Geometric Semantic Genetic Programming for the Prediction of Tennis Match Scores

William Venables
University of Exeter
Harrison Building
Exeter, EX4 4PY
wv207@exeter.ac.uk

ABSTRACT

Genetic Programming has been shown as an effective tool for acquiring models to problems of unknown formulation. A particularly useful application of this is symbolic regression, where the task is to predict numerical output values given an array of different inputs. A limitation of this however is the near infinite domain which they may operate in, which can lead to a challenging, potentially deceptive landscape to search. Geometric Semantic Genetic Programming (GSGP) is a proposed as a means of solving this, by making use of specially designed genetic operators to guarantee a conical landscape which is much easier to search. This paper presents a multi-population approach, using several GSGP instances to predict individual values of a desired vector output. This is applied to the challenging problem of predicting tennis match outcomes, and shows that GSGP can outperform both a geometric semantic hillclimber and a standard genetic programming implementation.

Keywords

Geometric Semantic; Genetic Programming; Genetic Algorithms; Evolutionary Computation; Symbolic Regression; Tennis Prediction;

1. INTRODUCTION

Genetic Programming (GP) has been shown to be a powerful meta-heuristic for solving numerous problem categories. Its ability to find a solution's formulation which maximises the fitness function to a problem with an unknown formation is extremely powerful, making it applicable to a number of generic problems such as symbolic regression and classification without the necessity for prior knowledge. However, a draw back of this methodology is the potentially infinite, multi-modal search space, which can inhibit the algorithms ability to converge to a near-global optimum. A potential solution to this is to constrain the crossover and mutation operators such that they are *geometric semantic*. Thus, Geometric Semantic Genetic Programming (GSGP) was proposed by Moraglio *et al.*[12]. The specially designed mutation and crossover operators used ensure that any offspring produced are related under a semantic distance which can be proven to give a conic fitness landscape which is more trivial to search.

This improvement has been applied to many different domains with success, ranging from medical applications [18, 3] through to the electoral redistricting problem [4]. However, the effectiveness of the approach relative to other techniques varies depending upon the problem itself. Whilst it is shown to be suitable for many different real-world scenarios, this paper shall aim to compare its effectiveness to other techniques on a practical problem with a formulation near impossible to model analytically.

Whilst a number of the discussed problems involve finding a single output value, many interesting topics involve optimising solutions whilst simultaneously finding multiple output values. A naïve solution would be to combine these objectives into a single, unified fitness metric. However, GSGP is not guaranteed to converge to a global optimum in a reasonable timeframe.[14] It is suggested that it may be more effective to decompose the problem into a series of subproblems and solve these simultaneously through GSGP. This would allow each instance to become more specialised and removes the need for an optimal solution to separated into into different distinct paths corresponding to each value. This paper thereby outlines MP-GSGP, a multi-population approach to the novel problem of tennis match outcome prediction.

2. LITERATURE REVIEW

The idea of generating evolving programs to solve a specific problem can be traced back to the works of Turing [16] and the well renowned Turing test. However, the first successful implementation of GP was arguably those performed by Forsyth in 1981, before being more famously being implemented in LISP and coined as GP by Koza's patent application in 1990, based on the prior works of his PhD supervisor John Holland [5, 9, 7].

A significant challenge associated with GP has been the infinitely sized domains in which they operate, which has been documented from near the field's inception [13]. This is an inherent difference between GP and the commonly associated Genetic Algorithms, which operate on a reduced domain size owing to their fixed representation size rather than the tree representation used within Genetic Programming. This problem is exacerbated by the multi-modal fitness landscapes being searched, which can make it challenging to solve problems. Whilst the former problem is simply a characteristic of this GP problem, GSGP is proposed as a solution to the latter by constraining the crossover and mutation operators to those within a subset of those available [12]. It defines Geometric Semantic Crossover $C : S, S \rightarrow S$ w.r.t. a comparative metric d as any offspring solution $o = C(p_1, p_2)$ which falls along the metric segment of the two parents p_1 and p_2. Similarly, Geometric Semantic Mutation $M : S \rightarrow S$ w.r.t d is defined as any offspring $o = M(p)$ such that o falls

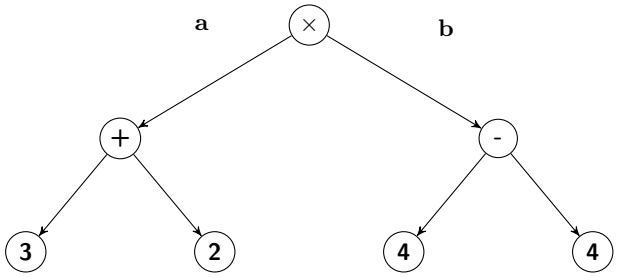

Figure 1: The tree structure here would result in the left side of the tree (labelled **a**) being evaluated unnecessarily, as side **b** would cause the result to always equal 0.

within the metric ball of the given parent. This ensures the landscape is always conical and therefore *very easy* to search.

Another issue commonly faced with Genetic Programming is the computational cost associated with evaluating the different solutions. Given many of the proposed operators involve the concatenation of trees, the evaluation times can increase exponentially if this is not accounted for. To combat this, many different approaches have been introduced. One such approach is to simplify the expressions and subtrees after each generation [8]. However, this approach can still face challenges when modelling particularly complex problems with multiple input variables. An alternative approach proposed by Handley is to use a directed acyclic graph as the population, rather than a forest of trees. However, whilst this aims to promote re-use of identical functions within solutions, it does nothing to avoid the evaluation of unnecessary subtrees. For example, if the structure given in Figure 1 was evaluated, the left side of the tree would be evaluated unnecessarily given the right side evaluates to 0. Whilst not a significant issue here, when this is scaled up to much larger trees this cost would have a more significant effect. Perhaps a simpler approach to this issue is to cache the subtrees (and corresponding fitness values) using a hash table [17]. This takes advantage of the advances in computational efficiency and storage capacity since the topics inception, though it can still reach limitations with the storage capacity of the machine. It becomes clear that the appropriate solution greatly depends upon the problem being solved.

An interesting progeny of the core GSGP approach has been the involvement of multiple populations. Lin *et al.* show that these can be used to give accuracy comparable to that of a singular GSGP implementation, but do so in a considerably shorter timeframe [10]. Additionally, it has been shown that a co-evolutionary approach via decomposition can be an appropriate means of solving a classification more accurately than the traditional *bagging* and *boosting* approaches [2]. Whilst these approaches aim to work in co-operation to provide each value of a solution, there is little literature exploring the usage of a having a single GSGP instance working towards solving a single output value, a lacuna this paper aims to fill.

3. MP-GSGP DESIGN

The proposed algorithm, referred to as MP-GSGP hereafter, is a means of solving general symbolic regression problems of which the desired output is a vector of independent values. Whilst this can be achieved through a singular GSGP approach, building a solution from a vector of GSGP trees where each tree gives a singular value can allow each tree to become more specialised than a unified approach encompassing the whole problem together. The hope is that such an approach should allow for simultaneous feature selection and optimisation on each output vector feature to thereby give a more adept output programs.

Whilst the meta-heuristic framework of GSGP provides a fixed starting point for the implementation, it remains to choose suitable approaches for crossover, mutation and selection. Whilst many alternatives to the semantic crossover have been produced (the reader is referred to Pawlak *et al.* for an in-depth review [14]) a design choice to use the original paper's method rather than an approximation in this work. Whereas the review suggests using approximate operators to achieve smaller program sizes and a better generalisation, using the exact crossover approach is more appropriate in drawing comparisons to other methods as it will allow the implementation to exhibit the full potential of GSGP. Thus, the MP-GSGP crossover is defined in Algorithm 1.

Algorithm 1: Crossover
A pseudocode outline for the Geometric Semantic Crossover operator which is used within MP-GSGP.

```
1: function Crossover(ParentA, ParentB)
2:     r ← random number in range [0-1]
3:     return r × ParentA + (1 − r) × ParentB
```

Consequently, the decision was also made to use the mutation operators from the initial paper. Hence, the mutation operator is defined in Algorithm 2 where `mr` represents the tuneable mutation step.

Algorithm 2: Mutation
A pseudocode outline for the Geometric Semantic Mutation operator which is used within MP-GSGP.

```
1: function Mutation(Parent, mr)
2:     A ← RandomFunction()
3:     B ← RandomFunction()
4:     return Parent + mr × (A × Parent − B × Parent)
```

The final choice to be made for the system's implementation is the selection operator, which can be an important characteristic of a GP algorithm for the purpose of balancing *exploitation* and *exploration*. The most common selection mechanism used (and that of the initial GSGP paper) is tournament selection, as this is insensitive to absolute values of fitness [6]. In the validation problem, a tournament size of 5 was chosen for the experiments. At each generation, the population is truncated to include the fittest solutions provided by `TruncationRatio`. The overall algorithmic structure for MP-GSGP is provided in Algorithm 3.

Algorithm 3 refers to RandomFunction, which are subtrees generated up to a maximum specified depth. These are used to generate both the initial population and within the mutation operator. It generates a subtree up to `MaxDepth` containing a random choice of input variables or numerical values, all falling in the range 0–1. A pseudocode outline for this is also given in Algorithm 4.

The literature review in Section 2 highlighted the need to manage the size of the solutions created. As this work does

Algorithm 3: MP-GSGP
A pseudocode outline for the MP-GSGP algorithm used on the validation problem. The structure is predominantly remains the same for the novel tennis prediction problem, with the only change being in the selection mechanism.

```
1: function EVOLVE(PROBLEM, FITNESS, PopSize, NumGenerations, TruncationRatio)
2:     Pop ← {RANDOMFUNCTION() | x ∈ [1, .., pop_size]}
3:     Best, BestFitness ← Null, ∞
4:     for iteration in NumGenerations do
5:         SortedPop ← Pop sorted by FITNESS
6:         temp ← first element of SortedPop
7:         if BestFitness < FITNESS(temp) then
8:             Best, BestFitness ← temp, FITNESS(temp)
9:         Parents ← First TruncationRatio × PopSize of SortedPop
10:        Pop ← [ ]
11:        while length of Pop < PopSize do
12:            Pop ← APPEND(Pop, TOURNAMENTSELECTION(Parents))
13:    return Best
```

```
1: function TRAINTREE(Subproblems, Inputs, FITNESS, PopSize, NumGenerations, TruncationRatio)
2:     Tree ← [ ]
3:     for Problem in Subproblems do
4:         Tree ← APPEND(Tree, Evolve(Problem, FITNESS, PopSize, NumGenerations, TruncationRatio))
5:     return Tree
```

```
1: function PREDICT(Tree, Inputs)
2:     Results ← [ ]
3:     for Input in Inputs do
4:         OutputVector ← [ ]
5:         for Function in Tree do
6:             OutputVector ← APPEND(OutputVector, FUNCTION(Input))
7:         Results ← APPEND(Results, OutputVector)
8:     return Results
```

Algorithm 5: RandomFunction
The pseudocode outline for random function generation, which is used extensively within MP-GSGP.

```
1: function RANDOMFUNCTION(Depth, InputValues)
2:     if Depth == 1 or rand() < 1/2^(Depth−1) then
3:         Terminals ← CONCATENATE(InputValues,
               {RAND()|x ∈ [1, .., LENGTH(InputValues)]})
4:         return CHOICE(Terminals)
5:     else
6:         return CHOICE(Operators)(
               RANDOMFUNCTION(Depth − 1),
               RANDOMFUNCTION(Depth − 1)
           )
```

aims to highlight the effectiveness of MP-GSGP as applied to a certain problem, the decision was made to implement the caching approach as this would allow for the easiest to comparison to other approaches.

4. VALIDATION

As the proposed tennis prediction system contains a significant number of input values, it is challenging to visualise these. Instead, the implementation was first tested against a smaller problem which could be more easily assessed by eye in order to validate the system was correct.

The chosen validation problem was the parameterised equation $(x, y, z) = (t, t^2, t^5 - t^4)$, and was performed over two separate experiments. The first passed the x and y values as inputs, with the MP-GSGP algorithm tasked with producing the z value. Intuitively, with only one output variable the algorithm would only be using a single population. Thus, this tested the implementation and validity of the GSGP implementation itself, ensuring the collated components worked together sufficiently.

The second experiment looked at ensuring the implementation could suitably decompose the problem and converge towards optimal values. For this, the singular input of x was provided, with the aims of producing the y and z values given in the above parameterised equation.

For the fitness metric, cumulative absolute error was chosen. As each output variable is working in only one dimension, the choice of fitness does not make a significant difference. However, if these were larger problems then it is recommended to use Euclidian over the slightly more efficient Manhattan distance used here [1]. However, the cumulative absolute error seemed an appropriate metric for the validation problem as it gives a landscape which is easy to analyse visually through charts, as is done so in the results.

Each experiment was run a total of 20 times, with both the number of training inputs and population size set to 500. The algorithm's mutation step was set to 0.1, and random functions were generated up to a maximum depth of 5. These results were recorded over 100 generations and are visualised in Figure 2.

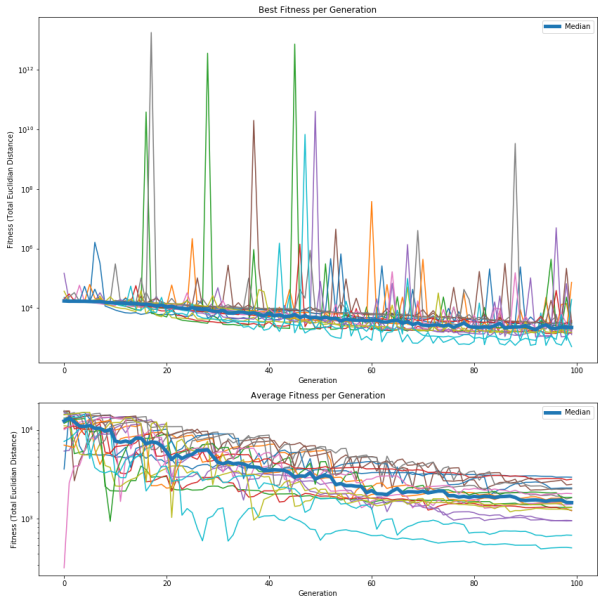

Figure 2: The best and mean fitness over time for the singular GSGP problem.

Whilst it may not seem necessary to provide these results, they show that there is a clear convergence towards an optimal solution to the problem over time. The validation of the implementation is always an important step when attempting a new problem. If the target problem does not appear to be converging towards an optimal result, the knowledge of a sound implementation can help to draw quicker conclusions. In some cases, for example, it may act to indicate the complexities of the problem itself are preventing a convergence, rather than the implementation which could be case with the novel problem given later on in Section 5.

In this situation, however, the results show a clear convergence towards an optimal solution over time, indicating that the algorithm is working as expected and has a sound implementation.

5. EXPERIMENTAL DESIGN

The prediction of tennis matches itself is a challenging task, owing to the complexities of the game and the vast number of variables which would have an influence on the outcome. Whilst an easier challenge would be a binary classification predicting the winner of the match, this paper looks at the more interesting task of attempting to predict the games won in each set through symbolic regression.

The dataset used was a collation of all the professional ATP male matches in 2018, courtesy of TENNIS-DATA.CO.UK [15]. This was processed in order to give a total of 47 different input variables, which are composed of integer or boolean values (represented themselves as integers, 0 being false and 1 being true). These are as follows; the ATP global ranking and ATP points for both opponents; an integer representing a type of court the match is held on (1–4); the set scores, court type, personal and opponents ranking for the past three matches each player has been in. To ensure the software would run on both the women's and men's matches, the decision was made to only include the first two sets of any men's match, rather than having separate versions with more inputs, or instead placeholder values where data does not exist.

For MP-GSGP, a semi-steady state approach was found to be the most effective in preliminary trials. At each generation, the bottom 50% of the population (250 of the 500) were replaced with new solutions obtained through mutation and crossover. This helped to preserve feasible solutions more effectively, whilst still allowing enough exploration of the landscape to take place. The selection mechanism for crossover was uniform stochastic sampling from the parent population with replacement, given in Algorithm 6.

Algorithm 6: SEMI-STOCHASTIC SELECTION
A pseudocode outline for the Semi-Stochastic selection mechanism which was found to be the most effective in the novel tennis score prediction problem.

```
1: function SEMI-STOCHASTIC SELECTION(Population, TruncationRatio)
2:     ParentPop ← TRUNCATE(Population,
           LENGTH(Population) × TruncationRatio)
3:     A, B ← SAMPLE(ParentPop, 2)
4:     return A, B
```

Analytically, it was discovered that all the tennis matches in the training and test datasets only contained scores in the range 0–7. Whilst tennis can have higher scores, it was decided to ignore these for the purpose of the experiment. As a result, the fitness function heavily penalised any values falling outside of this range by allocating a fitness of 1000. If they fell within the desired range, however, then the fitness was again the absolute error between the value and the desired target. This was evaluated over the entire training dataset and summated to give the total fitness. The same fitness function was used in all the methods.

For the purpose of comparison, MP-GSGP is also compared to the results achieved to a geometric semantic hillclimber, a standard genetic programming implementation and a theoretical accuracy rate achievable through stochastic sampling with prior knowledge. A greedy geometric semantic hillclimber used was based on some of the functionality developed for GSGP, starting with a random function and then using geometric semantic mutation to iteratively search and attempt to find a fitter solution within the landscape. The Multi-Population Genetic Programming implementation (MP-GP) was given the same parameters as MP-GSGP in order to allow for a fair comparison.

All methods trialled had access to five operators. Multiplication, addition, subtraction were implemented as one would expect, however there was also the inclusion of a safe division where should there be a division by 0, the maximum of the two input values was instead chosen. Finally, a less than operator was included, which returned 1 should $x < y$ and 0 otherwise. This was found to partially help constrain the values to the desired range and prevent exponential fitness sizes, thereby increasing the convergence ability and helping to create suitable solutions.

6. EXPERIMENTAL RESULTS

6.1 MP-GSGP

The results for the MP-GSGP can be visualised in Figures 3 and 4, showing the average and best results over time for each of the output values respectively. The average values provide an interesting point of discussion. It is clear that the mean for each generation is extremely high in comparison to the best values achieved, owing to the heavy fitness penalty for invalid values. Clearly, this indicates that a significant portion of the solutions generated by the algorithm fall outside of the desired range. This is perhaps one of the difficulties associated with this problem. If the desired output values were in the range -1–1, this would be easier to overcome due to the result of multiplication always being less the same or less than the input value. However, given the desired outputs are in the range 0–7 as described above, there is a significant chance that a multiplication applied to a nearby solution (say, 6) will result in a value higher than anything feasible. Whilst solutions could involve modifying the fitness function to constrain the results to scale between 0 and 1, this is would be changing the problem to work on non-discrete values. A further alternative would be to use one-hot encoding for the potential regression values and convert the regression values to a classification task.

This being said, the algorithm does appear to quickly converge in terms of the average value, and by referring to Figure 4 it is clear that the algorithm continues to improve the fitness of solutions over time. However, it is interesting to note that the average fitness appears to level out and increase slightly after 15 generations or so. It is believed that this is a result of the algorithm nearing the base of the conic landscape, and as such the mutations attempt to achieve a better solution. However, this instead leads to values falling outside the desired range and thereby gives higher fitness values which cannot be improved upon by crossover. *i.e* the algorithm appears to be placing too much too much on exploring the landscape at this than exploiting the knowledge it has gained, perhaps something which could be alleviated by reducing the mutation step of the operator.

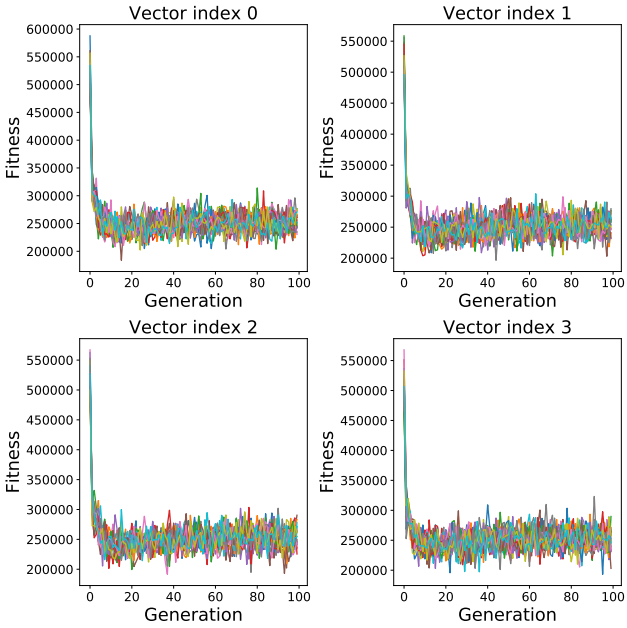

Figure 3: The average for each generation over 20 different runs of the MP-GSGP algorithm.

6.2 Comparative Methods

As the desired output values fall in the constrained range 1–7, and there are 4 potential values, we can work out the probability of achieving the correct result through random searches. For a player to win a set outright, they must score 6 or 7 with a difference of two from the other value in the set. This means the theoretical accuracy of a stochastic search with prior knowledge in order to predict the match outcome is $(\frac{1}{2} \cdot \frac{1}{4} + \frac{1}{2} \cdot \frac{1}{5}) \cdot \frac{1}{2} = 11.25\%$. However, this is assuming the players complete the sets fully without retiring early (through injury or fatigue), which is a possibility. Thus, in reality the accuracy would likely be marginally lower than this.

Given the accuracy rate seen in Table 1 by MP-GSGP is higher than this, it appears that MP-GSGP algorithm is an effective choice in combating the challenging problem of tennis score prediction. It is able to outperform both the standard genetic programming implementation and the hillclimber in terms of accuracy and the convergence of fitness towards the global optimum. This is partially due to the population's ability to more effectively exploit the prior knowledge it has acquired when developing the next generation of solutions.

However, whilst the results are higher than those of the comparative methods (as well as the theoretical accuracy from a stochastic prediction with prior knowledge), they are not considerably higher than these. This is likely due to the challenging problem domain. All methods are attempting to find a model to the problem, but given the complexities involved it may not be possible to find an extremely accurate solution through the given inputs. Nonetheless, as an experiment it does show MP-GSGP as a suitable method for uncovering a model fitting an unknown system.

The only measured aspect MP-GSGP lost out in was to MP-GP, in terms of the mean evaluation time. However, referring back to Table 1, this time is clearly within the same region as that of its counterpart. However, the time cost of such a method is something to consider. With each generation taking around 35 minutes to train as a result of the large computation time and number of inputs, over each of the output vector values, this is clearly an expensive method. Furthermore, outside of training the prediction of results also takes a significant time. For many situations this would be acceptable, however other techniques may prove more suitable for certain tasks. An example of this would be McConaghy's deterministic FFX algorithm, which performs symbolic regression without evolutionary techniques [11].

Figures 5 and 6 show the best fitness achieved by per generation for the MP-GP and GSHC algorithms respectively. The reader will note that GSHC has a maximum of 50,000 generations, which was chosen to give an equal number of fitness evaluations to both MP-GP and MP-GSGP. They indicate that GSHC is unable to find adept solutions to the problem without having a good staring point. This is a characteristic where MP-GSGP is more suited to, as it can take advantage of a good solution within the population and propagate its characteristics through the population via crossover.

A comparison of Figures 4 and 5 suggests that whilst GP is able to converge to a fitter solution over time, it gets stuck

Method	Mean Evaluation Time (min)	Mean Best Final Fitness	Best Achieved Fitness	Mean Test Dataset Accuracy
Hillclimber	44:36	801895	2191	9.26%
MP-GP	**35:11**	1670	1510	13.26%
MP-GSGP	35:32	**1547**	**1486**	**14.68%**

Table 1: The mean results achieved from the trials of the different approaches to solving the tennis score prediction problem.

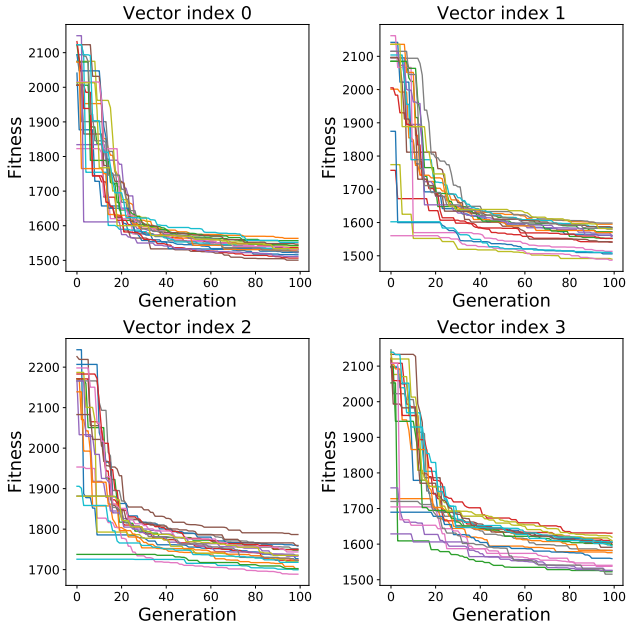

Figure 4: The best fitness which has been achieved at each generation over 20 different runs of the MP-GSGP algorithm.

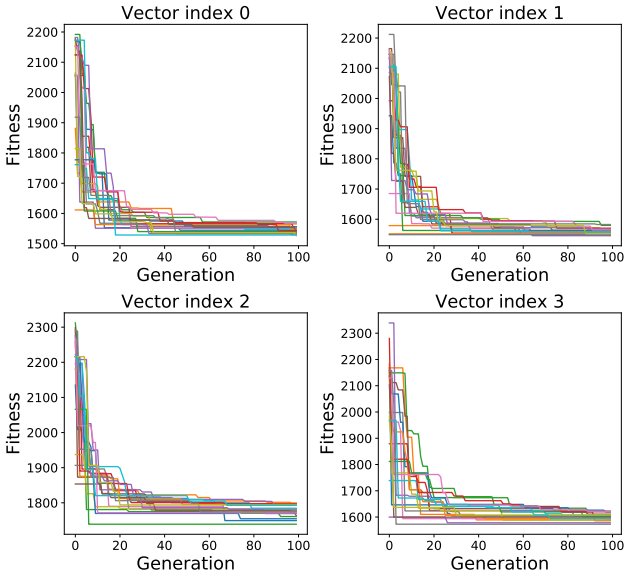

Figure 5: The best fitness which has been achieved at each generation over 20 different runs of the MP-GP algorithm.

in local optimums as a result of the multi-modal landscape in which it operates. Whilst MP-GSGP takes longer to converge with the chosen parameters, it is able to achieve higher fitness values by continually improving over time.

7. CONCLUSIONS AND FURTHER WORK

The method of GSGP is an effective evolutionary approach to symbolic regression. It is able to achieve better than stochastic results without being provided any prior knowledge about the problem, as well as outperforming other rival techniques over the majority of criteria assessed.

Despite its success, there are areas in which further works could improve the effectiveness of the implementation. A drawback to the algorithm is the size of solutions developed. Whilst in the given context it is likely an effective solution requires a complex model to be found, in other domains the most suitable solution may be one requiring only a few inputs to be compared and evaluated. Thus, a direction for further work may be to produce offspring of same or smaller size, whilst maintaining the geometric semantic characteristic and corresponding advantages.

Further to this, the algorithm here produced a number of solutions falling outside the desired range solutions which were penalised in the fitness function. Another advance would be to avoid these infeasible solutions in a manner which maintains the geometric semantic nature of results, without changing the nature of the problem. This may be possible through the use of a logistic function, or converting to a classification problem as briefly discussed earlier. However, time constraints inhibit the ability for these to be trialled and discussed here.

Another interesting area would be in the tuning of hyperparameters. Whilst the parameters in this paper were chosen as a result of trials, it would be interesting and beneficial to achieve these dynamically. A statistical approach to implementing an adaptive mutation step, for example, could help to explore more of the landscape at earlier stages of training before easing to allow the crossover to exploit more of the prior knowledge and attempt to reach the global optimum. This could potentially aid in reducing the aid in reducing the slight increase in fitness witnessed in the experiments. Furthermore, it may also be interesting to see the effect of a mutation or crossover only algorithm. Perhaps alternating the two over separate generations and looking closely at the results would help to further understand how GSGP works.

Nonetheless, these directions are not to discredit the suitability of GSGP but merely highlight areas which could be used to further understand the mechanisms involved, thus allowing for improvements in the effectiveness of the algorithms and thereby lead to it being utilised in more scenarios.

8. REFERENCES

[1] J. Albinati, G. L. Pappa, F. E. Otero, and L. O. V.

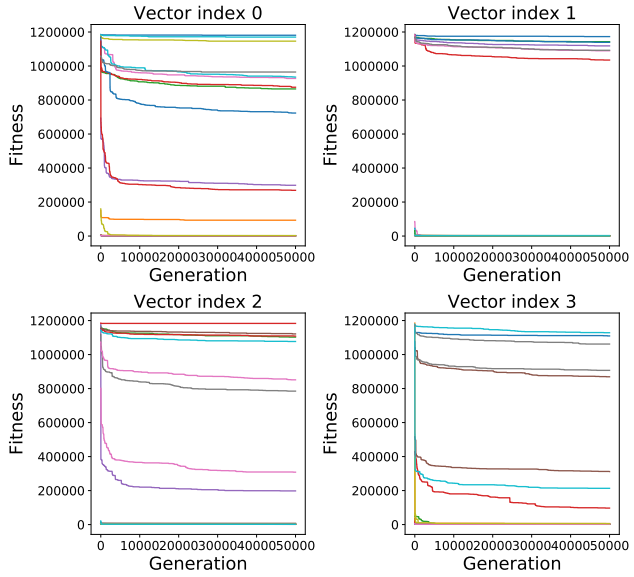

Figure 6: The best fitness which has been achieved at each generation over 20 different runs of the GSHC algorithm.

Oliveira. The effect of distinct geometric semantic crossover operators in regression problems. In *European Conference on Genetic Programming*, pages 3–15. Springer, 2015.

[2] D. A. Augusto, H. J. C. Barbosa, and N. F. F. Ebecken. Coevolutionary multi-population genetic programming for data classification. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 933–940, New York, NY, USA, 2010. ACM.

[3] M. Castelli, D. Castaldi, I. Giordani, S. Silva, L. Vanneschi, F. Archetti, and D. Maccagnola. An efficient implementation of geometric semantic genetic programming for anticoagulation level prediction in pharmacogenetics. In L. Correia, L. P. Reis, and J. Cascalho, editors, *Progress in Artificial Intelligence*, pages 78–89, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[4] M. Castelli, R. Henriques, and L. Vanneschi. A geometric semantic genetic programming system for the electoral redistricting problem. *Neurocomputing*, 154:200 – 207, 2015.

[5] R. Forsyth. BEAGLE a Darwinian approach to pattern recognition. *Kybernetes*, 10(3):159–166, 1981.

[6] E. Galván-López, B. Cody-Kenny, L. Trujillo, and A. Kattan. Using semantics in the selection mechanism in genetic programming: A simple method for promoting semantic diversity. In *IEEE Congress on Evolutionary Computation*, pages 2972–2979, June 2013.

[7] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[8] D. C. Hooper and N. S. Flann. Improving the accuracy and robustness of genetic programming through expression simplification. In *Proceedings of the 1st Annual Conference on Genetic Programming*, pages 428–428, Cambridge, MA, USA, 1996. MIT Press.

[9] J. R. Koza. Non-linear genetic algorithms for solving problems. United States Patent 4935877, 19 June 1990. filed may 20, 1988, issued june 19, 1990, 4,935,877. Australian patent 611,350 issued september 21, 1991. Canadian patent 1,311,561 issued december 15, 1992.

[10] J.-Y. Lin, H.-R. Ke, B.-C. Chien, and W.-P. Yang. Designing a classifier by a layered multi-population genetic programming approach. *Pattern Recognition*, 40(8):2211 – 2225, 2007. Part Special Issue on Visual Information Processing.

[11] T. McConaghy. *FFX: Fast, Scalable, Deterministic Symbolic Regression Technology*, pages 235–260. Springer New York, New York, NY, 2011.

[12] A. Moraglio, K. Krawiec, and C. G. Johnson. Geometric semantic genetic programming. In C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, editors, *Parallel Problem Solving from Nature - PPSN XII*, pages 21–31, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[13] U.-M. O'Reilly. *An analysis of genetic programming*. PhD thesis, Carleton University, 1995.

[14] T. P. Pawlak. Geometric semantic genetic programming is overkill. In M. I. Heywood, J. McDermott, M. Castelli, E. Costa, and K. Sim, editors, *Genetic Programming*, pages 246–260, Cham, 2016. Springer International Publishing.

[15] Tennis Betting. Tennis Results. http://www.tennis-data.co.uk/alldata.php, April 2019. Accessed 19-Apr-2019.

[16] A. M. Turing. Computing machinery and intelligence. *Mind*, 49:433–460, Jan. 01 1950.

[17] P. Wong and M. Zhang. Scheme: Caching subtrees in genetic programming. In *IEEE Congress on Evolutionary Computation*, pages 2678–2685. IEEE, 2008.

[18] Z. Zhu, A. K. Nandi, and M. W. Aslam. Adapted geometric semantic genetic programming for diabetes and breast cancer classification. In *IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–5, Sep. 2013.