Selection Bias and Generalisation Error in Genetic Programming

Jeannie Fitzgerald
Bio-Computing and Developmental Systems Group
University of Limerick
Ireland
Email: jeannie.fitzgerald@ul.ie

Conor Ryan
Bio-Computing and Developmental Systems Group
University of Limerick
Ireland
Email: conor.ryan@ul.ie

***Abstract* — There have been many studies undertaken to determine the efficacy of parameters and algorithmic components of Genetic Programming, but historically, generalization considerations have not been of central importance in such investigations. Recent contributions have stressed the importance of generalisation to the future development of the field. In this paper we investigate aspects of selection bias as a component of generalisation error, where selection bias refers to the method used by the learning system to select one hypothesis over another. Sources of potential bias include the replacement strategy chosen and the means of applying selection pressure. We investigate the effects on generalisation of two replacement strategies, together with tournament selection with a range of tournament sizes. Our results suggest that larger tournaments are more prone to overfitting than smaller ones, and that a small tournament combined with a generational replacement strategy produces relatively small solutions and is least likely to over-fit.**

Keywords — Genetic Programming; Generalisation; Tournament Size; Elitism; Replacement Strategy.

I. INTRODUCTION

Generalisation is one of the most important evaluation criteria for any machine learning (ML) system [17] and a vital concern of ML practitioners is how to develop learners which generalize well [14]. In this context, under-fitting and over-fitting can be understood to correspond to bias error and variance error respectively, where generalisation error can be decomposed into bias, variance and irreducible error components [1]. In this view of generalisation error, bias is the tendency of a learner to consistently learn the same wrong thing (under-fitting) and variance is the tendency to learn random things irrespective of the true signal (over-fitting) [3]. Or as Keijzer and Babovic [13] expressed it "the bias component (of generalisation error) reflects the generic ability of a method to fit a particular data set, while the variance error reflects the intrinsic variability of the method in arriving at a solution".

Genetic Programming (GP) is an evolutionary computation approach inspired by Darwinian principles, in which a population of candidate solutions is "evolved", where this evolutionary process realises a parallel guided search of the problem space. The guided aspect of the search is controlled through the application of genetic processes such as selection, reproduction, crossover and mutation, roughly corresponding to the Darwinian notions of natural selection, survival of the fittest, sexual reproduction and genetic mutation. Through these mechanisms, the population evolves increasingly better solutions until an optimum solution is found, or until the individuals in the population are no longer showing significant improvement.

It is typical of GP experiments that as training performance improves (decreasing bias), testing performance, or more particularly, the difference between training and test performance, deteriorates (increasing variance). As the purpose of training is to gain information concerning the target function from the training data, sensitivity to the training data is essential, and generally more sensitivity results in lower bias [8]. However, this in turn increases variance, and so there is a natural "biasvariance trade-off" [9] associated with function approximation and classification tasks.

One of the perceived disadvantages of GP, relative to other ML methods, is potentially poor generalisation performance, where, at least for regression tasks, GP has be categorised as a low bias / high variance learner [13]. Thus, the challenge for practitioners is to strive for an optimal situation where bias and variance are both balanced and relatively low, and where generalisation error (in terms of bias and variance components) is minimised. In early work Whigham [22] defined three types of inductive bias and mapped those to particular aspects of GP:

1) **Search Bias:** Search bias refers to the factors that control the way in which the learning system transforms one hypothesis into another. In terms of GP this bias is represented by the crossover and mutation operators and the maximum depth of any created program tree;
2) **Selection Bias:** Refers to the method used to select one hypothesis over another. In GP this relates to the chosen fitness function, and other selection biases such as tournament size, elitism and replacement strategy;
3) **Language Bias:** The representation language implicitly defines the space of all hypotheses that a learner can represent and therefore can ever learn, as it restricts the forms that those hypotheses may represent. In GP this relates to the function and terminal sets.

We believe that these definitions and mappings may be useful as a focus for research, to identify and understand the potential contribution that each of these inductive biases may make to the bias and variance components of generalisation error. In so doing, we may learn heuristics for improving generalisation performance in GP. In this empirical study, we have chosen to focus on aspects of *selection bias* including replacement strategy, tournament size and elitism. We investigate the possible effects on generalisation of two popular replacement strategies together with tournament selection with a range of tournament sizes, with and without elitism on nine different binary classification tasks. To our knowledge this is the first study on this topic which is focused on possible implications for generalisation with regard to classification problems. Aside from tournament selection, there are various other selection methods possible, but we choose to concentrate on tournament selection as it is the most popular method used by GP practitioners today. Of course, the most significant component of selection bias in GP is the fitness function chosen and we intend to address that aspect in future work. The remainder of this paper is laid out as follows: Section II provides details of important previous work concerning the identified selection biases, Section IV explains the set-up of the empirical study and details the results, and Section V outlines conclusions.

II. Background

In a study comparing genetic robustness of steady state versus generational approaches Jones and Soule [12] investigated the behaviour of both algorithms on a simple "two peaks" problem. In this type of problem the *fitness landscape* [24] consists of two peaks: a broad low peak and a high narrow peak, and the objective is for the population to discover the tall narrow peak. Jones and Soule determined that when a generational algorithm was used, once the low broad peak was discovered the system required code growth in order to move off that peak onto the taller peak. By contrast, when a population was encoded using the steady state approach, it naturally made a smooth transition from one peak to the other without requiring code growth. In that context, the term *robust* refers to a feature of individual programs, such that they are relatively impervious to small changes introduced by crossover or mutation, i.e. small changes in their structure will not lead to big changes in their behaviour. In this way, a robust individual, once on the slopes of the high narrow peak, will be less likely to produce offspring prone to "falling off" in the event of a small change to its structure.

Another investigation by Whitley et al. [23] compared steady-state with generational replacement strategies using tournament sizes of 2 and 7 on several popular GP problems including Artificial Ant, 11 Multiplexer and a symbolic regression problem. The results of that study showed that a generational strategy with tournament size of 2 was the worst performing whereas the steady-state strategy with tournament size of 2 was best overall. With regard to Grammatical Evolution (GE), Ryan et al. [18] demonstrated that the steady state approach delivered superior performance when used to solve various symbolic regression problems. The authors hypothesised that this was because GE has a tendency to produce some sub-optimal solutions which a generational algorithm may allow to filter through to later generations, but which are usually eliminated with a steady state algorithm.

In early work comparing various selection strategies for Genetic Algorithms (GAs) Goldberg and Deb [10] explained that when steady-state genetic algorithms use a selective strategy which involves replacing the worst individual coupled with tournament selection the actual selective pressure is much greater than the tournament size might suggest. Whitley et al. [23] suggested that in terms of the time it takes for the best individual to take over the population under selection, a tournament size of 2 under the steady state model behaves more like a tournament size of 7 in a generational model.

Xie [25] studied both tournament and population size from the perspectives of selection pressure and an individual's probability of being sampled. They concluded that tournament size has an effect on sampling probability such that larger tournaments result in higher sampling probability and conversely lower probability of selection. The issue of multiple sampling of individuals was shown not to be a serious problem whereas for binary tournaments the higher probability of individuals not being sampled at all may lead to sub-optimal solutions. Xie investigated tournaments of size 2, 4 and 7.

Fitzgerald and Ryan [6] proposed a method of self-adaptive tournament selection in which an initial small tournament size becomes progressively more elitist depending on how well the system is performing on the training data. They maintained a threshold value t such that if the mean training fitness failed to improve for t generations, the size of tournaments increased by 1. They reported that their self-adaptive approach produced improved accuracy on test data. There has been quite a lot of other research on tournament selection, see for example [19, 21, 25], much of which concentrates on effects and properties of tournament selection with regard to fitness distribution, diversity, selection pressure or sampling and is focused on training data. See [4] for a review of the topic.

In this study we instead examine the effects of various tournament sizes on generalisation for binary classification problems

III. Area Under the ROC Curve (AUC)

Originating from World War II where it was initially used to evaluate the performance of radio personnel at accurately reading radar images, the *Receiver Operating Characteristic* (ROC) is a tool which may be used to measure the performance of medical tests, radiographers, and classifiers. The area under the ROC curve, known as the *AUC* is a scalar value which captures the accuracy of the entity under scrutiny. The AUC is a non-parametric measure representing ROC performance independent of any threshold [2]. A perfect ROC will have an AUC of 1, whereas the ROC plot of a random classifier will result in an AUC of approximately 0.5.

There are various ways to calculate the AUC and we have chosen the Wilcoxon Mann Whitney approximation as it is

easy to calculate and has the advantage that it facilitates the estimation of confidence intervals [20].

IV. Experiments and Results

For these experiments we compare the performance of generational and steady-state replacement strategies with tournament sizes of 2, 3, 6, 9, 12 and 15 on nine different binary classification tasks. We report the AUC, best AUC, average training accuracy, best training accuracy, average test accuracy, best test accuracy and variance error. Each of these relates to the best-of-run individuals.

For each configuration we carried out 50 runs with identical random seeds. We choose a population size of 300 and terminated evolution after 60 generations. A crossover rate of 0.8 and a mutation rate of 0.2 were employed. A generational approach is indicated with "G" and a steady-state one with "S" with the tournament size appended, together with an elitism indicator: S3 represents a non elitist steady-state algorithm with tournament size of 3 whereas G2E represents an elitist generational replacement strategy with a tournament of size 2. Where elitism is applied, it is at the rate of 10%.

Both the generational and steady state algorithms used for these experiments behave in a similar way in that each successful crossover operation generates two offspring. However, in the case of steady state the offspring replace their parents whereas for the generational approach they are copied to the new population, leaving the parents available for re-selection. In both cases, offspring are copied regardless of fitness. Mutation functions in a similar fashion.

We have chosen 9 binary classification datasets from the UCI machine learning repository [7]. This relatively large number of benchmarks allows for a robust evaluation, providing some insulation against suitability or otherwise of particular dataset/s to the inductive bias of the GP paradigm. Thus, we can reasonably expect that any trends observed may have general application for binary classification.

TABLE I: DATA SETS

Data Set	Acronym	Features	Instances	%Minority
Biodegradation	BIO	42	1082	33
Blood Transfusion	BT	5	768	23
Liver Disorders	BUPA	7	345	42
Caravan	CAR	85	5946	6
German Credit	GC	23	1000	30
Habermans Survival	HS	4	306	36
Ionosphere	ION	34	348	36
Diabetes	PIMA	9	768	35
Wisconsin Breast Cancer	WBC	10	699	46

We know from machine learning that generalisation error is composed of bias and variance error:

$$generalisation\,error = bias + variance + noise \quad (1)$$

We will assume for simplicity that our noise component $= 0$ and then bias error is $= 1 - AUC$ or $100 - accuracy$ and $variance\,error = training\,accuracy - test\,accuracy$ where variance error (over-fitting) may have a negative value in the case where test accuracy is better than training accuracy. We acknowledge that this is a drastic simplification, however it may be a useful way to reason about our practical experiments. We choose AUC as the primary measure of performance in considering bias error for these experiments. See [16] for a detailed explanation of why AUC should be preferred to accuracy as a measure of classifier performance, and also [5] as to why overall classification accuracy is a poor and misleading choice, even when the data is even mildly unbalanced.

A. Results

Experimental results for the nine datasets are extensive and quite similar, so due to space restrictions we show full results for two datasets here. Aggregate results for all datasets are reported under various headings.

1) Variance: TableIII summarises the variance error (over-fitting) for each dataset, for each configuration, where "E" indicates elitism and "NE" no elitism. The lowest average over-fitting is highlighted in bold font.

Here we see that in 5 of the 9 datasets an non-elitist generational strategy produced the smallest variance error whereas in 8 of 9 cases a steady-state strategy resulted in the greatest variance error, of these 8, 6 were elitist set-ups.

These results would seem to suggest that a non-elitist generational approach is best to minimise over-fitting and that elitism is not helpful in this respect. Even though the steady-state algorithm employed here does not involve replacing the worst individuals with newly created offspring or mutants, it is likely that due to the overlapping nature of the algorithm it is essentially more elitist in nature, and thus there maybe be elitist effects similar to those reported by Whitley et al. [23], although perhaps to a lesser extent.

To clarify the effects of elitism, we ran further experiments using varying levels of elitism (20%,30% and 40%) for each of with both generational and steady-state replacement strategies with a tournament of size 2. The results of those particular experiments revealed that in 7 of 9 datasets the $G10$ configuration (generational with 10% elitism) exhibited the lowest variance error. Which would seem suggest that elitism contributes to variance error, and that if elitism is used, then small values should be preferred if we wish to minimise variance error.

2) Bias: For 8 of the 9 datasets there is a generational set-up that produces or shares the best average AUC score, and the majority of these are elitist. In contrast, only 4 of the 9 datasets have a steady-state configuration that achieves or shares best average AUC score.

Looking at the best *overall* AUC scores, the relative performance of the steady-state algorithm is somewhat better, where 8 of the 9 datasets have at lease one elitist generational configuration which achieves the top score and 6 of 9 have a steady-state configuration which does so. In 6 of the 9 datasets, higher percentages of elitism resulted in or shared the best result for average or best overall AUC.

TABLE II: TEST AUC AND VARIANCE ERROR

Data	Config.	Avg. AUC	Best AUC	Var. Err.
BIO	S10	0.82	0.90	**3.26**
	S20	0.80	0.90	4.76
	S30	0.82	0.89	5.12
	S40	0.81	0.90	4.68
	G10	0.78	0.88	4.03
	G20	**0.83**	0.90	5.54
	G30	**0.83**	**0.91**	5.11
	G40	0.80	0.90	4.10
BUPA	S10	0.71	0.85	2.58
	S20	0.72	**0.89**	3.60
	S30	0.74	0.86	1.78
	S40	0.72	**0.89**	2.44
	G10	**0.77**	**0.89**	**-3.93**
	G20	0.75	**0.89**	0.79
	G30	0.74	0.88	4.16
	G40	0.75	**0.89**	-0.99
BT	S10	0.72	0.77	0.47
	S20	0.72	0.77	0.48
	S30	0.72	**0.79**	0.55
	S40	0.72	0.78	0.57
	G10	0.72	**0.79**	**0.07**
	G20	**0.74**	0.79	0.23
	G30	**0.74**	0.78	0.80
	G40	0.73	0.79	0.13
CAR	S10	0.66	**0.78**	-0.47
	S20	0.67	0.76	-0.43
	S30	0.67	0.77	-0.49
	S40	0.66	0.77	-0.78
	G10	0.65	0.76	**-0.92**
	G20	0.67	**0.78**	-0.77
	G30	**0.68**	0.76	-0.42
	G40	0.66	0.76	-0.60
GC	S10	0.59	**0.75**	7.89
	S20	**0.64**	0.71	7.73
	S30	0.63	0.72	8.24
	S40	0.63	0.73	7.68
	G10	0.61	0.70	**6.36**
	G20	**0.64**	0.71	7.73
	G30	**0.64**	0.73	8.30
	G40	**0.64**	0.74	7.92
HS	S10	0.76	0.87	**-5.14**
	S20	0.73	0.87	-0.66
	S30	0.74	0.84	-1.74
	S40	0.75	0.89	-3.09
	G10	**0.78**	0.90	-4.68
	G20	0.75	0.87	-2.59
	G30	0.74	0.87	-1.28
	G40	0.77	0.88	-3.97
ION	S10	0.83	0.95	4.10
	S20	0.84	0.95	4.38
	S30	0.84	0.95	5.39
	S40	0.83	0.93	4.61
	G10	0.80	0.94	**2.61**
	G20	0.85	0.95	3.77
	G30	**0.86**	**0.97**	4.44
	G40	0.83	0.96	3.24
PIMA	S10	**0.65**	0.74	7.26
	S20	0.65	0.73	8.95
	S30	0.64	0.73	8.33
	S40	0.64	0.74	8.09
	G10	0.64	0.74	**6.26**
	G20	0.64	**0.78**	8.18
	G30	**0.65**	0,73	8.45
	G40	0.63	0.76	8.21
WBC	S10	**0.98**	1	-0.98
	S20	0.96	**1**	-0.73
	S30	0.93	0.99	-0.75
	S40	0.96	**1**	-0.78
	G10	**0.98**	0.99	**-1.23**
	G20	0.96	**1**	-0.81
	G30	0.94	0.99	-0.87
	G40	0.90	**1**	-0.84

TABLE III: VARIANCE ERROR BY DATASET

Dataset	S	G	SNE	GNE	SE	GE
BIO	5.54	5.45	5.43	**5.33**	5.65	5.56
BT	1.00	0.57	1.13	**0.49**	0.87	0.65
BUPA	7.01	3.02	7.01	3.10	7.01	**2.93**
CAR	-0.53	-0.43	**-0.54**	-0.46	-0.52	-0.40
GC	9.58	8.91	9.42	8.77	**4.38**	9.06
HS	-1.53	-1.70	**-2.43**	-1.54	-0.62	-1.86
ION	5.61	5.12	5.50	**4.95**	5.70	5.29
PIMA	10.13	9.60	9.92	**9.32**	10.31	9.88
WBC	-0.46	**-0.56**	-0.48	**-0.56**	-0.45	-0.55

TABLE IV: RESULTS BY TOURNAMENT SIZE

Tourn. Size	AUC	Best AUC	Avg.Train	StdDev	Best Train	Avg. Test	StdDev	Best Test	Avg Size	OverFit
2	**0.75**	**0.85**	77.87	2.20	82.22	75.68	3.62	81.92	157.40	**2.21**
3	**0.75**	**0.85**	78.89	2.18	83.29	75.71	3.69	80.60	183.79	3.22
6	0.74	0.84	79.53	2.53	84.65	75.55	3.91	82.42	200.88	4.00
9	0.74	0.84	79.58	2.77	84.84	75.48	4.07	82.73	200.65	4.07
12	0.74	0.84	79.36	2.87	84.79	75.10	4.28	82.45	197.70	4.24
15	0.74	70.84	79.38	2.78	85.07	75.19	4.17	82.57	200.34	4.37

Overall, an elitist generational strategy delivers better performance in terms of bias error. Good results for average and best AUC were achieved at various tournament sizes, but overall tournaments of size 2 delivered the best results.

3) Summary of Results: He we summarise the results, showing averages across the datasets under the relevant headings: tournament size, replacement strategy and elitism.

4) Tournament Size: In Table IV we show the averages results for tournament size across all datasets and configurations. These results indicate that smaller tournaments (2 or 3) result in better average and overall AUC scores and that the smallest tournament produces the least over-fitting and the smallest solutions, on average.

5) Replacement Strategy: Looking at results in Table V organised by replacement strategy, we can see that the Generational approach results in best average and overall AUC and produces smaller programs which have lower over-fitting.

6) Elitism: Finally, if we examine the results with respect to elitism as seen in Table VI we can observe that the application of elitism produces better results on training accuracy for smaller tournaments but that the same benefit is not apparent for larger tournaments. Equally good results for average AUC are achieved with both elitist and non-elitist strategies, with the elitist approach delivering the best overall AUC on average. With regard to test accuracy, the application of elitism would not appear to confer any significant advantage. The best configuration overall from the perspective of generalisation is a non-elitist strategy with a tournament size of 2.

Taking a more detailed look at the data, we observe that in 8 of the 9 problems a non-elitist strategy resulted in the least over-fitting overall. For the most part this was with a

TABLE V: AVERAGE RESULTS BY REPLACEMENT STRATEGY

Algorithm	AUC	Best AUC	Avg.Train	StdDev	Best Train	Avg. Test	StdDev	Best Test	Avg Size	OverFit
Steady-State	0.74	0.84	79.12	2.76	84.52	75.09	3.64	81.48	210.24	4.05
Generational	**0.75**	**0.85**	79.08	2.35	83.77	75.82	4.28	82.75	170.02	**3.32**

TABLE VI: AVERAGE RESULTS BY ELITISM

Size/Elite	AUC	Best AUC	Avg.Train	StdDev	Best Train	Avg. Test	StdDev	Best Test	Avg Size	OverFit
2E	**0.75**	**0.85**	78.75	2.13	82.96	75.84	3.73	82.74	154.15	2.89
3E	**0.75**	**0.85**	79.23	2.16	83.66	75.91	3.74	82.65	174.32	3.33
6E	0.74	0.84	79.46	2.49	84.55	75.53	3.81	82.09	181.93	3.93
9E	**0.75**	0.84	79.42	2.69	84.61	75.32	4.13	82.62	187.16	4.10
12E	0.74	0.84	79.17	2.89	84.86	75.07	4.30	82.42	188.51	4.10
15E	0.74	0.84	79.45	2.76	84.67	75.12	4.22	82.67	189.11	4.32
2	**0.75**	0.84	76.99	2.27	81.49	75.52	3.51	81.10	160.66	**1.53**
3	0.74	0.84	78.56	2.19	82.93	75.52	3.64	82.53	196.97	3.11
6	0.74	0.84	79.59	2.56	84.74	75.57	4.02	82.75	219.84	4.07
9	0.74	0.84	79.73	2.85	85.06	75.65	4.01	82.85	214.14	4.04
12	0.74	0.84	79.55	2.85	84.72	75.14	4.26	82.49	206.89	4.38
15	0.74	0.84	79.31	2.79	85.46	75.27	4.12	82.47	211.57	4.42

tournament of size 2. However, when we compare the effects of elitism on over-fitting across the range of tournament sizes for the same replacement strategy there is little difference.

7) Further Remarks on Replacement Strategy: In other experiments (not shown) which used an optimised GP set-up, results in terms of average AUC were sub-optimal for both the GC and CAR datasets using a generational approach. Investigation indicated that the average program size with the generational method was disproportionately smaller for both the GC and CAR datasets in comparison with a steady state configuration. As some of the optimisations used had been shown to produce smaller programs, we hypothesise that this combined with the observation that a generational replacement strategy produces smaller solutions, created a situation where the GP system was unable to evolve programs of sufficient size to encode a good solution for these datasets, both of which have relatively large numbers of attributes and instances. We informally confirmed this hypothesis by re-running the CAR experiments using a steady-state algorithm with reduced crossover probability, whereupon the performance in terms of average AUC was significantly degraded. This problem could be overcome by running evolution for more iterations/generations. Hunt et al. [11] demonstrated that for classification, GP has good scalability with regard to instances but does not scale well with as the number of attributes increases. Our experience here may hint that program size is a factor in scalability with regard to attributes.

V. CONCLUSIONS

Considering the extensive experimental results, we can make several interesting observations:

1) Average solution size is *universally smaller* for the generational replacement strategy. A possible reason for this phenomenon it that steady-state algorithms are *overlapping* in the sense that until such time as the new population is filled, all individuals in the population, including newly created offspring, are available for selection. Since crossover increases program size [15], these newly created offspring are likely to be larger than the individuals which they replace, and if these are in turn selected for crossover in the same iteration that they were created in, this will likely lead to an increase in average size over a generational strategy;
2) In 8 of the 9 problems, a non-elitist strategy resulted in the least over-fitting. Of these, 6 were associated with a generational algorithm;
3) in 7 of the 9 problems, a tournament size of 2 resulted in the least over-fitting;
4) For 7 of the 9 problems variance error is tending to increase with tournament size. This, taken together with the previous point, suggests that larger tournaments are more prone to over-fitting than smaller ones, and that a small generational algorithm is least likely to over-fit;
5) Elitism may reduce bias error but lead to increased variance error, and is thus a parameter which influences the bias variance trade-off;
6) The best performing configuration in terms of AUC score from the main experiments is the G2E configuration. Experiments with higher percentages of elite yielded even better results, on average;
7) When using a generational strategy together with parameter or algorithmic settings which my constrain program size, it may be necessary to run evolution for longer in order to achieve optimal results.

In this paper, we have examined the influence of selection bias introduced through choices of tournament size, replacement strategy and application of elitism on generalisation behaviour. The results of the empirical study illustrate that, at least for the 9 problems studied, a tournament size of 2 combined with a non-elitist generational replacement strategy produced the best results in terms of bias and variance error.

Acknowledgements: This work has been supported by the Science Foundation of Ireland. Grant No. 10/IN.1/I3031.

TABLE VII: PERFORMANCE OF BEST-OF-RUN TRAINED INDIVIDUALS, HS

	Method	AUC	Best AUC	Avg.Train	Best Train	Avg. Test	StdDev	Best Test	Avg Size	OverFit
HS	S2	0.76	0.87	75.14	79.65	**80.92**	2.58	84.21	223.04	**-5.14**
	S3	0.75	0.88	76.21	80.08	80.18	2.91	85.52	256.00	-3.96
	S6	0.76	0.88	76.99	83.55	79.63	2.67	84.21	280.60	-2.64
	S9	0.73	0.86	**77.80**	83.98	78.71	3.96	85.52	273.88	-0.91
	S12	0.73	0.87	77.62	83.11	78.66	3.57	85.52	303.80	-1.03
	S15	0.74	0.86	77.16	85.71	78.07	4.55	84.21	299.80	-0.91
	S2E	0.76	0.88	74.13	77.92	77.21	4.80	84.21	210.28	-3.08
	S3E	0.73	**0.91**	75.93	82.55	75.47	4.94	85.53	222.40	0.46
	S6E	0.74	0.90	77.03	84.42	76.92	4.80	85.53	228.21	0.10
	S9E	0.76	0.90	76.88	82.88	77.05	5.41	85.53	256.68	-0.17
	S12E	0.74	0.87	76.07	**88.34**	77.47	4.15	85.53	272.00	-1.40
	S15E	0.75	0.86	77.22	85.71	76.84	4.78	85.53	256.48	0.37
	G2	**0.78**	0.90	73.00	77.06	77.18	3.51	85.53	147.36	-4.68
	G3	0.76	0.90	74.59	80.52	77.55	3.55	84.21	193.64	-2.96
	G6	0.75	0.87	76.45	82.68	77.00	4.81	84.21	230.04	-0.54
	G9	0.76	0.88	76.80	82.25	76.76	4.94	85.53	214.36	0.04
	G12	0.73	0.89	77.15	83.98	77.44	5.01	84.21	225.28	-0.30
	G15	0.76	0.87	76.60	83.55	77.39	4.98	84.21	226.36	-0.79
	G2E	**0.78**	0.89	74.43	79.65	78.55	3.66	85.52	155.76	-4.11
	G3E	0.77	**0.91**	75.83	81.81	77.86	4.71	84.21	164.24	-2.04
	G6E	0.76	0.89	76.13	80.52	76.86	4.64	84.21	173.40	-0.73
	G9E	**0.78**	0.88	76.19	83.11	77.99	4.60	**86.84**	190.44	-1.80
	G12E	**0.78**	0.89	76.00	81.38	77.81	4.01	84.21	189.80	-1.85
	G15E	0.75	0.88	76.78	83.12	77.39	3.79	82.89	210.04	-0.61

TABLE VIII: PERFORMANCE OF BEST-OF-RUN TRAINED INDIVIDUALS, PIMA

	Method	AUC	Best AUC	Avg.Train	Best Train	Avg. Test	StdDev	Best Test	Avg Size	OverFit
PIMA	S2	**0.65**	0.74	70.26	75.00	63.00	3.32	68.75	186.44	7.26
	S3	0.62	0.73	71.81	77.65	62.13	4.05	68.75	234.08	9.68
	S6	0.63	0.73	72.99	80.38	62.26	3.56	71.87	271.68	10.74
	S9	0.62	0.73	72.66	78.64	61.56	3.89	69.27	258.88	11.10
	S12	0.62	0.72	**73.44**	78.22	62.24	3.50	70.83	238.12	10.80
	S15	0.63	0.71	72.87	80.56	61.69	3.34	68.75	263.00	11.17
	S2E	0.64	0.75	72.18	77.26	**63.79**	2.76	69.27	180.24	8.38
	S3E	0.62	0.73	72.57	79.51	62.99	4.11	71.88	199.48	9.58
	S6E	0.62	0.72	72.29	80.56	61.32	3.70	68.75	214.92	10.76
	S9E	0.62	0.73	73.15	79.16	62.22	4.43	71.35	252.72	10.92
	S12E	0.62	0.70	71.93	78.64	60.87	4.19	68.75	234.44	11.05
	S15E	0.63	0.72	72.60	77.08	61.82	4.21	68.22	242.80	10.79
	G2	0.64	0.74	68.48	74.13	62.22	3.65	69.27	156.80	**6.26**
	G3	0.64	0.77	71.11	76.04	62.81	3.57	72.92	184.84	8.30
	G6	0.63	0.75	72.59	76.21	62.83	4.00	71.35	216.88	9.76
	G9	0.63	0.74	73.01	79.00	62.61	3.29	70.31	217.84	10.39
	G12	0.64	0.74	72.94	77.60	62.54	4.13	69.27	195.04	10.40
	G15	0.63	0.72	73.29	**81.59**	61.64	4.27	70.31	195.84	11.65
	G2E	**0.65**	**0.75**	71.92	76.38	63.70	3.15	73.43	126.20	8.21
	G3E	0.64	0.72	72.58	78.29	63.02	3.81	**74.44**	154.96	9.56
	G6E	0.64	0.72	72.75	77.78	62.91	3.63	69.27	164.00	9.83
	G9E	**0.65**	0.74	73.17	78.47	63.68	3.86	70.31	173.81	9.49
	G12E	0.63	0.71	73.09	78.30	62.53	3.42	68.23	161.64	10.55
	G15E	0.63	0.70	72.88	78.65	62.23	3.57	69.37	172.92	10.64

REFERENCES

[1] Leo Breiman. Bias, variance, and arcing classifiers. *Statistics*, 1996.

[2] Christopher D. Brown and Herbert T. Davis. Receiver operating characteristics curves and related decision measures: A tutorial. *Chemometrics and Intelligent Laboratory Systems*, 80(1):24–38. 2006.

[3] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[4] Yongsheng Fang and Jun Li. A review of tournament selection in genetic programming. In Zhihua Cai et al. editors, *Advances in Computation and Intelligence*, volume 6382 of *LNCS*, pages 181–192. Springer Berlin Heidelberg, 2010.

[5] Jeannie Fitzgerald and Conor Ryan. A hybrid approach to the problem of class imbalance. In *International Conference on Soft Computing*, Brno, Czech Republic, June 2013.

[6] Jeannie Fitzgerald and Conor Ryan. Individualized self-adaptive genetic operators with adaptive selection in genetic programming. In *Nature and Biologically Inspired Computing (NaBIC), 2013*, pages 232–237. IEEE, 2013.

[7] A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

[8] Jerome H Friedman. On bias, variance, 0/1 loss, and the curse-of-dimensionality. *Data mining and knowledge discovery*, 1(1):55–77, 1997.

[9] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

[10] David E Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. *Urbana*, 51:61801–2996, 1991.

[11] Rachel Hunt, Kourosh Neshatian, and Mengjie Zhang. Scalability analysis of genetic programming classifiers. In Xiaodong Li, editor, *Proceedings of the 2012 IEEE Congress on Evolutionary Computation*, pages 509–516, Brisbane, Australia, 10-15 June 2012.

[12] Josh Jones and Terry Soule. Comparing genetic robustness in generational vs. steady state evolutionary algorithms. In *Proceedings of the 8th annual conference on genetic and evolutionary computation*, pages 143–150. ACM, 2006.

[13] Maarten Keijzer and Vladan Babovic. Genetic programming, ensemble methods and the bias/variance tradeoff–introductory investigations. In *Genetic Programming*, pages 76–90. Springer, 2000.

[14] Tim Kovacs and Andy J. Wills. *Encyclopedia of the Sciences of Learning*, chapter Generalization Versus Discrimination in Machine Learning. 2012.

[15] William B. Langdon. Size fair and homologous tree crossovers for tree genetic programming. *Genetic programming and evolvable machines*, 1(1-2):95–119, 2000.

[16] Charles X Ling, Jin Huang, and Harry Zhang. Auc: a better measure than accuracy in comparing learning algorithms. In *Advances in Artificial Intelligence*, pages 329–341. Springer, 2003.

[17] Tom M. Mitchell. *Machine learning*. McGraw-Hill, New York, NY. 1997.

[18] Conor Ryan, JJ Collins, and Michael O Neill. Grammatical evolution: Evolving programs for an arbitrary language. In *Genetic Programming*, pages 83–96. Springer, 1998.

[19] Artem Sokolov and Darrell Whitley. Unbiased tournament selection. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, GECCO '05, pages 1131–1138, New York, NY, USA, 2005. ACM.

[20] Paul Stober and Shi-Tao Yeh. (2007) An explicit functional form specification approach to estimate the area under a receiver operating characteristic (roc) curve. *Available at http://www2. sas. com/proceedings/sugi27/p226–227. pdf Accessed March*, Issue 7.

[21] Dirk Thierens. Selection schemes, elitist recombination, and selection intensity. *UU-CS*, (1998-48), 1998.

[22] Peter A Whigham. Search bias, language bias and genetic programming. In *Proceedings of the First Annual Conference on Genetic Programming*, pages 230–237. MIT Press, 1996.

[23] Darrell Whitley, Marc Richards, Ross Beveridge, and Andre da Motta Salles Barreto. Alternative evolutionary algorithms for evolving programs: Evolution strategies and steady state GP. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 919–926, New York, NY, USA, 2006. ACM.

[24] Sewall Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. In *Proceedings of the sixth international congress on genetics*, volume 1, pages 356–366, 1932.

[25] Huayang Xie. *An Analysis of Selection in Genetic Programming*. PhD thesis, Computer Science, Victoria University of Wellington, New Zealand, 2008.