Semantic-based Subtree Crossover Applied to Dynamic Problems

Nguyen Quang Uy
NCRA Group
UCD, Ireland
quanguyhn@gmail.com

Eoin Murphy
NCRA Group
UCD, Ireland
eoin.murphy@ucd.ie

Michael O'Neill
NCRA Group
UCD, Ireland
m.oneill@ucd.ie

Nguyen Xuan Hoai
Faculty of IT
Hanoi University
nxhoai@gmail.com

***Abstract*—Although many real world problems are dynamic in nature, the study of Genetic Programming in dynamic environments is still immature. This paper investigates the application of some recently proposed semantic-based crossover operators on a series of dynamic problems. The operators studied include Semantic Similarity based Crossover and the Most Semantic Similarity based Crossover. The experimental results show the advantage of using semantic-based crossovers when tackling dynamic problems.**

***Keywords*-Genetic Programming; Semantics; Crossover; Dynamic Problems;**

I. Introduction

In the field of Genetic Programming (GP) [24], [20], researchers have recently paid more attention to semantic information, with a dramatic increase in the number of publications (e.g., [14], [16], [17], [19], [18], [1], [23], [25], [27], [8]). Previously, most research has purely focused on syntactic aspects of GP representation. From a programmer's perspective, however, maintaining syntactic correctness is only one part of program construction: maintaining program semantic correctness is much more desirable. 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 approaches.

In a recent work, Uy et al. proposed a Semantic Similarity based Crossover (SSC) with the aim to improve semantic locality [27]. The basic idea of SSC is keeping a semantically small change in crossover by exchanging two semantically similar subtrees. The experiments in [27] show that SSC helps to significantly improve GP performance in comparison with standard crossover in solving a class of real valued symbolic regression problems. SSC has been extended to the Most Semantic Similarity based Crossover (MSSC) [26] to predict time series and the experimental results in [26] show that MSSC helps to improve the performance of GP in solving this problem. However, the problems in the previous studies [27], [26] are static problems. Therefore, it raises a question whether these semantic-based crossovers still maintain their superior performance when they are applied to dynamic problems? This paper aims to answer the above question.

The remainder of the paper is organised as follows. In the next section we review the literature on GP with semantics and a description of dynamic environments. The Semantic Similarity based Crossover (SSC) and the Most Semantic Similarity based Crossover (MSSC) are presented in Section III. It is followed by a section detailing our experimental settings. The experimental results are shown and discussed in Section V. The last section concludes the paper and highlights some potential future work.

II. Related Work

This section briefly reviews the relevant literature for this study. Firstly, a summary of ways in which semantics have been employed in GP is provided, and this is followed by an overview on the study of dynamic environments.

A. Semantics in Genetic Programming

Incorporating semantics into GP has recently been considered by a number of GP researchers. This results in a dramatic increase in the number of related publications. Generally, the work falls into three main strands:

1) using formal methods [14], [16], [17], [19], [18]
2) using grammars [28], [8], [9]
3) using structures such as GP trees [1], [23], [25], [27]

The first approach in a series of work [14], [16], [17] was advocated by Johnson. In this work, Abstract Interpretation and Model Checking are used to calculate/extract semantics. Then, semantic information is used to measure the fitness of individuals where it is unable to use traditional sample point fitness. Consequently, Katz and Peled used model checking to solve the Mutual Exclusion problem [19], [18]. Again, individuals' fitness is quantified through model checking. These formal methods have a strict mathematical foundation, that potentially may aid GP. Perhaps because of high complexity, however, these methods have seen only limited research despite the advocacy of Johnson [15]. Their main application to date has been in evolving control strategies.

The second methodology uses Attribute Grammars as the main formalism. By adding attributes to a grammar, some useful semantic information about individuals can be generated. This information can then be used to delete bad individuals [9], or to prevent generating semantically invalid ones [28], [8]. The attributes used to represent semantics are, however, problem dependent, and it is not always easy to design such attributes for a new problem.

In the last category, controlling the GP operators is the major theme. In [1], the authors investigated the effect of semantic diversity on Boolean domains, checking the semantic equivalence between offspring and parents by

transforming them to a canonical form, Reduced Ordered Binary Decision Diagrams (ROBDDs) [12]. This information is used to determine which offspring are copied to the next generation. The method improved GP performance, presumably because it increased semantic diversity. The method has also been applied to mutation [3] and to the initialisation phase of GP [2].

While, most of previous research on semantics in GP were focused on combinatorial and boolean problems [8], [1], [23], [19], research on real-valued domains [25], [27], [21] is only recently considered. Krawiec and Lichocki [21] based the semantics of individuals on fitness cases, using it to guide the crossover operator (*Approximating Geometric Crossover* - AGC). AGC turned out to be not significantly better than standard crossover (SC) on their tested real-valued problems, and only slightly better on Boolean domains.

Uy et al. [25] proposed Semantics Aware Crossover (SAC) with an aim to promote semantic diversity. SAC is based on checking semantic equivalence of subtrees. It showed limited improvement on some real-valued problems; SAC was subsequently extended to Semantic Similarity based Crossover (SSC) [27], which performed better than both SC and SAC on tested real valued regression problems [27]. The idea of SSC was then extended to the Most Semantic Similarity based Crossover (MSSC) in solving time series prediction problem. However, in [27], [26], SSC and MSSC were only tested on some static problems. The objective of this paper is to investigate how SSC and MSSC perform when they are applied to dynamic problems.

B. Dynamic Environment

Before examining the performance of an algorithm in dynamic problems, it is important to define exactly what is a dynamic problem. In a broad context, dynamic problems can mostly be defined as problems in which some elements under their domain vary with the progression of time [10]. There are many aspects in which a problem can be considered as dynamic [6], [11]. In the context of Genetic Programming, the inputs, the constraints, or even the objective of the problem itself could be changed over time. In this paper, we restrict the definition of dynamism where the objective function of the problem changed with respect to time. In other words, the objective of evolution becomes dependent on time:

$$f(x) = f(x,t)$$

where f is the fitness function, x is an individual and t is the current generation.

So far, the studies of applying evolutionary computation to dynamic environments have focused on two main approaches: Memory recall [5], [4] and promoting diversity [13], [7]. The idea of the first method is to equip algorithms with a mechanism to recall previous solutions, and the motivation for the second method is to prevent the premature convergence of the population of a searcher resulting in its failure to discover new areas of the search space. To the best of our knowledge, semantics has not been employed as a technique in solving dynamic problems. This paper aims to promote semantic locality of crossover and applies new semantic-based crossovers to dynamic problems.

III. Methods

This section presents the methods used in the study. A way to measure semantics in GP is presented, followed by a detailed description of the way in which attributes can be used to store semantics. We then present Semantic Similarity based Crossover and its more recent improvement, the Most Semantic Similarity based Crossover.

A. Measuring Semantics

Although, an exact definition of semantics is non-trivial, in the field of GP, semantics of an individual program is often understood as the behavior of that program with respect to a set of input values. In this paper, we follow the previous work in [27] in using sampling semantics to measure semantics of any subtree. Formally, the *Sampling Semantics* (SS) of a (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 sequence of points sampled from domain D, $P = (p_1, p_2, ..., p_N)$. Then, the *Sampling Semantics* of T on P in domain D is the corresponding sequence $S = (s_1, s_2, ..., s_N)$ where $s_i = F(p_i), i = 1, 2, ..., N$.

The optimal choice of N and P depends on the problem; we follow the approach of [27] in setting the number of points for evaluating the semantics equal to the number of fitness cases (20 points – Section IV) and in choosing the sequence of points P coincide to the fitness cases of the problem.

Based on SS, we define a *Sampling Semantic Distance* (SSD) between two subtrees. It differs from that in [27] in using the mean absolute difference in SS values, rather than the sum of absolute difference. Let $U = (u_1, u_2, ..., u_N)$ and $V = (v_1, v_2, ..., v_N)$ represent the SSs of two subtrees, S_1 and S_2; then the SSD between S_1 and S_2 is defined in equation 1:

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

We follow [27] in defining a semantic relationship, *Semantic Similarity* (SSi), on the basis that the exchanging of subtrees in crossover is most likely to be beneficial if they are not semantically identical, but also not too different. Two subtrees are semantically similar if their SSD lies within a positive interval. The formal definition of SSi between subtrees S_1 and S_2 is given in the following equation:

$$SSi(St_1, St_2) = \begin{array}{l} \textbf{if } \alpha < \mathrm{SSD}(St_1, St_2) < \beta \\ \qquad \textbf{then true} \\ \qquad \textbf{else false} \end{array}$$

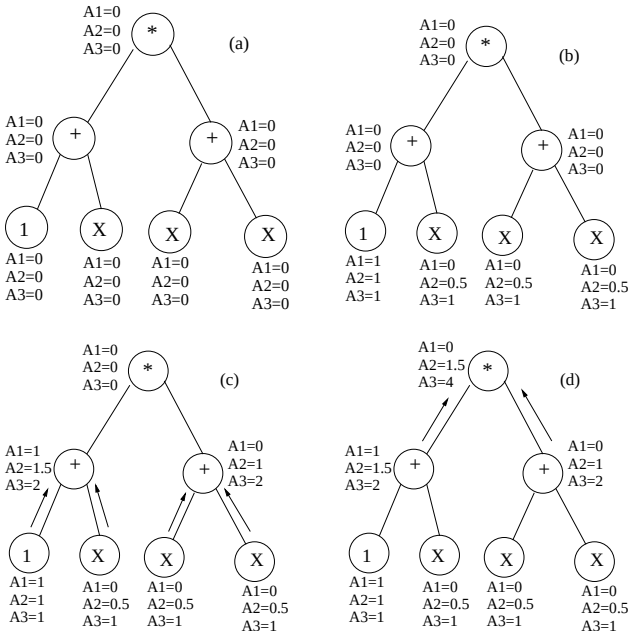

Figure 1. An individual in AGP and the Process of Evaluating its Attributes.

where α and β are two predefined constants, the *lower* and *upper* bounds for semantics sensitivity. In general, the best values for these semantic sensitivity bounds have been found to be problem dependent. In this work we set $\alpha = 10^{-3}$ and $\beta = 0.4$, which have been found to provide good performance in the case of SSC.

B. Attribute Based Representation

Since SS is expensive to compute, we reduce the cost by caching. The SS of each subtree is stored in the root node using attributes; the resulting GP system is known as *Attributes Genetic Programming* (AGP). In more detail, assume that the problem has N fitness cases; then N attributes are added to each node in the individual's tree. In figure 1 N is set to 3, so three attributes A_1, A_2, A_3 are added to every node, to cache the SS of the corresponding subtree.

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

C. Semantic Similarity-based Crossover

Inspired from the difficulty in designing an operator with the property of high locality in GP, Semantic Similarity based Crossover (SSC) was first proposed in [27] with the main objective being to improve the locality of 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 different values to see how these values affect SSC's performance in solving dynamic problems.

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;
choose a random crossover point Subtree1 in P1;
choose a random crossover point Subtree2 in P2;
execute crossover;
return true;
```

D. The Most Semantic Similarity-based Crossover

The Most Semantic Similarity based Crossover (MSSC) was proposed in [26] as an improvement of SSC in predicting time series. MSSC further exploits the main idea of SSC. The purpose of MSSC is to avoid the manual determination of the semantic sensitivities in SSC, but still keep a small semantic change in child individual(s) after crossover. MSSC works as follows. Firstly, N subtree pairs are randomly selected from the two parents. The Sampling Semantic distance (SSD) of two subtrees in each pair is calculated. The pair that have the smallest (most similar but not equivalent) semantic distance of two subtrees in N pairs is chosen for crossover. In MSSC, the concept of semantic equivalence is the same as in SAC. Algorithm

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 α < SD < Max then
        Max=SD;
        CrossPoint1=Subtree1;
        CrossPoint2=Subtree2;
Execute crossover by exchange the subtrees at
CrossPoint1 and CrossPoint2;
```

2 shows how MSSC works in detail. In the experiments of MSSC, the value of Max_Trial (TM) is again set as in SSC, and Extremal_Value is set to 10^6.

IV. Experimental Settings

This section outlines the settings used for our experiments. The dynamic problems used in this study are presented, followed by the GP parameters settings.

A. Dynamic Problems

In order to achieve dynamic problems, the static problems are extended so that their fitness functions change with respect to time. The population is allowed to evolve normally for a certain number of generations before the objective function changes. This number of generations is known as T. In our experiments in this paper, T is fixed at 10, meaning that the evolution continues for 10 generations before the fitness function changes. At this point, the entire population is evaluated on the new fitness functions and the evolution then continues for 10 more generations before the objective function changes once more. For the experiments in this paper, the four following dynamic problems are used.

F1 = 1. $x+x^2$
2. $x+x^2+x^3$
3. $x+x^2+x^3+x^4$
4. $x+x^2+x^3+x^4+x^5$
5. $x+x^2+x^3+x^4+x^5+x^6$

F2 = 1. $x+x^2+x^3+x^4+x^5+x^6$
2. $x+x^2+x^3+x^4+x^5$
3. $x+x^2+x^3+x^4$
4. $x+x^2+x^3$
5. $x+x^2$

F3 = 1. $\cos(x)+\sin(x)$
2. $\cos(x)+\sin(x)+\sin^2(x)$
3. $\cos(x)+\sin(x)+\sin^2(x)+\sin^3(x)$
4. $\cos(x)+\sin(x)+\sin^2(x)+\sin^3(x)+\sin^4(x)$
5. $\cos(x)+\sin(x)+\sin^2(x)+\sin^3(x)+\sin^4(x)+\sin^5(x)$

F4 = 1. $\cos(x)+\sin(x)$
2. $\cos(x)+\sin(x)+\sin(2x)$
3. $\cos(x)+\sin(x)+\sin(2x)+\sin(3x)$
4. $\cos(x)+\sin(x)+\sin(2x)+\sin(3x)+\sin(4x)$
5. $\cos(x)+\sin(x)+\sin(2x)+\sin(3x)+\sin(4x)+\sin(5x)$

These problems can be divided into two groups. The first group includes two polynomial problems and the second group comprises two trigonometrical problems. The first polynomial problem changes from simple form to complex form while the second one changes conversely. Both trigonometrical problems change from simple form to more complex form. The objective of the research is to study the behaviour of semantic-based crossovers with different types of changing environment.

Table I
Run and Evolutionary Parameter Values.

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	15
Non-terminals	+, -, *, / (protected one), sin, cos, exp, log (protected one)
Terminals	X, 1
Training set	20 random points in [-1,1]
Raw fitness	mean of absolute error on all fitness cases
Trials per treatment	100 independent runs for each value

B. GP Parameters Settings

The GP parameters used for our experiments are shown in Table I. Since all the above dynamic problems change the fitness function five times and each fitness function is evolved over 10 generations, the maximum number of generations is 50. Despite this being an experiment purely concerned with the 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.

For SSC, the lower bound of semantic sensitivity is set at 10^{-3} and the upper bound is set at 0.4. Six values of Max_Trial (MT), 4, 8, 12, 16, 20, 24 are tested. These configurations of SSC will be referred to as SSCX with X=4, 8, 12, 16, 20, 24. The same values of MT are used for MSSC and the similar conventional naming is used.

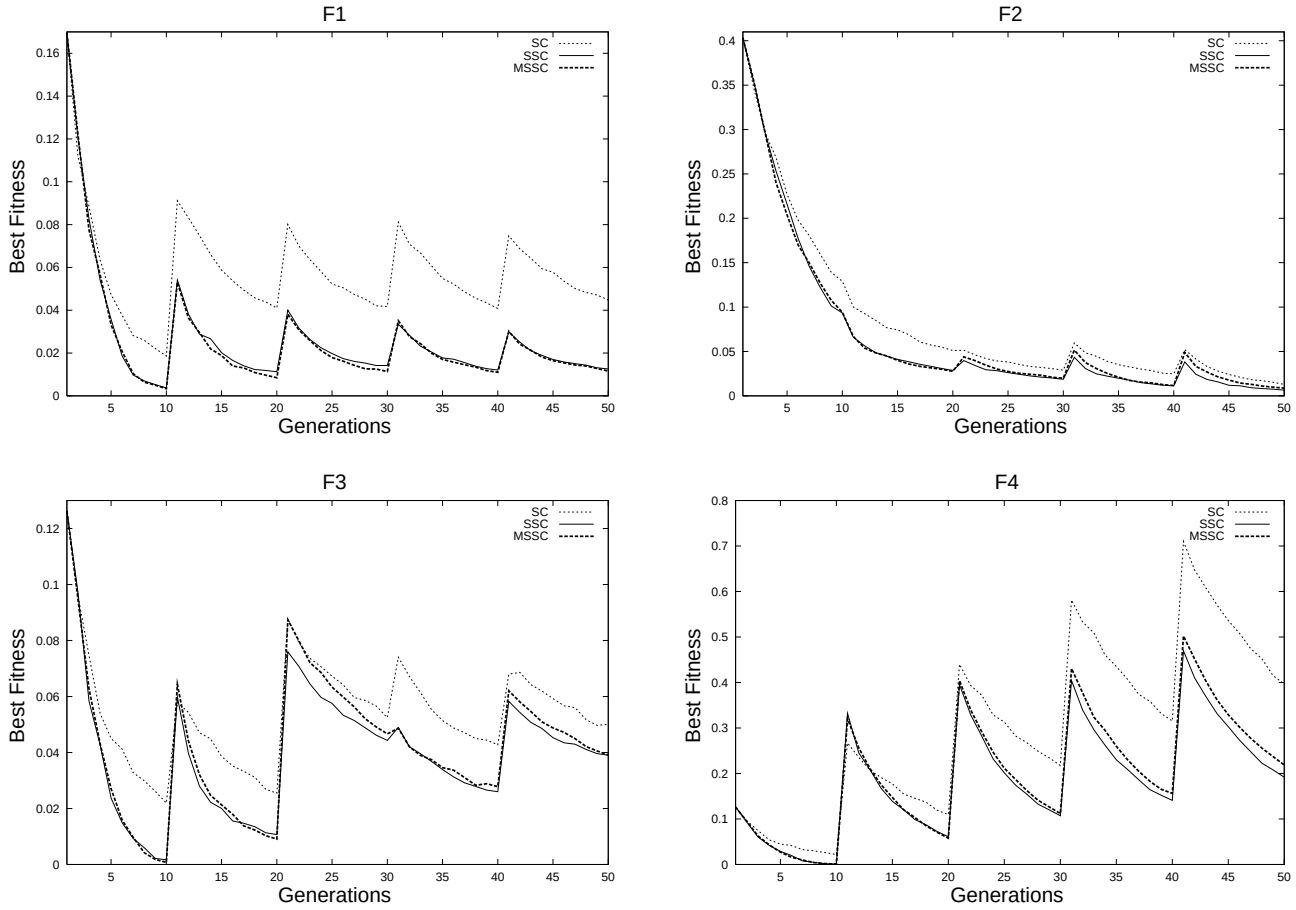

Figure 2. The plot of mean best fitness, MT=12 for both SSC and MSSC. The lower lines are better.

V. Results and Discussion

The results of the experiments are presented in this section. The best fitness across the runs is averaged over 100 runs and is plotted in Figure 2 (MT=12 for SSC and MSSC in this figure). Three properties of these graphs are analysed (see Table II).

Area under the curve (AUC) is the area covered by each line. The value of semantic-based crossover (SSC, and MSSC) is then divided by the value of standard crossover (SC). Since the objective is to minimise fitness then if the ratio is smaller than 1, it indicates that, generally, SSC and MSSC perform better than SC and smaller values are better. If these values are larger than 1 then SC is better than semantic-based crossovers.

Fall off (FO) is the immediate difference between fitness before and after a change in the environment occurs. This quantity presents the ability to react with the change in the environment of each crossover. The values are averaged across the runs and the ratio between semantic-based crossovers and standard crossover are calculated. If these ratios are smaller than 1, it means that semantic-based crossover react better with the change in the environment and vice versus.

Drawdown (DD) is the difference between fitness from the start of a period to the end of it. This value reflect the speed to improve fitness of each method. Again, the values are averaged across the runs and the ratio between standard crossover and semantic-based crossovers are calculated. If these ratios are smaller than 1, meaning that semantic-based crossovers improve fitness faster than standard crossover. Whereas, if these value are greater than 1, then standard crossover improves fitness faster than semantic-based crossovers in each period.

Interpreting the plots seen in Figure 2, we can see that semantic-based crossovers, both SSC and MSSC perform better than standard crossover in solving these dynamic problems. It can be seen from this figure that the mean best fitness of SSC and MSSC are often smaller than of SC. The figure also shows that when the problems change from simple form to complex form (F1, F3 and F4) the advantage of semantic-based crossovers over standard crossover seem greater than when the problem adapts from complex form to simple form (F2). Comparing between two semantic-based crossovers, SSC and MSSC, the figure shows that they perform mostly equally.

The values of AUC in Table II are consistent with Figure 2. It can be seen from this table that the values of AUC are always smaller than 1 and in many cases they are mostly equal to 0.5. This indicates that, in general, the populations of SSC and MSSC are fitter than SC across the dynamic runs. The table also shows that the superior performance of SSC and MSSC over SSC are greater on

Table II
THE ANALYSIS OF SOME PROPERTIES OF THE MEAN BEST FITNESS OF SEMANTIC-BASED CROSSOVERS IN COMPARISON WITH STANDARD CROSSOVER. THE VALUES FOR AUC, FO, DD ARE SMALLER THAN 1 INDICATE THAT SEMANTIC-BASED CROSSOVERS ARE BETTER THAN STANDARD CROSSOVER. CONVERSELY, IF THEY ARE GREATER THAN 1, STANDARD CROSSOVER IS BETTER.

Methods	F1			F2			F3			F4		
	AUC	FO	DD	AUC	FO	DD	AUC	FO	DD	AUC	FO	DD
SSC4	0.64	0.81	1.04	0.84	1.74	0.93	0.74	1.04	0.92	0.69	0.93	0.92
SSC8	0.57	0.78	1.05	0.82	1.24	0.96	0.72	1.09	0.88	0.63	0.95	0.88
SSC12	0.52	0.71	1.07	0.81	1.43	0.95	0.69	1.09	0.88	0.62	0.96	0.86
SSC16	0.45	0.63	1.13	0.78	1.20	0.97	0.72	1.10	0.89	0.63	0.96	0.88
SSC20	0.46	0.62	1.12	0.78	1.34	0.96	0.75	1.20	0.85	0.59	0.97	0.84
SSC24	0.44	0.62	1.13	0.78	1.45	0.95	0.73	1.14	0.88	0.64	1.00	0.82
MSSC4	0.72	0.89	0.99	0.87	1.11	0.98	0.71	1.01	0.91	0.78	0.94	0.97
MSSC8	0.52	0.75	1.05	0.80	1.69	0.93	0.75	1.11	0.90	0.66	0.97	0.87
MSSC12	0.44	0.66	1.10	0.80	1.97	0.92	0.77	1.22	0.83	0.67	0.99	0.85
MSSC16	0.44	0.63	1.13	0.81	2.36	0.90	0.82	1.38	0.79	0.71	1.00	0.81
MSSC20	0.43	0.61	1.14	0.82	2.58	0.89	0.80	1.43	0.73	0.82	1.11	0.79
MSSC24	0.42	0.51	1.22	0.82	2.67	0.88	0.84	1.42	0.76	0.85	1.13	0.79

F1, F3, F4 than on F2.

While the AUC values provide evidence that semantic-based crossovers performance is better than standard crossover across the evolutionary process, the values for FO are mixed. On two problems, F1 and F4, FO are often smaller than 1 and on two left problems, FO are greater than 1. Therefore, it is difficult to conclude about the ability to react with change in the environment of semantic-based crossovers compared to standard crossover.

Finally, the table shows that the speed to improve fitness in each period of semantic-based crossovers are often faster than for standard crossover. It can be observed from the table that the values for DD are often smaller than 1 with only exception on function F1. This result is understandable since the previous researches have shown that semantic-based crossovers, SSC and MSSC help to improve the performance of GP versus standard crossover. This can be seen as a reason that helps semantic-based crossovers perform better than standard crossover in dynamic problems.

In comparing between SSC and MSSC, the table again shows that they are most equal. For six values of Max_Trial (MT) tested, it can be seen that the values from 8 to 16 seem slightly better than 4, 20 or 24. When MT is too small (4), semantic-based crossovers can under perform while if they are too large, SSC and MSSC can be trapped into local optimals and poorly respond when the environment changed. This is evidenced by the fact that the values of FO are often greater with MT=20 and 24.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we apply two recently proposed semantic-based subtree crossover operators, Semantic Similarity based Crossover (SSC) and the Most Semantic Similarity based Crossover (MSSC), to dynamic problems. We compare the performance of GP with these two crossovers and GP with standard crossover (SC) in solving a class of dynamic symbolic regression problems. The mean best fitness of these crossovers are plotted and some important properties of these graphs are analysed. The results show that semantic-based crossovers help to improve the performance of GP compared to standard crossover in solving dynamic problems. Further analyses show that SSC and MSSC improve fitness faster than SC, although they do not respond to the change in the environment better than SC.

There are a number of avenues for future research that can be developed from this paper. Firstly, this paper studies dynamic problems with an environmental change which occurs at regular intervals. In the near future we aim to study these problems with different frequencies of change. Secondly, we want to combine semantic-based crossovers with some diversity promoting methods, such as fitness sharing [22] to see if they help to gain further improvements in dynamic environments. Last, but not least, we would like to create a class of dynamic benchmark problems suitable for GP.

ACKNOWLEDGMENT

Q.U. Nguyen is funded under a Postgraduate Scholarship from the Irish Research Council for Science Engineering and Technology (IRCSET). E. Murphy and M. O'Neill are supported by the Science Foundation Ireland under Grant No. 08/IN.1/I1868.

REFERENCES

[1] 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.

[2] L. Beadle and C. G. Johnson. Semantic analysis of program initialisation in genetic programming. *Genetic Programming and Evolvable Machines*, 10(3):307–337, Sep 2009.

[3] L. Beadle and C. G. Johnson. Semantically driven mutation in genetic programming. In A. Tyrrell, editor, *2009 IEEE Congress on Evolutionary Computation*, pages 1336–1342, Trondheim, Norway, 18-21 May 2009. IEEE Computational Intelligence Society, IEEE Press.

[4] C. N. Bendtsen and T. Krink. Dynamic memory model for non-stationary optimization. In D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton, editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 145–150. IEEE Press, 2002.

[5] J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 3, pages 1875–1882, 1999.

[6] J. Branke. *Evolutionary optimization in dynamic environments*. PhD thesis, University Karlsruhe, 2000.

[7] W. Cedeno and V. R. Vemuri. On the use of niching for dynamic landscapes. In W. Porto, editor, *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, pages 361–366, Piscataway, New Jersey, Apr. 1997. IEEE Press.

[8] 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.

[9] 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.

[10] I. Dempsey. *Grammatical Evolution in Dynamic Environments*. PhD thesis, University College Dublin, Ireland, 2007.

[11] I. Dempsey, M. O'Neill, and A. Brabazon. *Foundations in Grammatical Evolution for Dynamic Environments*. Springer, 2009.

[12] R. E.Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35:677–691, 1986.

[13] J. J. Grefenstette. Genetic algorithms for changing environments. In R. Maenner and B. Manderick, editors, *Proc. Parallel Problem Solving from Nature-2*, pages 137–144. North-Holland, 1992.

[14] 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.

[15] C. Johnson. Genetic programming with guaranteed constraints. In *Recent Advances in Soft Computing*, pages 134–140. The Nottingham Trent University, 2002.

[16] C. Johnson. What can automatic programming learn from theoretical computer science. In *Proceedings of the UK Workshop on Computational Intelligence*. University of Birmingham, 2002.

[17] 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.

[18] 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.

[19] 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.

[20] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, Massachusetts, 1992.

[21] 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.

[22] R. I. B. McKay. An investigation of fitness sharing in genetic programming. *The Australian Journal of Intelligent Information Processing Systems*, 7(1/2):43–51, July 2001.

[23] 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.

[24] 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).

[25] 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.

[26] N. Q. Uy, M. O'Neill, and N. X. Hoai. Predicting the tide with genetic programming and semantic-based crossovers. In *KSE 2010 The Second International Conference on Knowledge and Systems Engineering*, Hanoi, Vietname, 7-9 Oct. 2010. IEEE Computer Society, IEEE Press.

[27] 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.

[28] 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.