scenarios (2 and 4) where the logic of the semantic equivalence tests are reversed show no improvement over standard subtree crossover.

There are some interesting areas for future investigation. In contrast to the approach adopted in an earlier study on Boolean problems [2] the semantic aware crossover operators adopted here can be applied to both Boolean and real-valued domains. We also wish to examine the utility of these semantic aware operators to more difficult symbolic regression problems, and more classic benchmark problems from the literature to determine more clearly the generality of these findings.

Acknowledgements

This research was funded under a Government of Ireland Postgraduate Scholarship from the Irish Research Council for Science, Engineering and Technology (IRCSET).

References

1. Banzhaf, W., Nordin, P., Francone, F.D., Keller, R.E.: Genetic Programming: An Introduction - On the Automatic Evolution of Computer Programs and Its Applications. Morgan Kaufmann Publishers, San Francisco (1998)
2. Beadle, L., Johnson, C.G.: Semantically Driven Crossover in Genetic Programming. In: Proceedings of the IEEE World Congress on Computational Intelligence, pp. 111–116. IEEE Press, Los Alamitos (2008)
3. Johnson, C.G.: Deriving Genetic Programming Fitness Properties by Static Analysis. In: Foster, J.A., Lutton, E., Miller, J., Ryan, C., Tettamanzi, A.G.B. (eds.) EuroGP 2002. LNCS, vol. 2278, pp. 299–308. Springer, Heidelberg (2002)
4. Johnson, C.G.: Genetic Programming with Guaranteed Constraints. In: Lofti, A., John, B., Garibaldi, J. (eds.) Recent Advances in Soft Computing. Physica/Springer-Verlag, Heidelberg (2002)
5. Johnson, C.G.: Genetic Programming with Fitness based on Model Checking. In: Ebner, M., O'Neill, M., Ekárt, A., Vanneschi, L., Esparcia-Alcázar, A.I. (eds.) EuroGP 2007. LNCS, vol. 4445, pp. 114–124. Springer, Heidelberg (2007)

6. Johnson, C.G.: What Can Automatic Programming Learn from Theoretical Computer Science? In: Yao, X. (ed.) Proceedings of the UK Workshop on Computational Intelligence, University of Birmingham (2002)
7. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
8. Mahfoud, S.W.: Crowding Preselection Revisited. In: Manner, R., Manderick, B. (eds.) Parallel Problem Solving from Nature, vol. 2, pp. 27–36. Elsevier, Amsterdam (1992)
9. Mahfoud, S.W.: Niching Methods for Genetic Algorithms. Doctoral Dissertation at University of Illinois at Urbana-Champaign (1995)
10. McPhee, N.F., Ohs, B., Hutchison, T.: Semantic Building Blocks in Genetic Programming. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) EuroGP 2008. LNCS, vol. 4971, pp. 134–145. Springer, Heidelberg (2008)
11. Mori, N., McKay, R.I., Nguyen, X.H., Essam, D.: Equivalent Decision Simplification: A New Method for Simplifying Algebraic Expressions in Genetic Programming. In: Proceedings of 11th Asia-Pacific Workshop on Intelligent and Evolutionary Systems (2007)
12. Poli, R., Langdon, W.B., McPhee, N.F.: A Field Guide to Genetic Programming (2008), http://lulu.com http://www.gp-field-guide.org.uk
13. Nguyen, X.H., McKay, R.I., Essam, D.: Solving the Symbolic Regression Problem with Tree-Adjunct Grammar Guided Genetic Programming: The Comparative Results. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), pp. 1326–1331. IEEE Press, Los Alamitos (2002)
14. Krysztof, K., PremysBaw, P.: Potential Fitness for Genetic Programming. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2008), Late-Breaking Papers, pp. 2175–2180. ACM, New York (2008)
15. Langdon, W.B., Poli, R.: Fitness causes bloat: Mutation. In: Koza, J. (ed.) Late Breaking Papers at the GP 1997 Conference, Stanford, CA, USA, July 13-16, pp. 132–140. Stanford Bookstore (1997)
16. Langdon, W.B., Soule, T., Poli, R., Foster, J.A.: The evolution of size and shape. In: Spector, L., Langdon, W.B., O'Reilly, U.-M., Angeline, P.J. (eds.) Advances in Genetic Programming 3, ch. 8, pp. 163–190. MIT Press, Cambridge (1999)
17. Dignum, S., Poli, R.: Crossover, Sampling, Bloat and the Harmful Effects of Size Limits. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) EuroGP 2008. LNCS, vol. 4971, pp. 158–169. Springer, Heidelberg (2008)
18. Dignum, S., Poli, R.: Operator Equalisation and Bloat Free GP. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) EuroGP 2008. LNCS, vol. 4971, pp. 110–121. Springer, Heidelberg (2008)
19. Banzhaf, W., Langdon, W.B.: Some considerations on the reason for bloat. In: Genetic Programming and Evolvable Machines, vol. 3, pp. 81–91. Springer, Netherlands (2002)
20. Ghodrat, M.A., Givargis, T., Nicolau, A.: Equivalence Checking of Arithmetic Expressions. In: CASES 2005, San Francisco, California. ACM, New York (2005)