

# Multi-objective Techniques in Genetic Programming for Evolving Classifiers

Daniel Parrott, Xiaodong Li and Vic Ciesielski

School of Computer Science and Information Technology

RMIT University, Melbourne, Vic 3001, Australia

{dparrot, xiaodong, vc}@cs.rmit.edu.au

**Abstract-** The application of multi-objective evolutionary computation techniques to the genetic programming of classifiers has the potential to both improve the accuracy and decrease the training time of the classifiers. The performance of two such algorithms are investigated on the even 6-parity problem and the Wisconsin Breast Cancer, Iris and Wine data sets from the UCI repository. The first method explores the addition of an explicit size objective as a parsimony enforcement technique. The second represents a program's classification accuracy on each class as a separate objective. Both techniques give a lower error rate with less computational cost than was achieved using a standard GP with the same parameters.

## 1 Introduction

Genetic Programming (GP) [9] has successfully evolved data classifiers. Early work developed approaches for binary classifiers [6], with multi-class data sets classified using multiple binary classifiers in series [7]. Recently, classifiers for multi-class data have been evolved directly [12, 16, 15].

Evolutionary multi-objective (EMO) techniques have been applied to genetic programming for parsimony enforcement [3, 4] in order to overcome 'bloat', the tendency of genetic programs to develop unnecessary code or 'introns' [1, 10], and evolve programs for less computational cost. Diversity maintenance measures used in these EMO techniques also contribute to avoiding convergence to local minima.

Combining multi-objective techniques with the genetic programming of classifiers thus presents opportunities for evolving smaller classifiers that should generalize better [14] while searching the solution space more effectively. This would lead to classifiers that attain lower errors and are evolved for less computational cost than would be the case using the standard GP algorithm.

In this paper we investigate evolving GP classifiers using techniques derived from the NSGA-II multi-objective algorithm [5]. The parsimony enforcement technique of POPE-GP [2] is developed further and analyzed. A novel approach of decomposing a classification problem so that classification accuracy on each class is represented as a distinct objective is also explored. In addition to maintaining diversity, it is intended that by crossing-over partial solutions, each capable of classifying one class well, programs that can classify both or several classes well will be produced. The performance of these approaches is evaluated on the even-6 parity problem and several classification tasks from the UCI database. Both POPE-GP and decomposed

multi-objective GP (DecMO-GP) provide promising results compared to those given by the standard GP algorithm.

## 2 Related Work

Genetic programming of classifiers is a well established field of research but still suffers from several difficulties, such as bloat leading to ineffective crossover and mutation, long training times and error rates greater than those achieved by other classification methods on some tasks [6]. Multi-objective techniques for evolutionary computation have also received much attention. However, the combination of the two is not as well studied, although there has been interest in the use of EMO for parsimony enforcement [2, 3, 4]. The relevant bodies of work are generally distinct and will be treated as such here.

### 2.1 Genetic programming of classifiers

The application of genetic programming to binary classification problems has been demonstrated with reasonable success [6, 7, 12]. The evolved classifiers generally give numeric output, with a value below zero indicating membership of one class and a value greater than zero indicating membership of the other. Multi-class classification problems can then be approached by decomposition into a set of binary classification problems [7]. This entails training each binary classifier independently and thus increases the effort required for training. Recent research has also examined the evolution of a set of binary classifiers from a single run [13].

A GP-evolved program can be trained to classify data into multiple classes using by dividing the output into multiple ranges, each corresponding to a different class. In static range selection (SRS), the ranges are specified beforehand, introducing the difficulty of determining the appropriate range boundaries ahead of time. Dynamic range selection (DRS) attempts to overcome this by evolving the range boundaries along with the programs [12, 16, 15].

### 2.2 Pareto-based Multi-objective algorithms

Early evolutionary approaches to multi-objective problems used fixed weights to determine the relative importance of different objectives and hence required a priori knowledge of the solutions possible. These have given way to Pareto-based approaches that give a partial ordering of solutions based on Pareto dominance, relieving users of the need to predict the solutions possible. A solution is said to Pareto-dominate another if it is no worse than the other on any objective and better than the other on at least one objective. Formally, for a minimization problem,  $\mathbf{u}$  dominates  $\mathbf{v}$

( $\mathbf{u} >_d \mathbf{v}$ ) over the set of objectives  $\Theta$  iff

$$\forall \theta \in \Theta(u_\theta \leq v_\theta) \wedge \exists \theta \in \Theta(u_\theta < v_\theta), \quad (1)$$

where  $u_\theta$  is the fitness of solution  $\mathbf{u}$  on objective  $\theta$ .

Within a set of solutions  $\Phi$ , the first front  $\phi_1$  is the set of non-dominated solutions:

$$\phi_1 = \{\mathbf{v} : \nexists \mathbf{u} \in \Phi(\mathbf{u} >_d \mathbf{v})\} \quad (2)$$

Note that the true Pareto front is the first front of the set of all possible solutions.

The second front  $\phi_2$  of the set  $\Phi$  is then the first front of the set of remaining solutions  $\{\Phi - \phi_1\}$  and generally:

$$\phi_i = \{\mathbf{v} : \nexists \mathbf{u} \in \{\Phi - \bigcup_{n=0}^{i-1} \phi_n\}(\mathbf{u} >_d \mathbf{v})\}, \quad (3)$$

where  $\phi_0 \equiv \emptyset$ .

In prominent Pareto-based multi-objective evolutionary algorithms such as NSGA-II [5], SPEA-2 [17] and PAES [8], the members of fronts closer to the Pareto front have a greater chance of being selected for mutation, crossover or inclusion in the next generation. Within a front, solutions from less crowded areas are selected in preference to those from more crowded areas to maintain diversity and front coverage. The methods of selecting and maintaining diverse, low-front solutions are the key differences between the algorithms.

The Strength Pareto Evolutionary Algorithm 2 (SPEA2) [17] maintains an archive of all non-dominated solutions found so far. The solutions are assigned a fitness based on the number of solutions they dominate and are dominated by. Solutions with low distances to other individuals are eliminated when the archive exceeds a predefined size.

The Pareto Archive Evolutionary Strategy (PAES) [8] in its simplest form uses a (1+1) evolution strategy to generate a non-dominated front. A single randomly generated solution is used as a starting point and mutated to generate new solutions. New non-dominated solutions are added to the archive. Solutions from uncrowded areas are selected to mutate and generate new solutions from. Where the archive exceeds a specified limit, archive members are replaced by new members if the new members are from a less crowded area of the front.

NSGA-II (the Non-Dominated Sorting Genetic Algorithm-II) [5] uses tournament selection to choose solutions for mutation or crossover, with the solution from the lowest front in the tournament being selected. Where more than one solution belongs to the lowest front present in a tournament, the solution with the lowest 'crowding distance' (the sum of distances between a solution and its nearest neighbours in the same front) is selected. The algorithm is fully elitist - every iteration the child and parent populations are merged and the combined population Pareto-sorted into fronts. The combined population is then culled to half its size to create the new parent population.

The use of  $\epsilon$ -approximate Pareto sets has been suggested as a possible improvement to the Pareto-based EMO paradigm [11]. The Pareto-dominance criterion of equation

1 is hardened so that a solution  $\mathbf{u}$   $\epsilon$ -dominates another  $\mathbf{v}$  ( $\mathbf{u} >_{d_\epsilon} \mathbf{v}$ ) iff:

$$\forall \theta \in \Theta((1 + \epsilon) \cdot u_\theta \leq v_\theta) \wedge \exists \theta \in \Theta((1 + \epsilon) \cdot u_\theta(j) < v_\theta) \quad (4)$$

By combining this with an update operator designed for use with this dominance relation the authors create a framework for an algorithm with guaranteed convergence to the true Pareto front while maintaining diversity within the front.

### 2.3 Parsimony enforcement in GP

Bloat, where a large proportion of a program's code does not contribute to its fitness, is a well known problem within genetic programming [1, 10]. The increased program size leads to longer run times, impaired crossover and mutation and inferior generalization. A number of approaches have been attempted to combat this phenomenon, including simple tree size limitation, constant parsimony pressure (application of a fitness penalty to large programs) and adaptive parsimony pressure (where the fitness penalty is increased as the error decreases). All such techniques require foreknowledge or guess work to determine appropriate parameters.

Multi-objective techniques have been applied to parsimony enforcement by introducing low size as an objective in addition to high fitness. This approach has been used with the SPEA-2 algorithm [3], the NSGA-II algorithm in the Pseudo-Objective Parsimony Enforcement GP (POPE-GP) [2] and with the novel FOCUS algorithm [4], which included high diversity as an explicit objective in addition to low size. The FOCUS algorithm also allowed only a single solution at any point in the solution space and maintained members of the first front only using a steady-state algorithm. These approaches have provided solutions comparable to or better than those attained using standard GP and done so with lower computational cost on problems such as even n-parity and classification of the UCI Wisconsin Breast Cancer Database.

## 3 The POPE-GP and DecMO-GP algorithms

Two methods of utilising multi-objective techniques are covered - the POPE-GP algorithm [2] is developed and explored further and the Decomposed Multi-Objective GP algorithm is described. A derivative of the DecMO-GP algorithm utilising a simple parsimony enforcement technique, DecMO Parsimony-GP or DecMOP-GP, is also investigated.

These techniques are based on the NSGA-II algorithm and the reader is directed to Deb et al. [5] for details. The implementation used here differs from the original in that only a single solution is allowed at any point in the solution space. This modification was incorporated after populations in initial runs rapidly converged to a single poor but easy-to-find solution (e.g. with POPE-GP a solution consisting of a single node), other than a few unique individuals on

Table 1: Function set for the even 6-parity problem.

Name	Return Type	Arity	Argument Types	Description
AND	B	2	B,B	Logical AND operator
NAND	B	2	B,B	Logical NAND operator
OR	B	2	B,B	Logical OR operator
XOR	B	2	B,B	Logical XOR operator

Table 2: Terminal set for the even 6-parity problem.

Name	Return Type	Description
BitX	B	The value of bit x of the bit string.

the first front maintained by the algorithm’s elitism. As a result, it is possible that the size of the population at the end of a generation is less than the specified population size  $p$ , although the child population generated will still be of size  $p$ .

### 3.1 POPE-GP algorithm

The POPE-GP algorithm utilises the multi-objective method of NSGA-II, with minimising the classification error of the program as one objective and minimising its size as the second. As stated, only one individual is allowed at a  $(fitness, size)$  point in objective space. Where more than one solution occurs at a point, a solution is chosen at random to be maintained and the others deleted.

### 3.2 DecMO-GP algorithm

The Decomposed Multi-Objective Genetic Programming algorithm, DecMO-GP, decomposes the single objective of minimising classification error rate into an objective of minimising error for each class. The error  $e_i$  for the  $i$ th class  $C_i$  has been defined as the number of instances of class  $C_i$  not classified as members of class  $C_i$ :

$$e_i = \frac{|C_i - \{C_i \cap classified(C_i)\}|}{|C_i|} \quad (5)$$

This can lead to an individual achieving zero error on the  $i$ th objective by identifying all individuals as members of class  $i$ . However, doing so will result in the worst error possible on all other objectives. An individual using this strategy will be pushed to a lower front by individuals achieving zero error on the  $i$ th objective and low error on others. Selection pressure will thus minimise error across all classes simultaneously. Figure 1 showing the first and second fronts on a two-class classification task, gives an example of this. Here, the individual at (0,3) is in the first front, ahead of that at (0,5) which classifies the second class slightly worse. Ideally, the algorithm will give rise to an individual with a perfect score across all classes that hence classifies all classes correctly.

If this occurs, the first front will consist of a single individual that will be chosen as the classifier from the run. Where the first front has multiple individuals the choice of classifier is not so obvious. If one individual has a lower aggregate error across all classes than any other it will be

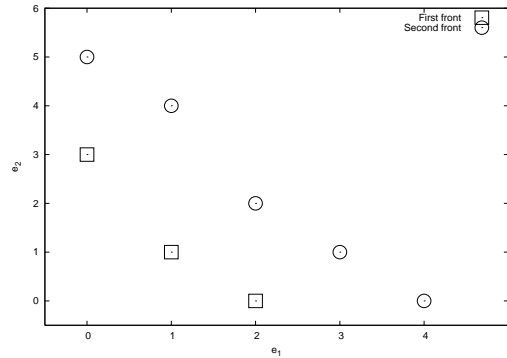


Figure 1: Example first and second fronts of a two-class DecMO-GP run.

chosen. Where the lowest aggregate error is achieved by more than one individual, the individual with the best balance of errors across classes is selected in order to obtain an individual that classifies all classes well. The best-balanced individual for an  $n$ -class classification problem is defined as that with the lowest sum of deviations  $s$  from the mean of its own errors across all  $n$  classes:

$$s = \sum_{i=1}^n \left| e_i - \frac{\sum_{i=1}^n e_i}{n} \right| \quad (6)$$

Finally, if the minimum value of  $s$  is achieved by more than one individual, one of these individuals is chosen at random. In the example shown in Figure 1, the individual at (1, 1) would be chosen ahead of that at (0, 2) by applying the best-balanced criteria.

There are several aims to decomposing the overall objective as described. First, it is intended that this strategy, combined with the aggressive elimination of duplicates in objective space, will result in a population with high diversity. This should result in a population that searches the solution space effectively and avoids the trap of converging to local optima. Of course, if the genotype of the population partially converges so that all solutions are similar despite high phenotypic diversity, a relatively small region of the solution space will be searched.

Second, it is intended that the strategy promote the evolution, retention and re-use of functional blocks of code. Evolving individuals that classify a single class relatively

Table 3: Function set for the database classification problems.

Name	Return Type	Arity	Argument Types	Description
+	D	2	D,D	Addition
-	D	2	D,D	Subtraction
*	D	2	D,D	Multiplication
/	D	2	D,D	Division
If	D	3	B,D,D	If arg 1 is true returns arg2 else arg3
<	B	2	D,D	true if arg1 < arg2
>	B	2	D,D	true if arg1 > arg2
=	B	2	D,D	true if arg1 == arg2
Between	B	3	D,D,D	true if arg1 < arg2 < arg3 or if arg1 > arg2 > arg3

Table 4: Terminal set for the database classification problems.

Name	Return Type	Description
AttrX	B	The value of attribute X for the instance.
RandX	B	A random number in range (0,100).

well may result in functional blocks of code capable of classifying a single class. These individuals will be kept by the elitism of the NSGA-II algorithm. By crossing-over individuals with complementary classification abilities, i.e. that correctly classify different classes, it is hoped that child individuals that can correctly classify both classes will be produced. Such an effect should lead to faster evolution of good individuals.

Third, where no single best solution is available, having several individuals each optimised for classification of a different class could give rise to systems utilising multiple programs to classify new data. Such systems could be similar to the hierarchical application of binary classifiers for multi-class systems (e.g. [13]) but could use multiple ‘good’ but different classifiers in conjunction to classify new data instances. While an interesting concept, this is not explored in the current paper.

### 3.3 DecMOP-GP algorithm

Keeping only a single individual at each point in phenotypic solution space gives the opportunity to choose which individual to keep. Strong selection pressure against size is introduced by keeping the smallest individual at each point. The algorithm created by applying this modification to the DecMO-GP algorithm is referred to as the Decomposed Multi Objective-Parsimony Genetic Programming, or DecMOP-GP. The modification is intended to promote the elimination of introns but could result in the premature elimination of promising but large arrangements of genetic program nodes, a risk common to all parsimony enforcement techniques.

## 4 Methodology

To determine the efficacy of the above algorithms each was run on several well known classification tasks:

- the even-6 parity problem;

- the Wisconsin Breast Cancer Database from the UCI repository;
- the Iris Database from the UCI repository;
- the Wine Database from the UCI repository;

The treatment of the data sets, function and terminal sets used for each classification task and the parameters used for the runs are given below.

### 4.1 Data

The task for the even 6-parity problem was to evolve a classifier that returns *true* when given a string of 6 bits with an even number of ones and returns *false* when given a bit string with an odd number of ones. Performance is analyzed over the set of 64 bit strings of length six.

The Wisconsin Breast Cancer database consists of 699 cases of breast cancer to be classified as benign or malignant based on nine numerical attributes. After removing 16 cases with missing attributes the data was split into a training set of 477 instances (70%) and a testing set of 206 (30%) instances with proportional representation of benign and malignant classes in the two sets. Attributes were not normalised.

The Iris database consists of three classes of iris with 50 instances of each. There are four numeric attributes with no missing data. Data was split 70/30 into a training set of 105 instances and a testing set of 45 instances with proportional representation of each class. Attributes were not normalised.

The Wine database consists of a chemical analysis of 178 wines from three different cultivars in the same Italian region. Wines are to be classified to a cultivar according to thirteen numerical attributes representing the results of the chemical analysis. The data set was again split 70/30 into a training set of 124 instances and a testing set of 54 instances with proportional representation of each class. Attributes were not normalised.

## 4.2 Function and Terminal Sets

For the even-6 parity problem the simple logical function set shown in Table 1 was used. The only terminal for the problem is a boolean operator assigned the value of one of the 6 positions on the bit string as shown in Table 2.

The function set for the Wisconsin Breast Cancer, Iris and Wine databases is shown in Table 3. The terminals are a random number and a randomly assigned attribute as shown in Table 4.

## 4.3 Parameters

Each algorithm was trained and tested on each data set over 30 runs. Parameters for these runs are shown in Table 5. Initial populations were generated using the ramped half-and-half method [9]. Tournament selection with a tournament size of two was used. Runs were stopped if an individual capable of correctly classifying the entire training set or all 6-bit strings was found. For the three-class Iris and Wine classification tasks, classes were differentiated using static range selection with boundaries at -15.0 and 15.0 .

Table 5: Parameters for the GP runs.

Parameter	Value
Population size	300
Max Generations	200
Crossover probability	90%
Mutation probability	10%
Max Depth	10

## 4.4 Performance measurement

The main performance measures of interest for these algorithms are the computational effort needed to evolve a classifier and the classification error of the evolved classifier. Also of interest for the data classification problems are the success of the algorithms in evolving solutions to the training data and the generalizability of the solutions.

For the even-6 parity problem the percentage of runs that found a solution within the maximum number of generations, the average number of node evaluations (i.e. sum of the number of functions and terminals evaluated over the course of the run) to a solution and the average size of the final solution were all recorded. These metrics indicate the probability of success and the computational effort needed to achieve it.

For the three data classification problems the training error, testing error, average size of the best individual and the average number of node evaluations per run (i.e. the number of node evaluations until a solution to the training data set was found or, if no solution found, the number of node evaluations over the course of the run) were recorded.

## 5 Results

The POPE-GP, DecMO-GP and DecMOP-GP outperformed the standard GP on all problems with the parameter settings used. In general, the average size of individuals

was smallest when using the POPE-GP algorithm, while the average size for individuals evolved by the DecMOP-GP slightly larger and that of DecMO-GP evolved individuals larger again. The standard GP had the largest average size of individuals by a significant margin. The relatively small average size of DecMO-GP individuals is surprising as no parsimony technique is used. Other characteristics of the results are pointed out in the following sections.

### 5.1 Even 6-parity

Results are shown in Figure 2 and Table 6. The POPE-GP algorithm clearly performed best on this problem (highest proportion of successful runs and lowest node evaluations) followed by the DecMOP-GP algorithm. The standard and DecMO-GP algorithms performed poorly.

Table 6: Summary of performance results on even 6-parity problem. Standard deviations are given in brackets.

Algorithm	Percent runs finding solution	Average size of solution
Standard GP	36.7%	455 (324)
POPE-GP	<b>93.3%</b>	<b>36.3</b> (36.4)
DecMO-GP	43.3%	328 (311)
DecMOP-GP	80.0%	78.8 (65.5)

### 5.2 Wisconsin Breast Cancer Database

Results are shown in Figure 3 and Table 7. DecMOP-GP achieved the lowest error on the test data while DecMO-GP trained slightly better and tested slightly worse. However, the differences in both the testing and training errors were slight between the three algorithms developed here and all significantly outperformed the standard GP. The average size of solution is smallest for the POPE-GP and then for the DecMOP-GP algorithm (excluding two large outliers created in the zeroth and sixth generations). The size of the DecMO-GP solutions, although roughly four times that of the two parsimony enforcement techniques, is only a quarter that of the standard GP.

### 5.3 Wine Database

Results are shown in Figure 4 and Table 8. The DecMO-GP trains and tests the best but it and the DecMOP-GP generalize poorly, with excellent training results relative to the other algorithms giving way to test results only slightly better than that of POPE-GP. Relative sizes follow the pattern described for the WBC data sets. Standard GP is again the worst performer.

### 5.4 Iris Database

Results are shown in Figure 5 and Table 9. The POPE-GP algorithm achieves the lowest error rate on test data; surpris-

Table 7: Summary of results on Wisconsin Breast Cancer Database classification. Standard deviations are given in brackets. \*Without two outliers with over 2000 nodes each average size drops to 37.1 and SD to 60.6 .

Algorithm	Training Error	Testing Error	Average size of solution
Standard GP	0.0102 (0.0034)	0.0618 (0.0113)	476 (301)
POPE-GP	0.0076 (0.0026)	0.0492 (0.0154)	<b>25.6</b> (12.9)
DecMO-GP	<b>0.0060</b> (0.0024)	0.0481 (0.0148)	120 (45.9)
DecMOP-GP	0.0073 (0.0036)	<b>0.0440</b> (0.0194)	148.3* (429.49)

Table 8: Summary of results on Wine Database classification.

Algorithm	Training Error	Testing Error	Average size of solution
Standard GP	0.0341 (0.0257)	0.1451 (0.0598)	593 (620)
POPE-GP	0.0191 (0.0209)	0.1179 (0.0480)	<b>31.3</b> (26.0)
DecMO-GP	<b>0.0013</b> (0.0030)	<b>0.0912</b> (0.0351)	172 (134)
DecMOP-GP	0.0021 (0.0042)	0.1006 (0.0352)	61.9 (42.3)

ingly it is much less than the training error rate achieved. The DecMO-GP algorithm performed better on training than the DecMOP-GP algorithm but worse on test data. These results may be a result of the data partitioning. Standard GP performed worst of all with training and test error rates two to four times those achieved by the other algorithms. Size followed the pattern described for the WBC data.

Table 9: Summary of results on Iris Database classification.

Algorithm	Training Error	Testing Error	Average size of solution
Standard GP	0.0454 (0.0584)	0.0393 (0.0749)	563 (478)
POPE-GP	0.0238 (0.0074)	0.0119 (0.0140)	<b>22.5</b> (21.4)
DecMO-GP	<b>0.0089</b> (0.0056)	0.0207 (0.0164)	113 (58.3)
DecMOP-GP	0.0143 (0.0055)	<b>0.0193</b> (0.0224)	33.5 (24.6)

## 6 Discussion

The three multi-objective variants of GP explored here all achieve lower error rates with less node evaluations than the standard GP over the data sets classified for the parameters used, as shown in Figures 2 through 4.

The low average sizes achieved by the DecMO-GP are surprising as no parsimony enforcement technique is used. With only one individual at a point there is less rationale for the development of introns to protect against the disruptive effect of mutation and crossover and less opportunity for swapping useless code between similar solutions. By thus decreasing the opportunity for intron development, individuals' size appears to have been reduced without explicit parsimony enforcement techniques. More research on this and the resulting diversity could give a better understanding of the development of introns. Experimenting with a single-point standard GP or  $n$ -point ( $n=1, 2, 3, \dots$ ) standard, DecMO- or POPE-GP could be of interest to investigate the diffusion of introns among similar individuals and its effects on bloat and diversity.

The results gained suggest a weak relationship between the size and generalizability of solutions generated by the DecMO-GP and DecMOP-GP algorithms, with the smaller solutions from the latter generalizing better. The DecMO-GP and DecMOP-GP algorithms are particularly at risk of overtraining as highlighted by the Wine data classification results, although all overtrain on this data set.

Understanding of the multi-objective algorithms could be increased by applying them to data sets with more classes. Combination with more advanced classification techniques such as dynamic range selection rather than static range selection could also provide a superior classifier and should be attempted.

Algorithm modifications may also lead to a better classifier than those explored here. Adaptations may include the use of multiple classifiers from a single run as suggested in Section 4; use of a steady state rather than generational algorithm as demonstrated by FOCUS; or intra-front selection based on minimum total error rather than crowding to focus the front on the area of interest.

While the intent of this research was to investigate the performance of the multi-objective classifier algorithms relative to standard GP, future work should compare the algorithms or improved versions thereof to classifiers derived using other techniques such as decision trees, logistic regression, neural nets or Bayesian classifiers.

## 7 Conclusions

The paper has investigated the use of multi-objective genetic programming techniques for application to classification problems. The performance of a recent algorithm, POPE-GP, and a novel algorithm utilising an error-minimisation objective for each class, the DecMO-GP and its parsimony-enforcing derivative DecMOP-GP, have been investigated on several well known problems - the even-6 parity problem and the Wisconsin Breast Cancer, Iris and Wine databases from the UCI repository.

All three of the new algorithms were superior to a standard implementation of the Genetic Programming algorithm on these problems with the data set partitioning and parameter sets as described. The new algorithms achieved lower error rates while utilising less computational effort.

It is apparent that the addition of a parsimony enforce-

ment technique to the DecMO-GP algorithm evolves classifiers with comparable classification error on test data sets for less computational effort. This is despite the original DecMO-GP training better. However, as indicated by the Wine data classification results, both algorithms are susceptible to overtraining.

Overall, these results show that multi-objective techniques can be applied to the genetic programming of classifiers to decrease both error rates and the computational effort needed to evolve GP classifiers. With a range of avenues open for investigation in this area, it is hoped that these techniques can be developed to be a useful addition to the evolutionary computation practitioner's toolbox.

## Acknowledgements

This work was sponsored by a grant from the Victorian Partnership for Advanced Computing.

## Bibliography

- [1] Nordin, P. Banzhaf, W.: Complexity Compression and Evolution. In *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*. Morgan Kaufmann (1995) 310-317
- [2] Bernstein, Y., Li, X., Ciesielski, V., Song, A.: Multiobjective Parsimony Enforcement for Superior Generalisation Performance. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, IEEE Press (2004) 83-89
- [3] Bleuler, S., Brack, M., Thiele, L. and Zitzler, E.: Multiobjective Genetic Programming: Reducing Bloat using SPEA2. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*. IEEE Press (2001) 536-543
- [4] De Jong, E., Pollack, J. B.: Multi-Objective Methods for Tree Size Control. *Genetic Programming and Evolvable Machines* 4(3) 211-233. Kluwer Academic Publishers (2003)
- [5] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2) 182-197. IEEE Press (2002).
- [6] Eggermont, J., Eiben, A.E., van Hemert, J.I.: A comparison of genetic programming variants for data classification. In *Proceedings of the Third Symposium on Intelligent Data Analysis (IDA-99)* LNCS 1642. Springer-Verlag (1999) 281-290
- [7] Freitas, A.A.: A Genetic Programming Framework for Two Data Mining Tasks: Classification and Generalized Rule Induction. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*. Morgan Kaufmann (1997) 96-101
- [8] Knowles, J., Corne, D.: Approximating the Nondominated Front Using the Pareto Archived Evolutionary Strategy. *Evolutionary Computation* 8(2) 149-172. MIT Press (2000)
- [9] Koza, R. *Genetic Programming: On the programming of computers by means of natural selection*. MIT Press, 1992.
- [10] Langdon, W.B., Poli, R.: Fitness Causes Bloat. In *Second Online World Conference on Soft Computing in Engineering Design and Manufacture*. Springer-Verlag (1997) 13-22
- [11] Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation* 10(3) 263-282. MIT Press (2002)
- [12] Loveard, T. and Ciesielski, V.: Representing Classification Problems in Genetic Programming. In *Proceedings of the 2001 Congress on Evolutionary Computation*, IEEE Press (2001), 1070-1077
- [13] McIntyre, A., Heywood, M.: On Multi-class Classification by Way of Niching. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2004*. Springer-Verlag (2004) 581-592
- [14] Rosca, J.: Generality versus Size in Genetic Programming. In *Proceedings of the Genetic Programming Conference 1996 (GP-96)*. MIT Press (1996) 381-387
- [15] Smart, W.R., Zhang, M.: Classification Strategies for Image Classification in Genetic Programming. In *Proceedings of Image and Vision Computing New Zealand 2003* 402-407
- [16] Song, A., Loveard, C., Ciesielski, V.: Towards Genetic Programming for Texture Classification. In *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence*. Springer-Verlag (2001) 461-472
- [17] Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, 2001.

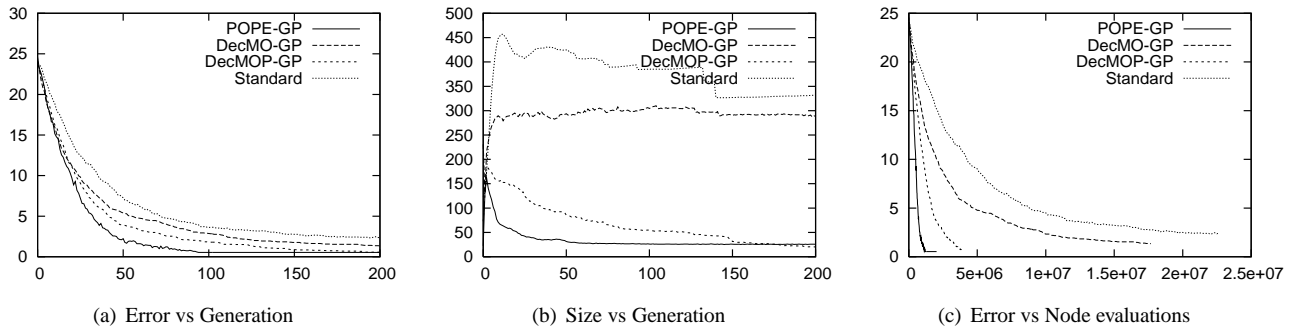


Figure 2: Results for the even 6 parity problem.

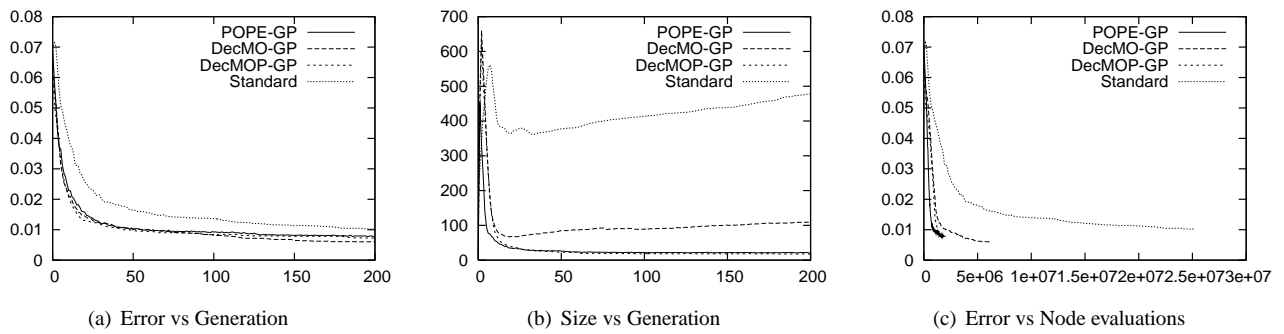


Figure 3: Results for Wisconsin Breast Cancer database classification.

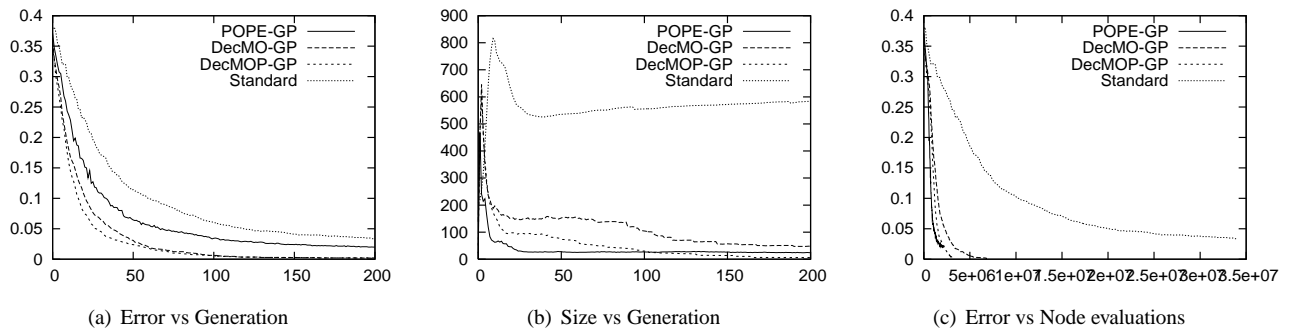


Figure 4: Results for Wine database classification.

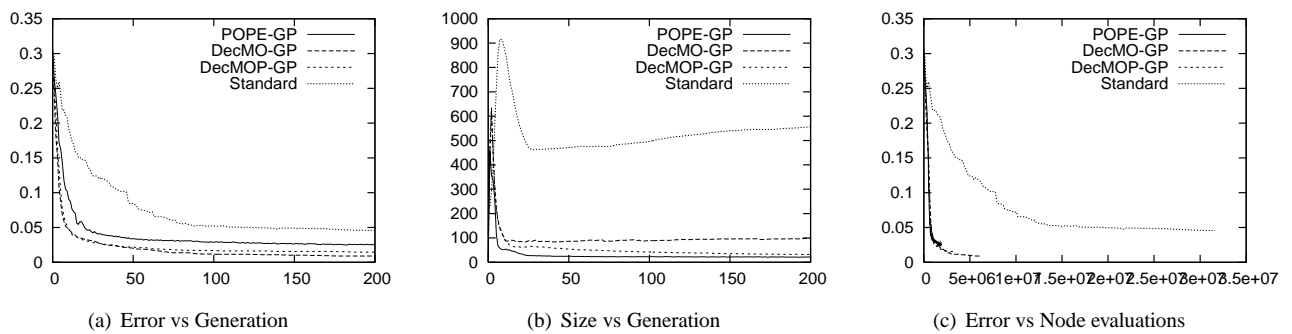


Figure 5: Results for Iris database classification.