Information Sciences 235 (2013) 195–213

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

On the roles of semantic locality of crossover in genetic programming

Nguyen Quang Uy [a], Nguyen Xuan Hoai [b,*], Michael O'Neill [c], R.I. McKay [d], Dao Ngoc Phong [e]

[a] Faculty of Information Technology, Military Technical Academy, Viet Nam
[b] Faculty of Information Technology, Hanoi University, Viet Nam
[c] Natural Computing Research and Applications Group, University College Dublin, Ireland
[d] School of Computer Science and Engineering, Seoul National University, South Korea
[e] Department of Information and Communication, Hanoi City Government, Viet Nam

ARTICLE INFO

Article history:
Received 5 June 2012
Received in revised form 4 January 2013
Accepted 8 February 2013
Available online 16 February 2013

Keywords:
Genetic Programming
Semantic
Locality
Crossover

ABSTRACT

Locality has long been seen as a crucial property for the efficiency of Evolutionary Algorithms in general, and Genetic Programming (GP) in particular. A number of studies investigating the effects of locality in GP can be found in the literature. The majority of the previous research on locality focuses on syntactic aspects, and operator semantic locality has not been thoroughly tested. In this paper, we investigate the role of semantic locality of crossover in GP. We follow McPhee in measuring the semantics of a subtree using the fitness cases. We use this to define a semantic distance metric. This semantic distance supports the design of some new crossover operators, concentrating on improving semantic locality. We study the impact of these semantically based crossovers on the behaviour of GP. The results show substantial advantages accruing from the use of semantic locality.

1. Introduction

Genetic Programming (GP) [38,28] is an evolutionary paradigm for finding solutions, often in the form of functional expressions, for a problem. To date, research has largely focused on syntactic aspects of GP representation, bringing valuable insights and contributions to the success of GP. However, from a programmer's perspective, maintaining syntactic correctness is just a part of program construction: programs must be correct not only syntactically, but also semantically. Thus incorporating semantic awareness in the GP evolutionary process could potentially improve performance and extend the applicability to problems that are difficult to deal with using purely syntactic approaches. Recently, several researchers have taken an interest in incorporating semantic information into GP, leading to a sharp increase in related publications (e.g. [19,21,24,2,34,49,5]).

Previous work in the broader field of evolutionary computation has shown that locality (i.e., small changes in genotype resulting in small changes in phenotype) deeply affect search performance [16,42,41]. With current GP representations, designing GP operators supporting this desirable property can be difficult. An important reason is the lack of a clear separation between syntax and semantics of individuals. Most GP genetic operators have been designed based on syntax alone; but small changes in syntax can lead to large changes in semantics. Thus ensuring semantic locality requires genetic operators that are semantically aware.

* Corresponding author.
E-mail addresses: quanguyhn@gmail.com (N.Q. Uy), nxhoai@gmail.com (N.X. Hoai), m.oneill@ucd.ie (M. O'Neill), rimsnucse@gmail.com (R.I. McKay), phongaptech@yahoo.com (D.N. Phong).

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.02.008

In GP, most research on locality has focused on syntax [16,42,41]. Recently, semantic diversity has attracted some attention [2,46,13], but limited work has been conducted on semantic locality [49]. In this paper, we extend our earlier work in [49] on incorporating semantics into GP crossover operators. The earlier paper introduced the use of semantic locality to generate controllable search. Here, we propose two improvements to the crossover operator of [49]. We also further investigate the effects of these operators on locality. In [49], we argued that semantic locality is a desirable property of crossover. However it has previously been argued that syntactic locality is valuable [16]. Perhaps semantic locality's value merely reflects its correlation with syntactic locality. Here, we extend this work to show that semantic locality is substantially more beneficial than syntactic locality. We apply these methods to two real-world problems in a time series prediction task, and in producing human-competitive approximations for the Gaussian *Q*-function.

The remainder of the paper is structured as follows. The next section reviews literature on semantic information in GP, on crossover operators, and on locality. Section 3 contains a detailed descriptions of our methods. The potential effects on GP performance of improving semantic locality in crossover are investigated in Section 4, followed by an analysis of semantic exchange in standard crossover and semantic based crossovers (Section 5). The comparison between semantic locality and syntactic locality is presented in Section 6. Section 7 presents two application of these ideas to two problems: time series prediction and approximating the Gaussian *Q*-function. The conclusions are drawn in Section 8, leading to suggestions for future research in Section 9.

2. Background

This section gives a brief overview of previous work on semantics in GP, on variants of GP crossover operators, and on locality in GP.

2.1. Semantics in Genetic Programming

In GP, semantic information has generally been used to provide additional guidance to the evolutionary search. The way semantics are exploited depends on the problem domain (Boolean or real-valued), individual representation (Grammar-, Tree- or Graph-based), and the search algorithm components (fitness measure, genetic operators, ...). There have been three main approaches for representing, extracting, and using semantics to guide the evolutionary process of GP:

1. Grammar-based [52,5,6].
2. Formal methods [19,21,25].
3. GP s-tree representation [2,34,46,48,29].

The first, using attribute grammars, is the most popular. Such grammars extend context-free grammars, providing context sensitivity through a finite set of attributes [26]. GP individuals expressed in the form of attribute grammar derivation trees can incorporate semantic information, which can be used to eliminate bad (i.e., less fit) individuals from the population [6] or to prevent the generation of semantically invalid individuals [52,5]. The attributes used to present semantics are generally problem-dependent, and it is not always obvious how to determine the attributes for a given problem.

Johnson, in his recent work, has advocated formal methods as a means to incorporate semantic information into the GP process [19,21]. In this work, semantic information, extracted by formal methods, is used to quantify the fitness of individuals on some problems for which traditional sample-point-based fitness measure are unavailable or misleading.

In other work, Keijzer [25] used interval analysis to check whether an individual is defined over the whole range of input values – if an individual is undefined anywhere, that individual can be assigned minimal fitness or simply eliminated from the population. This allowed Keijzer to avoid discontinuities arising from protected operators, improving the evolvability of the system. The advantage of formal methods lies in their rigorous mathematical foundations, potentially helping GP to evolve computer programs. However they are high in complexity and difficult to implement, possibly explaining the limited number of related publications since the advocacy of Johnson [20]. Their main application to date has been in evolving control strategies.

Methods for extracting semantics from expression trees depend on the problem domain. The finite inputs of Boolean domains mean that semantics can be accurately estimated in a number of ways. Beadle and Johnson [2] checked the semantic equivalence of the offspring produced by crossover with their parents by converting them to Reduced Ordered Binary Decision Diagrams (ROBDDs). If the conversion of the two trees leads to the same ROBDD, they are semantically equivalent. This semantic equivalence checking is used to decide if new individuals are copied to the next generation. If offspring are equivalent to their parents, they are discarded and the crossover is restarted. The authors argued that this enforces semantic diversity in the evolving population, and consequently leads to improvement in GP performance [2].

By contrast, in [34], semantic information is extracted from a Boolean expression tree by enumerating all possible inputs. Two aspects of semantics were evaluated in each tree: the semantics of subtrees and the semantics of context (the remainder of an individual after removing a subtree). The variation of these semantic components throughout the GP evolutionary process was experimentally measured. Special attention was paid to fixed-semantic subtrees: subtrees where the semantics of the tree does not change when this subtree is replaced by another subtree. The authors showed that there may be many such fixed semantic subtrees when the tree size increases during the evolutionary process. Thus it becomes increasingly difficult

to change the semantics of trees with crossover and mutation once the trees have become large. This reduces the semantic diversity [34].

In [46], we proposed a crossover operator, Semantics Aware Crossover (SAC), based on checking the semantic equivalence of subtrees. SAC was tested on a family of real-valued symbolic regression problems, and shown to improve GP performance. SAC was extended to Semantic Similarity based Crossover (SSC) in [48,49]. The performance of SSC was superior to both SC and SAC. SAC and SSC differ from Beadle and Johnson's approach [2] in three ways. The problem domain is real-valued rather than Boolean; for real-valued domains, it is not generally feasible to check semantic equivalence by reducing to a canonical form like a ROBDD. Also, the crossover operators (SAC or SSC) are guided not by the semantics of the whole program tree but by subtrees – inspired by the recent work of [34] for calculating subtree semantics. However, for real domains, measuring semantics by enumerating all possible inputs as in [34] is infeasible, so the semantics must be approximated. Finally, SSC differs from [2] in targeting locality as well as diversity.

Recently, Krawiec and Lichocki proposed a measure for the semantics of individuals based on fitness cases [29]. The semantics is defined as a vector, of which each element is the output of the individual on one fitness case. This was used to guide crossover in a method similar to Soft Brood Selection (SBS) [1], known as Approximating Geometric Crossover (AGC). A number of children are generated by a crossover operation, the children semantically most similar to their parents being added to the next generation. The experiments were conducted on both real and Boolean regression problems, with the conclusion that AGC is no better than SC on real-valued problems, and only slightly superior on Boolean. Moraglio et al. [35] developed AGC to proposed Geometric Semantic Genetic Programming (GSGP). In GSGP, the offspring were generated in crossover by a convex combination of their parent semantics. The experimental results shown that GSGP obtains better performance than standard GP though it exacerbates code bloat.

2.2. Alternative crossovers in genetic programming

It is well-known that crossover is the primary operator for GP [27]. Much research has concentrated on the efficiency of crossover, resulting in new and improved operators which might be classified into three categories as follows:

1. Crossover based on syntax (structure).
2. Crossover based on context.
3. Crossover based on semantics.

Most of the early modifications to SC were based on syntax [27,37,36]. Koza [27] proposed a crossover that is 90% biased to function nodes and 10% to terminal nodes as crossover points. Although this method encourages the exchange of more genetic material (bigger subtrees) between the participating individuals, it risks exacerbating bloat and thus making it more difficult to refine solutions in later generations [2]. O'Reilly and Oppacher [36] introduced a height-fair crossover, in which all subtree heights in the two parents are recorded, and one subtree height is randomly selected. The crossover sites in both parents are then restricted to that particular height. Poli and Langdon [37,30] introduced one-point crossover and uniform crossover. In these methods, when two parents are selected for crossover, they are aligned based on their shapes. By aligning two parents, the common shape of these parents (starting from their roots) can be determined. The crossover points are then randomly selected from the nodes that lie in the common shape region. This kind of crossover has been shown especially effective on Boolean problems. It causes a larger exchange of genetic material in earlier generations (in these generations the common shape is often very small), and yet can tune the solutions in later generations (when the common shapes are bigger) [37,30]. Recently, Lin and Chen proposed a FuzzyTree crossover for multi-valued stock valuation [31]. Two crossover points are randomly selected from the two parents. If they are internal nodes, the standard subtree crossover is executed. However, if they are leaf nodes, the crossover is based on the exchange two fuzzy numbers. The authors showed that the FuzzyTree crossover is more effective than the subtree crossover in their problems.

Beside syntax, context has been used as extra information for the selection of crossover points [1,33]. This class of crossover operators is perhaps closest to semantic based crossovers. Altenberg [1] proposed a new crossover inspired by the observation that in most animal species, breeding occurs more often than the number of surviving offspring might suggest. In other words, viable offspring are not always produced as a consequence of breeding. This method is known as Soft Brood Selection (SBS). Two parents are selected for crossover, then N random crossover operations are performed to generate a brood of $2N$ children. The children are then evaluated and sorted based on their fitness. The two best children are copied to the next generation, the rest being discarded.

Majeed and Ryan [33] proposed Context Aware Crossover (CAC). In CAC, after the two parents have been selected for crossover, one subtree is randomly chosen in the first parent. This subtree is then crossed over into all possible locations in the second, then all generated children are fitness evaluated. The best child (based on fitness) is selected as the result, and copied to the next generation. This process is then repeated for the second parent. The advantage of these context-based crossovers is the increased probability of producing better children. On the other hand, it can be very time consuming to evaluate the context of each subtree.

To the best of our knowledge, the only previous use of semantics in crossover are those previously discussed in SubSection 2.1. They include Beadle and Johnson's [2] Semantics Driven Crossover for Boolean problems, Krawiec and Lichocki's

[29] Approximating Geometric Crossover, and our previous work [46] on Semantics Aware Crossover (SAC) and on Semantic Similarity-based Crossover (SSC) [48,49].

2.3. Locality in genetic programming

In evolutionary computation, locality (small change in genotype corresponding to small change in phenotype) plays a crucial role in the efficiency of an algorithm [11,16,42,10]. Maintaining diversity is necessary but not sufficient for good algorithm performance; it should be accompanied by measures to ensure locality. Previous work on locality in GP (such as [16]) has largely focused on syntactic aspects, with the exception of our recent studies [48,49].

The importance of locality was brought into full focus by Rothlauf [42]. Metrics for locality of the genotype–phenotype mapping are derived from metrics on both genotypic and phenotypic spaces. Rothlauf distinguished high locality representations, where a small change in genotype leads to a small change in phenotype, from low locality representations (small changes in genotype may lead to large changes in phenotype). Although not completely formalised, this corresponds to the mathematical notion of continuity (or to the notion of uniform continuity when bounds are independent of the location in the search space). Rothlauf argued that a high locality representation is necessary for efficient evolutionary search [42].

Although locality may be desirable for efficient search, designing such a representation is not a trivial task, and with current GP representations, there may be no guarantee of locality. Hoai et al. [16] formally proved that their genotype–phenotype mapping was syntactically local, but did not investigate, or attempt to control, semantic locality.

In this paper, we extend the concept of locality of a representation [42] to the locality of a search operator. We emphasise semantic locality rather than the syntactic locality that has dominated previous work. We study semantic locality through a continuous domain of values rather than restricting to a two valued (low, high) metric as in [42], We demonstrate that design of crossover operators achieving higher semantic locality is feasible for tree-based GP, and that it leads to an improvement in GP performance.

3. Methods

In this section, we first describe the necessary components for designing semantics-based crossovers, starting from a measure we call subtree semantics (SS). SS is used to define a semantic distance, and then to formulate the semantic relationship (similarity). We review the semantic similarity based crossover (SSC) proposed in [49], highlighting some deficiencies. Finally, we detail the designs of two new semantics-based crossovers that overcome these deficiencies.

3.1. Measuring semantics

Although, an exact definition of semantics is non-trivial, in GP the semantics of an individual program is generally understood as the behaviour of that program with respect to a set of input values. We use a simple way to quantify semantics in GP based on the fitness cases of the problem, extending Mcphee's subtree semantics [34] from Boolean to real domains. It is defined for any (sub) tree as follows:

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

This way of measuring subtree semantics is similar to our preliminary work in [46,48,49], where *Sampling Subtree Semantics* was measured by sampling random points from the problem domain. The difference lies merely in which points are used: random points or fitness cases.

Based on SS we define the *Subtree Semantics Distance* (SSD) between two subtrees. Previously [46,48], we defined the Sampling Semantics Distance as the sum of the absolute differences. While the experiments in [46,48] showed that this semantic distance is beneficial, its value depends on the number of sampling points N. To remedy this, we normalise, defining the SSD as the mean of the absolute differences between subtrees. In other words, let $P = \{p_1, p_2, \ldots, p_N\}$ and $Q = \{q_1, q_2, \ldots, q_N\}$ be the SS of $Subtree_1(St_1)$ and $Subtree_2(St_2)$, the SSD between St_1 and St_2 is defined as:

$$SSD(S_1, S_2) = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N} \tag{1}$$

3.2. Semantic relationships

As in [46,48], we next define a semantic relationship between subtrees: *Semantic Similarity* (SSi).[1] The intuition behind semantic similarity is that exchange of subtrees is most likely to be beneficial if the subtrees are semantically not identical, but also not too dissimilar. Two subtrees, St_1 and St_2, are semantically similar if their SSD lies within a positive interval – formally:

$$SSi(St_1, St_2) = \textbf{if } \alpha < \mathrm{SSD}(St_1, St_2) < \beta$$

[1] The word 'similarity' is used here in its ordinary English meaning, where A is similar to B implies that A is not the same as B, as opposed to a common mathematical convention in which similarity includes equivalence.

here α and β are two predefined constants, known as the *lower* and *upper bounds* for semantic sensitivity (LBSS and UBSS). Conceivably, the best values for α and β might be problem dependent, and might also be self-adapted as in Section 3.4.

We note that there is a biological motivation for this approach. In mammals, the Major Histocompatibility Complex (MHC) genes (on chromosome 6 in humans) play a major role in the immune response, and thus are a key part of our defences against disease, and subject to strong and rapidly-changing evolutionary pressures. However they also play an important role both in mate selection (partners in the same species, but with dissimilar MHC genes, are preferred) [4]), and in speciation, because too-great differences in MHC may cause an immune response from the mother to the foetus [9]. Thus in this case at least, biology also appears to favour crossovers with semantic similarity lying in a specific range.

We conclude this section by highlighting some important differences between semantics and fitness. Firstly, fitness reflects how good (close to the target function) an individual is, while the SS of a subtree only reflects how the subtree behaves on specific input values from the environment. Second, fitness is measured for the whole individual, while SS is mainly used to encapsulate the semantics of subtrees. The last and most important difference is the objective: fitness is used for individual selection while SS is used to guide crossover.

3.3. Improving semantic locality using semantic similarity-based crossover

Previous research [42,16] showed the crucial role of locality (of representation and operators) in EC and GP, but focused purely on syntactic locality. In [48], we proposed a semantics based crossover, called *Semantic Similarity based Crossover* (SSC),[2] that achieves higher locality than SC. SSC is an extension of the semantic diversity crossover, Semantic Aware Crossover [46], in two directions. First, when two subtrees are selected for crossover, their semantic similarity (SS), rather than semantic equivalence (SE), is checked. Since SS is usually more difficult to satisfy than semantic inequivalence, repeated failures may occur. Therefore SSC uses multiple trials to find semantically similar pairs, only reverting to random selection after passing a bound on the number of trials. The number of trials to find a semantically similar pair is called *Max_Trial* (MT). Algorithm 1 shows how SSC works in detail [48,49].

Algorithm 1.

Algorithm 1: Semantic Similarity based Crossover

```
select Parent 1 P1;
select Parent 2 P2;
Count=0;
while Count<Max_Trial do
    choose a random crossover point Subtree1 in P1;
    choose a random crossover point Subtree2 in P2;
    calculate SSD between Subtree2 and Subtree1;
    if Subtree1 is similar to Subtree2 then
        execute crossover;
        add the children to the new population;
        return true;
    else
        Count=Count+1;
if Count=Max_Trial then
    choose a random crossover point Subtree1 in P1;
    choose a random crossover point Subtree2 in P2;
    execute crossover;
    return true;
```

The motivation for SSC is to encourage exchange of semantically different, but not substantially different, subtrees. In other words, while forcing a change in the semantics of the individuals in the population, we want to keep this change relatively small. SSC was detailed and thoroughly analysed in [49]. The results in [49] showed that SSC helps GP to find the solution more reliably and quickly than SC and other fitness based crossovers (Soft Brood Selection, No Same Mate Selection) and the semantic (diversity) aware crossover (SAC). The deeper analysis in [49] showed that the main reason for the superior performance of SSC over these crossovers is that SSC causes smoother semantic change from parents to children, which creates higher constructive effect and less bloat.

However, these advantages of SSC over other crossover operators come at a cost, especially when compared with SC: it has extra parameters to tune, namely the lower and upper bounds for semantic sensitivity (LBSS and UBSS). The analysis of SSC's parameter sensitivity in [49] has shown that while the performance of SSC is only slightly sensitive to LBSS,[3] it is quite

[2] Table A.13 in Appendix A provides a list of abbreviations in the paper.

[3] Since in practice, we usually keep LBSS fixed at a rather small positive value such as 10^{-3}.

sensitive to UBSS, perhaps in a problem dependent way. In the following subsections, we propose two methods to overcome these problems, resulting in two new operators.

3.4. Self-adaptation of upper bound on semantic sensitivity

The first approach to fix the problem of tuning UBSS for SSC is to self-adapt it during evolution. It has been long known that finding appropriate values for evolutionary parameters is challenging. Self-adaptation has been used right from the start. Most previous work focused on controlling operator application rates or step sizes [15], or adjusting population size [17] or individual size [43].

In this subsection, we present a method for self-adjusting semantic sensitivity in SSC. The motivation is to ensure that as many as possible SSC operations exchange semantically similar subtrees. If the UBSS is too small (e.g. 10^{-2}), SSC may not work, because it is very difficult to select subtrees pairs that can be swapped. Conversely, if it is too large (e.g. 100) SSC would behave much the same as SC, with almost all pairs satisfying the condition. We introduce *Self-Adaptive Successful Execution* (SASE), whose aim is to ensure that a large enough portion, but not all, SSC operations exchange similar subtree pairs. Formally, the UBSS is controlled by the following equation:

$$\beta^{t+1} = \begin{cases} c_i \cdot \beta^t, & \text{if } p_s^t < \psi; \\ c_d \cdot \beta^t, & \text{if } p_s^t > \psi; \\ \beta^t, & \text{if } p_s^t = \psi; \end{cases} \tag{2}$$

where β^t is the UBSS at generation t, and p_s^t is the proportion of SSC operations that successfully exchange at generation t. c_i is the size increment, and c_d the decrement. The LBSS at generation $t+1$ is calculated from the UBSS as: α^{t+1}=$10^{-3} \cdot \beta^{t+1}$. The initial value of UBSS was set at 0.4 in these experiments. ψ is a threshold presents for the percentage of SSC we expect to successfully exchange to semantic similar subtrees. Several values of ψ will be investigated in the following sections.

3.5. Most semantically similar crossover

The second method for handling the problem of UBSS tuning for SSC is simply to remove UBSS. Instead, we use competition of possible crossover points within each parent to ensure the likelihood of maintaining small change in semantics. The resultant new semantic similarity based crossover is called *Most Semantically Similar Crossover* (MSSC) and it works as follows: N subtree pairs are randomly selected from the two parents. The subtree semantic distance (SSD) of the subtrees in each pair is calculated. SSD-equivalent pairs are excluded, then the pair having the smallest SSD is chosen for crossover. Algorithm 2 shows how MSSC works in detail. In the experiments with MSSC, several values of Max_Trial was tested, and Extremal_Value was set to 10^6.

Algorithm 2.

Algorithm 2: The Most Semantic Similarity based Crossover

```
select Parent 1 P1;
select Parent 2 P2;
Count=0;
Max=Extremal_Value;
while Count<Max_Trial do
    choose a random crossover point Subtree1 in P1;
    choose a random crossover point Subtree2 in P2;
    SD=SSD(Subtree1, Subtree2)
    if SD is less than Max and greater than LBSS then
        Max=SD;
        CrossPoint1=Subtree1;
        CrossPoint2=Subtree2;
Execute crossover by exchange of subtrees at CrossPoint1 and
CrossPoint2;
```

4. The impact of semantic locality of crossover on GP performance

In previous work [49], we showed that a crossover improving semantic locality (SSC) favourably compares with standard crossover. In addition, it outperforms the crossover for promoting semantic diversity [46], and fitness-based crossovers such as No Same Mate Selection [14] and Soft Brood Selection [1]. This section compares SASE and MSSC with SSC. We first describe the experimental settings and then present a performance comparison.

Table 1
Symbolic regression functions.

Functions	Fitcases
$F_1 = x^4 + x^3 + x^2 + x$	20 Random points $\subseteq [-1, 1]$
$F_2 = x^5 + x^4 + x^3 + x^2 + x$	20 Random points $\subseteq [-1, 1]$
$F_3 = sin(x) + sin(x + x^2)$	20 Random points $\subseteq [0, 1]$
$F_4 = sin(x^2)cos(x) - 1$	20 Random points $\subseteq [0, \frac{\pi}{2}]$
$F_5 = log(x + 1) + log(x^2 + 1)$	20 Random points $\subseteq [0, 2]$
$F_6 = \sqrt{x}$	20 Random points $\subseteq [0, 4]$
$F_7 = sin(x) + sin(y^2)$	100 Random points $\subseteq [0, 1] \times [0, 1]$
$F_8 = 2sin(x)cos(y)$	100 Random points $\subseteq [0, 1] \times [0, 1]$

Table 2
Parameter setting.

Parameters	Value
Population size	500
Generation	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 versions), sin, cos, exp, log (protected versions)
Terminals	X, 1 for single variable problems, and X, Y for bivariable problems
Raw fitness	Sum of absolute error on all fitness cases
Trials per treatment	100 Independent runs for each value

4.1. Experimental settings

To compare the effects of these operators on GP performance, we tested them on eight real-valued symbolic regression problems. These problems are divided into three groups: polynomial functions; trigonometric, logarithm and square-root functions; and bivariate functions. Two, F_1, F_2 have been previously used in [46,48] for testing SAC and SSC. Most of the rest are taken from Keijzer [25], and Johnson [22]. These functions are shown in Table 1, and the basic parameters used for our experiments are given in Table 2. The parameter settings are similar to our previous work [46,48,49]. Although these experiments study crossover, we have retained mutation at a low rate, to put crossover in the context of a normal GP run. Note that the raw fitness function is the sum of the absolute error over all fitness cases.

The semantic sensitivities for the tested crossover were set as follows. For SSC, LBSS was fixed at 10^{-3}, and we used three values for UBSS: 0.4, 0.5, and 0.6 (the same values were used in [49]).[4] The value of Max_Trial for SSC was set at 12. This was calibrated from preliminary experiments as a suitable value, though again it is not a particularly sensitive parameter. The three tested configurations of SSC are denoted as SSCX with X = 04, 05, 06.

For SASE, the size of decrement (c_d) was fixed at 0.9 and the size of increment was set as $1/c_i$. Four configurations of SASE were tested. In the first three configurations, different cutoff ratios for successful exchanges were tested: 65%, 75%, and 85%. These configurations of SASE are called SASEX (X = 65, 75, 85). In the final version of SASE, the rate of successful SSC is itself adapted. The motivation is that, at the beginning of the search process, GP needs to be exploratory, so the rate of successful SSC in earlier generations should be low. Later, SSC should be encouraged to be more exploitative, so the rate of successful SSC should increase. The change was scheduled using the following equation:

$$R\_SSC = \alpha + (\beta - \alpha) \cdot \frac{Current\_Gen}{Max\_Gen} \tag{3}$$

where *Current_Gen* is the current generation and *Max_Gen* is the maximum number of generations set for the algorithm. *R_SSC* is the rate of successful SSC in the current generation. In our experiments, α was set at 65% and β was set at 85%. This version of SASE is referred to as SASES.

For MSSC, the lower bound semantic sensitivity was set as in SSC at 10^{-3}. Three values of Max_Trial: 12, 16, 20 were tested resulting in three configurations of MSSC referred as MSSCX with X = 12, 16, 20.

[4] Similarly, the performance of SSC is robust to any sufficiently small value of LBSS.

Table 3
Comparison of SASE and MSSC with SSC and SC in mean best fitness.

Xovers	F1	F2	F3	F4	F5	F6	F7	F8
SC	0.30	0.40	0.11	0.27	0.15	0.25	1.67	1.19
SSC04	*0.16*	*0.20*	*0.06*	*0.16*	*0.08*	*0.13*	*1.06*	*0.62*
SSC05	*0.15*	*0.21*	*0.06*	*0.17*	*0.10*	*0.12*	*1.13*	*0.71*
SSC06	*0.20*	*0.21*	*0.06*	*0.18*	*0.08*	*0.12*	*0.97*	*0.61*
SASE65	*0.13*	*0.20*	*0.03*	*0.14*	*0.08*	***0***.09	***0***.58	*0.37*
SASE75	*0.14*	*0.20*	***0***.04	*0.16*	*0.07*	*0.10*	***0***.62	*0.46*
SASE85	*0.13*	*0.21*	*0.05*	*0.17*	*0.06*	*0.10*	***0***.07	***0***.44
SASES	*0.13*	*0.19*	***0***.03	*0.14*	*0.07*	***0***.09	***0***.55	***0***.38
MSSC12	*0.14*	*0.17*	***0***.04	***0***.12	*0.07*	*0.10*	***0***.31	***0***.28
MSSC16	*0.15*	*0.17*	***0***.04	***0***.12	*0.07*	***0***.09	***0***.54	*0.51*
MSSC20	*0.14*	*0.18*	***0***.03	***0***.12	*0.06*	***0***.09	***0***.50	*0.47*

4.2. Results and discussion

For all systems, a classic performance metric, mean best fitness, was recorded. Table 3 shows the best fitness found, averaged over all 100 runs of each GP system.[5] We tested the statistical significance of the results in Table 3 using a Wilcoxon signed-rank test with the confidence level of 95%. The results in Table 3 can be interpreted as follows:

1. If a run of SSC, SASE, or MSSC is not significantly better than SC, it is printed in plain face.
2. If a run of SSC, SASE, or MSSC is significantly better than SC, its is printed in italic face.
3. If a run of SASE or MSSC is significantly better than both SC and SSC, it is printed bold and italic.

It can be seen from Table 3 that both SASE and MSSC help to further improve GP performance. This is reflected by the better fitness of solutions found by SASE and MSSC compared with SSC. In both cases, there was little variation with parameter settings – i.e. the parameter values are robust.

The improvement is particularly striking for the two bivariate functions, F_7 and F_8. The Wilcoxon signed-rank test results confirm that all SSC based crossovers are significantly better than SC, and that in many cases, SASE and MSSC were significantly better than SSC, while MSSC was slightly better than SASE. To conclude, the results confirm that both these improvement not only overcome a weakness of SSC (by eliminating manual tuning of extra parameters) but also further improve SSC performance.[6]

5. Semantic locality of crossover operators

In previous work, we provided analyses of some important properties of SSC [49]. The semantic locality and constructive effect of SSC were argued to be the main reason behind the superior performance of SSC. In this section we further analyse semantic locality in SSC and MSSC to shed light on its impact on semantic based crossovers.[7] We start with an analysis of the semantic exchange between two subtrees in SC, investigating its constructive and semantic locality effects. We then look at the semantic exchange of subtrees in SSC and MSSC. Finally, we investigate the effects of semantic locality on other aspects of GP.

5.1. Semantic exchange in the standard subtree crossover

To investigate the semantic exchange in crossover, we collected data from the runs in the previous section. Let SSD be the subtree semantic distance between the two subtrees exchanged by SC. We divide SC crossovers into seven groups, as follows:

1. SC0 if SSD$< 10^{-3}$.
2. SC1 if $10^{-3} \leqslant$SSD<0.1.
3. SC2 if $0.1 \leqslant$ SSD < 0.2.
4. SC3 if $0.2 \leqslant$ SSD < 0.4.
5. SC4 if $0.4 \leqslant$ SSD < 0.8.
6. SC5 if $0.8 \leqslant$ SSD < 1.6.
7. SC6 if $1.6 \leqslant$ SSD.

[5] For the better comparison, the performance of standard crossover (SC) is also presented in this table.
[6] in Figs. 2 and 3 in Appendix B we draw the mean best fitness by generations and some properties of these crossovers that support for the results in this section.
[7] Since SASE works similarly to SSC except for using a dynamic UBSS, we do not analyse it further in the remainder of this paper.

Table 4
The percentage of standard crossover (SC) events falling into each group.

Xovers	F1	F2	F3	F4	F5	F6	F7	F8
SC0	14.3	14.7	17.1	13.2	14.0	14.3	12.3	12.7
SC1	2.74	2.79	4.59	0.95	1.13	0.06	4.77	5.43
SC2	3.22	3.06	5.62	3.75	3.71	2.43	4.85	4.35
SC3	4.69	4.50	10.9	11.5	6.27	2.85	28.9	24.7
SC4	21.9	22.3	31.5	25.4	11.7	11.3	28.7	30.0
SC5	38.3	38.4	17.6	24.6	34.0	35.2	14.2	15.8
SC6	14.7	14.1	12.4	20.6	29.0	33.3	6.21	6.79

Table 5
The change in semantics from parents to children using standard crossover.

Xovers	F1	F2	F3	F4	F5	F6	F7	F8
SC0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SC1	0.22	0.21	0.15	0.06	0.11	0.15	0.08	0.04
SC2	0.21	0.19	0.21	0.11	0.16	0.17	0.10	0.06
SC3	0.36	0.35	0.31	0.21	0.18	0.33	0.13	0.14
SC4	0.58	0.59	0.49	0.36	0.32	0.58	0.21	0.21
SC5	0.73	0.91	0.71	0.49	0.45	0.74	0.36	0.43
SC6	2.07	2.21	1.43	0.71	1.23	1.38	0.98	1.04

Table 6
Constructive events of standard crossover.

Xovers	F1	F2	F3	F4	F5	F6	F7	F8
SC0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SC1	9.41	9.39	8.39	8.04	9.40	7.83	12.1	12.1
SC2	6.75	6.94	5.01	5.43	7.70	6.03	7.40	7.34
SC3	4.46	4.49	3.58	3.54	5.81	3.82	5.81	5.63
SC4	2.54	2.71	2.59	2.54	3.96	2.81	4.33	4.15
SC5	1.82	1.95	1.42	1.60	1.98	1.94	2.98	2.87
SC6	1.31	1.39	1.22	1.27	1.63	1.52	2.32	2.17

The intermediate ranges are those which would be accepted by SSC.

We first recorded the proportion of crossovers (using SC) that fell into each group, averaged over runs and generations; the results are shown in Table 4. We can see that the proportion of crossover events with small semantic distance was very low. Most crossovers either exchange semantically identical subtrees (12–15%), resulting in wasted effort, or exchange highly dissimilar subtrees (50–70%). Smoother crossovers, exploring the search space locally, are comparatively rare.

We next recorded the SSD between parents and children in standard crossover, averaged over generations and runs. The results are shown in Table 5. We see that when SC exchanges equivalent subtrees, there is generally no change in the semantics between parents and children. When SC exchanges semantically similar subtrees, the change in semantics between parents and children is relatively smooth, by comparison with the exchange of dissimilar subtrees. Hence small semantic changes leads to smooth movement in the search space. This is important because it is difficult to achieve semantic locality through syntactic controls.

This smooth semantic movement leads to more constructive crossovers, as we can see from Table 6, which records the percentage of constructive events.[8] Constructive events were overwhelmingly concentrated in SC1–SC3 (4–12%), SC4...SC6 contributing much lower percentages, and SC0 contributing none.[9]

5.2. Semantic exchange in semantic based crossovers and its effects

The previous subsection showed that only a tiny portion of SC crossovers exchange semantically similar subtrees. Since SSC and MSSC aim to increase this exchange, we now examine how successful SSC and MSSC are in achieving their aim. As before, the upper bound semantic sensitivity of SSC was set at 0.4 and Max_Trial of MSSC was set at 12. SSC and MSSC were

[8] A crossover is considered constructive if both children have better fitness than their parents.

[9] Fig. 4 in Appendix B presents the contructive rate of SC1, SC3 and SC5 over the course of the evolutionary process.

Table 7
The percentage of SSC events falling into each group.

Xovers	F1	F2	F3	F4	F5	F6	F7	F8
SSC0	3.16	3.24	2.98	1.79	2.45	4.87	0.57	0.35
SSC1	24.1	24.1	24.5	6.36	12.1	8.71	20.5	11.7
SSC2	26.4	25.2	24.1	20.5	30.3	26.9	12.8	12.0
SSC3	30.8	31.1	41.5	60.7	43.3	30.2	64.5	74.5
SSC4	4.61	4.66	3.43	3.87	1.98	4.26	0.73	0.63
SSC5	8.08	8.50	1.91	3.44	5.54	12.7	0.53	0.48
SSC6	2.82	3.17	1.56	3.23	5.27	12.2	0.31	0.25

Table 8
The percentage of MSSC events falling into each group.

Xovers	F1	F2	F3	F4	F5	F6	F7	F8
MSSC0	1.42	1.51	2.16	1.85	3.15	2.40	0.93	1.08
MSSC1	40.3	40.9	51.3	16.1	22.7	13.3	57.0	50.4
MSSC2	21.1	20.3	20.0	31.7	27.4	25.3	18.2	17.4
MSSC3	20.4	19.6	17.4	35.5	22.3	24.4	20.3	25.8
MSSC4	10.9	11.3	5.90	8.8	7.7	16.3	2.31	3.23
MSSC5	4.77	5.17	1.98	3.69	11.1	13.0	0.87	1.22
MSSC6	0.97	1.17	1.16	2.24	5.42	5.30	0.44	0.69

Table 9
Some properties of SC, SSC and MSSC.

Properties	Xovers	F1	F2	F3	F4	F5	F6	F7	F8
Dif	SC	68.3	67.9	65.8	63.2	71.6	61.5	81.0	81.4
	SSC	83.1	84.5	81.7	81.4	87.3	79.2	93.5	94.1
	MSSC	86.2	87.3	83.2	84.9	90.1	91.3	95.8	95.2
Sem	SC	0.93	1.05	0.72	0.61	0.68	1.07	0.23	0.26
	SSC	0.48	0.44	0.26	0.25	0.27	0.50	0.11	0.10
	MSSC	0.42	0.41	0.22	0.21	0.22	0.43	0.09	0.08
Fit	SC	9.21	10.5	7.27	6.23	8.31	9.36	11.6	13.1
	SSC	4.23	4.49	2.74	2.82	3.12	4.32	5.58	6.40
	MSSC	3.81	4.23	2.58	2.65	3.02	4.16	5.37	6.18
Cons	SC	2.27	2.23	2.28	1.98	2.82	1.88	4.36	4.19
	SSC	5.12	5.16	4.87	4.36	5.43	4.60	6.73	6.40
	MSSC	6.25	6.62	6.14	5.96	6.84	6.14	8.04	8.12

divided into seven groups. The percentage of SSC and MSSC events that lie in each of the seven groups, averaged over all generations and runs, are shown in Tables 7 and 8.

We see from Tables 7 and 8 that the proportion of exchanges of semantically similar subtrees is substantially higher than for SC; 80–90% of SSC and MSSC exchanges lie in SSC1, SSC2 or SSC3. Exchange of semantically equivalent subtrees, and of semantically dissimilar subtrees, are both virtually eliminated. Thus SSC and MSSC achieve their design objective, of ensuring limited semantic change. Comparing SSC and MSSC, we see that they are quite similar. However, as will be seen below, MSSC is often better than SSC on some important aspects of GP.

The increase in the proportion of exchanges of semantically similar subtrees in SSC and MSSC results in a number of potentially valuable impacts. Table 9 shows that SSC and MSSC generated substantially more children that differed semantically from their parents (Dif row), often 20% higher. However even SC was generally able to change the semantics (60–80%), in strong contrast to McPhee's results on Boolean problems [34], where 60–70% of SC operation did not change semantics. This may help to explain why SAC and NSM (which only promote semantic diversity) showed only intermittent improvement on the results of SC.

Semantic locality of the crossover operator also leads to smoother movement in the search space. The Sem row of the table shows the average SSD between parents and children for SC, SSC and MSSC; the latter is generally substantially smaller, so that SSC and MSSC move more smoothly in the semantic space than SC. Row Fit of the table shows the average change of fitness before and after executing SC, SSC and MSSC – these values are smaller for SSC and MSSC. This substantially increases the constructive effect of crossover. Row Cons shows the percentage of these constructive events; SSC and MSSC were substantially more constructive than SC on all functions. The table also shows that MSSC is often better than SSC.

In summary, controlling crossover to ensure semantic locality results in a number of changes in GP dynamics. These shed light on the reasons for the performance gains seen in Section 4.

Table 10
The mean best fitness of SC, SySC, and SSC.

Xovers	F1	F2	F3	F4	F5	F6	F7	F8
SC	0.30	0.40	0.11	0.27	0.15	0.25	1.67	1.19
SySC6	0.33	0.43	0.15	0.30	0.16	0.21	1.69	1.17
SySC8	0.31	0.43	0.15	0.31	0.15	0.28	1.87	0.95
SySC10	0.33	0.45	0.14	0.30	0.13	0.24	1.40	1.18
SSC04	*0.16*	*0.20*	*0.06*	*0.16*	*0.08*	*0.13*	*1.06*	*0.62*
SSC05	*0.15*	*0.21*	*0.06*	*0.17*	*0.10*	*0.12*	*1.13*	*0.71*
SSC06	*0.20*	*0.21*	*0.06*	*0.18*	*0.08*	*0.12*	*0.97*	*0.61*

6. Semantic locality versus syntactic locality

The previous section have shown that improving semantic locality of genetic operators leads to a significant improvement in GP performance. This raises the question: which of semantic and syntactic locality is more important? In [16], it was shown that syntactic locality is important in a problem where the syntax of an individual effectively defines its fitness. Here, we investigate the effects of syntactic and semantic locality on more typical test problems (where syntax is only loosely correlated with fitness). First we define a syntactic version of SSC. Then we compare its effects on performance and search space with the semantically-based crossovers.

6.1. Syntactic similarity-based crossover

To define our syntactic crossover, we need a syntactic distance. We use the Ekart–Nemeth distance [8], using their algorithm to compute it.[10] From this, a syntactic similarity relationship between two (sub) trees is defined in a similar way to the semantic counterpart, the only difference being the use of syntactic (Ekart–Nemeth) distance rather than semantic distance. In other words, two subtrees are said to be syntactically similar if the syntactic distance (SyD) between them lies in a specific range. Formally, two subtrees S_1 and S_2 are syntactically similar (SySi) if

$$SySi(S_1, S_2) = \text{if } \alpha < \text{SyD}(S_1, S_2) < \beta$$

where α and β are two predefined constants, the *lower* and *upper* bounds for syntactic sensitivity. Here, α was set to 0, and several values of β were tested. Based on SySi, a syntactic similarity based crossover is proposed. The crossover for improving syntactic locality, *Syntactic Similarity based crossover* (SySC), is based on SSC, being computed the same except for the different similarity measure. Based on preliminary trials, we set the *Max_Trial* of SySC to 4.

6.2. Performance analysis

This subsection compares the performance of SySC, SSC and SC.[11] The problems are as in the previous sections. The basic GP parameter settings are given in Table 2. For SSC, three configurations similar to those in Section 4 are used. They are abbreviated as SSCX with $X = 04$, 05 and 06. For SySC, the the upper syntactic sensitivities were set at 6, 8, 10. These three values were calibrated from our experiments as good values for the performance of SySC. In total, three configurations of SySC were tested, denoted as SySCX with $X = 6$, 8, 10.

To compare the performance of the three operators, we used the mean best fitness. The results are shown in Table 10. Syntactically-bounded crossover (SySC) does not improve GP performance. The mean best fitness of SySC is generally equal to SC. In some cases SySC is better than SC and in other cases SC is better than SySC. The differences are not statistically significant. Conversely, SSC performs much better than both SC and SySC. This is consistent with the results in the previous section.

We also statistically tested the performance of SSC versus SC and SySC using the Wilcoxon signed-rank test with a confidence level of 95% (Table 10). If the improvement of SSC over SC and SySC is significant, the results of SSC are printed in italic face. The results in this table confirm the significant improvement of SSC over both SC and SySC. Thus despite some advantages for SySC in simplicity of implementation, SSC is a better bet than SySC in improving GP performance.

6.3. Fitness landscape analysis

Section 6.2 showed that improving semantic locality is more important than syntactic locality in improving GP performance. This subsection further investigates semantic versus syntactic locality of operators by analysing their fitness landscapes.

[10] The basic idea in calculating the Ekart–Nemeth distance is to overlay the two trees on the same shape, and count the number of positions having different content as the tree distance.

[11] We used SSC rather than MSSC for comparison, because of the closer analogy between it and SySC.

6.3.1. Experimental settings

To experimentally investigate the fitness landscape in the preceding problems we use a method proposed by Weinberger [51] known as an autocorrelation function.[12] Most previous work studied the fitness landscape of mutation [50,23], where a random walk can easily be generated by applying the mutation operator. Since this research focuses on crossover, we followed Riley and Ciesielski [39] in creating a random walk of N steps using Algorithm 3.

Algorithm 3.

Algorithm 3: Crossover Based Random Walk

```
Select individual I_0 randomly from the search space;
s=1;
repeat
    Select individual I_s ≠ I_{s-1} randomly from the search space;
    Generate p randomly in [0,1];
    if p < P_crossover then
        cross I_s with I_{s-1}, generating two neighbours I'_1 and I'_2 of I_{s-1};
        I_s = I'_1; (Discarding I'_2)
    Generate p randomly in [0,1];
    if p < P_mutation then
        mutate I_s;
    s = s + 1;
until s > N ;
```

We set $P_{crossover} = 0.9$ and $P_{mutation} = 0.05$ as before. We compare the fitness landscapes of SC, SySC, SSC and MSSC. For each, a random walk of 10000 steps was generated. 500 random walks were conducted for each problem. The lower semantic sensitivity used for SSC was 10^{-3} and the upper 0.4. The Max_Trial of SSC was set to 12. For MSSC we set Max_Trial = 12; for SySC we set the upper syntactic sensitivity to 8.

6.3.2. Results and dicussion

The results of the autocorrelation analysis are shown in Table 11.[13] Improving syntactic locality has limited impact on the fitness landscape. The autocorrelation value for SySC is only slightly better than for SC, with the only exceptions being for $h = 1$ and $h = 2, 4$ on function $F5$. A Wilcoxon signed-rank test with confidence level 95% gives mixed results, only some values for SySC are significantly greater than for SC with $h = 16$ and $h = 32$. That is why SySC does not improve much on SC.

Conversely, improving semantic locality of crossover smooths out the fitness landscape. The autocorrelation values for SSC and MSSC are always much greater than for SC and SySC. In all cases, SSC/MSSC differences from SC/SySC are significant at the 95% level (Wilcoxon signed-rank). MSSC is always more autocorrelated than SSC, providing an explanation for the different performance of these two operators.

7. Application of semantics based crossovers

This section presents two applications of semantics-based crossovers. The first applies semantics-based crossovers to a time series prediction problem and the second uses them in approximating the Gaussian Q-function.

7.1. Solving a time series prediction problem

In this subsection we test the usefulness of semantically based crossover on a realistic-scale application, Mackey–Glass time series prediction.[14] We compare the ability of GP to solve this problem with SC, SSC, SASE and MSSC.

7.1.1. Experimental settings

The requirement in time series prediction is to estimate the future values of a series based on its past values. In other words, it is to find a function F such that:

$$x(t+1) = F(x(t), x(t-\alpha), \ldots, x(t-N\alpha)) \tag{4}$$

This problem is generally considered quite tough, as there is no exact closed form solution [17]. We follow the approach of [12] in setting $\alpha = 6$ and $N = 8$. That is, our task is to estimate the value of the series at time $t+1$ given the 8 values at times $t, t-6, \ldots, t-42$ in the past.

[12] Appendix C gives a detailed description of the autocorrelation function.
[13] We omit the similar results of $F3, F4, F7$ and $F8$ to save space.
[14] Appendix D gives a detailed description of the Mackey–Glass time series.

Table 11
Autocorrelation analysis (larger values are better).

Distance (h)	Crossovers	F1	F2	F5	F6
	SC	0.604	0.605	0.651	0.639
1	SySC	0.600	0.599	0.648	0.638
	SSC	0.660	0.661	0.699	0.685
	MSSC	0.694	0.695	0.717	0.699
	SC	0.477	0.478	0.549	0.539
2	SySC	0.484	0.484	0.527	0.550
	SSC	0.551	0.552	0.599	0.599
	MSSC	0.593	0.594	0.631	0.625
	SC	0.345	0.346	0.417	0.416
4	SySC	0.354	0.354	0.409	0.437
	SSC	0.423	0.425	0.479	0.484
	MSSC	0.469	0.471	0.525	0.527
	SC	0.219	0.219	0.278	0.283
8	SySC	0.234	0.234	0.293	0.321
	SSC	0.285	0.286	0.341	0.355
	MSSC	0.333	0.334	0.410	0.409
	SC	0.111	0.112	0.159	0.160
16	SySC	0.127	0.127	0.179	0.185
	SSC	0.165	0.166	0.214	0.218
	MSSC	0.207	0.208	0.288	0.291
	SC	0.048	0.049	0.081	0.071
32	SySC	0.066	0.066	0.099	0.101
	SSC	0.088	0.088	0.125	0.131
	MSSC	0.108	0.109	0.177	0.175

Table 12
Training and test set accuracy and size for SC and other crossovers in Mackey–Glass prediction.

Xovers	Training	Test 0	Test 1	Test 2	Test 3	Size
SC	0.27	0.29	1.21	1.48	2.94	61.3
SSC	*0.20*	*0.22*	*1.05*	1.26	*1.85*	*58.2*
SASE	*0.19*	*0.21*	*1.01*	*0.99*	*1.88*	55.4
MSSC	*0.17*	*0.19*	*0.94*	*0.79*	*1.84*	*51.9*

The experimental settings are as in Table 2, the only difference being the terminal set, which here consists of one constant (1) and 8 variables $(X_1, X_2, \ldots, X_8)$ representing the time series values at $x(t), x(t-6), \ldots, x(t-42)$. The semantic sensitivities of SSC were $\alpha = 10^{-3}$ and $\beta = 0.4$. Max_Trial for SSC, and MSSC was set to 12. For SASE we used SASES.

We took the first 2320 points of Mackey–Glass time series as the data set. The first 1320 points were used for constructing the training set. These points are divided into 30 blocks of 44 consecutive points (indexed from 0 to 43). For each block, a fitness case for the training set is extracted as follows. The point at position 43 is the output and the 8 points at positions 42, 36, 30, ..., 0 are the inputs. Four test sets were designed and they will be referred to as Test k with $k = 0, 1, 2, 3$ in the following subsection. Each test set is extracted from the set of 1320 points which is comprised of the last $1320 - 10^k$ points of the first 1320 points of the series and the first 10^k points of the remaining 1000 points of the series, with $k = 0, 1, 2, 3$. For example, with $k = 3$, the set of points, upon which the test set is extracted, is built from the last 320 points of the first 1320 points (used for extracting the training set) and the remaining 1000 points of the whole data set. For each set of these 1320 points, a test set is extracted in a similar way as for the training set.

7.1.2. Results and discussion

For each run, we used the training set to provide the fitness function, selecting at the end the most accurate individual. This individual was tested on the four test sets. The results were averaged over 100 runs. We also measured the average size (number of nodes) of the fittest individual, again averaged over 100 runs. The overall results are shown in Table 12.

In this problem, semantic locality promotion (SSC, SASE and especially MSSC) had a positive effect, especially on longer horizons. That is, more local search helps to generate more general solutions (but also more accurate on the training set – this is not an issue of over-fitting).

We tested the statistical significance of differences from SC using the Wilcoxon signed-rank test with a confidence level of 95% (Table 12). If a method is significantly better than SC, the value is printed italic face. It can be seen from the table that most differences arising from locality promotion are significant.

Overall, locality-promotion not only enhances the accurate fitting of this time series, but also promotes good generalisation, especially for longer timescales.

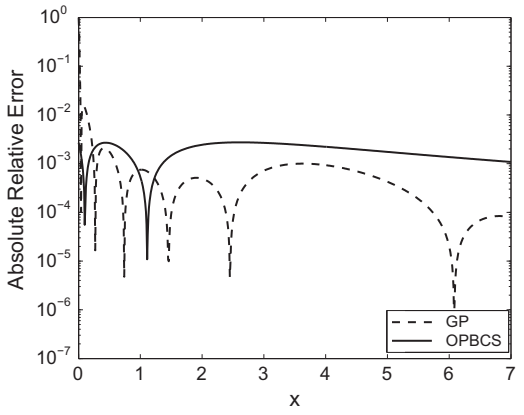

Fig. 1. Relative errors of the approximation found by GP and of OPBCS.

7.2. Approximating the Gaussian Q-function

This subsection illustrates another application of MSSC, in approximating the Gaussian Q-function.[15] These improved approximations to the Q-function have been previously reported in [7]; here we focus on the effects of MSSC.[16]

7.2.1. Experimental settings

For Q-function approximation, we follow previous work in the literature in using absolute relative error (ARE), so that the fitness function is:

$$\text{ARE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|f_i - y_i|}{f_i} \tag{5}$$

where N is the number of data samples (fitness cases), f_i is the value of the Q-function, and y_i is the function value of the individual at the ith point in the sample set. We used a sample set of 400 equidistant points in the interval $[0,8]$, together with the function values computed from Eq. (E.1). We used $\{+,-,*,/,\text{sqrt},\exp,\log\}$ as nonterminals, and $\{X,\pi,\text{ERC}\}$ as the terminal set, with protected versions of/, sqrt and log. ERC was an ephemeral random constant from $(-1,1)$. Population size was 3000, tournament size 9, and there were 50 trials.

7.2.2. Results and discussion

We aimed to find better solutions than the most accurate form found to date [3], known as OPBCS, so we accepted the best solution resulting from a run only if its ARE was less than that of OPBCS: 0.001747. With SC, we were unable to find any better solutions. However with MSSC, we found 3 runs yielding better solutions. The best solution found with SC had an ARE of 0.001923; with MSSC, the best solution had an ARE of 0.000863. Its form was:

$$\text{Best\_Solution} = \frac{f(x)}{g(x)(h_1(x)h_2(x)+x)} \tag{6}$$

where

$$f(x) = 0.283673x - 0.000547 \tag{7}$$

and

$$g(x) = \sqrt{e^{x^2}}(0.767402x - ln\sqrt{x+3.141592} + 0.998554) \tag{8}$$

and

$$h_1(x) = \sqrt{\left|\frac{ln(ax) - x - ln|b - \frac{x+c}{dx+e}|}{e^{x^2}}\right|} \tag{9}$$

with $a = 0.014896,\ b = 2.382099,\ c = 0.148960,\ d = 1.316667,\ e = 0.329776$ and

$$h_2(x) = \frac{0.058918x - 0.000929}{1.316667x + 0.332324} \tag{10}$$

[15] Since the previous results show that MSSC is generally better than SSC and SASE, we omit the latter two.
[16] Appendix E gives a detailed description of the Gaussian Q-function.

We visually show the performance of this approximation by plotting the relative error (along with that of OPBCS) in Fig. 1. The approximation found by MSSC is almost uniformly better than OPBCS, especially when $x > 1.5$. This demonstrates the practical effectiveness of GP with MSSC.

8. Conclusions

In this paper, we have extended the work in [49], further exploring the role of semantic locality in GP crossover. We proposed two improvements, SASE and MSSC, of SSC, and provided a deeper analysis of the semantic exchange generated by these operators. We also showed that semantic locality is more beneficial in GP crossover than syntactic locality. Together with the evidence in [49], that semantic locality is more beneficial than semantic diversity, we conclude that semantic locality is a truly important property that should be considered in designing GP crossover operators. Finally, we showed that MSSC improves GP performance on two difficult real-world problems: time series prediction problem and approximating the Gaussian Q-function.

9. Future work

In the near future, we plan to extend this work in a number of ways. Firstly our problem domain so far has been mainly real-valued symbolic regression. However this is not a serious limitation of the method, as any classification problem could be recast as a real-valued regression problem (e.g. see [18] and references therein). We are planning to apply semantic similarity based crossovers to classification problems. We also plan to work on Boolean problems. Even though we could treat boolean problems as binary classification problems with binary data attributes, and could therefore solve them directly, semantic similarity calibrated for a Boolean domain could potentially be a better solution (see [47] for some preliminary results).

At a more theoretical level, we would like to investigate the relationship between improving semantic locality and GP bloat. Previous work has shown that semantics-based crossovers reduce GP bloat and improve generalisation capacity [45]. We would like to understand why.

Finally, we plan to further investigate the relationship between syntactic and semantic change in semantically-based crossover.

Acknowledgment

The work in this paper was funded by The Vietnam National Foundation for Science and Technology Development (NAFOSTED), under Grant No. 102.01-2011.08. The ICT of Seoul National University provided research facilities for this study.

Appendix A. The list of abbreviations

See Table A.13.

Appendix B. Some additional figures

This appendix presents some additional figures for the results in Sections 4 and 5. For all Figures in this appendix, we use SSC with the upper bound semantic sensitivity set to 0.4 (SSC04). With SASE we use the version where the rate of successful SSC is adapted (SASES) and with MSSC, the *Max_Trial* was set as 12.

Fig. 2 shows the mean best fitness of four crossovers over the course of the evolutionary process on some function including F3, F4, F7 and F8. It can be seen from this figure that the improvement of SSC, SASE and MSSC over standard crossover are maintained over the whole evolutionary process, none of these crossovers becoming trapped in local optima.

Table A.13
Abbreviations in the paper.

Abbreviation	Meaning
GP	Genetic Programming
SS	Sampling Semantics
SSD	Subtree Semantic Distance
SC	Standard Crossover
SAC	Semantic Aware Crossover
SSC	Semantic Similarity based Crossover
LBSS	The lower bound semantic sensitivity
UBSS	The upper bound semantic sensitivity
SASE	Self-Adaptive Successful Execution
MSSC	The Most Semantic Similarity based Crossover
SySC	Syntactic Similarity based Crossover

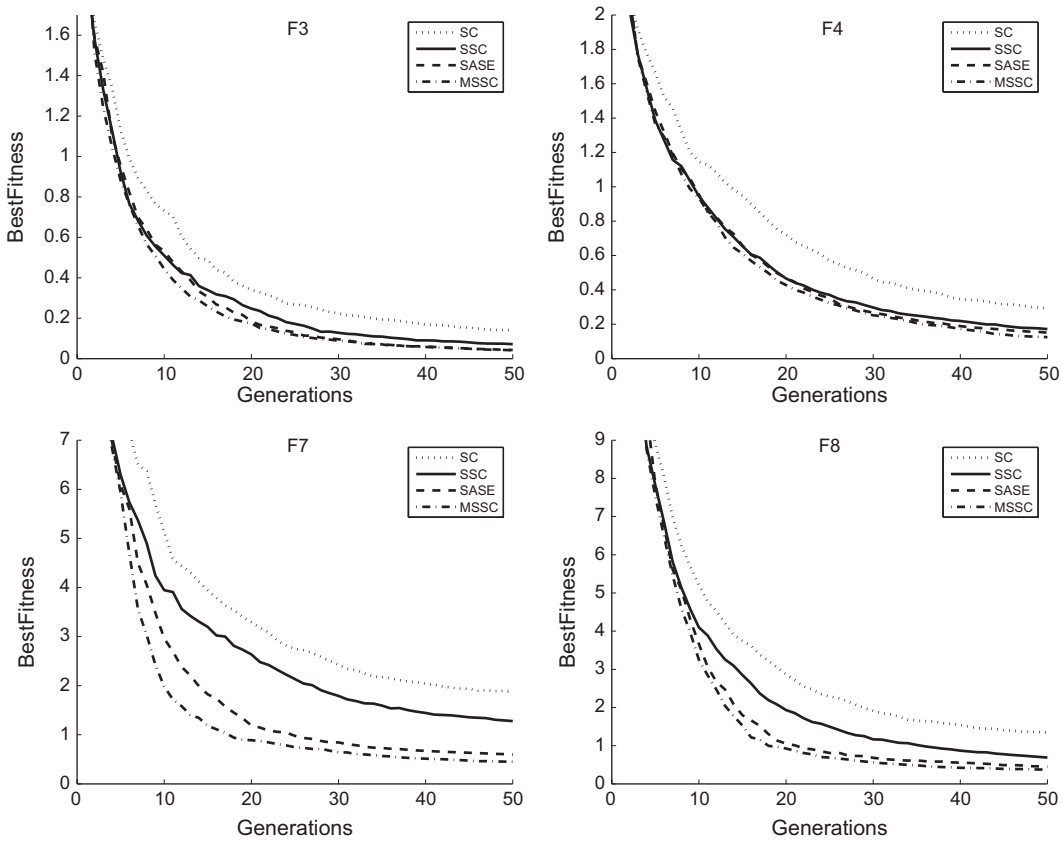

Fig. 2. Mean best fitness of four methods over generations on some functions.

Some detailed properties of the semantics-based crossover on function F3 are plotted in Fig. 3. It shows that semantics-based crossovers maintain better diversity (Fig. 3a), and higher locality (Fig. 3b), generate less bloat (except for SSC), (Fig. 3c) and have a better constructive ratio (Fig. 3d) on this problem.

Fig. 4 presents the constructive rates of three levels (SC1, SC3 and SC5) of standard crossover. It can be observed that SC1 has a much higher constructive ratio than SC3 and SC5. Thus greater locality (smaller changes as in SC1) leads to higher constructivity in crossover.

Appendix C. Autocorrelation function

The autocorrelation function was proposed by Weinberger [51]. For a given fitness landscape f (the fitness function), a starting point s_0 is randomly selected. Random sampling is used to generate a random walk of N steps $F = \{f(s_i)\}_{i=0}^{N}$. The autocorrelation function is:

$$\rho(h) = \frac{R(h)}{s_f^2} \tag{C.1}$$

where h is the distance between two points in the random walk, and s_f^2 is the variance of the sequence, calculated as:

$$s_f^2 = \frac{\sum_{i=0}^{N}(f(s_i) - m_F)^2}{N+1} \tag{C.2}$$

and $R(h)$ is the auto-covariance function of sequence F. For each $h, R(h)$ is estimated by:

$$R(h) = \frac{\sum_{i=0}^{N-h}(f(s_i) - m_F) \cdot (f(s_{i+h}) - m_F)}{N-h+1} \tag{C.3}$$

where m_F is mean of fitness sequence $F, m_F = \frac{1}{N+1}\sum_{i=0}^{N}(f(s_i))$. The autocorrelation function indicates the correlation between points that are separated by a distance h. A smooth landscape is highly correlated, as the fitness difference between a point and its neighbours is small. In this case the autocorrelation function is greater. Conversely, if a fitness landscape is rugged, the fitness difference between neighbouring points is high, and the autocorrelation function is smaller.

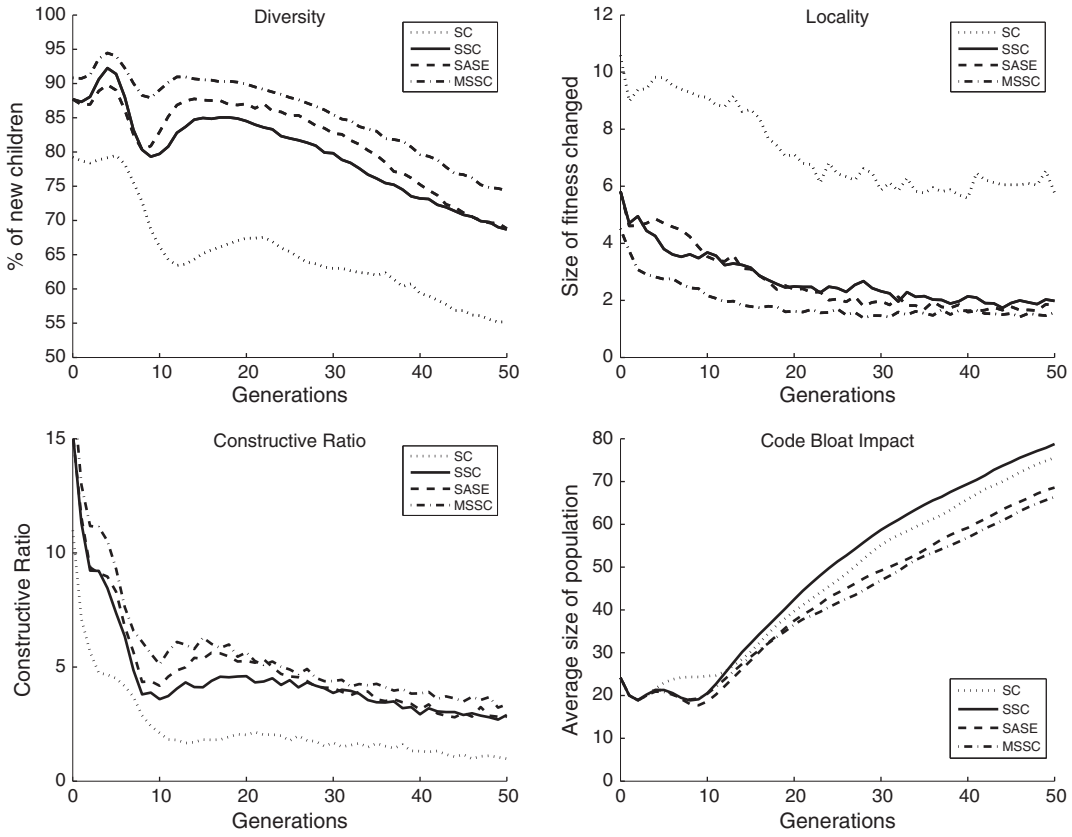

Fig. 3. The diversity, locality, average of population size and constructive ratio of four methods on functions F3 over generations.

Appendix D. The Mackey–Glass time series

Mackey and Glass introduced their time series in 1977 [32]. It has been widely used as a benchmark for the generalisation ability of a variety of machine learning methods. The Mackey–Glass equation is:

$$x(t+1) = x(t) - bx(t) + a\frac{x(t-\tau)}{1+x(t-\tau)^{10}} \tag{D.1}$$

With $a = 0.2,\ b = 0.1$ and $\tau = 17$, the time series is aperiodic, non-convergent, and completely chaotic. Fig. 5 plots the first 2000 points.

Appendix E. The Gaussian *Q*-function

The Gaussian *Q*-function is important across of a wide range of fields, because of the generally accepted assumption, that unknown system noise is Gaussian [44]. Thus computing expectations implies integrating over the tail of the Gaussian, so that we need to compute the function $Q(x)$ defined by:

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-y^2/2}\, dy \tag{E.1}$$

A detailed discussion of the importance of the Q function, of the difficulties in approximating it, and of previous work on this problem, are included in [7]. As with Mackey–Glass, there can be no closed form solution [40]. The most accurate form found to date [3], is:

$$Q(x) \approx \frac{1}{(1-a)x + a\sqrt{x^2+b}} \cdot \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} \tag{E.2}$$

with $a = 0.339$, $b = 5.510$; it is known as the OPBCS approximation.

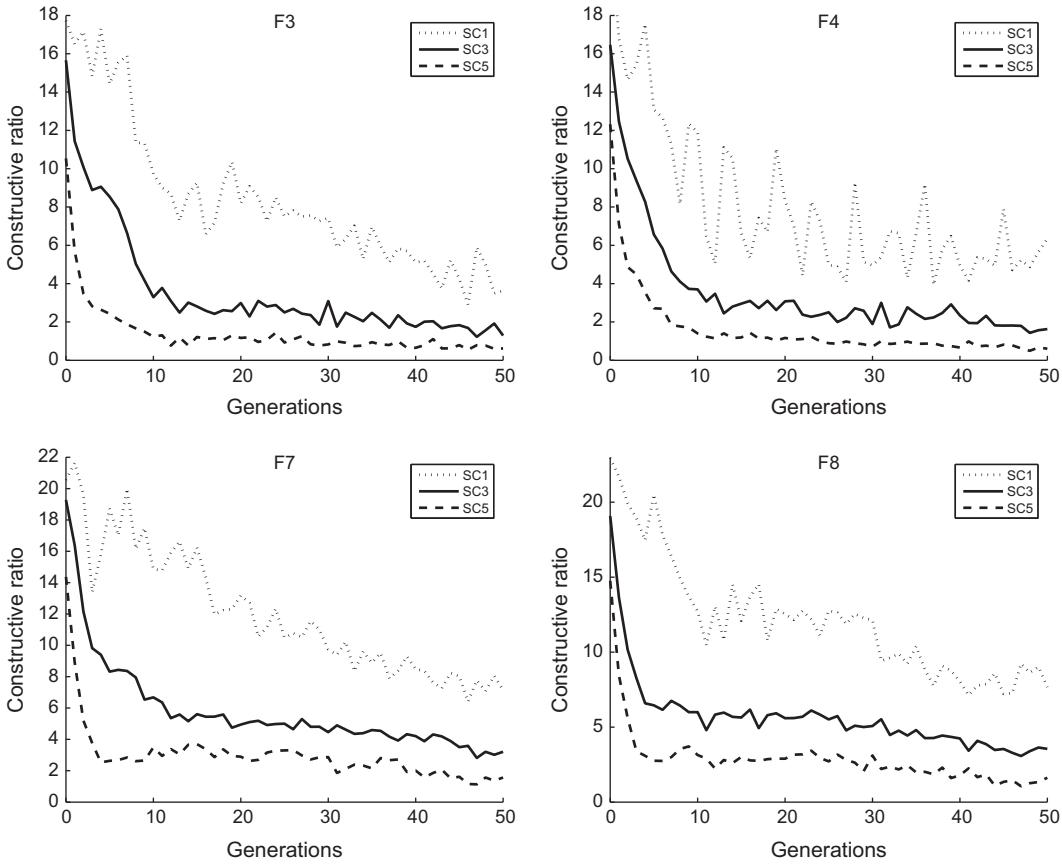

Fig. 4. The constructive rate of three groups of standard crossover.

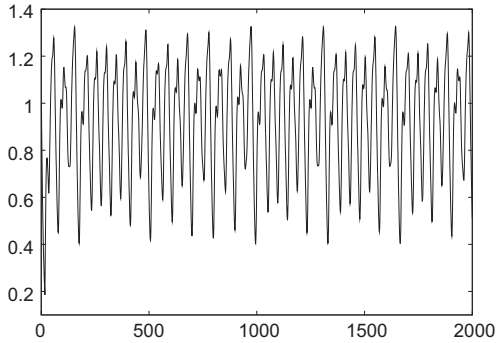

Fig. 5. Mackey–Glass first 2000 points ($a = 0.2,\ b = 0.1$ and $\tau = 17$).

References

[1] L. Altenberg, The evolution of evolvability in genetic programming, in: Advances in Genetic Programming, MIT Press, 1994, pp. 47–74.
[2] L. Beadle, C. Johnson, Semantically driven crossover in genetic programming, in: Proceedings of the 2008 I EEE World Congress on Computational Intelligence, IEEE Press, 2008, pp. 111–116.
[3] P. Borjesson, C. Sundberg, Simple approximations of the error function $q(x)$ for communications applications, IEEE Transactions on Communications 27 (1979) 639–643.
[4] C. Wedekind, T. Seebeck, F. Bettens, A. Paepke, Mhc-dependent mate preferences in humans, Proceedings of Biological Sciences 260 (1995) 245–249.
[5] R. Cleary, M. O'Neill, An attribute grammar decoder for the 01 multi-constrained knapsack problem, in: Proceedings of the 2005 European Conference on Evolutionary Computation in Combinatorial Optimization, Springer, 2005, pp. 34–45.
[6] M. de la Cruz Echeanda, A.O. de la Puente, M. Alfonseca, Attribute grammar evolution, in: Proceedings of the IWINAC 2005, Springer Verlag, Berlin, Heidelberg, 2005, pp. 182–191.
[7] N.P. Dao, Q.U. Nguyen, X.H. Nguyen, R.I.B. McKay, Evolving approximations for the gaussian q-function by genetic programming with semantic based crossover, in: Proceedings of the IEEE World Congress on Computational Intelligence, IEEE Press, 2012, pp. 1924–1929.

[8] A. Ekart, S.Z. Nemeth, A metric for genetic programs and fitness sharing, in: R. Poli, W. Banzhaf, W.B. L angdon, J.F. Miller, P. Nordin, T.C. Fogarty (Eds.), Genetic Programming, Proceedings of EuroGP'2000, LNCS, vol. 1802, Springer-Verlag, Edinburgh, 2000, pp. 259–270.
[9] M. Elliot, B. Crespi, Placental invasiveness mediates the evolution of hybrid inviability in mammals, The American Naturalist 168 (1) (2006) 114.
[10] E. Galvan-Lopez, J. McDermott, M. O'Neill, A. Brabazon, Defining locality as a problem difficulty measure in genetic programming, Genetic Programming and Evolvable Machines 12 (4) (2011) 365–401.
[11] J. Gottlieb, G. Raidl, The effects of locality on the dynamics of decoder-based evolutionary search, in: Proceedings of the 2000 Genetic and Evolutionary Computation Conference, ACM, 2000, pp. 283–290.
[12] E. Group, Evolutionary and Neural Computation for Time Series Prediction Minisite, Website, 2005. <http://tracer.uc3m.es/tws/TimeSeriesWeb/repo.html>.
[13] S. Gustafson, E.K. Burke, G. Kendall, Sampling of unique structures and behaviours in genetic programming, in: Proceedings of 7th European Conference on Genetic Programming, LNCS, vol. 3003, Springer-Verlag, 2004, pp. 279–288.
[14] S. Gustafson, E.K. Burke, N. Krasnogor, On improving genetic programming for symbolic regression, Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 1, IEEE Press, 2005, pp. 912–919.
[15] R. Harper, A. Blair, A self-selecting crossover operator, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE Press, 2006, pp. 5569–5576.
[16] N.X. Hoai, B. McKay, D. Essam, Representation and structural difficulty in genetic programming, IEEE Transactions on Evolutionary Computation 10 (2) (2006) 157–166.
[17] T. Hu, W. Banzhaf, The role of population size in rate of evolution in genetic programming, in: Proceedings of the EuroGP'2009, LNCS, vol. 5481, Springer, 2009, pp. 85–96.
[18] H. Jabeen, A.R. Baig, Review of classification using genetic programming, International Journal of Engineering Science and Technology 2 (2) (2010) 94–103.
[19] C. Johnson, Deriving genetic programming fitness properties by static analysis, in: Proceedings of the 4th European Conference on Genetic Programming (EuroGP'2002)., LNCS, vol. 2278, Springer, 2002, pp. 299–308.
[20] C. Johnson, Genetic programming with guaranteed constraints, in: Proceedings of the 4th International Conference on Recent Advances in Soft Computing (2002), The Nottingham Trent University, 2002b, pp. 134–140.
[21] C. Johnson, Genetic programming with fitness based on model checking, in: Proceedings of the 10th European Conference on Genetic Programming (EuroGP2007), LNCS, vol. 4445, Springer, 2007, pp. 114–124.
[22] C. Johnson, Genetic programming crossover: does it cross over?, in: Proceedings of the 12th European Conference on Genetic Programming (EuroGP2009), LNCS, vol. 5481, Springer, 2009, pp. 97–108.
[23] T. Jones, Evolutionary Algorithms, Fitness Landscapes and Search. Ph.D. thesis, University of New Mexico, Albuquerque, NM, 1995.
[24] G. Katz, D. Peled, Model checking-based genetic programming with an application to mutual exclusion, Tools and Algorithms for the Construction and Analysis of Systems 4963 (2008) 141–156.
[25] M. Keijzer, Improving symbolic regression with interval arithmetic and linear scaling, in: Proceedings of EuroGP'2003, LNCS, vol. 2610, Springer-Verlag, 2003, pp. 70–82.
[26] D. Knuth, Semantics of context-free languages, Mathematical Systems Theory 2 (1968) 95.
[27] J. Koza, Genetic Programming: On the Programming of Computers by Natural Selection, MIT Press, MA, 1992.
[28] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, The MIT Press, Cambridge, MA, 1992.
[29] K. Krawiec, P. Lichocki, Approximating geometric crossover in semantic space, in: Proceedings of Genetic and Evolutionary Computation Conference, GECCO'2009, ACM, 2009, pp. 987–994.
[30] W.B. Langdon, Size fair and homologous tree genetic programming crossovers, in: Proceedings of the 1999 Genetic and Evolutionary Computation Conference, Morgan Kaufman, 1999, pp. 1092–1097.
[31] P.C. Lin, J.S. Chen, Fuzzytree crossover for multi-valued stock valuation, Information Sciences 177 (5) (2007) 1193–1203.
[32] M.C. Mackey, L. Glass, Oscillation and chaos in physiological control systems, Science 197 (1977) 287–289.
[33] H. Majeed, C. Ryan, A less destructive, context-aware crossover operator for gp, in: Proceedings of the 9th European Conference on Genetic Programming (EuroGP2006), LNCS, vol. 5, LNCS, 2006, pp. 36–48.
[34] N. McPhee, B. Ohs, T. Hutchison, Semantic building blocks in genetic programming, in: Proceedings of 11th European Conference on Genetic Programming (EuroGP2008), LNCS, vol. 4971, Springer, 2008, pp. 134–145.
[35] A. Moraglio, K. Krawiec, C.G. Johnson, Geometric semantic genetic programming, in: Parallel Problem Solving from Nature, PPSN XII (Part 1), Lecture Notes in Computer Science, vol. 7491, Springer, Taormina, Italy, 2012, pp. 21–31.
[36] U.M. O'Reilly, F. Oppacher, Program search with a hierarchical variable length representation: genetic programming, simulated annealing and hill climbing, in: Parallel Problem Solving from Nature, LNCS, vol. 866, Springer-Verlag, 1994, pp. 397–406.
[37] R. Poli, W.B. Langdon, Genetic programming with one-point crossover, in: Proceedings 2007 of Soft Computing in Engineering Design and Manufacturing Conference, Springer-Verlag, 1997, pp. 180–189.
[38] R. Poli, W.B. Langdon, N. McPhee, A Field Guide to Genetic Programming, Lulu Enterprises Ltd., UK, 2008.
[39] J. Riley, V. Ciesielski, Fitness landscape analysis for evolutionary non-photorealistic rendering, in: IEEE Congress on Evolutionary Computation (CEC 2010), IEEE Press, Barcelona, Spain, 2010, pp. 1–9.
[40] M. Rosenlicht, Liouville's theorem on functions with elementary integrals, Pacific Journal of Mathematics 24 (1) (1968) 153–161.
[41] F. Rothlauf, Representations for Genetic and Evolutionary Algorithms, 2nd ed., Springer, 2006.
[42] F. Rothlauf, M. Oetzel, On the locality of grammatical evolution, in: Proceedings of the 9th European Conference on Genetic Programming (EuroGP2006)., LNCS, vol. 3905, Springer, 2006, pp. 320–330.
[43] S. Silva, S. Dignum, Extending operator equalisation: fitness based self adaptive length distribution for bloat free GP, in: Proceedings of the 12th European Conference on Genetic Programming, LNCS, vol. 5481, Springer, 2009, pp. 159–170.
[44] M. Simon, Probability Distributions Involving Gaussian Random Variables: A Handbook for Engineers and Scientists, Kluwer Academics, 2002.
[45] N.Q. Uy, N.T. Hien, N.X. Hoai, M. O'Neill, Improving the generalisation ability of genetic programming with semantic similarity based crossover, in: Proceedings of EuroGP'2010, LNCS, vol. 6021, Springer, 2010, pp. 184–195.
[46] N.Q. Uy, N.X. Hoai, M. O'Neill, Semantic aware crossover for genetic programming: the case for real-valued function regression, in: Proceedings of EuroGP'09, LNCS, vol. 5481, Springer, 2009, pp. 292–302.
[47] N.Q. Uy, N.X. Hoai, M. O'Neill, B. Mckay, Semantics based crossover for boolean problems, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2010, ACM, Portland, Oregon, 2010, pp. 869–876.
[48] N.Q. Uy, M. O'Neill, N.X. Hoai, B. McKay, E.G. Lopez, Semantic similarity based crossover in GP: the case for real-valued function regression, in: Evolution Artificielle, 9th International Conference (EA2009), vol. 5975 of LNCS, 2009b, pp. 170–181.
[49] N.Q. Uy, M. O'Neill, N.X. Hoai, B. McKay, E.G. Lopez, Semantically-based crossover in genetic programming: application to real-valued symbolic regression, Genetic Programming and Evolvable Machines 12 (2) (2011) 19–91.
[50] L. Vanneschi, M. Tomassini, P. Collard, M. Clergue, Fitness distance correlation in structural mutation genetic programming, in: C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, E. Costa (Eds.), Genetic Programming, Proceedings of EuroGP'2003. Essex, LNCS, vol. 2610, Springer-Verlag, 2003, pp. 455–464.
[51] E.D. Weinberger, Correlated and uncorrelated fitness landscapes and how to tell the difference, Biological Cybernetics 63 (1990) 325–336.
[52] M.L. Wong, K.S. Leung, An induction system that learns programs in different programming languages using genetic programming and logic grammars, in: Proceedings of the 1995 IEEE International Conference on Tools with Artificial Intelligence, 1995, pp. 380–387.