CONCLUSIONS AND FUTURE WORK

This dissertation focuses on the selection stage in the evolution and the code bloat problem of GP. The overall goal was to improve GP performance by using semantic information. This goal was successfully achieved by developing a number of new methods. The dissertation has the following main contributions.

- Three semantic tournament selection are proposed, including TS-R, TS-S and TS-P. For further improvements, the best method, TS-S is combined with RDO, and the resulting method is called TS-RDO.
- A novel semantic approximation technique (SAT) is proposed. Besides, two other versions of SAT are also introduced.
- Two methods, SA and DA based on semantic approximation technique for reducing GP code bloat are proposed. Additionally, three other bloat control methods based on the variants of SAT, including SAT-GP, SAS-GP and PP-AT are introduced.

However, the dissertation is subject to some limitations. First, the proposed methods are based on the concepts of sampling semantics that is only defined for the problems in which the input and output are continuous real-valued vectors. Second, the dissertation lacks examining the distribution of GP error vectors to select an appropriate statistical hypothesis test. Third, two approaches for reducing GP code bloat, SA and DA add two more parameters (max depth of $sTree$ and the portion of GP population for pruning) to GP systems.

Building upon this research, there are a number of directions for future work arisen from the dissertation. Firstly, we will conduct research to reduce the above limitations of the dissertation. Secondly, we want to expand the use of statistical analysis in other phases of the GP algorithm, for example, in model selection [129]. Thirdly, SAT was used for lessening code bloat in GP. Nevertheless, this technique can also be used for designing new genetic operators to be similar to RDO [93]. Finally, in terms of applications, all proposed methods in the dissertation can be applied to any problem domain where the output is a single real-valued number. In the future, we will extend them to a wider range of real-world applications including classification and problems of bigger datasets to better understand their weakness and strength.

INTRODUCTION

Genetic Programming (GP) is considered as a machine learning method that allows computer programs encoded as a set of tree structures to be evolved using an evolutionary algorithm. A GP system is started by initializing a population of individuals. The population is then evolved for a number of generations using genetic operators such as crossover and mutation. At each generation, the individuals are evaluated using a fitness function, and a selection schema is used to choose better individuals to create the next population. The evolutionary process is continued until a desired solution is found or when the maximum number of generations is reached.

To enhance GP performance, the dissertation focuses on two main objectives, including improving selection performance and addressing code bloat phenomenon in GP. In order to achieve these objectives, several new methods are proposed in this research by incorporating semantics into GP evolutionary process. The main contributions of the dissertation are outlined as follows.

- Three new semantics based tournament selection methods are proposed. A novel comparison between individuals based on a statistical analysis of their semantics is introduced. From that, three variants of the selection strategy are proposed. These methods promote semantic diversity and reduce code bloat in GP.
- A semantic approximation technique is proposed. We propose a new technique that allows to grow a small (sub)tree with the semantics approximate to a given target semantics.
- New bloat control methods based on semantic approximation are introduced. Inspired by the semantic approximation technique, a number of methods for reducing GP code bloat are proposed and evaluated on a large set of regression problems and a real-world time series forecasting.

The dissertation is organised into three chapters except for introduction, conclusion, future work, bibliography and appendix. Chapter 1 gives the backgrounds related to this research. Chapter 2 presents the proposed forms of tournament selection, and Chapter 3 introduces a new proposed semantics approximation technique and several methods for reducing code bloat.

Chapter 1

BACKGROUNDS

1.1 Genetic Programming

Genetic Programming (GP) is an Evolutionary Algorithm (EA) that automatically finds the solutions of unknown structure for a problem[50,96]. It is also considered as a metaheuristic-based machine learning method which finds solutions in form of computer programs for a given problem through an evolutionary process. Technically, GP is considered as an evolutionary algorithm, so it shares a number of common characteristics with other EAs. Algorithm 1 presents the algorithm of GP.

Algorithm 1: GP algorithm

1. Randomly create an initial population of programs from the available primitives.

repeat

2. Execute each program and evaluate its fitness.
3. Select one or two program(s) from the population with a probability based on fitness to participate in genetic operators.
4. Create new individual program(s) by applying genetic operators with specified probabilities.

until *an acceptable solution is found or other stopping condition is met.*

return the best-so-far individual.

The first step in running a GP system is to create an initial population of computer programs. GP then finds out how well a program works by running it, and then comparing its behaviour to some objectives of the problem (step 2). Those programs that do well are chosen to breed (step 3) and produce new programs for the next generation (step 4). Generation by generation, GP transforms populations of programs into new, hopefully better, populations of programs by repeating steps 2-4 until a termination condition is met.

1.2 Semantics in GP

Semantics is a broad concept used in different research fields. In the context of GP, we are mostly interested in the behavior of the individuals (what they 'do'). To specify what individual behavior is, researchers have recently intro-

in the generalization ability. Moreover, the solution complexity of SA and DA is much simple than the solution complexity of RF that is often the combination of dozens or hundreds of trees.

3.6 Applying semantic methods for time series forecasting

The above analysis, we used the generalized version of SAT, in which $sTree$ is a small randomly generated tree. Besides, there are some variants of SAT that can be $sTree$ is a random terminal taken from the terminal set, and $sTree$ is a small tree taken from the pre-defined library. Based on that, we have proposed a new method called SAT-GP [C2] in which $sTree$ is a random terminal that taken from the terminal set, and a new other method, namely SAS-GP [C5] in which $sTree$ is a small tree taken from a pre-defined library of subprograms.

Moreover, the semantic approximation technique can be applied to other bloat control methods. We combine this semantic approximation technique with Prune and Plant operator [2] to create a new operator called PP-AT [C6]. PP-AT is an extension of Prune and Plant. PP-AT selects a random subtree and then replaces it with an approximate tree. The approximate tree is grown from a random terminal so that the semantics of it is the most similar to the semantics of the selected subtree. Moreover, this subtree is also grown in the population as a new another child.

For an extension, we applied the proposed semantic methods for reducing code bloat on a real-world time series forecasting problem taken from Kaggle competition with different GP parameter settings. However, due to the limited space, the results of this section is only summarized. The experimental results showed that TS-S and SAT-based methods usually achieved better performance in comparison to GP. For PP-AT, although it has not achieved good performance like TS-S and SAT-based methods, it has inherited the benefits and improved the performance of PP.

3.7 Conclusion

In this chapter, we proposed a new technique for generating a tree that is semantically similar to a target semantic vector. Based on that, we proposed two approaches for lessening GP code bloat. Besides, some other versions of SAT are introduced. From that, several other methods for reducing code bloat are proposed, including SAT-GP, SAS-GP and PP-AT. The results illustrated that all proposed bloat control methods based on semantics help GP system increase the performance and reducing code bloat.

than GP. The average running time of SA and DA are significantly smaller than that of GP on most tested problems. Comparing between various versions of SA and DA we can see that SA20, SAD, DA20 and DAD often run faster than SA10 and DA10.

Overall, the results in this section show that SA and DA improve the training error and the testing error compared to GP and the recent bloat control methods (PP and TS-S). Moreover, the solutions obtained by SA and DA are much simpler, and their average running time are much less than that of GP on most tested functions.

3.5 Comparing with Machine Learning Algorithms

This section compares the results of the proposed methods with four popular machine learning algorithms including Linear Regression (LR), Support Vector Regression (SVR), Decision Tree (DT) and Random Forest (RF).

The testing error of the proposed models and four machine learning systems are presented in Table 3.8. The experimental results show that our proposed

Table 3.8: Comparison of the testing error of GP and machine learning systems. The best results are underlined.

Pro	GP	SA10	SA20	SAD	DA10	DA20	DAD	LR	SVR	DT	RF
F1	1.69	1.28	1.05	1.44	0.80	1.68	1.95	1.85	1.64	1.50	1.45
F2	0.30	0.27	0.25	0.24	0.28	0.26	0.26	0.26	0.25	0.30	0.24
F3	10.17	4.41	5.44	5.44	4.38	4.67	5.68	6.61	5.37	7.59	5.83
F5	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.00
F6	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01
F9	0.31	0.06	0.73	3.44	0.01	0.01	1.40	5.18	5.17	4.44	5.24
F13	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.03
F15	2.19	2.18	2.18	2.18	2.20	2.18	2.18	2.17	2.17	2.23	2.18
F16	0.75	0.27	0.27	0.28	0.26	0.23	0.26	0.22	0.32	0.31	0.23
F17	0.61	0.60	0.57	0.58	0.59	0.57	0.57	0.60	0.64	0.70	0.54
F18	0.36	0.21	0.29	0.32	0.16	0.17	0.18	0.15	0.37	0.16	0.13
F22	0.28	0.18	0.61	0.76	0.21	0.34	0.52	0.76	1.14	0.14	0.15
F23	1.44	0.65	0.87	0.99	0.52	0.51	0.53	1.84	1.02	0.56	0.56
F24	2.69	2.42	2.10	2.04	2.31	2.08	1.97	1.83	2.47	2.53	2.04
F25	1.77	1.26	1.13	1.13	1.30	1.30	1.34	1.58	1.22	1.15	1.14
F26	1.04	1.02	1.02	1.02	1.03	1.02	1.02	1.31	1.02	3.35	1.67

methods are often better than three machine learning algorithms including LR, SVR and DT and they are as good as the best machine learning algorithm (RF)

duced several concepts related to semantics in GP [67,82,92,93] as following.

Let $p \in \mathbb{P}$ be a program from a set $\mathbb{P}$ of all possible programs. When applied to an input $in \in \mathbb{I}$, a program p produces certain output $p(in)$.

Definition 1.1. *A* ***semantic space*** *of a program set $\mathbb{P}$ is a set $\mathbb{S}$ such that a semantic mapping exists for it, i.e., there exists a function $s : \mathbb{P} \rightarrow \mathbb{S}$ that maps every program $p \in \mathbb{P}$ into its semantics $s(p) \in \mathbb{S}$ and has the following property:*

$$s(p_1) = s(p_2) \Leftrightarrow \forall in \in \mathbb{I} : p_1(in) = p_2(in)$$

Definition 1.1 indicates that each program in $\mathbb{P}$ has thus exactly one semantics, but two different programs can have the same semantics.

The semantic space $\mathbb{S}$ enumerates all behaviors of programs for all possible input data. That means semantics is complete in capturing the entire information on program behavior. In GP, semantics is typically contextualized within a specific programming task that is to be solved in a given program set $\mathbb{P}$. As a machine learning technique, GP evolves programs based on a finite training set of fitness cases [54,71,116]. Assuming that this set is the only available data that specifies the desired outcome of the sought program, naturally, an instance of the semantics of a program is the vector of outputs that the program produces for these fitness cases as Definition 1.2.

Definition 1.2. *Let $\mathbb{K} = \{k_1, k_2, ...k_n\}$ be the fitness cases of the problem. The semantics $s(p)$ of a program p is the vector of output values obtained by running p on all fitness cases.*

$$s(p) = (p(k_1), p(k_2), \ldots, p(k_n)),\ for\ i = 1, 2, \ldots, n.$$

In Definition 1.2, semantics may be viewed as a point in $n-$dimensional semantic space, where n is the number of fitness cases. The semantics of a program consists of a finite sample of outputs with respect to a set of training values. Hence, this definition is not a complete description of the behavior of programs, and it is also called *sampling semantics* [78,79]. Moreover, the definition is only valid for programs whose output is a single real-valued number, as in symbolic regression. However, this definition is widely accepted and extensively used for designing many new techniques in GP [54,67,73,79,82,93,110]. The studies in the dissertation use this definition.

Formally, a semantic distance between two points in a semantic space is defined as Definition 1.3.

Definition 1.3. *A **semantic distance** between two points in the semantic space $\mathbb{S}$ is any function:*

$$d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$$

that is non-negative, symmetric, and fulfills the properties of the identity of indiscernibles and triangle inequality.

Interestingly, the fitness function is some kind of distance measure. Thus, semantics can be computed every time a program is evaluated, and it is essentially free to obtain. Moreover, a part of tree program is also a program, so semantics can be calculated in (almost) every node of the tree.

Based on Definition 1.2, the error vector of an individual is calculated by comparing the semantic vector with the target output of the problem. More precisely, the error vector of an individual is defined as:

Definition 1.4. *Let $s(p) = (s_1, s_2, ...s_n)$ be the semantics of an individual p and $y = (y_1, y_2, ...y_n)$ be the target output of the problem on n fitness cases. The error vector $e(p)$ of a program p is a vector of n elements calculated as follows.*

$$e(p) = (|s_1 - y_1|, |s_2 - y_2|, \ldots, |s_n - y_n|)$$

Overall, semantics indicates the behavior of a program (individual) and can be represented by program outputs with all possible inputs.

1.3 Semantic Backpropagation

The semantic backpropagation algorithm was proposed by Krawiec and Pawlak [53,93] to determine the desired semantics for an intermediate node of an individual. The algorithm starts by assigning the target of the problem (the output of the training set) to the semantics of the root node and then propagates the semantic target backwards through the program tree. At each node, the algorithm calculates the desired semantics of the node so that when we replace the subtree at this node by a new tree that has semantics equally to the desired semantics, the semantics of entire individual will match the target semantics. Figure 1.8 illustrates the process of using the semantic backpropagation algorithm to calculate the desired semantics for the blue node N.

The semantic backpropagation technique is then used for designing several genetics operators in GP [53,93]. Among these, Random Desired Operator (RDO) is the most effective operator. A parent is selected by a selection

For SA and DA, 20% and dynamic configurations often achieve the simplest solutions.

Table 3.5: Average size of solutions

Pro	GP	RDO	PP	TS-S	SA10	SA20	SAD	DA10	DA20	DAD
F1	295.5	$\mathbf{167.7}^+$	$\mathbf{66.9}^+$	$\mathbf{135.0}^+$	$\mathbf{89.3}^+$	$\mathbf{19.9}^+$	$\mathbf{18.2}^+$	$\mathbf{79.4}^+$	$\mathbf{17.3}^+$	$\underline{\mathbf{13.3}}^+$
F2	171.0	$\mathbf{115.9}^+$	$\mathbf{28.3}^+$	$\mathbf{31.2}^+$	$\mathbf{69.8}^+$	$\underline{\mathbf{19.2}}^+$	$\mathbf{22.9}^+$	$\mathbf{53.3}^+$	$\underline{\mathbf{17.9}}^+$	$\mathbf{20.8}^+$
F3	228.3	$\mathbf{115.7}^+$	$\mathbf{44.8}^+$	$\mathbf{126.7}^+$	$\mathbf{82.8}^+$	$\mathbf{23.7}^+$	$\mathbf{16.5}^+$	$\mathbf{72.5}^+$	$\mathbf{26.8}^+$	$\underline{\mathbf{13.3}}^+$
F5	100.9	$\mathbf{43.7}^+$	$\mathbf{23.9}^+$	$\mathbf{62.4}^+$	$\mathbf{51.9}^+$	$\mathbf{15.0}^+$	$\underline{\mathbf{14.9}}^+$	$\mathbf{52.4}^+$	$\mathbf{21.1}^+$	$\underline{\mathbf{11.3}}^+$
F6	152.3	$\underline{\mathbf{12.6}}^+$	$\mathbf{33.1}^+$	$\mathbf{40.3}^+$	$\mathbf{81.7}^+$	$\mathbf{39.5}^+$	$\mathbf{31.8}^+$	$\mathbf{64.1}^+$	$\mathbf{36.9}^+$	$\mathbf{31.5}^+$
F9	187.3	$\mathbf{70.2}^+$	$\mathbf{19.4}^+$	$\mathbf{84.5}^+$	$\mathbf{67.2}^+$	$\mathbf{18.4}^+$	$\mathbf{13.4}^+$	$\mathbf{52.1}^+$	$\mathbf{13.1}^+$	$\underline{\mathbf{10.1}}^+$
F13	153.6	$\mathbf{57.4}^+$	$\mathbf{21.5}^+$	$\mathbf{46.2}^+$	$\mathbf{70.1}^+$	$\mathbf{23.0}^+$	$\underline{\mathbf{18.5}}^+$	$\mathbf{72.3}^+$	$\mathbf{18.6}^+$	$\mathbf{19.6}^+$
F15	237.8	$\mathbf{91.0}^+$	$\mathbf{30.4}^+$	$\mathbf{169.5}^+$	$\mathbf{80.3}^+$	$\mathbf{15.6}^+$	$\mathbf{12.0}^+$	$\mathbf{68.4}^+$	$\mathbf{19.2}^+$	$\underline{\mathbf{8.9}}^+$
F16	196.4	$\mathbf{148.4}^+$	$\mathbf{21.5}^+$	209.6	$\mathbf{52.6}^+$	$\underline{\mathbf{8.8}}^+$	$\mathbf{9.2}^+$	$\mathbf{63.8}^+$	$\mathbf{16.3}^+$	$\mathbf{12.8}^+$
F17	192	$\mathbf{140.7}^+$	$\mathbf{10.2}^+$	$\mathbf{72.3}^+$	$\mathbf{60.0}^+$	$\mathbf{9.6}^+$	$\underline{\mathbf{7.2}}^+$	$\mathbf{77.3}^+$	$\mathbf{17.4}^+$	$\mathbf{12.4}^+$
F18	151.7	164.6	$\mathbf{19.9}^+$	151.9	$\mathbf{55.0}^+$	$\mathbf{14.6}^+$	$\mathbf{13.4}^+$	$\mathbf{73.7}^+$	$\mathbf{21.9}^+$	$\underline{\mathbf{13.3}}^+$
F23	187.4	$\mathbf{156.3}$	$\mathbf{10.3}^+$	$\mathbf{48.1}^+$	$\mathbf{53.2}^+$	$\mathbf{10.3}^+$	$\underline{\mathbf{7.6}}^+$	$\mathbf{69.6}^+$	$\mathbf{16.3}^+$	$\mathbf{10.4}^+$
F24	192.6	$\mathbf{161.6}$	$\mathbf{10.0}^+$	$\mathbf{45.8}^+$	$\mathbf{61.6}^+$	$\mathbf{11.6}^+$	$\underline{\mathbf{7.9}}^+$	$\mathbf{76.6}^+$	$\mathbf{17.5}^+$	$\mathbf{15.2}^+$
F25	177.5	$\mathbf{141.6}^+$	$\mathbf{12.0}^+$	$\mathbf{49.4}^+$	$\mathbf{62.8}^+$	$\mathbf{9.0}^+$	$\underline{\mathbf{8.1}}^+$	$\mathbf{66.0}^+$	$\mathbf{19.2}^+$	$\mathbf{12.7}^+$
F26	177.2	$\mathbf{25.8}^+$	$\mathbf{14.2}^+$	$\mathbf{130.6}^+$	$\mathbf{16.1}^+$	$\underline{\mathbf{7.0}}^+$	$\underline{\mathbf{7.0}}^+$	$\mathbf{29.8}^+$	$\mathbf{11.1}^+$	$\mathbf{8.4}^+$

The last metric we examine in this section is the average running time of the

Table 3.6: Average running time in seconds

Pro	GP	RDO	PP	TS-S	SA10	SA20	SAD	DA10	DA20	DAD
F1	3.6	18.7^-	$\mathbf{1.1}^+$	$\mathbf{1.3}^+$	$\mathbf{1.0}^+$	$\mathbf{0.7}^+$	$\mathbf{0.8}^+$	$\mathbf{0.9}^+$	$\underline{\mathbf{0.4}}^+$	$\mathbf{1.0}^+$
F2	2.7	17.5^-	$\mathbf{1.1}^+$	$\mathbf{0.7}^+$	$\mathbf{1.4}^+$	$\mathbf{0.6}^+$	$\mathbf{0.7}^+$	$\mathbf{1.3}^+$	$\mathbf{0.8}^+$	$\underline{\mathbf{0.5}}^+$
F3	2.7	15.9^-	$\mathbf{0.9}^+$	$\mathbf{1.6}^+$	$\mathbf{1.0}^+$	$\mathbf{0.6}^+$	$\mathbf{1.1}^+$	$\mathbf{0.9}^+$	$\underline{\mathbf{0.4}}^+$	$\mathbf{0.8}^+$
F5	31.5	468.7^-	$\mathbf{6.9}^+$	$\mathbf{20.4}^+$	$\mathbf{16.4}^+$	$\mathbf{6.1}^+$	$\underline{\mathbf{3.1}}^+$	$\mathbf{20.5}^+$	$\mathbf{9.3}^+$	$\mathbf{8.6}^+$
F6	14.5	70.2^-	$\mathbf{3.2}^+$	$\underline{\mathbf{1.4}}^+$	$\mathbf{2.4}^+$	$\mathbf{2.1}^+$	$\mathbf{10.2}^+$	$\mathbf{2.1}^+$	$\mathbf{1.9}^+$	$\mathbf{2.7}^+$
F9	63.2	882.7^-	$\mathbf{15.0}^+$	$\mathbf{27.7}^+$	$\mathbf{16.6}^+$	$\underline{\mathbf{6.4}}^+$	$\mathbf{8.5}^+$	$\mathbf{18.5}^+$	$\mathbf{7.2}^+$	$\mathbf{10.5}^+$
F13	77.7	773.1^-	$\mathbf{19.4}^+$	$\mathbf{31.6}^+$	$\mathbf{27.4}^+$	$\mathbf{11.5}^+$	$\underline{\mathbf{8.5}}^+$	$\mathbf{32.2}^+$	$\mathbf{12.5}^+$	$\mathbf{11.5}^+$
F15	82.7	1232.6^-	$\mathbf{15.3}^+$	$\mathbf{61.7}^+$	$\mathbf{22.8}^+$	$\underline{\mathbf{7.1}}^+$	$\mathbf{7.4}^+$	$\mathbf{26.6}^+$	$\mathbf{9.1}^+$	$\mathbf{9.9}^+$
F16	46.0	629.8^-	$\mathbf{7.0}^+$	55.7	$\mathbf{7.2}^+$	$\underline{\mathbf{3.4}}^+$	$\mathbf{5.3}^+$	$\mathbf{15.3}^+$	$\mathbf{5.9}^+$	$\mathbf{7.0}^+$
F17	8.4	45.5^-	$\mathbf{2.6}^+$	$\mathbf{3.2}^+$	$\mathbf{2.9}^+$	$\underline{\mathbf{1.3}}^+$	$\mathbf{2.9}^+$	$\mathbf{5.6}^+$	$\mathbf{1.4}^+$	$\mathbf{3.7}^+$
F18	43.8	768.8^-	$\mathbf{10.2}^+$	$\mathbf{40.4}$	$\mathbf{12.9}^+$	$\underline{\mathbf{6.3}}^+$	$\mathbf{8.1}^+$	$\mathbf{19.1}^+$	$\mathbf{9.1}^+$	$\mathbf{11.5}^+$
F23	4.1	35.2^-	$\underline{\mathbf{0.6}}^+$	$\mathbf{0.9}^+$	$\mathbf{1.2}^+$	$\mathbf{0.8}^+$	$\mathbf{1.2}^+$	$\mathbf{2.9}^+$	$\mathbf{1.0}^+$	$\mathbf{2.2}^+$
F24	4.0	33.8^-	$\mathbf{0.6}^+$	$\mathbf{1.0}^+$	$\mathbf{1.3}^+$	$\mathbf{0.5}^+$	$\mathbf{1.1}^+$	$\mathbf{2.8}^+$	$\underline{\mathbf{0.4}}^+$	$\mathbf{0.8}^+$
F25	4.0	32.4^-	$\mathbf{0.6}^+$	$\mathbf{1.0}^+$	$\mathbf{1.3}^+$	$\mathbf{0.5}^+$	$\mathbf{1.2}^+$	$\mathbf{3.0}$	$\underline{\mathbf{0.5}}^+$	$\mathbf{0.9}^+$
F26	268.1	9334.5^-	$\mathbf{33.0}^+$	$\mathbf{237.0}$	$\underline{\mathbf{18.1}}^+$	$\mathbf{19.3}^+$	$\mathbf{30.8}^+$	$\mathbf{84.7}^+$	$\mathbf{20.0}^+$	$\mathbf{42.6}^+$

tested systems. It can be observed Table 3.6 that both SA and DA run faster

Table 3.2: Mean of the best fitness

Pro	GP	RDO	PP	TS-S	SA10	SA20	SAD	DA10	DA20	DAD
F1	0.47	**0.07**$^{+}$	1.60^{-}	0.97^{-}	0.52	0.89^{-}	1.30^{-}	**0.41**	0.97^{-}	1.17^{-}
F2	0.08	**0.02**$^{+}$	0.17^{-}	0.16^{-}	0.09	0.16^{-}	0.19^{-}	0.09	0.15^{-}	0.17^{-}
F3	1.91	**0.06**$^{+}$	4.45^{-}	**1.79**	**1.08**$^{+}$	2.33	4.12^{-}	**0.96**$^{+}$	2.2	3.58^{-}
F5	0.01	**0.01**	0.01^{-}	**0.01**	**0.01**$^{+}$	0.01^{-}	0.01^{-}	**0.01**	0.01	0.01
F6	0.12	**0.00**$^{+}$	0.23^{-}	0.26^{-}	**0.09**	**0.07**$^{+}$	**0.06**$^{+}$	**0.05**$^{+}$	**0.03**$^{+}$	**0.01**$^{+}$
F9	0.51	**0.05**$^{+}$	1.26^{-}	0.91^{-}	**0.06**$^{+}$	0.83	1.88^{-}	**0.13**$^{+}$	**0.37**	1.04^{-}
F13	0.03	**0.03**	**0.03**$^{+}$	0.04^{-}	**0.03**	**0.03**$^{+}$	**0.03**$^{+}$	**0.03**$^{+}$	**0.03**$^{+}$	**0.03**$^{+}$
F15	0.38	**0.32**	0.51^{-}	**0.37**	**0.35**	0.48^{-}	0.49^{-}	**0.35**	0.46^{-}	0.48^{-}
F16	0.41	**0.11**$^{+}$	1.03^{-}	**0.40**	**0.17**$^{+}$	**0.22**$^{+}$	**0.22**$^{+}$	**0.14**$^{+}$	**0.17**$^{+}$	**0.18**$^{+}$
F17	0.47	**0.39**$^{+}$	0.52^{-}	0.51^{-}	0.48^{-}	0.52^{-}	0.53^{-}	**0.46**	0.50^{-}	0.51^{-}
F18	0.4	**0.13**$^{+}$	1.32^{-}	0.42	**0.19**$^{+}$	**0.27**$^{+}$	**0.30**	**0.15**$^{+}$	**0.16**$^{+}$	**0.17**$^{+}$
F23	0.82	**0.22**$^{+}$	1.20^{-}	0.94	**0.65**$^{+}$	0.87	0.98^{-}	**0.45**$^{+}$	**0.52**$^{+}$	**0.57**$^{+}$
F24	1.68	**0.88**$^{+}$	2.05^{-}	1.93^{-}	1.7	1.99^{-}	2.05^{-}	**1.51**$^{+}$	1.83^{-}	1.93^{-}
F25	0.91	**0.56**$^{+}$	1.19^{-}	1.13^{-}	**0.90**	1.11^{-}	1.11^{-}	**0.84**$^{+}$	1.01^{-}	1.04^{-}
F26	1.51	1.51	1.53^{-}	**1.50**$^{+}$	1.52	1.53^{-}	1.53^{-}	**1.51**	1.52^{-}	1.52^{-}

Table 3.4: Median of testing error

Pro	GP	RDO	PP	TS-S	SA10	SA20	SAD	DA10	DA20	DAD
F1	1.69	3.16^{-}	1.76	**1.35**$^{+}$	**1.28**$^{+}$	**1.05**$^{+}$	**1.44**$^{+}$	**0.80**$^{+}$	**1.68**$^{+}$	1.95
F2	0.30	0.36^{-}	**0.25**$^{+}$	**0.26**$^{+}$	**0.27**$^{+}$	**0.25**$^{+}$	**0.24**$^{+}$	**0.28**	**0.26**$^{+}$	**0.26**$^{+}$
F3	10.17	**1.92**$^{+}$	**8.00**	**6.66**	**4.41**$^{+}$	**5.44**$^{+}$	**5.44**$^{+}$	**4.38**$^{+}$	**4.67**$^{+}$	**5.68**$^{+}$
F5	0.01	0.01	0.01	**0.01**$^{+}$	**0.01**$^{+}$	0.01	0.01	**0.01**$^{+}$	0.01	0.01
F6	0.01	**0.00**$^{+}$	**0.01**	0.01	**0.00**	**0.00**$^{+}$	**0.00**$^{+}$	**0.00**$^{+}$	**0.00**$^{+}$	**0.00**$^{+}$
F9	0.31	**0.01**$^{+}$	2.18	0.33	**0.06**$^{+}$	0.73	3.44^{-}	**0.01**$^{+}$	**0.01**$^{+}$	1.40
F13	0.03	**0.03**	**0.03**$^{+}$	**0.03**	**0.03**	**0.03**$^{+}$	**0.03**$^{+}$	**0.03**$^{+}$	**0.03**$^{+}$	**0.03**$^{+}$
F15	2.19	**2.18**	**2.18**	2.19	**2.18**	**2.18**$^{+}$	**2.18**$^{+}$	2.2	**2.18**	**2.18**$^{+}$
F16	0.75	**0.29**$^{+}$	1.28	0.83	**0.27**$^{+}$	**0.27**$^{+}$	**0.28**$^{+}$	**0.26**$^{+}$	**0.23**$^{+}$	**0.26**$^{+}$
F17	0.61	0.66^{-}	**0.57**$^{+}$	**0.58**$^{+}$	**0.60**	**0.57**$^{+}$	**0.58**$^{+}$	**0.59**	**0.57**$^{+}$	**0.57**$^{+}$
F18	0.36	**0.14**$^{+}$	1.60^{-}	0.45	**0.21**$^{+}$	**0.29**$^{+}$	**0.32**$^{+}$	**0.16**$^{+}$	**0.17**$^{+}$	**0.18**$^{+}$
F23	1.44	**1.19**	**1.30**	**1.14**$^{+}$	**0.65**$^{+}$	**0.87**$^{+}$	**0.99**$^{+}$	**0.52**$^{+}$	**0.51**$^{+}$	**0.53**$^{+}$
F24	2.69	9.69^{-}	**2.14**$^{+}$	**2.41**	**2.42**	**2.10**$^{+}$	**2.04**$^{+}$	**2.31**	**2.08**$^{+}$	**1.97**$^{+}$
F25	1.77	3.91	**1.21**$^{+}$	**1.34**$^{+}$	**1.26**$^{+}$	**1.13**$^{+}$	**1.13**$^{+}$	**1.30**$^{+}$	**1.30**$^{+}$	**1.34**$^{+}$
F26	1.04	**1.03**	**1.02**$^{+}$	**1.03**	**1.02**$^{+}$	**1.02**$^{+}$	**1.02**$^{+}$	**1.03**	**1.02**$^{+}$	**1.02**$^{+}$

The main objective for performing bloat control is to reduce the complexity of the solutions. To validate if SA and DA achieve this objective, we recorded the size of the final solution and presented it in Table 3.5. The table shows that all tested methods achieve the goal to find simpler solutions compared to GP.

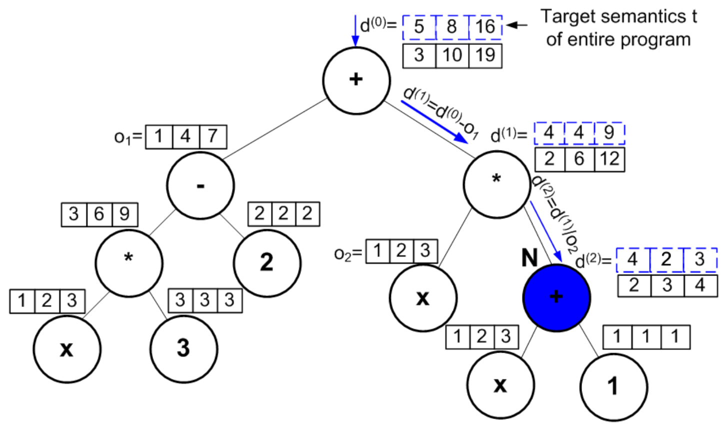

Figure 1.8: An example of calculating the desired semantics.

scheme, and a random subtree *subTree* is chosen. The semantic backpropagation algorithm is used to identify the desired semantics of *subTree*. After that, a procedure is called to search in a pre-defined library of trees for a *newTree* that is the most semantically closest to the desired semantics. Finally, *newTree* is replaced for *subTree* to create a new individual.

1.4 Statistical Hypothesis Test

In Chapter 2, we propose the use of a statistic hypothesis test to analyse the error vectors of individuals competing in a tournament. Practically, a Wilcoxon signed-rank test is employed the experiments. The Wilcoxon signed-rank test is a non-parametric statistical hypothesis test used when comparing two related samples to assess whether their population mean ranks differ [43]. In practice, the *p_ value* is often calculated from the test. The *p_ value* is defined as the probability of obtaining a result equal to or more extreme than what was actually observed, when the null hypothesis is true [100].

1.5 Conclusion

Chapter 1 presents the knowledge directly related to the research of the dissertation. Firstly, a more detailed GP introduction has been given. Next, some foundation concepts of the dissertation including the semantics of a program, semantic distance and vector errors of the program are also presented. Then, the semantic backpropagation algorithm is introduced. Finally, the chapter presents a statistical hypothesis test.

Chapter 2

TOURNAMENT SELECTION USING SEMANTICS

2.1 Introduction

Selection mechanism plays a very important role in GP performance. Among several selection techniques, tournament selection is often considered the most popular. Standard tournament selection randomly selects a set of individuals from the population, and the individual with the best fitness value is chosen as the winner. However, an opportunity exists to enhance tournament selection as the standard approach ignores finer-grained semantics which can be collected during GP program execution. In this chapter, we introduce the use of a statistical test into GP tournament selection that utilizes information from the individual's error vector, and three variants of the selection strategy are proposed.

2.2 Tournament Selection based on Semantics

This section presents three statistics tournament selection techniques using a statistical hypothesis test.

Algorithm 3: Statistics Tournament Selection with Random

Input: Tournament size: $TourSize$, Critical value: $alpha$.
Output: The winner individual.
$A \longleftarrow RandomIndividual()$;
for $i \leftarrow 1$ **to** $TourSize$ **do**
    $B \longleftarrow RandomIndividual()$;
    $sample1 \longleftarrow Error(A)$;
    $sample2 \longleftarrow Error(B)$;
    $p\_value \longleftarrow Testing(sample1, sample2)$;
    if $p\_value < alpha$ **then**
        $A \longleftarrow GetBetterFitness(A, B)$;
    else
        $A \longleftarrow GetRandom(A, B)$;
    end
end
The winner individual $\longleftarrow$A ;

Table 3.1: Evolutionary parameter values

Parameters	Value
Population size	500
Generations	100
Tournament size	3
Crossover, mutation probability	0.9; 0.1
Function set	$+, -, *, /, sin, cos$
Terminal set	$X_1, X_2, ..., X_n$
Initial Max depth	6
Max depth	17
Max depth of mutation tree	15
Raw fitness	root mean squared error on all fitness cases
Trials per treatment	30 independent runs for each value
Elitism	Copy the best individual to the next generation.

and DA (referred to as DAD) was also tested in which the individuals with the size greater than the average of the population are selected for pruning. The $newTree$ was grown from $sTree$ with the max depth of 2. We used a pre-defined library of 1000 subprograms with max depth of 2. Wilcoxon signed rank test with the confidence level of 95% is used across all the result tables in this chapter. All symbols in the result tables are set the same Chapter 2.

3.4 Performance Analysis

This section analyses the performance of the proposed methods using four popular metrics: training error, testing error, solution size and running time.

Training error is first analyzed in this section. The mean of the best fitness values in the training process across 30 runs is presented in Table 3.2. This table shows that the training error of SA and DA is often better than that of GP, PP and TS-S, especially with configurations 10%. This result is very impressive since the previous researches showed that bloat control methods often negatively affect the ability of GP to fit the training data.

The second metric is the generalization ability of the tested methods through comparing their testing error. The median of these values was calculated and shown in Table 3.4. The table shows that SA and DA outperform GP on the unseen data, especially 20% and dynamic configurations. Perhaps, the reason for the convincing result of them on the testing data is that these techniques obtain smaller fitness and simple solutions (Table 3.5) than the other methods.

lessen GP code bloat and enhance its ability to fit the training data. This technique is called *Desired Approximation* (DA). Algorithm 7 describes DA.

Algorithm 7: Desired Approximation

Input: Population size: N, Number of pruning: $k\%$.
Output: a solution of the problem.
$i \longleftarrow 0$;
$\mathbb{P}_0 \longleftarrow InitializePopulation()$;
Estimate fitness of all individuals in $\mathbb{P}_0$;
repeat
  $i \longleftarrow i + 1$;
  $\mathbb{P}'_i \longleftarrow GenerateNextPop(\mathbb{P}_{i-1})$;
  $pool \longleftarrow$ get $k\%$ of the largest individuals of $\mathbb{P}'_i$;
  $\mathbb{P}_i \longleftarrow \mathbb{P}'_i - pool$;
  foreach $I' \in pool$ **do**
    $subTree \longleftarrow RandomSubtree(I')$;
    $D \longleftarrow DesiredSemantics(subTree)$;
    $newTree \longleftarrow SemanticApproximation(D)$;
    $I \longleftarrow Substitute(I', subTree, newTree)$;
    $\mathbb{P}_i \longleftarrow \mathbb{P}_i \cup I$;
  Estimate fitness of all individuals in $\mathbb{P}_i$;
until *Termination condition met*;
return the best-so-far individual;

The structure of Algorithm 7 is very similar to that of SA. The main difference is in the second loop. First, the desired semantics of *subTree* is calculated by using the semantic backpropagation algorithm instead of the semantics of *subTree*. Second, *newTree* is grown to approximate the desired semantics D of *subTree* instead of its semantics S.

3.3 Experimental Settings

We tested SA and DA on twenty-six regression problems with the same dataset of Chapter 2 (Table 2.1). The GP parameters used in our experiments are shown in Table 3.1. The raw fitness is the root mean squared error. For each problem and each parameter setting, 30 runs were performed.

We compared SA and DA with standard GP (referred to as GP), Prune and Plant (PP) [2], TS-S and RDO [93]. The probability of PP operator was set to 0.5. For SA and DA, 10% and 20% of the largest individuals in the population were selected for pruning. The corresponding versions were shorted as SA10, SA20, DA10 and DA20. Moreover, a dynamic version of SA (shortened as SAD)

The first proposed method is called *Statistics Tournament Selection with Random* and shortened as TS-R. The main objective of TS-R is to promote the semantic diversity of GP population compared to standard tournament selection. Algorithm 3 presents the detailed description of TS-R. The process of TS-R is similar to standard tournament selection. However, instead of using the fitness value for comparing, a statistical test is applied to the error vector of these individuals. For a pair of individuals, if the test shows that the individuals are different, then the individual with better fitness value is considered as the winner. Conversely, if the test confirms that two individuals are not different, a random individual is selected from the pair. After that, the winner is tested against other individuals in the tournament size.

The second proposed tournament selection is called *Statistics Tournament Selection with Size* and shortened as TS-S. TS-S is similar to TS-R in the objective of promoting diversity. Moreover, TS-S also aims at reducing the code growth in GP population. In TS-S, if two individuals involved in the test are not statistically different, then the smaller individual will be the winner.

Algorithm 4: Statistics Tournament Selection with Size

Input: Tournament size: $TourSize$, Critical value: $alpha$.
Output: The winner individual.
$A \longleftarrow RandomIndividual()$;
for $i \leftarrow 1$ **to** $TourSize$ **do**
  $B \longleftarrow RandomIndividual()$;
  $sample1 \longleftarrow Error(A)$;
  $sample2 \longleftarrow Error(B)$;
  $p\_value \longleftarrow Testing(sample1, sample2)$;
  if $p\_value < alpha$ **then**
    $A \longleftarrow GetBetterFitness(A, B)$;
  else
    $A \longleftarrow GetSmallerSize(A, B)$;
  end
end
The winner individual $\longleftarrow$A ;

The third tournament selection method is called *Statistics Tournament Selection with Probability* and shorted as TS-P. Algorithm 5 presents the algorithm of TS-P. This technique is different from TS-R and TS-S in which it does not rely on the critical value to decide the winner. Instead, TS-P uses the *p_value* as the probability to select the winner. In other words, the better fitness individual is selected with the probability of $1- p\_value$ while the worse fitness individual has the probability of p_value to be selected.

Algorithm 5: Statistics Tournament Selection with Probability

```
Input: Tournament size: TourSize.
Output: The winner individual.
A ⟵ RandomIndividual();
for i ← 1 to TourSize do
    B ⟵ RandomIndividual();
    sample1 ⟵ Error(A);
    sample2 ⟵ Error(B);
    p_value ⟵ Testing(sample1, sample2);
    A ⟵ GetBetterWithProbability(A, B, p_value);
end
The winner individual ⟵A ;
```

2.3 Experimental Settings

In order to evaluate the proposed methods, we tested them on a large number of problems including twenty-six regression problems and the noisy version of them. The detailed description is presented in Table 2.1[1]. The GP parameters used for our experiments are shown in Table 2.2. The raw fitness is the mean of absolute errors on all fitness cases. In all experiment, three popular values of tournament size (referred to as tour-size hereafter) including 3, 5 and 7 were tested[2]. The critical value in Wilcoxon test is set to 0.05. For each problem and each parameter setting, 100 runs were performed.

We employ the Friedman's test and a post-hoc analysis on the results in all result tables in the following sections. In the tables, if the result of a method is significantly better than GP with standard tournament selection (GP), this result is marked + at the end. Conversely, if it is significantly worse compared to GP, this result is marked – at the end. Additionally, if it is the best (lowest) value, it is printed underline, and if the result of a method is better than that of GP, it is printed bold face.

2.4 Results and Discussions

We divided our experiment into three sets. The first set aims at investigating the performance of three variants of semantic tournament selection based on

[1]Since the space limitation, we only show the results of 15 problems in this summary,
[2]and only show in this chapter the results with tour-size=3 and tour-size=7.

as close to s as possible. Let $q = (q_1, q_2, ...q_n)$ be the semantics of $sTree$, then the semantics of $newTree$ is $p = (\theta \cdot q_1, \theta \cdot q_2, ..., \theta \cdot q_n)$. To approximate s, we need to find θ so that the squared Euclidean distance between two vectors s and p is minimal. In other words, we need to minimize function $f(\theta) = \sum_{i=1}^{n}(\theta \cdot q_i - s_i)^2$ with respect to θ. The quadratic function $f(\theta)$ achieves the minimal value at θ^* calculated in Equation 3.1:

$$\theta^* = \frac{\sum_{i=1}^{n} q_i s_i}{\sum_{i=1}^{n} q_i^2} \tag{3.1}$$

After finding θ^*, $newTree = \theta^* \cdot sTree$ is grown, and this tree is called the approximate tree of the semantic vector s.

Based on SAT, we continuously propose two techniques for reducing code bloat in GP. The first technique is called *Subtree Approximation* (SA). After generating the next population, k% largest individuals in the population are selected for pruning. Next, for each selected individual, a random subtree is chosen and replaced by an approximate tree of smaller size. The approximate tree is grown so that the semantics of it is the most similar to the semantics of the selected subtree. Algorithm 6 presents this technique in detail.

Algorithm 6: Subtree Approximation

```
Input: Population size: N, Number of pruning: k%.
Output: a solution of the problem.
i ⟵ 0;
ℙ0 ⟵ InitializePopulation();
Estimate fitness of all individuals in ℙ0;
repeat
    i ⟵ i + 1;
    ℙ'i ⟵ GenerateNextPop(ℙi−1);
    pool ⟵ get k% of the largest individuals of ℙ'i;
    ℙi ⟵ ℙ'i − pool;
    foreach I' ∈ pool do
        subTree ⟵ RandomSubtree(I');
        S ⟵ Semantics(subTree)
        newTree ⟵ SemanticApproximation(S);
        I ⟵ Substitute(I', subTree, newTree);
        ℙi ⟵ ℙi ∪ I;
    Estimate fitness of all individuals in ℙi;
until Termination condition met;
return the best-so-far individual;
```

The second technique attempts to achieve two objectives simultaneously:

Chapter 3

SEMANTIC APPROXIMATION FOR REDUCING CODE BLOAT

3.1 Introduction

Code bloat is a phenomenon in Genetic Programming (GP) characterized by the increase in individual size during the evolutionary process without a corresponding improvement in fitness. Bloat negatively affects GP performance, since large individuals are more time consuming to evaluate and harder to interpret. This chapter introduces a semantic approximation technique that allows to grow a (sub)tree being semantically approximate to a given target semantics. Based on that, two approaches for reducing GP code bloat are introduced. The bloat control methods are tested on a large set of regression problems and a real-world time series forecasting. Experimental results show that these methods improve GP performance and specifically reduce code bloat.

3.2 Methods

This section introduces a novel proposed approach to grow for a tree of approximate semantics to the target semantics. This approach is called the *Semantic Approximation Technique* (SAT).

Let $s = (s_1, s_2, ..., s_n)$ be the target semantics, then the objective of SAT is to grow a tree in the form: $newTree = \theta \cdot sTree$ so that the semantics of $newTree$ is

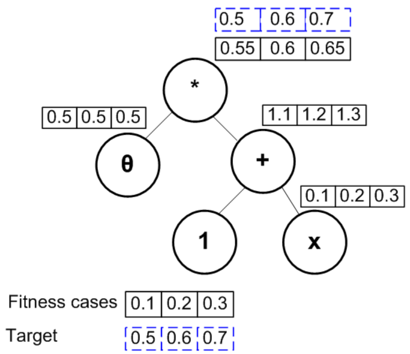

Figure 3.1: An example of Semantic Approximation

statistical analysis in comparison with standard tournament selection. The second set attempts to improve the performance of the semantic selection strategy through its combination with a state of the art semantic crossover operator [93]. The third set of experiments examines the performance of the strategies on noisy instances of the problems.

Table 2.1: Problems for testing statistics tournament selection techniques

Abbreviation	Name	features	Training	Testing
A. **Benchmarking Problems**				
F1	korns-11	5	20	20
F2	korns-12	5	20	20
F3	korns-14	5	20	20
F4	vladislavleva-2	1	100	221
F5	vladislavleva-4	5	500	500
F6	vladislavleva-6	2	30	93636
F7	vladislavleva-8	2	50	1089
F8	korns-1	5	1000	1000
F9	korns-2	5	1000	1000
F10	korns-3	5	1000	1000
F11	korns-4	5	1000	1000
F12	korns-11	5	1000	1000
F13	korns-12	5	1000	1000
F14	korns-14	5	1000	1000
F15	korns-15	5	1000	1000
B. **UCI Problems**				
F16	airfoil_self_noise	5	800	703
F17	casp	9	100	100
F18	ccpp	4	1000	1000
F19	wpbc	31	100	98
F20	3D_spatial_network	3	750	750
F21	protein_Tertiary_Structure	9	1000	1000
F22	yacht_hydrodynamics	6	160	148
F23	slump_test_Compressive	7	50	53
F24	slump_test_FLOW	7	50	53
F25	slump_test_SLUMP	7	50	53
F26	Appliances_energy_prediction	26	5000	9235

2.4.1 Performance Analysis of Statistics Tournament Selection

This subsection analyses the performance of three statistics tournament selection methods and compares them with GP and semantics in selection (SiS)

Table 2.2: Evolutionary Parameter Values.

Parameters	Value
Population size	500
Generations	100
Tournament size	3, 5, 7
Crossover, mutation probability	0.9; 0.1
Function set	$+, -, *, /, sin, cos$
Terminal set	$X_1, X_2, ..., X_n$
Initial Max depth	6
Max depth	17
Max depth of mutation tree	15
Raw fitness	mean absolute error on all fitness cases
Trials per treatment	100 independent runs for each value
Elitism	Copy the best individual to the next generation.

by Galvan-Lopez et al [29].

The first metric is the mean best fitness values on the training data and presented in Table 2.3. This table shows that three new selection methods did not help to improve the performance of GP on the training data. By contrast, the training error of standard tournament selection is often significantly better than that of statistics tournament selections. This result is not very surprising

Table 2.3: Mean of best fitness with tour-size=3 (left) and 7 (right)

Pro	GP	SiS	TS-R	TS-S	TS-P	GP	SiS	TS-R	TS-S	TS-P
F1	2.01	**1.91**	2.74$^-$	2.98$^-$	2.70$^-$	1.46	1.50	2.29$^-$	3.13$^-$	2.29$^-$
F2	0.24	**0.24**	0.39$^-$	0.56$^-$	0.31$^-$	0.23	**0.22**	0.35$^-$	0.55$^-$	0.26$^-$
F3	5.19	**4.94**	6.62$^-$	6.36$^-$	6.15$^-$	4.62	**3.62**	5.66$^-$	6.29$^-$	4.93$^-$
F5	0.126	0.133$^-$	0.130	**0.126**	0.127	0.124	0.126	0.129	0.127	**0.123**
F6	0.44	0.46	0.76$^-$	0.99$^-$	0.59$^-$	0.33	**0.31**	0.62$^-$	1.09$^-$	0.48$^-$
F9	1.48	1.50$^-$	1.98	1.96	**1.32**	1.62	1.90	**1.42**$^+$	2.30$^-$	1.66
F13	0.87	0.87	0.89$^-$	0.88$^-$	0.88$^-$	0.88	0.89	0.89$^-$	0.89$^-$	**0.88**$^+$
F15	2.55	2.85	2.64	**2.45**	2.61	2.17	2.33	2.23	2.29	2.37
F16	9.74	**9.13**	10.19	9.83	10.39	8.04	8.15	8.77	8.40	8.65
F17	3.69	3.75	4.05$^-$	4.11$^-$	3.97$^-$	3.39	3.46	3.89$^-$	4.11$^-$	3.82$^-$
F18	10.62	11.51	11.61	11.43	12.04	9.72	**9.07**	11.05	**9.41**	10.06
F23	4.24	4.36	5.35$^-$	4.66	5.01$^-$	3.47	**3.47**	4.58$^-$	7.22$^-$	4.18$^-$
F24	8.99	9.18	10.73$^-$	10.91$^-$	10.35$^-$	8.08	**8.05**	10.22$^-$	12.14$^-$	9.47$^-$
F25	4.98	5.00	6.18$^-$	6.69$^-$	5.86$^-$	4.47	**4.44**	5.79$^-$	7.18$^-$	5.40$^-$
F26	52.00	52.14	52.10	52.18$^-$	52.07	51.77	51.84	51.97	52.09$^-$	51.94

Table 2.11: Average running time with tour-size=3 (left) and 7 (right)

Pro	GP	neatGP	TS-S	RDO	TS-RDO	GP	neatGP	TS-S	RDO	TS-RDO
F1	4	863$^-$	**1**$^+$	32$^-$	10$^-$	3	863$^-$	**2**$^+$	34$^-$	9$^-$
F2	3	501$^-$	**1**$^+$	29$^-$	10$^-$	2	501$^-$	**1**$^+$	32$^-$	10$^-$
F3	4	831$^-$	**1**$^+$	29$^-$	11$^-$	2	831$^-$	**2**	32$^-$	11$^-$
F5	20	177$^-$	**18**	690$^-$	556$^-$	24	177$^-$	25	627$^-$	595$^-$
F6	3	522$^-$	**1**$^+$	104$^-$	85$^-$	17	522$^-$	**2**$^+$	142$^-$	86$^-$
F9	49	584$^-$	**36**	1762$^-$	1430$^-$	53	584$^-$	71	1755$^-$	1556$^-$
F13	69	396$^-$	**36**$^+$	1697$^-$	1235$^-$	69	396$^-$	**61**	1530$^-$	1259$^-$
F15	54	672$^-$	**48**	1478$^-$	1228$^-$	50	672$^-$	74$^-$	1390$^-$	1255$^-$
F16	27	1081$^-$	32	1195$^-$	1050$^-$	39	1081$^-$	58	1602$^-$	1072$^-$
F17	9	398$^-$	**4**$^+$	137$^-$	65$^-$	8	398$^-$	**6**	192$^-$	67$^-$
F18	36	821$^-$	37	1782$^-$	1282$^-$	40	821$^-$	62$^-$	2293$^-$	1406$^-$
F23	5	365$^-$	**2**$^+$	68$^-$	31$^-$	3	365$^-$	**2**	81$^-$	31$^-$
F24	4	367$^-$	**1**$^+$	53$^-$	30$^-$	3	367$^-$	**2**	103$^-$	28$^-$
F25	4	395$^-$	**1**$^+$	53$^-$	27$^-$	4	395$^-$	**2**	78$^-$	25$^-$
F26	522	**408**	**484**	40 389$^-$	35 838$^-$	474	**408**	614$^-$	39 577$^-$	41 971$^-$

The last experimental result analysed in this chapter is the average running time of the five methods. Apparently, TS-S is often the fastest system among all tested methods, especially with tour-size=3. This is not surprising since the code growth of TS-S's population is much less than GP. For TS-RDO, although it is slower than GP, its execution time has been considerably reduced compared to RDO. Besides, the time complexity of the statistics tournament selection methods is $T(n) = O(k.n.log(n))$, consequently, the selection step of them is slower than that of GP. It is possible to further reduce the computational time of TS-S by conducting the statistical test on only a subset of the fitness cases.

2.5 Conclusion

In this chapter, we proposed three variations of tournament selection that employ statistical analysis of these semantic vectors to select the winner for the mating pool. The proposed techniques aim at enhancing the semantic diversity and reducing the code bloat in GP population. In the experimental results, we observed that the proposed techniques especially TS-S was better than standard tournament selection and neatGP in improving GP generalisation and reducing GP code growth. The combined method, TS-RDO improves GP performance compared with TS-S and RDO. Additionally, these proposed methods have a good ability to perform well on noisy problems.

ing error is mostly achieved by TS-RDO on all problems with both values of the tournament size. For TS-S, the performance of it is also robust and more consistent than on the noiseless data.

Table 2.9: Average of solutions size with tour-size=3 (left) and 7 (right)

Pro	GP	neatGP	TS-S	RDO	TS-RDO	GP	neatGP	TS-S	RDO	TS-RDO
F1	280	**124**$^+$	**121**$^+$	**238**$^+$	**78**$^+$	286	**124**$^+$	**100**$^+$	**219**$^+$	**56**$^+$
F2	169	**60**$^+$	**35**$^+$	174$^-$	**80**$^+$	160	**60**$^+$	**37**$^+$	167$^-$	**47**$^+$
F3	263	**112**$^+$	**124**$^+$	**153**$^+$	**59**$^+$	262	**112**$^+$	**98**$^+$	**169**$^+$	**48**$^+$
F5	89	**12**$^+$	**53**$^+$	**49**$^+$	**23**$^+$	91	**12**$^+$	**42**$^+$	**46**$^+$	**13**$^+$
F6	167	**45**$^+$	**50**$^+$	**40**$^+$	**20**$^+$	137	**45**$^+$	**37**$^+$	**50**$^+$	**18**$^+$
F9	166	**62**$^+$	**73**$^+$	**53**$^+$	**36**$^+$	227	**62**$^+$	**74**$^+$	**74**$^+$	**37**$^+$
F13	169	**49**$^+$	**32**$^+$	**142**$^+$	**22**$^+$	161	**49**$^+$	**29**$^+$	**113**$^+$	**17**$^+$
F15	155	**58**$^+$	**112**$^+$	**53**$^+$	**40**$^+$	157	**58**$^+$	**84**$^+$	**45**$^+$	**35**$^+$
F16	200	**103**$^+$	**152**$^+$	279$^-$	**186**$^+$	262	**103**$^+$	**203**$^+$	326$^-$	**165**$^+$
F17	207	**62**$^+$	**50**$^+$	**207**$^+$	**106**$^+$	230	**62**$^+$	**39**$^+$	247$^-$	**81**$^+$
F18	160	**71**$^+$	**119**$^+$	305$^-$	196$^-$	226	**71**$^+$	**132**$^+$	380$^-$	**178**$^+$
F23	160	**55**$^+$	**56**$^+$	245$^-$	**118**$^+$	204	**55**$^+$	**24**$^+$	286$^-$	**84**$^+$
F24	164	**68**$^+$	**45**$^+$	240$^-$	**97**$^+$	220	**68**$^+$	**20**$^+$	291$^-$	**46**$^+$
F25	170	**63**$^+$	**31**$^+$	227$^-$	**92**$^+$	226	**63**$^+$	**22**$^+$	265$^-$	**63**$^+$
F26	161	**40**$^+$	**107**$^+$	**70**$^+$	**50**$^+$	249	**40**$^+$	**107**$^+$	**58**$^+$	**29**$^+$

Table 2.10: Median of testing error with tour-size=3 (left) and 7 (right)

Pro	GP	neatGP	TS-S	RDO	TS-RDO	GP	neatGP	TS-S	RDO	TS-RDO
F1	9.68	13.1$^-$	**5.88**$^+$	10.3	**7.99**	9.19	13.1$^-$	**5.13**$^+$	10.2	**6.53**$^+$
F2	0.92	**0.84**	**0.81**	1.17$^-$	1.01$^-$	0.90	**0.84**	**0.79**$^+$	1.14$^-$	0.92
F3	29.6	32.2	**15.9**$^+$	**7.06**$^+$	**6.28**$^+$	34.8	**32.2**	**16.8**$^+$	**7.30**$^+$	**6.28**$^+$
F5	0.14	0.14	**0.139**$^+$	0.141$^-$	0.14	0.14	0.14	**0.137**$^+$	0.14	0.14
F6	2.14	2.19	**2.10**	4.03$^-$	**1.36**$^+$	2.22	**2.19**	**2.07**$^+$	2.71	**1.39**$^+$
F9	5.52	5.68	**5.34**	**5.19**	**5.04**	5.49	5.68	**5.24**	**4.98**$^+$	**5.10**$^+$
F13	0.90	**0.90**	**0.90**$^+$	0.91	**0.90**	0.90	**0.90**	**0.90**$^+$	0.90	**0.90**$^+$
F15	5.01	6.21$^-$	5.07	**4.15**$^+$	**4.12**$^+$	4.20	6.21$^-$	**4.13**	**4.13**$^+$	**4.12**$^+$
F16	36.4	**36.3**	37.2	**12.0**$^+$	**11.4**$^+$	34.5	36.3	37.5	**12.1**$^+$	**11.55**$^+$
F17	5.61	**5.45**	**5.42**$^+$	6.41$^-$	**5.34**$^+$	5.70	**5.45**	**5.30**$^+$	6.55$^-$	**5.26**$^+$
F18	48.3	52.9$^-$	**46.6**	**37.8**$^+$	**36.8**$^+$	48.0	52.9$^-$	**46.3**	**38.6**$^+$	**36.7**$^+$
F23	8.84	9.15	**5.81**$^+$	8.96	**6.07**$^+$	7.52	9.15$^-$	9.19$^-$	9.29$^-$	**6.01**$^+$
F24	20.2	**19.1**	**17.1**$^+$	23.7	**16.5**$^+$	22.7	**19.1**	**16.9**$^+$	29.1	**16.0**$^+$
F25	9.36	9.42	**8.38**$^+$	14.8$^-$	**8.00**$^+$	9.58	**9.42**	**8.50**$^+$	13.9$^-$	**7.38**$^+$
F26	46.25	46.58	**46.23**	**46.26**	**46.16**	46.83	**46.58**	**46.61**	**46.72**	**46.62**

since the statistics tournament selection techniques impose less pressure on the improving training error compared to standard tournament selection.

The second metric used in the comparison is the generalisation ability of the tested methods. The median of testing error was calculated, and the results are shown in Table 2.4. We can see that the testing error of three statistics tournament selection are often smaller than that of GP. Among three statistics tournament selection, the performance of TS-S is the best on the testing data.

Table 2.4: Median of testing error with tour-size=3 (left) and 7 (right)

Pro	GP	SiS	TS-R	TS-S	TS-P	GP	SiS	TS-R	TS-S	TS-P
F1	8.55	9.06	**5.31**$^+$	**3.93**$^+$	**6.68**$^+$	11.3	**10.2**	**6.13**$^+$	**3.97**$^+$	**6.18**$^+$
F2	0.96	0.99	**0.88**	**0.81**$^+$	**0.92**	0.98	1.00	**0.89**$^+$	**0.82**$^+$	**0.98**
F3	33.4	34.2	**15.7**$^+$	**14.2**$^+$	**16.7**$^+$	33.4	34.2	**14.8**$^+$	**13.5**$^+$	**16.7**$^+$
F5	0.135	0.137$^-$	0.136	**0.131**	**0.135**	0.135	0.135	**0.135**	**0.130**$^+$	**0.131**
F6	1.78	**1.45**	1.89	1.90	**1.63**	1.34	**1.21**	1.64	1.97	1.62
F9	1.66	**1.63**	**1.61**	**1.58**$^+$	**1.57**	1.71	**1.66**	**1.59**$^+$	**1.59**$^+$	**1.61**
F13	0.88	**0.88**	**0.87**	**0.87**$^+$	**0.88**	0.88	**0.88**	**0.87**$^+$	**0.87**$^+$	**0.88**
F15	4.95	5.32	5.14	**4.74**	**4.90**	4.23	4.43	4.32	**3.92**	4.68
F16	20.9	**20.8**	27.1	25.5	27.8	18.8	23.7	24.6	24.5	24.2
F17	4.99	**4.93**	**4.90**	**4.78**$^+$	**4.87**	4.93	4.99	**4.91**	**4.70**$^+$	**4.88**
F18	8.72	9.07	10.1	10.1	11.1	8.70	**8.31**	10.3	8.79	8.88
F23	7.74	**7.24**	7.99	**5.89**$^+$	7.90	6.65	7.37	**6.63**	8.04$^-$	**6.01**
F24	16.7	18.4	**15.8**	**14.9**$^+$	18.1	17.5	18.3	**15.5**$^+$	**13.3**$^+$	**16.4**
F25	8.70	**8.44**	**8.31**	**7.99**$^+$	**8.48**	8.89	**8.43**	**7.99**$^+$	**8.40**$^+$	**8.46**$^+$
F26	46.05	46.14	**46.04**	**46.04**$^+$	46.14	50.33	**47.94**	**49.67**	**48.54**$^+$	**47.18**

The third metric is the average size of their solutions. These values are presented in Table 2.5. While the solutions found by SiS are often as complex as those found GP, the solutions found by statistics tournament selection are simpler than those of GP and SiS. Especially, the size of the solutions of TS-S is always much smaller than that of GP on all problems. This provides a reason partially explaining why the performance of TS-S on the testing data is better than other techniques in Table 2.4 following the Occam Razor principle [75].

We also measured the semantic distance between parents and their children of GP, SiS and TS-S and presented in Table 2.6. This information shows the ability of a method to discover different areas in the search space. Apparently, both SiS and TS-S maintained higher semantic diversity compared to GP. TS-S and SiS preserved better semantic diversity than GP on 24 and 22 problems,

Table 2.5: Average of solution's size with tour-size=3 (left) and 7 (right)

Pro	GP	SiS	TS-R	TS-S	TS-P	GP	SiS	TS-R	TS-S	TS-P
F1	280	**276**	**244**	**121**$^+$	**238**$^+$	286	292	**264**	**100**$^+$	**259**
F2	169	170	**130**$^+$	**35**$^+$	**148**$^+$	160	173	**150**	**37**$^+$	169
F3	263	270	**262**	**124**$^+$	277	262	276	278	**98**$^+$	277
F5	89	**78**	91	**53**$^+$	105	91	**85**	93	**42**$^+$	111
F6	167	**156**	**141**$^+$	**50**$^+$	**159**	137	151	**134**	**37**$^+$	145
F9	166	**141**	**129**$^+$	**73**$^+$	**140**	227	**176**$^+$	**161**$^+$	**74**$^+$	**166**$^+$
F13	169	**146**	**119**$^+$	**32**$^+$	**135**$^+$	161	**152**	**117**$^+$	**29**$^+$	**152**
F15	155	**132**	**147**	**112**$^+$	163	157	**150**	**133**	**84**$^+$	**156**
F16	200	**184**	**193**	**152**$^+$	**189**	262	**252**	**260**	**203**$^+$	276
F17	207	**182**	**168**	**50**$^+$	**165**	230	**220**	**192**	**39**$^+$	**192**
F18	160	**150**	165	**119**$^+$	165	226	**207**	**206**	**132**$^+$	**203**
F23	160	**159**	**132**$^+$	**56**$^+$	**134**$^+$	204	**200**	**149**$^+$	**24**$^+$	**155**$^+$
F24	164	168	**115**$^+$	**45**$^+$	**137**	220	**195**	**131**$^+$	**20**$^+$	**155**$^+$
F25	170	**167**	**111**$^+$	**31**$^+$	**138**$^+$	226	**205**	**132**$^+$	**22**$^+$	**157**$^+$
F26	161	**129**	**159**	**107**$^+$	**156**	249	**213**	**226**	**107**$^+$	**211**

respectively. These results show that TS-S achieved one of its objective in enhancing semantic diversity of GP population.

Table 2.6: Average semantic distance with tour size=3. Bold indicates the value of SiS and TS-S is greater than the value of GP.

Pro	GP	SiS	TS-S	Pro	GP	SiS	TS-S
F1	2.42	**8.93**	**3.76**	F16	71.08	**101.43**	**78.42**
F2	0.42	**1.60**	**0.43**	F17	60.91	13.52	**136.99**
F3	7.01	**19.73**	5.02	F18	105.55	**366.87**	**123.34**
F5	0.07	**0.63**	**0.10**	F23	42.25	29.08	**53.12**
F6	0.99	**1.58**	**1.03**	F24	43.10	**44.05**	**80.74**
F9	8.79	**16.88**	**8.86**	F25	37.19	17.49	**41.42**
F13	2.81	**3.86**	**10.69**	F26	54.86	**56.66**	**78.18**
F15	10.36	**12.28**	**12.79**				

Overall, the proposed methods find simpler solutions and generalize better on unseen data even though they do not improve the training error. Particularly, the solutions found by TS-S are much less complex than those of GP. Moreover, the generalization ability of TS-S is also better compared to GP and SiS.

2.4.2 *Combining Semantic Tournament Selection with Semantic Crossover*

We present an improvement of TS-S performance by combining this technique with RDO [93], and the resulting method is called TS-RDO. TS-RDO is compared with TS-S, neatGP [112], RDO [93] and GP. The results of these methods on the testing data are shown in Table 2.8. It can be seen that the combined method, TS-RDO improved the performance of TS-S and RDO. TS-RDO achieved the best result among the five tested techniques.

Table 2.8: Median of testing error with tour-size=3 (left) and 7 (right)

Pro	GP	neatGP	TS-S	RDO	TS-RDO	GP	neatGP	TS-S	RDO	TS-RDO
F1	8.55	12.5$^-$	**3.93**$^+$	8.91	**4.19**$^+$	11.3	12.5	**3.97**$^+$	**8.88**	**4.52**$^+$
F2	0.96	**0.84**$^+$	**0.81**$^+$	1.17$^-$	0.97	0.98	**0.84**$^+$	**0.82**$^+$	1.19$^-$	**0.96**
F3	33.4	**32.2**	**14.2**$^+$	**3.73**$^+$	**1.61**$^+$	33.4	**32.2**	**13.5**$^+$	**5.92**$^+$	**1.87**$^+$
F5	0.135	0.135	**0.131**	0.14	0.14	0.135	0.135	**0.131**	0.14	0.14$^-$
F6	1.78	**1.74**	1.90	**0.00**$^+$	**0.00**$^+$	1.34	1.74$^-$	1.97	**0.00**$^+$	**0.00**$^+$
F9	1.66	2.41	**1.58**	**0.01**$^+$	**0.11**$^+$	1.71	2.41	**1.59**	**0.23**$^+$	**0.23**$^+$
F13	0.88	**0.87**	**0.87**$^+$	0.88	**0.87**	0.88	**0.87**$^+$	**0.87**$^+$	**0.88**	**0.87**$^+$
F15	4.95	5.92	**4.74**	**3.24**$^+$	**3.24**$^+$	4.23	5.92$^-$	**3.92**	**3.24**$^+$	**3.24**$^+$
F16	20.9	33.7$^-$	25.5	**6.11**$^+$	**5.75**$^+$	18.8	33.7$^-$	24.5	**6.46**$^+$	**5.91**$^+$
F17	4.99	**4.95**	**4.78**$^+$	5.50$^-$	**4.85**	4.93	4.95	**4.70**$^+$	5.63$^-$	**4.74**$^+$
F18	8.72	28.49$^-$	10.18	**3.56**$^+$	**3.56**$^+$	8.70	28.49$^-$	8.79	**3.63**$^+$	**3.61**$^+$
F23	7.74	8.44	**5.89**$^+$	**5.72**	**4.32**$^+$	6.65	8.44$^-$	8.04$^-$	6.84	**4.03**$^+$
F24	16.7	17.7	**14.9**$^+$	22.2$^-$	**14.8**	17.5	17.7	**13.3**$^+$	23.3	**14.3**$^+$
F25	8.70	8.89	**7.99**$^+$	12.2$^-$	**8.13**	8.89	**8.89**	**8.40**$^+$	16.0$^-$	**7.07**$^+$
F26	46.05	47.26	**46.05**	46.75	**45.84**	50.33	**47.26**	**47.63**	**46.10**	**45.40**$^+$

In terms of the complexity, the average size of the solutions is presented in Table 2.9. TS-RDO is the best technique regarding to the solutions size. This method achieved the best result on most problems.

Overall, TS-RDO improves the testing error, and further reduces the size of the solutions compared to TS-S. Moreover, this technique performs better than both RDO and neatGP, two recently proposed methods for improving GP performance and reducing GP code bloat.

2.4.3 *Performance Analysis on The Noisy Data*

This subsection investigates the performance of five methods in Subsection 2.4.2 on the noisy data. The testing error on the noisy data is shown in Table 2.10. It can be observed from this table that TS-RDO performs slightly more consist on the noisy data compared to the noiseless data. The best test-