Combining Semantically-Effective and Geometric Crossover Operators for Genetic Programming

Tomasz P. Pawlak

Institute of Computing Science, Poznan University of Technology, Poznań, Poland
tpawlak@cs.put.poznan.pl

Abstract. We propose a way to combine two distinct general patterns for designing semantic crossover operators for genetic programming: geometric semantic approach and semantically-effective approach. In the experimental part we show the synergistic effects of combining these two approaches, which we explain by a major fraction of crossover acts performed by geometric semantic crossover operators being semantically ineffective. The results of the combined approach show significant improvement of performance and high resistance to a premature convergence.

Keywords: Semantics, taxonomy, neutrality, brood selection, experiment.

1 Introduction

Genetic Programming (GP), as a method of automatic induction of discrete structures from an arbitrary given set of building blocks, guided by a fitness function has been known for over 25 years [13,29]. Among other applications, like synthesis of electronic circuits [14,15] or design of 3D structures [21,1], GP is mostly used for induction of programs from a set of instructions. To accurately assess the program's fitness, it must be run multiple times on a set of program inputs to compare the produced outputs with the desired target outputs. Typically the divergence between the target output and the actual program output is measured by a minimized fitness function and the pair of program input and the target program output is refered to as *fitness case* [30,19,25,16,8].

The tuple that consists of a program output for each fitness case is known as *semantics*[1]. Consequently, the target program output extracted from a set of fitness cases is also a semantics, however to distinguish this one, we refer to it as *target semantics*. The main role of program semantics is to describe program behavior in a concise way.

In recent years we observe a growing interest in semantics in genetic programming. Most applications of semantics are related to designing novel genetic operators, such as crossover [30,19,25,16,27,2] and mutation [25,27,26,4], however

[1] The common definition involves a vector instead of tuple [30,19,25,16,8], however vector is by definition a tuple of numbers and we consider programs that operate on an arbitrary data, thus we keep the tuple in the definition as more general.

T. Bartz-Beielstein et al. (Eds.): PPSN XIII 2014, LNCS 8672, pp. 454–464, 2014.

some researchers also analyze the impact of semantic initialization of the initial population [3,10] and selection [8].

In this paper we focus on semantic crossover operators. We distinguish two general patterns of designing semantic crossover operators, which are *semantically-effective* and *geometric* patterns. In the first one semantics is used to reject the offspring having the same or similar semantics to any of its parents, while in the second one the crossover is guided on producing offspring having semantics that is a certain combination of semantics of parents. We present the way to combine these two approaches and experimentally confirm advantages of this combination.

2 Taxonomy of Semantic Crossover Operators

We divide the collection of semantic crossover operators for tree-based GP, depending on they use program semantics for either providing *effective* changes, or exploiting *geometric* properties of search space. This taxonomy shall not be confused with the one proposed in [32], since the latter one divides semantic operators into operators making syntactic changes with *indirect* impact on the semantics, and manipulating syntax to *directly* influence semantics.

Naturally we do not include in this taxonomy non-semantic crossover operators, such as tree-swapping crossover [13], context-preserving crossover [6], one-point crossover [28], size-fair crossover [20] and depth-based crossover [9].

Semantically-Effective Crossover Operators

The first class of crossover operators consists of operators that use semantics mainly for the purpose of rejecting offspring that are semantically equal to any of their parents. In this sense these operators produce *semantically-effective* offspring w.r.t. its parents by rejecting the *ineffective* ones. They are commonly referred in the literature to as 'semantically-driven' [2] or 'semantic-aware' [30], since to the best of our knowledge they were historically the first semantics-based crossover operators. However the historical names are too general, thus we decided to refer to these operators to as *semantically-effective*.

The term 'semantically-effective' shall not be considered as an opposition to *neutrality* in GP, which refers to two notions: (i) neutral genes, e.g., introns [12,7], and (ii) mutation performing genotypic changes that are not reflected in phenotype [12] or fitness change [7]. In contrast semantically effective crossover (i) does not negate that an offspring may contain introns, and (ii) does not perform a (mutational) change in the genotype, instead it combines genotypes of two parents. Moreover the 'effectiveness' of crossover is defined on the semantic level, which is fundamentally different from what is meant by 'phenotype' in these early studies [12,7], i.e., a syntax tree.

One of the first of semantically-effective crossovers is *semantically driven crossover* (SDC) by Beadle et al. [2], which involves reduced ordered binary decision diagrams (ROBDD) as a representation for semantics of Boolean programs.

Thanks to the uniqueness property of ROBDD, their operator is able to prevent of crossing over parents having semantically equal subtrees rooted at crossover points. Later Quang et al. [30] extended this idea to the domain of real function synthesis by proposing *semantics aware crossover* (SAC). Their operator discards all crossover acts of parents, whose subtrees rooted at crossover points are semantically less distant than a given threshold, called semantic sensitivity. Thus SDC and SAC are conceptually equivalent, however adapted for different domains.

To some extent another operator by Quang et al. [30], namely *semantic similarity based crossover* (SSC) can be also included in this class of operators. SSC operates similarly to SAC, however it adds second threshold to the semantic distance between parents' subtrees to prevent breeding offspring unrelated to its parents. In this sense SSC reduces the chaotic characteristics of crossover. The same authors [31], arguing that not only the effectiveness but also the magnitude of change is important, proposed the *most semantic similarity based crossover* (MSSC), which operate similarly to SSC, however instead of using the second threshold, it promotes the exchange of subtrees that are the most similar but different.

Semantic Geometric Crossover Operators

We distinguish a separate class of crossover operators that guide the crossover act to produce offspring having semantics that is a kind of geometric combination of semantics of its parents. Such a combination is usually a point on the segment spanned over semantics of parents. If crossover operator guarantees that the semantics of offspring lies on this segment, we refer to it as *geometric crossover*. This definition is consistent with [24,18,27]. Although most of the operators presented in this section only approximate geometric crossover, we would not distinguish another class for them.

Note that the geometric crossover requires a way to appoint a segment between parent semantics. This imposes important restrictions to the representation of semantics, as it must be an object in a normed vector space. However if this condition is met, an (exact) geometric crossover is guaranteed to breed offspring that is not worse, than the worst of its parents [18].

In this group, special attention should be paid to *semantic geometric crossover* (SGX) by Moraglio et al. [25], which guarantees that the semantics of an offspring lies on the segment spanned over semantics of its parents, i.e., SGX is exact geometric crossover. Unfortunately SGX causes exponential in time bloat, which puts its practical applications into question, even when using implementation techniques that allow to handle exponentially increasing in time program structures in linearly-increasing data structures [5].

The next operator that is worth to mention is *approximately geometric semantic crossover* by Krawiec and Lichocki (KLX) [16]. KLX is indeed a meta-operator that runs in a loop some other (secondary) crossover operator, and from all the offspring produced by the secondary operator, chooses the most geometric according to an arbitrary geometry measure.

Algorithm 1. Krawiec Lichocki Crossover (KLX). p_1, p_2 are parents, k is a number of crossover attempts, SX is a secondary crossover, MostGeometric is Eq. 1.

```
1: function KLX(p1, p2, k)
2:     O ← ∅                                          ▷ Offspring candidates
3:     for all i ∈ 1..k do
4:         O ← O ∪ SX(s(p1), s(p2))
5:     o1 ← MostGeometric(O, p1, p2)
6:     o2 ← MostGeometric(O\{o1}, p1, p2)
7:     return {o1, o2}
8: end function
```

Other operators in this group are: *locally geometric semantic crossover* [19,17], that approximates geometric recombination of parents at the level of the homologous crossover point, and *approximately geometric semantic crossover* [27,18], that propagates the desired geometric semantics of offspring back trough the tree of parent program, in order to place an adequate subtree at the level of crossover point.

3 Semantically-Effective Geometric Crossover Operator

One may ask, whether is it possible to combine features of semantically-effective and geometric crossover operators to benefit from advantages of both of them?

To achieve that we can equip almost any geometric crossover operator, with a procedure that prevents the operator from producing semantically ineffective offspring w.r.t. its parents.

To be more specific, we propose such a procedure for SX+ variant of KLX operator described in [16]. The original KLX algorithm is shown in Algorithm 1. It operates by running k times a secondary crossover operator, e.g., tree-swapping crossover [13], and finally choosing the most geometric offspring that it produced, according to the formula:

$$\begin{aligned}&\textsc{MostGeometric}(O, p_1, p_2) = \\ &\quad \arg\min_{o \in O} \underbrace{d(s(p_1), s(o)) + d(s(o), s(p_2))}_{\text{distance sum}} + \underbrace{|d(s(p_1), s(o)) - d(s(o), s(p_2))|}_{\text{penalty}} \end{aligned} \quad (1)$$

where O is a set of candidate offspring, p_1, p_2 are parent programs, $s(p)$ is semantics of program p, $d(\cdot, \cdot)$ is a distance metric. The formula consists of two major parts: the first one is sum of distances between the semantics of candidate offspring o and both parents; the second one is a penalty for choosing offspring that is not equidistant from parents.

To combine KLX with principles of semantically effective crossover, we propose to replace MostGeometric calls in Algorithm 1 with the formula:

$$\begin{aligned}\textsc{MostGeometric}^{+}(O, p_1, p_2) &= \begin{cases} \textsc{MostGeometric}(O', p_1, p_2) & O' \neq \emptyset \\ \textsc{MostGeometric}(O, p_1, p_2) & O' = \emptyset \end{cases} \\ \text{where} \quad O' &= \{o : o \in O, s(o) \neq s(p_1), s(o) \neq s(p_2)\} \end{aligned} \quad (2)$$

Table 1. Parameters of evolution

Parameter	Value
Population size	1024
Fitness function	*Symbolic regression*: Mean absolute error
	Boolean domain: Hamming distance
Termination condition	At most 100 generations or find of individual having fitness 0
Initialization method	Ramped Half-and-Half, height range $2-6$, up to 100 retries
Selection method	Tournament selection, size 7
Max program height	17
Crossover probability	1.0
Number of attempts	KLX$^+$, KLX, SAC, GPX$^+$: 10, GPX: n/a
Semantic sensitivity	10^{-6}
Instructions	*Symbolic regression*: x, $+$, $-$, $\times$, /, sin, cos, exp, log, ERC[a]
	Boolean domain: D1...D11[b], and, or, nand, nor, ERC
Number of runs	30

[a] log and / are protected. log is defined as $\log|x|$; / returns 0 if divisor is 0.
[b] Number of inputs depends on a problem instance.

Table 2. Benchmarks; in symbolic regression there are 20 *equidistant* fitness cases in the given range

Symbolic regression			Boolean domain		
Problem	Definition (formula)	Range	Problem	Instance (bits)	Fitness cases
Septic	$x^7-2x^6+x^5-x^4+x^3-2x^2+x$	$[-1,1]$	Even parity	PAR5	32
Nonic	$\sum_{i=1}^{9} x^i$	$[-1,1]$		PAR6	64
R1	$(x+1)^3/(x^2-x+1)$	$[-1,1]$		PAR7	128
R2	$(x^5-3x^3+1)/(x^2+1)$	$[-1,1]$	Multiplexer	MUX6	64
R3	$(x^6+x^5)/(x^4+x^3+x^2+x+1)$	$[-1,1]$		MUX11	2048
Nguyen6	$\sin(x)+\sin(x+x^2)$	$[-1,1]$	Majority	MAJ6	64
Nguyen7	$\log(x+1)+\log(x^2+1)$	$[0,2]$		MAJ7	128
Keijzer1	$0.3x\sin(2\pi x)$	$[-1,1]$	Comparator	CMP6	64
Keijzer4	$x^3e^{-x}\cos(x)\sin(x)(\sin^2(x)\cos(x)-1)$	$[0,10]$		CMP8	256

This change restricts the set of candidate offspring by removing offspring that is semantically equal to any of the parents. Only if all candidates are semantically equal, the algorithm uses the whole candidate set. For semantics represented by floating point numbers, it is natural to compare semantics in Eq. 2 with a threshold. To be consistent with [30], we call it *semantic sensitivity*. We denote the augmented variant of KLX as KLX$^+$.

4 The Experiment

We attempt to experimentally verify whether combining features of geometric and semantically-effective crossover operators pays off. In order to do that we prepare a run of GP employed with KLX$^+$ and we compare it to GP running 'bare'

Table 3. Average fitness and 95% confidence interval achieved by best-of-run individual as of 100 generation. Best values are marked in bold.

Problem	KLX$^+$		KLX		SAC		GPX$^+$		GPX	
Keijzer1	**0.005**	±0.002	0.015	±0.003	0.008	±0.003	0.009	±0.003	0.013	±0.004
Keijzer4	**0.030**	±0.011	0.139	±0.014	0.049	±0.013	0.045	±0.014	0.041	±0.010
Nguyen6	**0.001**	±0.000	0.005	±0.002	0.002	±0.001	0.002	±0.001	0.004	±0.002
Nguyen7	**0.001**	±0.000	0.005	±0.002	0.002	±0.001	0.003	±0.001	0.005	±0.002
Nonic	**0.006**	±0.002	0.024	±0.003	0.012	±0.003	0.014	±0.004	0.018	±0.004
Septic	**0.013**	±0.005	0.063	±0.026	0.020	±0.004	0.027	±0.009	0.032	±0.008
R1	**0.006**	±0.002	0.028	±0.006	0.012	±0.003	0.022	±0.006	0.022	±0.005
R2	**0.015**	±0.004	0.028	±0.005	0.017	±0.004	0.021	±0.005	0.028	±0.007
R3	**0.001**	±0.000	0.006	±0.001	0.002	±0.000	0.003	±0.001	0.003	±0.001
CMP6	**0.000**	±0.000	0.533	±0.221	0.267	±0.158	0.100	±0.107	0.867	±0.303
CMP8	**0.067**	±0.128	7.300	±0.956	7.000	±1.070	6.167	±0.570	9.433	±1.182
MAJ6	**0.000**	±0.000	**0.000**	±0.000	**0.000**	±0.000	**0.000**	±0.000	**0.000**	±0.000
MAJ7	**0.000**	±0.000	0.133	±0.153	0.100	±0.142	0.367	±0.269	0.567	±0.272
MUX6	**1.700**	±0.717	5.667	±0.642	3.133	±0.732	3.200	±0.770	3.567	±0.683
MUX11	**102.567**	±12.470	160.067	±14.555	114.000	±8.641	118.467	±7.202	118.467	±7.048
PAR5	**0.000**	±0.000	2.033	±0.510	3.500	±0.604	2.433	±0.440	3.400	±0.567
PAR6	**5.000**	±0.947	13.367	±0.792	11.567	±1.204	13.167	±0.795	14.200	±0.893
PAR7	**23.467**	±1.044	40.967	±1.047	35.967	±1.752	40.533	±0.799	41.033	±1.899

KLX. In addition we add two control setups: tree-swapping crossover (GPX) [13], which acts as a reference point, and GPX$^+$ – GPX that rejects semantically ineffective offspring and retries crossover act until it is effective or a given number of attempts is exceeded. We added GPX$^+$ to check whether the achievements of KLX$^+$ are due to the combination of properties of geometric and semantically-effective crossovers, or solely due to restrictions for ineffective offspring. Note that GPX$^+$ is not equivalent to SAC [30], as the latter one performs equivalence check in the crossover points instead of whole program trees, i.e., SAC allows an exchange of semantically different subtrees, however it does not take into account, how this exchange influences the semantics of entire trees, which could not change at all. However, as an previously published operator, SAC is good reference point, therefore we add it as the last setup. Details of evolutionary parameters can be found in Table 1, parameters not included there, are set to ECJ's defaults [22]. Note that the experiment does not include any mutation operator, since it is focused on the examination of properties of crossover operators.

We run evolution of 18 commonly used benchmark problems shown in Table 2: 9 symbolic regression [13,23] and 9 Boolean function synthesis [13,33] benchmarks.

Performance

Figure 1 presents the plots of average and 95% confidence interval of the best-of-generation fitness achieved by each operator and Table 3 shows average and 95% confidence interval of the best-of-run fitness.

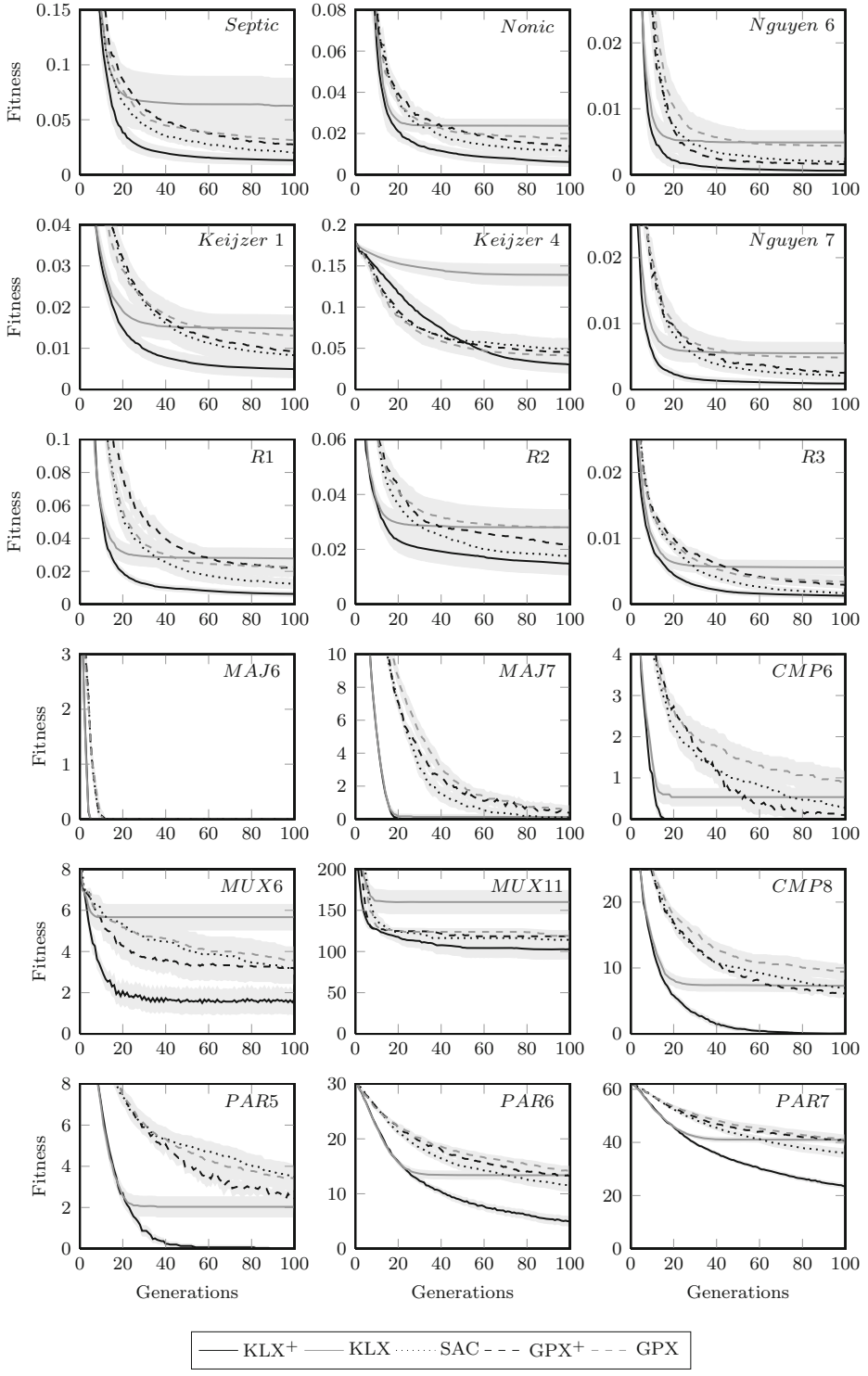

Fig. 1. Fitness achieved by best-of-generation individual averaged over 30 runs. Shadings are 95% confidence intervals.

Table 4. Post-hoc analysis of Friedman's test using symmetry test: p-values of incorrectly judging operator in row as outranking operator in column. Significant p-values are marked in bold ($\alpha = 0.05$) and visualized as arcs in outranking graph.

	GPX	GPX$^+$	KLX	KLX$^+$	SAC
GPX			0.976		
GPX$^+$	0.151		**0.033**		
KLX					
KLX$^+$	**0.000**	**0.004**	**0.000**		0.052
SAC	**0.017**	0.926	**0.002**		

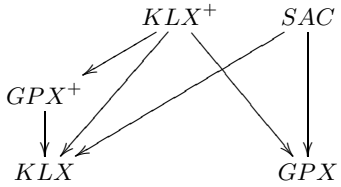

Table 5. Fraction and 95% conf. interval of ineffective crossover acts (the lowest in bold)

Domain	KLX$^+$	KLX	SAC	GPX$^+$	GPX
Symbolic regression	**0.001** ± 0.0000	0.883 ± 0.0001	0.067 ± 0.0001	0.002 ± 0.0000	0.113 ± 0.0001
Boolean domain	**0.000** ± 0.0000	0.902 ± 0.0001	0.649 ± 0.0002	0.001 ± 0.0000	0.525 ± 0.0002

It is clear that KLX$^+$ converges the quickest and in all benchmark problems achieves the best fitness at the end of evolution. Moreover, thanks to quick convergence, KLX$^+$'s fitness achieved after 30 generations, in 15 out of 18 benchmarks remains unbeatable by any other operator at the end of evolution.

To verify statistical significance of obtained results, we performed Friedman's test for multiple achievements of multiple subjects [11] on data shown in Table 3. The calculated p-value is 6.52×10^{-9}, thus assuming critical value $\alpha = 0.05$, the test is conclusive, that there is at least one statistical difference in the results. Therefore we present in Table 4 a post-hoc analysis of Friedman's test, using the symmetry test, and an outranking graph for the operators.

The analysis shows that KLX$^+$ outranks all other operators except SAC, however the p-value for outranking SAC, i.e., 0.052, is very close to the critical value of $\alpha = 0.05$. From the graph we can see that semantically-effective SAC outranks GPX and KLX, and semantically-effective GPX$^+$ outranks KLX only. Perhaps the low efficiency of KLX can be surprising, however we believe that KLX is likely to stick in local optima due to lack of routine that prevents from producing ineffective offspring, that exists in superior KLX$^+$. This leads us to conclusion that rejection of semantically-ineffective offspring is an important factor that influences GP performance.

Semantic Effectiveness

An observant reader may ask why semantic-effectiveness has such a significant influence on operator's performance. To answer this question, we presented in Table 5 fraction of ineffective crossover acts done by each operator separately during all evolutionary runs of previous experiment.

Values obtained by KLX are clearly the highest and show that nearly 90% of crossover acts done by KLX are actually ineffective. We believe that this is

mainly caused by bias of Eq. 1, which prefers offspring having the same semantics as a parent over the offspring that is nearly-distant from one of the parents, but non-geometric[2]. The high fraction of ineffective offspring clearly explains the observed premature convergence of KLX.

On the other hand SAC maintains low percentage of ineffective offspring, however only for symbolic regression. We hypothesize that high value for Boolean domain is due to our previous observation, that SAC performs equivalence check only at the level of crossover point, and since Boolean instructions are likely to return unchanged output if only one input changes, it likely makes the crossover operation ineffective at higher parts of program tree. This seems to be consistent with results of GPX$^+$, which are very low for both domains, and significantly lower than its unbiased counterpart – GPX.

5 Conclusions

We divided semantic crossover operators for tree-based GP into two classes depending on the purpose of using program semantics: to reject semantically ineffective offspring, or to geometrically combine parent programs. Moreover we proposed a way to combine features of these two classes in a single operator and experimentally demonstrated that this combination performs better and is less likely to suffer from premature convergence than each approach solely. We believe that the proposed method can be successfully applied to other geometric semantic crossover operators as well.

Acknowledgment. We would like to thank Wojciech Jaśkowski and Krzysztof Krawiec for their time and the valuable comments on this study. Work supported by Polish National Science Centre grant no. DEC-2012/07/N/ST6/03066.

References

1. Al-Sakran, S.H., Koza, J.R., Jones, L.W.: Automated re-invention of a previously patented optical lens system using genetic programming. In: Keijzer, M., Tettamanzi, A.G.B., Collet, P., van Hemert, J., Tomassini, M. (eds.) EuroGP 2005. LNCS, vol. 3447, pp. 25–37. Springer, Heidelberg (2005)
2. Beadle, L., Johnson, C.: Semantically driven crossover in genetic programming. In: IEEE CEC 2008, pp. 111–116. IEEE Press (2008)
3. Beadle, L., Johnson, C.G.: Semantic analysis of program initialisation in genetic programming. Genetic Programming and Evolvable Machines 10(3), 307–337 (2009)
4. Beadle, L., Johnson, C.G.: Semantically driven mutation in genetic programming. In: IEEE CEC 2009, pp. 1336–1342. IEEE Press (2009)

[2] Eq. 1 evaluates to $2d(s(p_1), s(p_2))$ for offspring semantically equal to any of its parents.

5. Castelli, M., Castaldi, D., Giordani, I., Silva, S., Vanneschi, L., Archetti, F., Maccagnola, D.: An efficient implementation of geometric semantic genetic programming for anticoagulation level prediction in pharmacogenetics. In: Correia, L., Reis, L.P., Cascalho, J. (eds.) EPIA 2013. LNCS, vol. 8154, pp. 78–89. Springer, Heidelberg (2013)
6. D'haeseleer, P.: Context preserving crossover in genetic programming. In: IEEE CEC 1994, vol. 1, pp. 256–261. IEEE Press (1994)
7. Ferreira, C.: Genetic representation and genetic neutrality in gene expression programming. Advances in Complex Systems 5(4), 389–408 (2002)
8. Galvan-Lopez, E., et al.: Using semantics in the selection mechanism in genetic programming: A simple method for promoting semantic diversity. In: IEEE CEC 2013, vol. 1, pp. 2972–2979 (2013)
9. Harries, K., Smith, P.: Exploring alternative operators and search strategies in genetic programming. In: GP 1997, pp. 147–155. Morgan Kaufmann (1997)
10. Jackson, D.: Phenotypic diversity in initial genetic programming populations. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (eds.) EuroGP 2010. LNCS, vol. 6021, pp. 98–109. Springer, Heidelberg (2010)
11. Kanji, G.: 100 Statistical Tests. SAGE Publications (1999)
12. Keller, R.E., Banzhaf, W.: Genetic programming using genotype-phenotype mapping from linear genomes into linear phenotypes. In: GP 1996, pp. 116–122. MIT Press (1996)
13. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
14. Koza, J.R., et al.: Genetic Programming 3: Darwinian Invention and Problem Solving. Morgan Kaufman (April 1999)
15. Koza, J.R., et al.: Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers (2003)
16. Krawiec, K., Lichocki, P.: Approximating geometric crossover in semantic space. In: GECCO 2009, pp. 987–994. ACM (2009)
17. Krawiec, K., Pawlak, T.: Quantitative analysis of locally geometric semantic crossover. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 397–406. Springer, Heidelberg (2012)
18. Krawiec, K., Pawlak, T.: Approximating geometric crossover by semantic backpropagation. In: GECCO 2013, pp. 941–948. ACM (2013)
19. Krawiec, K., Pawlak, T.: Locally geometric semantic crossover: A study on the roles of semantics and homology in recombination operators. Genetic Programming and Evolvable Machines 14(1), 31–63 (2013)
20. Langdon, W.B.: Size fair and homologous tree genetic programming crossovers. Genetic Programming and Evolvable Machines 1(1/2), 95–119 (2000)
21. Lohn, J., Hornby, G., Linden, D.: An evolved antenna for deployment on Nasa's Space Technology 5 Mission. In: Genetic Programming Theory and Practice II, ch. 18, pp. 301–315. Springer (2004)
22. Luke, S.: The ECJ Owner's Manual – A User Manual for the ECJ Evolutionary Computation Library, zeroth edition, online version 0.2 edition (October 2010)
23. McDermott, J., et al.: Genetic programming needs better benchmarks. In: GECCO 2012, pp. 791–798. ACM (2012)
24. Moraglio, A.: Abstract convex evolutionary search. In: FOGA XI, pp. 151–162. ACM (2011)

25. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric semantic genetic programming. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 21–31. Springer, Heidelberg (2012)
26. Nguyen, Q.U., Nguyen, X.H., O'Neill, M.: Semantics based mutation in genetic programming: The case for real-valued symbolic regression. In: Mendel 2009, pp. 73–91 (2009)
27. Pawlak, T.P., Wieloch, B., Krawiec, K.: Semantic backpropagation for designing search operators in genetic programming. IEEE Transactions on Evolutionary Computation (2014)
28. Poli, R., Langdon, W.B.: Schema theory for genetic programming with one-point crossover and point mutation. Evolutionary Computation 6(3), 231–252 (1998)
29. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming. Lulu Enterprises, UK Ltd. (2008)
30. Uy, N.Q., et al.: Semantically-based crossover in genetic programming: Application to real-valued symbolic regression. Genetic Programming and Evolvable Machines 12(2), 91–119 (2011)
31. Uy, N.Q., et al.: On the roles of semantic locality of crossover in genetic programming. Information Sciences 235, 195–213 (2013)
32. Vanneschi, L., Castelli, M., Silva, S.: A survey of semantic methods in genetic programming. Genetic Programming and Evolvable Machines (online first)
33. Walker, J.A., Miller, J.F.: Investigating the performance of module acquisition in cartesian genetic programming. In: GECCO 2005, vol. 2, pp. 1649–1656. ACM Press (2005)