THE RELATIONSHIP BETWEEN SEMANTIC DISTANCE AND PERFORMANCE IN DYNAMIC SYMBOLIC REGRESSION PROBLEMS

Clíodhna Tuite, Michael O'Neill, Anthony Brabazon

Complex and Adaptive Systems Laboratory
University College Dublin, Ireland
cliodhna.tuite@gmail.com, m.oneill@ucd.ie, anthony.brabazon@ucd.ie

Abstract: *Several methods which apply genetic programming (GP) in dynamic optimization environments implicitly assume that the smaller the semantic change in the optimization goal, the better the adaptation of the GP population to the new target. However, GP searches over genetic operator-based fitness landscapes. As such, relative distances between solution points in genetic operator-based landscapes may not be related to the semantic distances between points. Our experiments examine whether decreasing the semantic distance between first- and second-period target functions in symbolic regression problems result in improved performance in the second period. As a control, we also investigate how re-initializing the GP population in the second period performs in comparison with using a continuous GP population across the two periods. We find that decreasing the semantic distance does result in better performance in the second period, and that re-initializing the GP population underperforms a continuous population at low semantic distances.*

Keywords: *Genetic programming, dynamic environments, symbolic regression, semantic distance*

1 Introduction

One of the most active research areas in the field of evolutionary computation, as measured by publication output, is evolutionary dynamic optimization [10]. These dynamic optimization problems require an evolutionary algorithm to track a moving optimum over time, which presents an added challenge [10]. We can tackle optimization problems with a technique such as Genetic Programming (GP) which evolves a population of candidate solutions. One strategy to deal with dynamic optimization problems is to continuously maintain the evolving population of solutions [6, 15], instead of running GP 'from scratch' each time the optimization goal changes (by re-initializing our population of candidate solutions). An assumption of techniques like this is that a continuous GP population will adapt better to a relatively small semantic change in the optimization goal, than to a relatively large semantic change. However, GP searches genetic-operator-based landscapes of solutions: as a result, solution points which are relatively semantically close may be relatively distant when viewed in operator-based search landscapes – a complex issue which complicates an important assumption. Thus, motivated by the inclusion of performance in dynamic environments as one of the key open issues for GP [12], we investigate this issue in an illustrative and tractable problem domain – the popular application area of *symbolic regression*.

1.1 Related Work

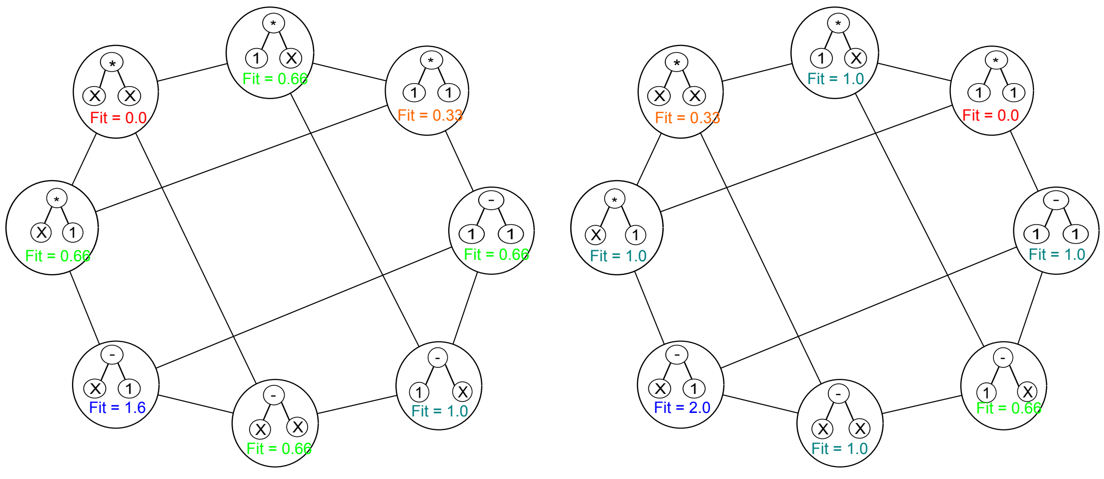
About one third of GP papers surveyed in [9] which perform benchmarking use symbolic regression, making it the most widely used class of GP benchmark. Nguyen et al. proposed two new semantically based crossover operators, and tested them on ten symbolic regression problems in [11]. Many authors have come up with techniques to help evolutionary algorithms deal with change, including incorporating a memory of past solutions in the search population [2, 6, 16], employing multiple populations in search [4, 16], or the use of mutation-based techniques such as hypermutation or random immigrants [5]. Recently, Loginov and Heywood [8] compared continuous GP evolution to population re-initialization for foreign exchange trading using decision trees composed of technical indicators. They found that automatic change detection followed by population re-initialization outperformed a continuous population.

1.2 Background: Fitness Landscapes and Search

We now describe the background to the issue of the difference between distance in operator-based fitness landscapes, and semantic distance, in more detail.

Genetic programming searches over genetic operator-based landscapes of potential solutions. In the view of fitness landscapes popularized by Jones [7], each genetic operator has associated with it a unique fitness landscape. Search in GP is carried out by genetic operators, guided by the fitness of individuals in the population. In a GP set-up employing crossover and mutation, search therefore proceeds on the crossover-based fitness landscape when crossover takes place, and on the mutation-based landscape when mutation takes place. Landscapes are represented as graphs, and each vertex in the graph represents a point, or a multiset of points (in the case of crossover) in the search space. The neighbourhood of a point in such a landscape is comprised of the other points in the landscape that can be reached from that point with a single use of the operator [7]. Neighbours are connected by edges in the landscape graph. The edges between vertices are labeled with operator transition probabilities, which determine the probability that one vertex can be transformed into the other.

When the optimal solution changes (as in dynamic problems), relative operator-based distances to the new solution may differ to relative semantic distances. To illustrate this, Fig. 1 depicts two fitness landscapes for point mutation. We are assuming a simple set-up for GP with point mutation as the only genetic operator. Fig. 1(a) shows the fitness of points in the landscape in the first time period, when the target function is x^2. Fig. 1(b) shows the same graph with the fitnesses having changed in the second time period, after the target function has changed to 1. In this tree-based GP set-up, the terminal set consists of the variable x and the integer value of 1, and the function set consists of the arithmetic multiplication and subtraction operators. All individuals are of depth one, and only full trees are allowed. Mutation operates by choosing one node in the tree at random, and mutating it to the only other value allowed at that node – the other member of the terminal set if we are at a leaf node, or else the other member of the function set. Transition probabilities are equal for all edges – since there is a 0.33 probability of point mutation transforming any of the individuals represented in the graph to one of it's neighbours.



(a) (b)

Figure 1: Point-mutation based fitness landscape at two points in time: (a) Period 1 (b) Period 2

Fitness for each potential solution is measured as the mean absolute error between the target function and the potential solution. We sample the input variable x three times over the interval $[-1, 1]$, at -1, 0 and 1. When the target function changes from x^2 to 1, the location of the target function in the fitness landscape moves a distance of two (that is, two edges are traversed when moving in the landscape between the vertices containing x^2 and 1). Semantic distance is also measured using mean absolute error, which means that 1 is the closest potential solution in the graph to x^2, as measured by semantic distance. If at the end of the first period, the entire GP population has converged on the target function x^2, then in the second period, it will take longer than the semantic distance between the nodes would suggest, for GP, using point mutation, to find the new target function (two edges need to be traversed).

1.3 Our Principal Investigation

Of course, when searching with GP, we have a population of solutions, and they won't necessarily all have converged on the target at the end of the first period. Our main question is: does this and other aspects of the GP search process allow the population to adapt well to small semantic changes in a dynamic environment, even though there is a difference between distance in operator-based landscapes, and semantic distance? Before describing the experimental methodology in our current research, we briefly describe results from our previous work [14].

1.4 Experimental Results from Prior Research

In order to investigate our question, we previously measured standard tree-based GP performance on a set of two-period symbolic regression problems [14]. We used one of thirteen target functions in the first period of evolution, and all first-period target functions were built on and expanded from the symbolic regression problems for single-instance problems described in [9]. We found that, for some of the thirteen sets of runs examined, continuous evolution didn't outperform re-initializing the population from scratch. We also found that GP search for second-period functions at a low semantic distance from the first-period target function did not always outperform search for functions which were semantically more distant from the first-period target function. Performance was measured using offline performance in the second period of evolution. The formula for offline performance is taken from [3], and is reproduced in equation 1, where e_t^* is the lowest error found so far at generation t, and T is the total number of generations. From now on, we refer to offline performance as offline error, and since we are minimizing error in symbolic regression problems, lower offline error is better.

$$\mathit{OfflineError} = \frac{\sum_{t=1}^{T} e_t^*}{T} \tag{1}$$

These results were interpreted in light of the genetic operator-based search of GP – potential solutions which are semantically close are not necessarily close in the operator-based search landscape. However, because we only had a relatively small number of first period target functions (13), for reasons of robustness we wished to follow up on this work with a greater number of first-period target functions, and a greater number of runs. The next section outlines the approach taken in this current work, to this end.

2 Outline of Experimental Set-up

In our current work, we keep an even distribution of pairs of target functions in five semantic distance 'buckets' – the buckets are evenly spaced in the range from zero to two and are fully described below. This allows us to compare the performance of GP populations during the second period of evolution, across second-period functions at differing semantic distances from the first-period target function. We investigate whether or not GP populations perform better when the semantic distance from the first to the second-period target function is relatively small. GP run parameters are listed in table 1. We perform two sets of experiments. Both sets of experiments involve the following steps:

1. Generate 15,000 pairs of target functions, using ramped half-and-half tree initialization, with different depth limits than those used for population initialization. The first function in the pair is used as the target in the first period of evolution, and the second serves as the target function in the second period. Measure the semantic (mean absolute) distance between all 15,000 pairs of functions. Ensure an even distribution of function-pairs across five semantic distance-buckets. Semantic distance-buckets have the following lower and upper bounds:

   - $0.2 \rightarrow 0.4$
   - $0.6 \rightarrow 0.8$
   - $1.0 \rightarrow 1.2$
   - $1.4 \rightarrow 1.6$
   - $1.8 \rightarrow 2.0$

   The maximum semantic distance between successive targets is two. We experimented with larger maximum distances, but the density of paired-functions at greater semantic distances reduces greatly, and furthermore, the trend found in our results roughly carries on to larger semantic distances.

   First set of experiments There are 15,000 first-period and 15,000 second period functions, with 3,000 function-pairs in each of the five distance-buckets.

   Second set of experiments We perform 100 *sets of runs*, or 'run-sets'. There are 100 first-period and 15,000 second-period functions. Each run-set has a common first-period target function, but 150 different second-period target functions, with 30 functions each in each of the five distance-buckets. Like in the first experiments, we perform 15,000 paired runs, but this time, there are 100 first-period functions each paired with 150 second-period functions, for a total of 100 times 150, or 15,000 runs.

Table 1: GP Run Parameters

Population Size	100	Number of gens (each time period)	20
Crossover Rate	0.9	Subtree Mutation Rate	0.05
Elitism	0.05	Initialization	Ramped half & half
Max individual initial tree depth	6	Max individual tree depth possible	10
X Range	[-1, 1]	Number of points sampled in range	20
Function Set: Binary Operators	+, -, *, /	Function Set: Unary Operators	cos, sin, ln, exp
Terminal Set	X, 1	Fitness Calculation	Mean Absolute Error

The reason for the approach taken in our second set of experiments is described in detail in section 3. (Briefly, our first set of experiments were open to a bias that stemmed from the differing average 'difficulty' of the first-period target functions over the five semantic-distance buckets.)

2. GP is run over two periods, that are both 20 generations long, on all 15,000 pairs of functions. Two approaches are adopted with respect to the GP population:
   (i) The (continuous) population carries over from the first to the second period.
   (ii) The GP population is *re-initialized* at the beginning of the second period (we use the terms 're-initialized population' and 'restart population' interchangeably throughout the rest of the text).

   We are particularly interested in whether using a continuous population is better than re-initializing the population at the beginning of the second period for the closer distance-buckets. If continuous GP is most successfully adapting to small changes in the target, the advantage to using a continuous population should be greatest at small semantic distances.
3. We record the lowest mean absolute error achieved at each generation for each two-period run (for both continuous and re-initialized populations), which we use to calculate offline error in both periods (see equation 1, which shows the formula used to calculate offline error).

Ramped Half-and-Half Target Function Generation: In order to limit any unfair advantage to re-initialized populations after the change in the target function, we altered the parameters and operation of ramped half-and-half when using it to generate target functions. Ramped half-and-half is ad hoc [1] – it has biases in the types of trees it generates – as do other tree generation techniques [13]. We used different ranges of possible tree depths than that which were used to initialize a search population. When initializing the search population, the minimum tree depth was two, and the maximum depth was six. In contrast, the minimum target function tree depth used was five, and the maximum was eight. Furthermore, when generating target functions, we forced function trees to have at least depth five. That is, when using the 'grow' initialization method to generate function trees, we discarded any trees which were returned with a depth of less than five, and generated a new tree with depth at least five to replace such a tree that broke our constraint on the minimum allowable depth limit. In an attempt to determine whether biases still remained in favour of restart populations, after running our second set of experiments, we measured the number of runs for which the best solution in the re-initialized population at the beginning of the second period was within a mean absolute error of 0.00001 of the target function, and in which the best solution in the continuous population had a mean absolute error of greater than 0.00001. This was the case in only 2.2% of runs.

In the next section, we describe our experimental results.

3 First Set of Experimental Results

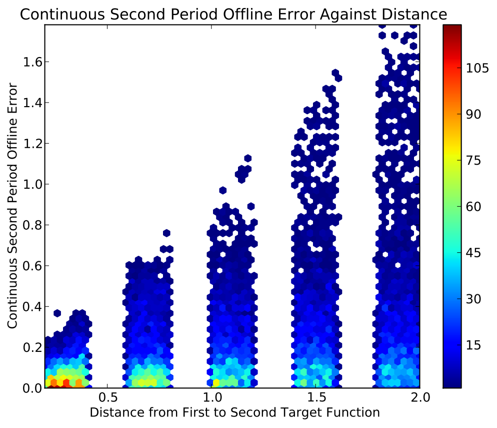
We show a hexagonally binned histogram of results over all 15,000 runs for the continuous population in Fig. 2(a) – with offline error in the second period on the vertical axis, and the distance of the second-period target function from the first-period target function on the horizontal axis. (Note that a bin must have at least one data point to take a colour other than white.) If being semantically closer to the first-period function confers an advantage on the continuous population in the second period of evolution, then we would expect to see a heavier concentration of points at lower offline errors for functions in the closer semantic distance buckets, and conversely, a higher concentration of points at higher offline errors for functions in the more distant buckets.

3.1 Evidence That Lower Semantic Distance from the First-Period Target Confers an Advantage

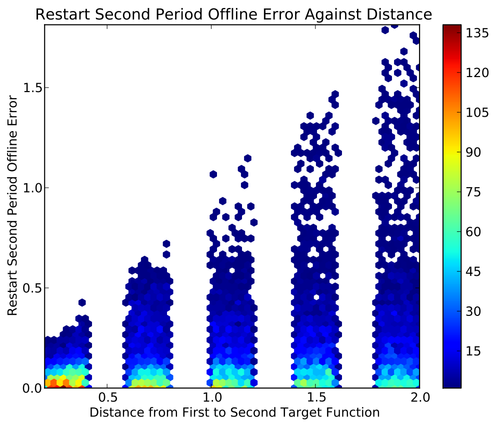
As can be seen, we do see such a result in Fig. 2(a). Another obvious thing we notice is the roughly diagonal line through the plot, above which there are no coloured bins. This line roughly equates inter-function distance with second-period offline error – the worst-case offline error in the second period roughly equals the semantic distance between the first and second target functions. Intuitively, we might expect no points to fall above this line. If the solution in the first period is close to optimal, and if evolution in the second period cannot find a solution that is fitter against the second target than the best solution found in the first period, then elitism will select the solution found in the first period into the population at each generation in the second period. Provided no fitter solution is found throughout the second period, then the error at each generation in the second period should equal the mean absolute distance between the first- and second-period target functions.

3.2 Semantically Close Functions are also Easy to Find using an Initialized Population

We show a hexagonal histogram plotting offline error in the second period against distance between the first and second targets for the restart populations, in Fig. 2(b). A roughly straight line through the diagonal is present in this figure, similar to that observed in Fig. 2(a). This is unexpected – for restart populations, there is no best solution carried over from the first period which, provided near-optimal performance in the first period, will have error equal to the distance between targets during the second period. Furthermore, in Fig. 3, we see that smaller semantic distance between the first- and second-period target functions consistently resulted in lower offline error in the *first* period.



(a)



(b)

Figure 2: Hexagonallly Binned Histogram: Distance versus offline error (a) continuous population (b) restart population

Fig. 4 displays the mean and standard error of the mean, of the best individual at each generation, over all runs in all five buckets, for continuous and restart populations. Each plot is labelled with either 'continuous' or 'restart', followed by the lower bound of the relevant semantic-distance-bucket. (The exact ranges of each of the semantic-distance buckets are as follows: $0.2 \rightarrow 0.4$, $0.6 \rightarrow 0.8$, $1.0 \rightarrow 1.2$, $1.4 \rightarrow 1.6$, and $1.8 \rightarrow 2.0$.) As can be seen, smaller semantic distance between the first and second target functions consistently results in better performance in the second period for the continuous populations: average best error at each generation is lowest for the closest distance-bucket, second-lowest for the second-closest distance-bucket, and so on. However, this is also true for restart populations, which is consistent with the results in Fig. 2(b), discussed above. The average best error in the *first* period of evolution is lowest for the closest distance-bucket, second lowest for the second-closest distance-bucket, and so on, which is consistent with the results in Fig. 3. Fig. 4 also shows that the average best error for the re-initialized populations is lower at every generation in the second period, than that of the continuous population, for all semantic distance buckets.

To summarize, we have found a relationship between worst-case offline error and semantic distance for first-period populations and re-initialized second-period populations. We have also found that average best error at every generation in the first period of evolution is lower, the smaller the semantic distance between functions, and the same result applies to the average best error for re-initialized populations in the second period. We conclude that there is a relationship, on average, between the semantic distance of a function to another random function, and the difficulty of finding the function from scratch using GP. Compared with functions in smaller distance-buckets, there is a greater chance that the first or second-period functions in greater distance-buckets will be difficult for GP to find from scratch.

We performed an analysis of the size and depth of trees in the both periods of evolution, to investigate whether first or second-period functions falling into the semantically more-distant buckets may have had larger or deeper trees, thus offering a potential explanation to their increasing 'difficulty'. Average tree size increased as semantic distance increased, for target functions in both the first and second period, and graphs showing

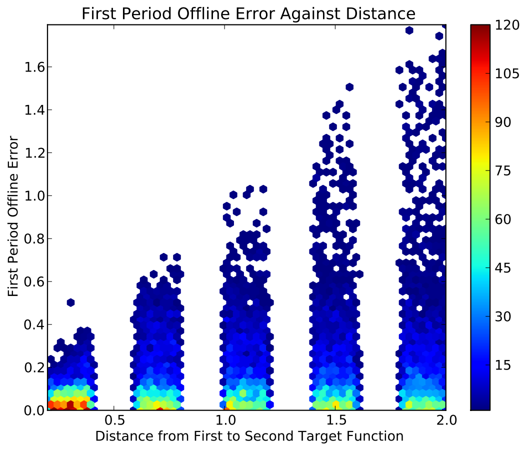


Figure 3: First Experiments:
Hexagonally Binned Histogram:
Distance versus offline error
– first period

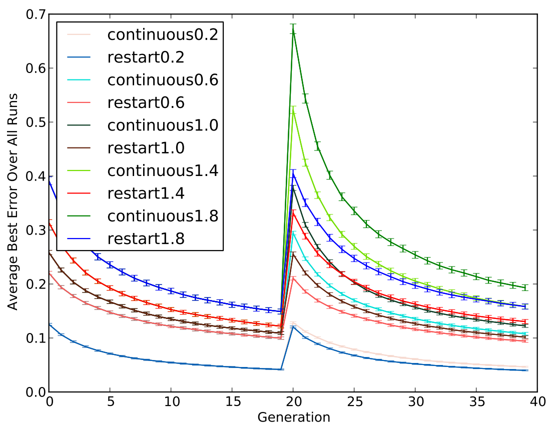


Figure 4: First Experiments:
Mean best solution over the generations:
all distance-buckets:
restart and continuous populations

these results are show in Figs. 5(a) and 5(b). Average tree depth did not really change as semantic distance increased.

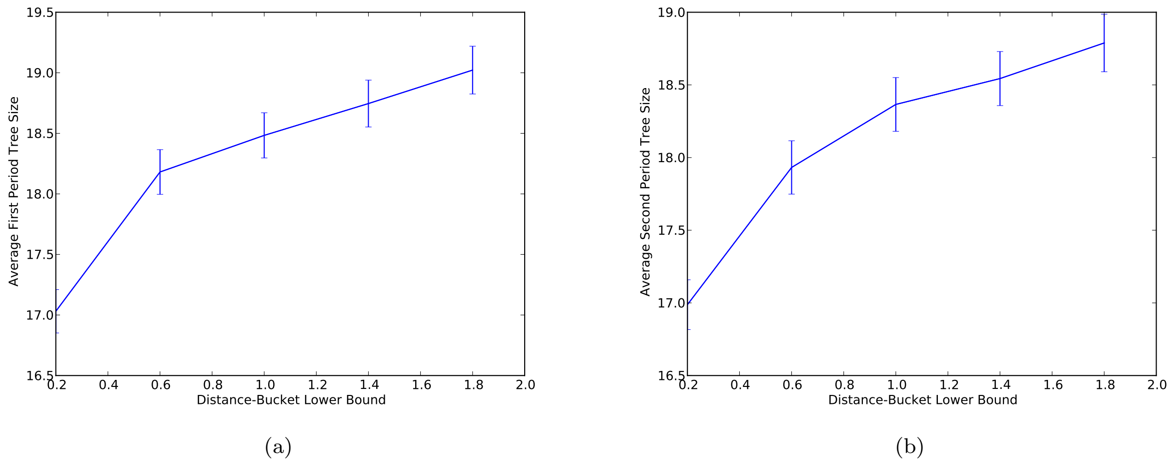


Figure 5: Mean (and std. error) tree size (a) first-period functions (b) second-period functions

4 Second Set of Experimental Results

4.1 Controlling for the Easiness of the First-Period Target Function

In order to control for influence of the 'easiness' or 'hardness' of the target function in the first period, we performed 100 sets of experiments (or 100 'run-sets'); for each 'run-set', the first-period target was always the same. Therefore, in the first period, the distribution across 'easy' and 'difficult' functions was the same for each distance-bucket. For each run-set, there were 30 second-period functions in each of the five distance-buckets. Our second set of experiments had the advantage that, when we examined performance in the second period, we knew that, for each first-second-function pairing, there was an equivalent pairing of the first function with a second function from *each* of the other four distance-buckets. Therefore, when we examined the performance in the second period across the five buckets, we knew that second functions in closer distance-buckets were not given a headstart (on average), due to the higher prevalence of 'easy' functions in the first period for closer distance-buckets than for further distance-buckets. We did not ensure that there were an equal number of easy and difficult functions across distance-buckets in the second period. The procedure to do so would be complex, and it is not clear that it would be faesible, at least for an appreciable number of experiments. Furthermore, as far as the comparison between restart and continuous goes, target functions in the second period were identical

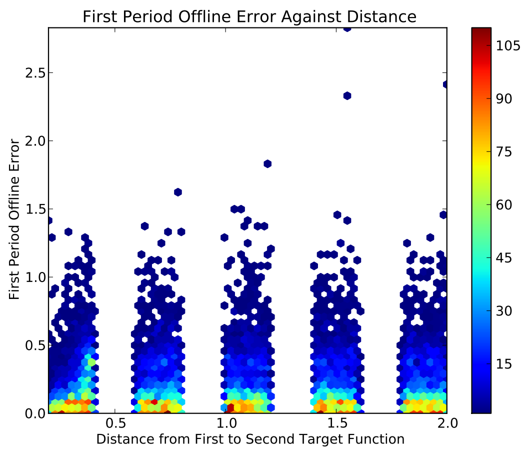


Figure 6: Second Experiments:
Hexagonally Binned Histogram:
Distance versus offline performance
– first period

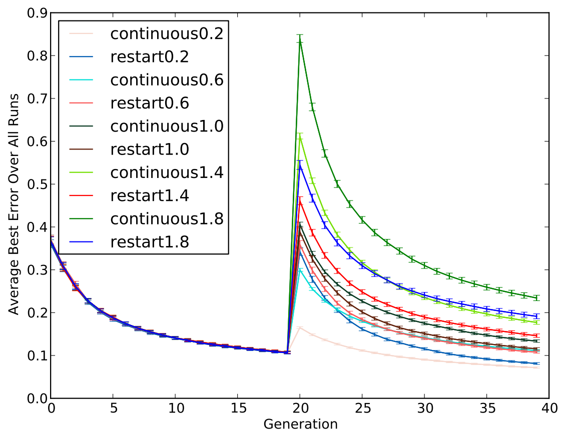


Figure 7: Second Experiments:
Mean best solution over the generations:
all distance-buckets:
restart and continuous populations

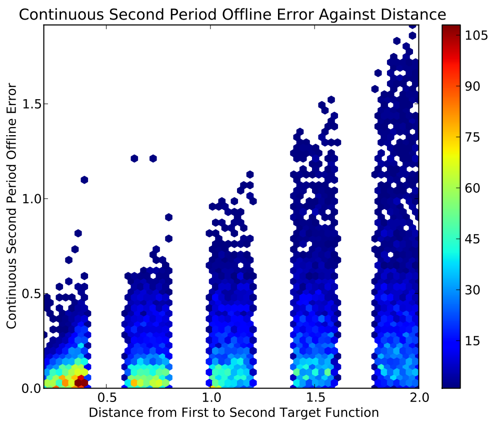
for continuous and restart populations.

In order to generate second-period functions, we built up to one and a half million functions, and measured their semantic distance from the first-period function. Sometimes, we were simply unable to find a sufficient number of second-period functions to populate each semantic-distance-bucket. If, after trying 1.5 million functions, we had not populated all distance-buckets, we moved on to the next experiment. (This resulted in approximately 13% of experiments being terminated, and replaced with another experiment).

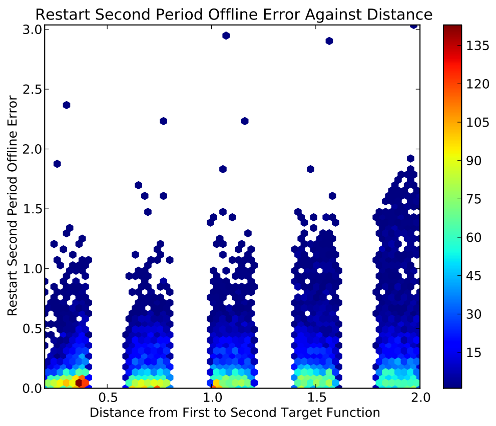
4.2 Lower Semantic Distance from the First-Period Target Function Confers an Advantage

Fig. 6 shows a plot of offline error in the first period of evolution against the semantic distance between the first and second target functions. As can be seen, the semantic distance between targets was unrelated to offline error in the first period, unlike in our first set of experiments. A plot showing the mean best error over the generations is shown in Fig. 7 (error bars show the standard error of the mean). On average, continuous populations performed better than re-initialized populations at every generation in the second period for the smallest bucket. For the second smallest bucket, continuous populations outperformed re-initialized populations on average in the earlier generations, before being overtaken in the later generations by re-initialized populations. (For the entire period, taken as a whole, continuous populations performed better.) For the three largest buckets, re-initializing was a better strategy. For continuous populations, at every generation, performance in the second period on functions in closer semantic buckets was better on average than for functions in further-away semantic buckets. This was also true of restart populations. Second-period target functions in closer distance-buckets were more likely to be relatively easy to find for a re-initialized population, which followed on from the relationship between function-hardness and semantic-distance discovered in section 3.2. Given this information, better performance for restart populations on functions in closer distance-buckets was not surprising. The primary issue we wished to investigate was, whether or not using a continuous population was sufficiently advantageous over using a re-initialized population, for semantically close buckets. We can see that this was indeed the case.

We show hexagonally-binned histograms of offline error against semantic distance for continuous and re-initialized populations in Figs. 8(a) and 8(b). If being semantically closer to the first-period function conferred an advantage on the continuous population in the second period of evolution, then we would expect to see a heavier concentration of points at lower offline errors for functions in the closer semantic distance buckets, and conversely, a higher concentration of points at higher offline errors for functions in the more distant buckets. But given the relationship between the 'easiness' of finding a target function from scratch, and the semantic distance between that function and another target function, we expect to see the same effect for restart populations. (That is to say, we expect more points at lower offline errors for closer distance-buckets, and more points at higher offine errors for more distant buckets). The question is, is the effect more pronounced for continuous populations? We observe this effect for both restart and continuous populations, but the relationship between offline error and semantic distance is clearly weaker for restart populations than for continuous populations, as we can see in Figs. 8(a) and 8(b).



(a)



(b)

Figure 8: Second Experiments: Hexagonallly Binned Histogram: Distance v. offline error (a) continuous (b) restart

5 Summary of Key Results

We have established the following key results through our experiments:

1. There is a relationship, on average, between the performance of an initialized GP population on a target function, and the semantic distance of that function to another random function. As a result of this, re-initalized populations perform better after a change in the target function when the semantic distance of the change is low. This is not because the magnitude of semantic chance is small; but rather it results from the relationship between lower intra-function semantic distance and greater ease in finding a target function from scratch.

2. The lower the semantic distance between the first and second period target functions, the better the performance in the second period, for continuous GP populations. Furthermore, *at low semantic distances, continuous populations in the second period outperform restart populations.*

6 Conclusions

While genetic programming searches over genetic-operator based fitness landscapes, potential solutions that are relatively close in operator-based landscapes may be relatively semantically distant, and vice-versa. In dynamic settings, when the optimization goal changes, it might seem reasonable to naively expect that changes of a smaller semantic magnitude will be easier to adapt to than larger changes. We investigated whether this was the case for the popular application area of symbolic regression.

We found that when the optimization goal changes a small amount (in terms of semantic distance) continuous populations outperform re-initialized populations. This is encouraging: even in light of the difference between relative operator-based and semantic distances, our results show that using continuous-population-GP can be an effective approach to dynamic optimization problems. However, in the process of disovering this result, we noticed that when comparing the impact on performance of small versus large changes to the optimization goal (in terms of semantic distance), not only did continuous populations perform better when the change is small; this same result also applied to re-initialized populations. This latter result was initially surprising. Upon further investigation, we discovered that this effect in re-initialized populations resulted from a relationship between function hardness and semantic distance. After controlling for the relationship between function hardness and semantic distance across distance-buckets in the first period, we conclude that continuous populations adapt better than restart populations to smaller semantic changes, and that both continuous and restart populations adapt better to smaller changes in the optimization goal than larger changes, though the effect is less pronounced for restart populations; an important illustration of the pitfalls in examining these subtle relationships.

In future, we would like to more closely examine the exact semantic distance at which continuous offline error falls below (outperforms) that for a restart population. We would also like to examine how our results change, if at all, when examining shorter and longer first and second periods of evolution. We would also like to experiment with different target function-generation techniques, and further examine techniques to enable an equal distribution of target function difficulty across buckets in the second period.

Acknowledgement: We gratefully acknowledge the financial support of Science Foundation Ireland under

Grant Number 08/SRC/FM1389.

References

[1] Böhm W., Geyer-Schulz A.: Exact uniform initialization for genetic programming. In: Proceedings of the Foundations of Genetic Algorithms Workshop (FOGA'4), pp. 255–269. International Society of Genetic Algorithms (1996)

[2] Branke J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, volume 3. IEEE (1999)

[3] Branke J.: *Evolutionary optimization in dynamic environments.* Kluwer Academic Publishers (2002)

[4] Branke J., Kaußler T., Schmidt C., Schmeck H.: A multi-population approach to dynamic optimization problems. In: Adaptive computing in design and manufacturing, pp. 299 – 307. (2000)

[5] Cobb H.G., Grefenstette J.J.: Genetic algorithms for tracking changing environments. In: Proceedings of the 5th International Conference on Genetic Algorithms, pp. 523–530. Morgan Kaufmann (1993)

[6] Dempsey I., O'Neill M., Brabazon A.: Adaptive trading with grammatical evolution. In: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, pp. 2587–2592. IEEE. (2006)

[7] Jones T.: *Evolutionary Algorithms, Fitness Landscapes and Search.* PhD thesis, Uni. of New Mexico (1995)

[8] Loginov A., Heywood M.I.: On the impact of streaming interface heuristics on gp trading agents: An fx benchmarking study. In: Proceedings of the 15th annual conference on genetic evolutionary computation, pp. 1341–1348. ACM. (2013)

[9] McDermott J. et. al: Genetic programming needs better benchmarks. In: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference, pp. 791–798. ACM (2012)

[10] Nguyen T.T., Yang S., Branke J.: Evolutionary dynamic optimization: A survey of the state of the art. Swarm and Evolutionary Computation, pp. 1–24 (2012)

[11] Nguyen, Q.U., Nguyen X.H., O'Neill, M., McKay, R.I, Galván-López, E.: Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genetic Programming and Evolvable Machines, **12**(2), 91–119. Springer. (2011)

[12] O'Neill M., Vanneschi L., Gustafson S., Banzhaf W.: Open issues in genetic programming. Genetic Programming and Evolvable Machines, **11**(3-4), 339–363 (2010)

[13] Poli R., Langdon W.B., McPhee N.F.: *A field guide to genetic programming.* Lulu Enterprises Uk Ltd. (2008)

[14] Tuite C., O'Neill M., Brabazon A.: Towards a dynamic benchmark for genetic programming. In: Proceedings of 15th annual conference companion on Genetic and evolutionary computation, pp. 151–152. ACM (2013)

[15] Wagner N., Michalewicz Z., Khouja M., McGregor R.R.: Time series forecasting for dynamic environments: the dyfor genetic program model. Evolutionary Computation, IEEE Transactions on, **11**(4), 433–452 (2007)

[16] Yang S.: Genetic algorithms with memory-and elitism-based immigrants in dynamic environments. Evolutionary Computation, IEEE Transactions on, **16**(3), 385–416 (2008)