Improving the Generalisation Ability of Genetic Programming with Semantic Similarity based Crossover

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

[1]Natural Computing Research & Applications Group, University College Dublin, Ireland
[2]School of Information Technology, Vietnamese Military Technical Academy, Vietnam
quanguyhn@gmail.com,hien_cpqn@yahoo.com,nxhoai@gmail.com,m.oneill@ucd.ie

Abstract. This paper examines the impact of semantic control on the ability of Genetic Programming (GP) to generalise via a semantic based crossover operator (Semantic Similarity based Crossover - SSC). The use of validation sets is also investigated for both standard crossover and SSC. All GP systems are tested on a number of real-valued symbolic regression problems. The experimental results show that while using validation sets barely improve generalisation ability of GP, by using semantics, the performance of Genetic Programming is enhanced both on training and testing data. Further recorded statistics shows that the size of the evolved solutions by using SSC are often smaller than ones obtained from GP systems that do not use semantics. This can be seen as one of the reasons for the success of SSC in improving the generalisation ability of GP.

Key words: Genetic Programming, Semantics, Generalisation, Crossover

1 Introduction

Genetic Programming (GP) [23, 17] researchers are in recent times paying increasing attention to semantic information, with a dramatic increase in the number of publications (e.g., [11–13, 15, 14, 2, 21, 24, 25, 3]). Previously, research has focused on syntactic aspects of GP representation. From a programmer's perspective, however, maintaining syntactic correctness is only one part of program construction: not only must programs be syntactically correct but also semantically correct. Thus incorporating semantic awareness in the GP evolutionary process could improve its performance, extending the applicability of GP to problems that are difficult with purely syntactic GP.

In the field of Machine Learning (ML), generalisation has been seen as one of the most desirable properties for learning machines [22]. As GP could be seen as a (evolutionary) machine learning methodology, it is very important to guarantee that the solutions GP finds, not only work well on training data but also on the unseen data [5]. Surprisingly, a lot of GP researchers only report results on training data. While overfitting the training data to get the exact solutions is suitable in some cases, for most of learning problems in reality it would be not enough without considering their generalisation over unseen data. Some recent works (e.g. [5, 26, 9]) have showed that the ability of GP to generalise could be poor. The awareness of the ability of GP to generalise is also important in the context of performance comparison between different GP systems. It has been recently shown in [5] that an enhanced GP system performance might

be remarkably better than standard GP on training data, but not significantly better on unseen data.

The previous research on improving the ability of GP to generalise is mostly focused on reducing the solution size [26, 9, 20]. The motivation for such an approach is that GP usually bloats, with solution complexity (size) increasing rapidly during the evolutionary process. The high complexity solutions are often poor in their ability to generalise as they contradict Ockham's razor principles in Machine Learning [22] (simple solutions are prefered). To the best of our knowledge, there has not been any work on the effect of semantic control on the ability of GP to generalise. In this paper, we demonstrate a new and semantic based approach to improve GP in finding solutions that have better properties of generalisation. In particular, we test if a recently proposed semantics based crossover, namely Semantic Similarity based Crossover (SSC) [25], could improve the ability of GP to generalise. The experimental results show the effectiveness of the SSC approach in comparison with both standard GP and the validation set based method. The remainder of the paper is organised as follows. In the next section we review the literature on GP with semantics and GP generalisation. The semantics based crossover (SSC) is described in Section 3 followed by the experimental settings. The experimental results are shown and discussed in Section 5. The last section concludes the paper and highlights some future work.

2 Related Work

Although generalisation of learned solutions is the primary interest of any learning machine [22], it was not seriously considered in the field of GP for a long time. Before Kushchu published his work on the generalisation ability of GP [19], there were rather few research dealing with the GP generalisation aspect. Francone et al. [8] proposed a new GP system called Compiling GP (CGP) and the authors compared its generalisation ability with that of other ML techniques. The results show that the ability of CGP to generalise compares favourably with a number of more traditional ML methods. Furthermore, the influence of using extensive mutation on the ablity of CGP to generalise was investigated and the experimental results show positive effects [1].

Recently, the issue of generalisation in GP is deservedly receiving increased attention. Mahler et al. [20] experimented with Tarpeian Control on some symbolic regression problems and tested the side effects of this method on the generalisation ability of GP. The results were inconsistent and problem dependent, i.e., it can either increase or reduce the generalisation power of solutions found by GP. Gagne et al. [9] investigated two methods to improve generalisation in GP-based learning: the selection of the best of run individuals using a three datasets method (training, validation, and test sets), and the application of parsimony pressure in order to reduce the complexity of the solutions. Their experimental results indicate that using a validation set could slightly improve the stability of the best of run solutions on the test sets. Costa et al. [4] proposed a new GP system called relaxed Genetic Programming (RGP) with generalisation ability better than standard GP.

More recently, Costelloe and Ryan [5] showed the important role of generalisation on GP. They experimentally showed that a technique like Linear Scaling [16] may only

be significantly better than standard GP on training data but not superior on testing data. They proposed an approach to improve GP generalisation by combining Linear Scaling and the No Same Mate strategy [10]. Vanneschi and Gustafson [26] improved GP generalisation using a crossover based similarity measure. Their method is to keep a list of over-fitting individuals and to prevent any individual entering the next generation if it is similar (based on structural distance or a subtree crossover based similarity measure) to one individual in the list. The method was then tested on a real-life drug discovery regression problem and the experimental results showed improvements on the ability to generalise. Most research on improving the ability of GP to generalise has been purely focused on reducing the complexity of the solution and semantic control has never been considered as an approach to enhance ability of GP to generalise.

The use of semantics in GP has recently attracted increasing attention by researchers in the field. There are three main approaches to representing, extracting, and using semantics to guide the evolutionary process: (a) using grammar-based approaches [27, 3, 6], (b) using formal methods [11, 13, 15], and (c) based on GP s-tree representations [2, 21, 24]. In [25], a more detailed review of semantics usage and control in GP is given.

Most of previous research on semantics in GP were focused on combinatorial and boolean problems such as the Knapsack problem [3], Boolean problems [2, 21], and Mutual Exclusion problems [15]. Recently, researchers have investigated the effect of semantic control in GP for problems in real-valued domains [24, 25, 18]. Krawiec [18] proposed a way to measure the semantics of an individual that is based on fitness cases. This semantics is then used to guide crossover (*Approximating Geometric Crossover* - AGC). The experiments conducted on both real-valued and boolean regression problems show that AGC is not better than standard subtree crossover (SC) on the tested real-valued problems and only slightly better than SC on the boolean ones. Uy et al. [24] proposed a new crossover operator, namely 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 was empirically shown to improve GP performance. SAC was then extended to Semantic Similarity based Crossover (SSC) [25]. The experimental results show that the performance of SSC is superior than both of SC and SAC on the tested problems. However, the performance measure was more focused on finding exact solutions (overfitting). It is interesting to see if this semantic based operator could also help to improve the ability of GP to generalise.

3 Semantic Similarity based Crossover

Semantic Similarity based Crossover (SSC) [25] is inspired and extended from earlier research on Semantics Aware Crossover (SAC) [24]. SSC described in this paper is almost identical to that described by Uy et al. [25] with a slightly modified semantic distance measure. Since SSC operates on the semantics of subtrees, first a defintion of subtree semantics is needed. Formally, the *Sampling Semantics* of any (sub)tree is defined as follows:

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

The value of N depends on the problems. If it is too small, the approximate semantics might be too coarse-grained and not sufficiently accurate. If N is too big, the approximate semantics might be more accurate, but more time consuming to measure. The choice of P is also important. If the members of P are too closely related to the GP function set (for example, π for trigonometric functions, or e for logarithmic functions), then the semantics might be misleading. For this reason, choosing them randomly may be the best solution. In this paper, the number of points for evaluating *Sampling Semantics* is set as the number of fitness cases of problems (30 points for $F_1, F3$ and F_5, 60 points for $F_2, F4$ and F_6, see Section 4), and we choose the set of points P uniformly randomly from the problem domain.

Based on *Sampling Semantics* (SS), we define a *Sampling Semantics Distance* between two subtrees. In the previous work [25], *Sampling Semantics Distance* (SSD) was defined as the sum of absolute difference of all values of SS. While the experiments show that this kind of SSD is acceptable, it has undoubted weakness that the value of SSD strongly depends of the number of SS points (N) [25]. To soften this drawback, in this paper we use the mean of absolute distance as the SSD between subtrees. In other words, let $U = \{u_1, u_2, ..., u_N\}$ and $V = \{v_1, v_2, ..., v_N\}$ be the SS of $Subtree_1(St_1)$ and $Subtree_2(St_2)$ on the same set of evaluating values, then the SSD between St_1 and St_2 is defined as follows:

$$SSD(St_1, St_2) = \frac{|u_1 - v_1| + |u_2 - v_2| + ... + |u_N - v_N|}{N} \tag{1}$$

Thanks to SSD, a relationship known as *Semantic Similarity* is defined. The intuition behind semantic similarity is that exchange of subtrees is most likely to be beneficial if the two subtrees are not semantically identical, but also they are not too semantically dissimilar. Two subtrees are semantically similar on a domain if their SSD on the same set of points in that domain lies within a positive interval. The formal definition of semantic similarity (SSi) between subtrees St_1 and St_2 is as follows:

$$\begin{aligned} SSi(St_1, St_2) = \ & \textbf{if } \alpha < \text{SSD}(St_1, St_2) < \beta \\ & \textbf{then } \text{true} \\ & \textbf{else } \text{false} \end{aligned}$$

here α and β are two predefined constants, known as the *lower* and *upper bounds* for semantic sensitivity, respectively. Conceivably, the best values for *lower* and *upper bound semantic sensitivity* might be problem dependent. However we strongly suspect that for almost any symbolic regression problem, there is a range of values that is appropriate [25]. The investigation of the effect of different semantic sensitivities on SSC performance is beyond the scope of this paper. In this paper, we set $\alpha = 10^{-4}$ and $\beta = 0.4$ which are good values found in the literature [25].

Inspired from the difficulty in designing an operator with the property of high locality in GP, SSC was proposed with the main objective being to improve the locality of

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;
    generate a number of random points (P) on the problem domain;
    calculate the SSD between Subtree1 and Subtree2 on P
    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;
```

crossover. SSC is in fact an extension of SAC in two ways. Firstly, when two subtrees are selected for crossover, their semantic similarity, rather than semantic equivalence as in SAC, is checked. Secondly, semantic similarity is more difficult to satisfy than semantic equivalence, so repeated failures may occur. Thus SSC uses multiple trials to find a semantically similar pair, only reverting to random selection after passing a bound on the number of trials. Algorithm 1 shows how SSC operates in detail. In our experiments, the value of Max_Trial was set to 12, with this value having been calibrated by earlier experimental results.

4 Experimental Setup

To investigate the impact of SSC on the ability of GP to generalise, we used six real-valued symbolic regression problems. The tested problems, training and testing data are shown in Table 1. These functions were taken from some other work on GP learning generalisation [5, 7, 16]. It is noted that the testing sets are often much larger than the training sets and in some cases they contain values that are not in the training intervals (F_5, F_6). This makes the experimental setting more general.

The GP parameters used for our experiments are shown in Table 2. Despite this being an experiment purely concerned with generalisation ability of crossover, we have retained mutation with a small rate in the system because the aim of the experiment is to study crossover in the context of a normal GP run. Our experiments were conducted on four configurations as follows:

Table 1. Symbolic Regression Functions.

Functions	Training Data	Testing Data
$F_1 = x^4 + x^3 + x^2 + x$	30 random points $\subseteq$ [-1,1]	100 $\subseteq$[-1:0.02:1]
$F_2 = x^3 - x^2 - x - 1$	60 random points $\subseteq$ [-1,1]	100 $\subseteq$[-1:0.02:1]
$F_3 = arcsin(x)$	30 random points $\subseteq$ [-1,1]	200 $\subseteq$[-1:0.01:1]
$F_4 = \sqrt{x}$	60 random points $\subseteq$ [0,4]	200 $\subseteq$[0:0.02:4]
$F_5 = 0.3sin(2\pi x)$	30 random points $\subseteq$ [-1,1]	100 $\subseteq$[-0.5:0.02:1.5]
$F_6 = cos(3x)$	60 random points $\subseteq$ [-1,1]	200 $\subseteq$[0:0.01:2]

Table 2. Run and Evolutionary Parameter Values.

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

1. Standard Crossover (SC): The fitness is measured as the error rate on the whole training set. The best-of-run individual is the individual with the lowest error rate on the training set in entire evolutionary time. This individual was then tested on the testing data set to give the result for solution generalisation capacity of the run.
2. Standard Crossover with Validation (SCV): The training set is randomly divided into 2 (for each run): 67% is used for training (training set) and the remaining 33% is used for validating (validation set). At each generation the fitness of individuals is measured on the training set and this fitness is used for tournament selection. At the same time, a two-objective trial (fitness and size of an individual) is conducted in order to extract a set of non-dominated individuals (the Pareto front). The individuals in the Pareto front are then evaluated on the validation set, with the best of run individual selected as the one of these with the smallest error rate on the validation set. This configuration is similar to the validation configuration in [9].
3. Semantic Similarity based Crossover (SSC): This configuration is similar to Configuration 1 with only one difference is that SSC is used stead of SC.

Table 3. Number of solutions of four schemas.

Fs	Ms	Training				Validating				Testing			
		GS	MS	BS	US	GS	MS	BS	US	GS	MS	BS	US
F1	SC	22	76	2	0	-	-	-	-	16	72	11	1
	SSC	49	50	1	0	-	-	-	-	39	59	3	1
	SCV	28	71	1	0	29	65	6	0	14	71	14	1
	SSCV	52	48	0	0	56	44	0	0	32	58	9	1
F2	SC	4	81	15	0	-	-	-	-	5	69	26	0
	SSC	15	85	0	0	-	-	-	-	12	86	2	0
	SCV	7	82	11	0	6	76	18	0	3	66	31	0
	SSCV	18	79	3	0	18	79	3	0	13	74	13	0
F3	SC	62	38	0	0	-	-	-	-	37	62	1	0
	SSC	88	12	0	0	-	-	-	-	71	29	0	0
	SCV	65	35	0	0	68	32	0	0	24	74	20	0
	SSCV	90	10	0	0	90	10	0	0	54	46	0	0
F4	SC	22	77	1	0	-	-	-	-	4	88	7	1
	SSC	34	64	0	0	-	-	-	-	9	91	1	0
	SCV	21	78	1	0	24	74	2	0	2	94	4	0
	SSCV	29	70	1	0	38	60	2	0	2	95	3	0
F5	SC	0	99	1	0	-	-	-	-	0	4	94	2
	SSC	5	95	0	0	-	-	-	-	1	5	91	3
	SCV	1	99	0	0	0	94	6	0	0	2	92	6
	SSCV	4	96	0	0	5	91	4	0	0	2	93	5
F6	SC	49	45	6	0	-	-	-	-	40	8	41	11
	SSC	61	38	1	0	-	-	-	-	54	7	34	5
	SCV	47	46	7	0	48	45	7	0	38	7	45	10
	SSCV	59	38	3	0	58	37	5	0	51	4	40	5

4. Semantic Similarity based Crossover with Validation (SSCV): This configuration is similar to Configuration 2 but with SSC rather than SC.

5 Results and Discussion

To examine and compare the generalisation performance of these methods, we use a new performance metric to measure the quality of solution of a run. For each run, we select the best individual (based on its fitness on the training data sets or the validating sets) as the final solution of the run. This solution is then tested on the testing sets. We define $\varepsilon = 5.10^{-3}$ as a constant to determine the quality of a solution. For a solution with fitness ft on the training sets (or validating sets or testing sets respectively), we classify it into four categories

1. A good solution (GS) if $ft < \varepsilon$
2. A moderate solution (MS) if $\varepsilon \leq ft < 10\varepsilon$

Table 4. Mean and Standard Deviation of the average of best fitness on three data sets. Note that the values are scaled by 10^2.

Functions	Methods	Training		Validating		Testing	
		Mean	Std	Mean	Std	Mean	Std
F1	SC	1.54	1.23	-	-	6.13	34.1
	SSC	**0.75**	0.99	-	-	**2.17**	9.10
	SCV	1.30	1.09	1.57	1.79	3.86	10.2
	SSCV	**0.67**	0.76	0.80	1.01	**3.19**	9.61
F2	SC	3.07	1.81	-	-	3.88	2.42
	SSC	**1.38**	0.84	-	-	**1.82**	1.27
	SCV	2.92	1.83	3.16	2.27	4.14	2.58
	SSCV	**1.62**	1.32	1.64	1.42	**2.53**	2.45
F3	SC	0.61	0.70	-	-	1.00	1.10
	SSC	**0.25**	0.23	-	-	**0.49**	0.44
	SCV	0.55	0.54	0.54	0.79	1.06	1.01
	SSCV	**0.23**	0.21	0.28	0.60	**0.62**	0.55
F4	SC	1.29	0.99	-	-	1.84	1.50
	SSC	**0.85**	0.75	-	-	**1.37**	1.11
	SCV	1.19	0.94	1.29	1.12	1.86	1.52
	SSCV	**0.99**	1.07	1.17	1.76	**1.56**	1.29
F5	SC	2.40	0.87	-	-	16.2	12.7
	SSC	**1.98**	0.83	-	-	**14.1**	12.6
	SCV	2.35	0.78	3.01	1.19	18.4	19.6
	SSCV	**1.79**	0.84	2.57	1.37	16.1	13.6
F6	SC	1.37	1.77	-	-	20.0	28.5
	SSC	**0.66**	0.98	-	-	**16.2**	22.2
	SCV	1.46	1.83	1.48	2.09	23.3	40.0
	SSCV	**0.84**	1.37	1.19	2.32	**17.8**	25.6

3. A bad solution (BS) if $10\varepsilon \leq ft < 100\varepsilon$
4. An Unacceptable solution (US) if $100\varepsilon \leq ft$

The number of each category of solutions found on three data sets are shown in Table 3. It can be seen from this table that SSC is consistently better than SC on the training sets. These results are consistent with those in [25] where SSC was shown to be significantly better than both SC and SAC. The results also show that SSC found good solutions more often than SC on all problems. The number of moderate and bad solutions of SSC are also significantly less than ones of SC. It is noted that none of the methods scored unacceptable solutions on the training sets. This means that on the tested problems, it is rather easy for all GP systems to overfit the training data. The table also shows that by using validation sets, the solutions selected at the end of the runs have validation errors almost similar to training errors and the solution quality of SSCV is also consistently better than one of SCV.

The results on test sets show some deterioration in the quality of the solutions for all methods. The table, however, also shows that the performance of SSC on the test sets is still better than SC. SSC generate more good solutions and less bad and unacceptable solution on the test sets regardless of how the test sets are designed. It confirms that the generalisation power of GP is increased when equiped with SSC. In other words, by adding semantic control via SSC, the performance of GP is improved not only on training data but also on unseen data. On the testing sets, solution quality of SCV and SSCV are slightly worse than SC and SSC respectively. It seems that the generalisation ability of both SC and SSC are not enhanced when the validation sets are used. It is not entirely surprising as it was also shown that the use of a validation set only improves the stability of the best-of-run solutions on the test sets and the improvement was not significant [9].

Table 5. The average size of population and the good solutions on training and testing sets

Fs	ASP				ASGSTr				ASGSTs			
	SC	SSC	SCV	SSCV	SC	SSC	SCV	SSCV	SC	SSC	SCV	SSCV
F1	52.9	43.2	50.1	43.4	64.6	59.8	62.1	60.2	30.8	30.4	20.7	23.1
F2	58.0	55.6	57.6	55.4	83.1	43.1	59.2	77.8	38.5	29.6	38.0	58.0
F3	43.2	40.3	41.8	42.2	58.7	63.5	64.2	62.5	61.5	55.6	48.1	53.0
F4	51.5	48.0	51.7	48.9	65.3	52.3	54.7	49.2	73.8	80.7	53.2	57.0
F5	65.5	63.7	65.7	63.8	NA	50.4	87.5	51.2	NA	90	NA	NA
F6	55.6	42.0	55.8	42.0	64.6	51.5	63.3	41.8	25.9	23.3	31.2	23.3

The second performance metric used here is the mean and standard deviation of the best fitness on three data sets. These results are presented in Table 4 (after the values are scaled by 10^2). A Ranked Wilcoxon Test was also conducted to analyse if the use of SSC results in significantly better solution quality over SC. The confidence interval is 95% and the results are printed bold face if they are statistically significant. The results in this table are consistent with those in Table 3, i.e., SSC is significantly better than SC on all tested functions both on the training and test sets. Unlike some other techniques for improving GP generalisation [5], at least on similar testing problems, SSC does not only improve the solution performance on the traning sets (overfitting) but also on the test sets (generalisation). The table also shows that using validation sets does not help to increase the power of SC or SSC. The superiority of SSCV over SC is mostly related to its semantic control mechanism (with an exception on function F_5).

Since there is a strong correlation between the complexity of solutions and their ability to generalise (Ockham's razor or Minimum Description Length - MDL principle [22], statistics on solution size were also recorded and analysed. This includes the average size of a solution in the population (ASP), the average size of the good solutions on the training sets (ASGSTr) and the average size of the good solutions on the test sets (ASGSTs). These results are depicted in Table 5. In this table, when a method could not find any good solution, NA is printed instead. It can be seen that the average

size of a solution in the population for SSC is constantly smaller than SC. It means that SSC not only helps to improve GP solution quality but also to reduce code bloat. This is important as the primary motivation of SSC is to design a crossover operator based on semantic control but not to reduce size or code bloat (i.e., the control exerted on the semantic level seems to have a positive consequence for the syntactic aspects of the evolving programs). While the reason for reducing code bloat of SSC is not investigated in this paper, it seems that the better individuals in SSC tend to be smaller than ones of SC. The results of the average size of the good solutions on the training sets give more evidence for this conclusion. These results show that the ASGSTr of SSC is often smaller than one of SC (with one exception on function F_3). These results can be considered as one of the underlining reasons for the improvement in generalisation power that SSC brings to a GP system.

The results in Table 5, contrary to those in Tables 3 and 4, also show the remarkable effect of the use of validation sets. It shows that the good solutions found by using validation sets (either with SC or SSC), are often smaller in size than without validation. It is understandable as the methods with validation sets tend to select smaller solutions to measure error on the test sets. The results are consistent with Gagne [9], where it was also shown that the use of a validation set helps to reduce the size of the best individuals of runs.

6 Conclusions and Future Work

In this paper, we investigated the impact of semantic control on the ability of GP to generalise by using a recently proposed semantic based crossover, *Semantic Similarity based Crossover* (SSC). The traditional approach for improving generalisation in the field of GP in particular and Machine Learning in general by using validation sets was also examined. Four GP systems were tested on a number of real-valued symbolic regression problems. The empirical results shows a significant positive impact of semantic control in GP on its generalisation ability, and limited effects of using validation sets were observed (except in terms of the average size of good solutions). Further analysis on the average size of individuals in the population and the solutions shows that using semantics (via SSC) also helps to reduce GP code bloat. This leads to both significant GP performance improvement and its ability to find simpler solutions.

Although the experiments provide strong evidence for the important role of semantic control in reducing GP code bloat which leads to the improvement of GP generalisation, it offers no explanation. We are aiming to investigate such causal relationship in the future. It might be very important as it creates a bridge between semantic and syntactic aspects of the GP evolutionary process. Furthermore, and perhap equally important, we are planning to find the answer for the limited impact of validation sets in the GP learning process as found on the problems examined here. Last but not least, we are also intending to do a more comprehensive comparison between SSC generalisation ability with some other generalisation methods in the GP literature (e.g. Tarpeian Control, Relaxed GP, Linear Scaling and No same mate etc).

Acknowledgements

This paper was funded under a Postgraduate Scholarship from the Irish Research Council for Science Engineering and Technology (IRCSET).

References

1. W. Banzhaf, F. D. Francone, and P. Nordin. The effect of extensive use of the mutation operator on generalization in genetic programming using sparse data sets. In *Parallel Problem Solving from Nature IV, Proceedings of the International Conference on Evolutionary Computation*, volume 1141 of *LNCS*, pages 300–309, Berlin, Germany, 22-26 Sept. 1996. Springer Verlag.
2. L. Beadle and C. Johnson. Semantically driven crossover in genetic programming. In *Proceedings of the IEEE World Congress on Computational Intelligence*, pages 111–116. IEEE Press, 2008.
3. R. Cleary and M. O'Neill. An attribute grammar decoder for the 01 multi-constrained knapsack problem. In *Proceedings of the Evolutionary Computation in Combinatorial Optimization*, pages 34–45. Springer Verlag, April 2005.
4. L. E. D. Costa and J.-A. Landry. Relaxed genetic programming. In *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, volume 1, pages 937–938, Seattle, Washington, USA, July 2006. ACM Press.
5. D. Costelloe and C. Ryan. On improving generalisation in genetic programming. In *Proceedings of the 12th European Conference on Genetic Programming, EuroGP 2009*, volume 5481 of *LNCS*, pages 61–72, Tuebingen, April 2009. Springer.
6. M. de la Cruz Echeanda, A. O. de la Puente, and M. Alfonseca. Attribute grammar evolution. In *Proceedings of the IWINAC 2005*, pages 182–191. Springer Verlag Berlin Heidelberg, 2005.
7. N. Foreman and M. Evett. Preventing overfitting in GP with canary functions. In *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 2, pages 1779–1780, Washington DC, USA, June 2005. ACM Press.
8. F. D. Francone, P. Nordin, and W. Banzhaf. Benchmarking the generalization capabilities of a compiling genetic programming system using sparse data sets. In *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 72–80, Stanford University, CA, USA, 28–31 July 1996. MIT Press.
9. C. Gagne, M. Schoenauer, M. Parizeau, and M. Tomassini. Genetic programming, validation sets, and parsimony pressure. In *Proceedings of the 9th European Conference on Genetic Programming*, volume 3905 of *Lecture Notes in Computer Science*, pages 109–120, Budapest, Hungary, April 2006. Springer.
10. S. Gustafson, E. K. Burke, and N. Krasnogor. On improving genetic programming for symbolic regression. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 912–919, Edinburgh, UK, 2005. IEEE Press.
11. C. Johnson. Deriving genetic programming fitness properties by static analysis. In *Proceedings of the 4th European Conference on Genetic Programming (EuroGP2002)*, pages 299–308. Springer, 2002.
12. C. Johnson. What can automatic programming learn from theoretical computer science. In *Proceedings of the UK Workshop on Computational Intelligence*. University of Birmingham, 2002.
13. C. Johnson. Genetic programming with fitness based on model checking. In *Proceedings of the 10th European Conference on Genetic Programming (EuroGP2002)*, pages 114–124. Springer, 2007.

14. G. Katz and D. Peled. Genetic programming and model checking: Synthesizing new mutual exclusion algorithms. *Automated Technology for Verification and Analysis, Lecture Notes in Computer Science*, 5311:33–47, 2008.
15. G. Katz and D. Peled. Model checking-based genetic programming with an application to mutual exclusion. *Tools and Algorithms for the Construction and Analysis of Systems*, 4963:141–156, 2008.
16. M. Keijzer. Improving symbolic regression with interval arithmetic and linear scaling. In *Proceedings of EuroGP'2003*, pages 70–82. Springer-Verlag, April 2003.
17. J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, Massachusetts, 1992.
18. K. Krawiec and P. Lichocki. Approximating geometric crossover in semantic space. In F. Rothlauf, editor, *Genetic and Evolutionary Computation Conference, GECCO 2009, Proceedings, Montreal, Québec, Canada, July 8-12, 2009*, pages 987–994. ACM, 2009.
19. I. Kushchu. An evaluation of evolutionary generalisation in genetic programming. *Artificial Intelligence Review*, 18(1):3–14, September 2002.
20. S. Mahler, D. Robilliard, and C. Fonlupt. Tarpeian bloat control and generalization accuracy. In *Proceedings of the 8th European Conference on Genetic Programming*, volume 3447 of *Lecture Notes in Computer Science*, Lausanne, Switzerland, April 2005. Springer.
21. N. McPhee, B. Ohs, and T. Hutchison. Semantic building blocks in genetic programming. In *Proceedings of 11th European Conference on Genetic Programming*, pages 134–145. Springer, 2008.
22. T. Mitchell. *Machine Learning*. McGraw Hill, New York, 1996.
23. R. Poli, W. B. Langdon, and N. F. McPhee. *A Field Guide to Genetic Programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008. (With contributions by J. R. Koza).
24. N. Q. Uy, N. X. Hoai, and M. O'Neill. Semantic aware crossover for genetic programming: the case for real-valued function regression. In *Proceedings of EuroGP09*, pages 292–302. Springer, April 2009.
25. N. Q. Uy, M. O'Neill, N. X. Hoai, B. McKay, and E. G. Lopez. Semantic similarity based crossover in GP: The case for real-valued function regression. In P. Collet, editor, *Evolution Artificielle, 9th International Conference*, Lecture Notes in Computer Science, pages 13–24, October 2009.
26. L. Vanneschi and S. Gustafson. Using crossover based similarity measure to improve genetic programming generalization ability. In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1139–1146, Montreal, 8-12 July 2009. ACM.
27. M. L. Wong and K. S. Leung. An induction system that learns programs in different programming languages using genetic programming and logic grammars. In *Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence*, 1995.