Evolving Diverse Ensembles using Genetic Programming for Classification with Unbalanced Data

Urvesh Bhowan, Mark Johnston, Mengjie Zhang and Xin Yao

***Abstract*—In classification, machine learning algorithms can suffer a performance bias when data sets are unbalanced. Data sets are unbalanced when at least one class is represented by only a small number of training examples (called the minority class) while the other class(es) make up the majority. In this scenario, classifiers can have good accuracy on the majority class but very poor accuracy on the minority class(es). This paper proposes a Multi-objective Genetic Programming (MOGP) approach to evolving accurate and diverse *ensembles* of genetic program classifiers with good performance on both the minority and majority classes. The evolved ensembles comprise of non-dominated solutions in the population where individual members vote on class membership. This paper evaluates the effectiveness of two popular Pareto-based fitness strategies in the MOGP algorithm (SPEA2 and NSGAII), and investigates techniques to encourage diversity between solutions in the evolved ensembles. Experimental results on six (binary) class imbalance problems show that the evolved ensembles outperform their individual members, as well as single-predictor methods such as canonical GP, Naive Bayes and Support Vector Machines, on highly-unbalanced tasks. This highlights the importance of developing an effective fitness evaluation strategy in the underlying MOGP algorithm to evolve good ensemble members.**

***Index Terms*—Genetic Programming, Classification, Class Imbalance Learning, Multi-objective Machine Learning**

I. Introduction

Classification with unbalanced data presently represents a major obstacle in machine learning (ML) [1][2][3]. Data sets are unbalanced when the learning examples from at least one class are *rare*. In binary classification, the class with the smaller number of examples is called the minority class, while the other class is the majority class. Unbalanced data sets are common; fraud detection [4], medical diagnostics [5], and image recognition [6] are only a few examples.

Genetic Programming (GP) is an evolutionary technique based on the principles of evolution or natural selection which has been widely successful in evolving reliable and accurate classifiers to solve a range of real-world classification problems [7]. However, GP, like many other ML techniques, can evolve classifiers "biased" toward the majority class when data is unbalanced [4][2][3]. Biased classifiers have strong classification accuracy on one class but weak accuracy on the other. This can occur because typical training criteria such as the overall accuracy or error rate can be influenced by the larger majority class.

Good accuracy on the minority class is usually *at least* as important as, or in some scenarios more important than, the majority class accuracy. However, these two learning objectives are usually in conflict; increasing the accuracy of one class can result in lower accuracy on the other. Evolutionary multi-objective optimisation (EMO) is a useful technique to capture this trade-off in the learning process [8][9][10]. EMO is often advantageous over canonical (single-objective) optimisation techniques because a *front* of the best trade-off (non-dominated) solutions along the objectives can be evolved simultaneously in a single optimisation run, without requiring the objective preference to be specified *a priori*.

In this paper we develop a multi-objective GP (MOGP) approach to classification with unbalanced data, using the minority and majority class accuracy as competing objectives in the learning process. Our first goal is to compare two popular Pareto-based fitness schemes in the MOGP algorithm, namely, SPEA2 [9] and NSGAII [10], across a number of classification tasks with unbalanced data. Recent work has shown that while NSGAII can be effective in evolving a good set of non-dominated solutions in some tasks, this performance needs to be improved for difficult classification problems [11]. We hypothesise that SPEA2 can evolve better-performing classifiers on these tasks as this strategy is known to better exploit the middle-region of the frontier; whereas NSGAII tends to reward exploration at the end-regions.

Another key advantage in simultaneously evolving a set of highly-accurate classifiers along the minority and majority class trade-off frontier is that the combined classification ability of these non-dominated solutions can be used *co-operatively* in an ensemble. However, for an ensemble to be more accurate than any of its individual members, the ensemble members must be diverse, i.e., make different errors on different inputs. The second goal of this paper is to adapt the MOGP approach to evolve diverse solutions which can be successfully combined into an ensemble, to further improve classification performance. We will investigate whether the stochastic way in which new solutions are created in the evolutionary process is sufficient to evolve diverse ensembles, and compare two measures in the fitness function to encourage diversity among the evolved solutions, namely, negative correlation learning [12][13][14] and pairwise failure crediting [15]. Finally, we compare the classification performance of the evolved MOGP ensembles using three different voting

U. Bhowan, M. Johnston and M. Zhang are with the Evolutionary Computation Research Group, Victoria University of Wellington, New Zealand.

X. Yao is with CERCIA, School of Computer Science, The University of Birmingham, UK.

strategies to canonical (single-objective) GP approaches and two other learning algorithms, Naive Bayes and Support Vector Machines, on the tasks.

The rest of this paper is organised as follows. Section II outlines the related work. Section III discusses the GP framework and the MOGP approaches. Section IV presents the experimental results comparing the two MOGP fitness schemes. Section V adapts the MOGP approach to evolving ensembles. Section VI discusses the MOGP ensemble results, and compares these to canonical GP and other machine learning approaches. Section VII outlines some further discussions in this work. Section VIII concludes this paper and gives directions for future work.

II. Related Work

In this section we discuss the related work for class imbalance and ensemble learning, and their limitations.

A. Overview of Related Work for Class Imbalance

Addressing the class imbalance learning bias tends to involve two major aspects [1]. The first uses various sampling techniques to create an artificially balanced distribution of class examples for training. Common approaches include random over-sampling of the minority class [2], random under-sampling of the majority class [3], editing or removing noisy or atypical majority class examples [16], or synthetic over-sampling (SMOTE) to create "new" minority examples by interpolating between similar known examples [17]. Bagging and boosting techniques which train multiple classifiers using different (usually balanced) subsets of class examples (bootstrap samples), are also popular sampling-based methods [4][18][19][20][21]. While these approaches can be effective, some sampling algorithms can incur a computational overhead when the samples are dynamically composed (such as active learning), and lead to over-fitting if active learning is not assumed [1].

The second aspect uses cost adjustment within the learning algorithm to factor in the uneven distribution of class examples, using the unbalanced data set "as is" in training. In GP, this includes using fixed misclassification costs for minority and majority class examples [5][22] or better training criteria such as the *area under the receiver operating characteristics (ROC) curve* (known as AUC) [3], in the fitness function. The AUC is a useful performance measure across *multiple* true positive (TP) and false positive (FP) rates. In GP, some work has also focused on developing new fitness functions for classification with unbalanced data [23]; while some approaches combine sampling and cost adjustment [24][25]. While many of these methods show improved minority class performances, particularly when the AUC is used in the fitness function, they have limitations. Adapting the fitness function for cost-adjustment can require that misclassification costs for each class be determined *a priori*, while some improved measures such the AUC can significantly increase training times due to the computational overhead required to calculate these measures in fitness evaluation [23].

B. Ensemble-based Learning

Much work has shown that ensemble-based machine learning approaches to classification can outperform canonical single-predictor classifiers [8][13][26][27]. By considering the contributions or outputs of multiple accurate classifiers, each able to specialise on different parts of the input-space, the ensemble can improve generalisation ability and reduce the risk of overfitting; while single-predictor approaches are required to map the full input-space using only one classifier. However, constructing good ensembles is a difficult problem. In [26], the authors discuss the two main techniques (among others) to generate diverse and accurate ensembles. The first involves manipulating or dividing up the input-space into many subsets which are used to train the different ensemble members, such as bagging and boosting techniques [4][18][27]. In [27], several bagging and boosting methods with different ensemble sizes are compared; results using 26 benchmark UCI tasks show that the ensembles usually outperform single-predictors and that small ensembles (10-25 predictors) give the best results. However, techniques such as bagging which partition the input space can suffer from similar limitations to other sampling-based techniques (e.g. over-fitting).

The second approach involves injecting randomness into the learning process. This technique is favoured in evolutionary algorithms (EAs) due to its inherent stochastic and population-based nature. EAs have been combined with bagging and boosting techniques for ensemble diversity [19][28] (discussed in the next section). Many EA approaches use an additional penalty term in the fitness function such as negative correlation learning (NCL) [15][13][29] to encourage ensemble diversity, or use cooperative co-evolutionary methods (such as "teaming" in GP) [30][31]. These approaches differ from typical bagging and boosting techniques as most of them typically use the *full* training set in learning to promote interaction and cooperation in the ensemble, whereas bagging techniques sample the data into smaller subsets during learning.

EMO approaches which use a diversity objective in the fitness function typically build the ensembles using the set of non-dominated individuals in the population [15][13][32][29]. In [29], the training accuracy is traded-off against the NCL in a two-objective EMO (called DIVACE) to evolve neural network (NN) ensembles. In [15], two diversity measures are compared using DIVACE to evolve NN ensembles: NCL and a measure called pairwise failure crediting (PFC). PFC shows better generalisation than NCL on two benchmark UCI tasks. In [13], EMO with three objectives is used to evolve a Radial Basis Function (RBF) network ensemble: the accuracy, NCL and a new regularization term to penalise large network weights to improve generalisation. The new approach is shown to outperform a two-objective version using only accuracy and NCL, particularly on noisy problems.

Some EMO approaches use other mechanisms for diversity [8][32][33]. In [8], the structure of the NN models (e.g. number of hidden nodes) is varied for diversity, and this is traded-off against their error rates where small accurate models are preferred. The authors conclude that choosing good non-dominated solutions for the ensembles is difficult and can be

problem-specific; a region of the frontier is manually selected for each of their classification tasks. In [32][33][34], two multi-objective formulations are proposed to evolve neuro-ensembles. The first splits the training set into two subsets and uses the error on the subsets as the learning objectives, while the second adds Gaussian noise to the training set as the second objective. The first formulation shows better results than the second, and these methods are competitive compared to NCL on two (binary) benchmark classification tasks.

Non-Pareto EAs with anti-correlation measures in the fitness function have also shown success in ensemble learning [35][36][37]. In [35], a new measure, root quartic NCL, shows better results than traditional NCL using a grammar-guided GP on a 6-multiplexer problem. Both measures are theoretically analysed to explain how each creates diversity in a population: the new measure creates widely separated but small clusters of points in the population, while NCL increases the distances of the points to the overall mean of the points. In [37], a fitness function is developed which first assigns a *difficulty* weighting to each training example, and uses the sum of example weights correctly learned by a given individual as the fitness of that individual. The example weights correspond to the number of individuals which incorrectly classify that example.

Cooperative co-evolution (or teaming) methods have also shown success in ensemble learning in GP [30][31]. In [30], teams of individuals in linear GP are applied to two benchmark classification tasks and a regression task, and several ensemble combination schemes are evaluated (discussed in the subsection below). In [31], four methods which vary the way selection and replacement is performed on teams and team members, are compared on two multiclass UCI classification tasks. In canonical teaming, selection and replacement is done exclusively between teams or individuals, while the new "orthogonal evolution" algorithms use individual selection with team replacement, and vice versa. The new methods produce better results than canonical methods. However, it is important to note a major difference between teaming in GP and the methods used is this paper. Teaming typically produces teams of weak individuals that cooperate strongly together, as shown in both [30] and [31]. In this paper, the GP classifiers are "strong" individuals (i.e. relatively accurate classifiers) where the two diversity-based fitness functions encourage cooperation between individuals.

Ensemble Selection: Much work also addresses how to choose which learned base classifiers to include in the final ensemble [28][30][37][38][39][40]. In [38], the fittest individuals in the population are selected for the ensemble using a weighted average of the accuracy and diversity of each individual. In [28] and [39], weights which specify the importance of each ensemble member's contribution are optimised (post-training) using a validation set. Both works use a genetic algorithm (GA) to optimise the ensembles, while [39] also compares two weighted-vote schemes. These include a fitness-weighted majority-vote and a recursive least-squared (RLS) algorithm to minimise the error of the ensemble. These works show that the weighted/GA-optimised ensembles perform better than traditional majority voting where all base classifiers contribute equally [28][39]. A probabilistic ensemble pruning approach is proposed in [40] to approximate the member weights, called "expectation propagation".

In [30], several ensemble combination schemes are evaluated in a GP-based teaming approach. These include the average of the members outputs, a majority vote and two winner-takes-all schemes; and two weighting schemes where weights of teams are co-evolved in parallel (with teams), or optimized after each generation (for the best team) using a perceptron learner. The combination schemes tend to be problem-specific (none produced the best results for all three benchmark problems), but the weighting schemes usually improved performances (compared to without), while the winner-takes-all performed the worst (as the ensemble output is reduced to only one member).

In [37], offline and online ensemble selection algorithms (off-EEL and on-EEL) are proposed. Given a pool of learned base classifiers sorted by fitness, each classifier is removed from the pool and copied into the ensemble where, at each step, the ensemble is evaluated. Once the pool is empty, the ensemble with the best performance is taken as the final ensemble. Off-EEL is run once after the training cycle, while on-EEL is performed at each generation. Off-EEL performs better than on-EEL on six UCI classification tasks, as on-EEL can be prone to noise. This paper uses off-EEL [37] to build MOGP ensembles and compares these results to an accuracy-based ensemble selection technique [41].

C. Ensemble Learning for Class Imbalance

Combining ensemble learning with sampling techniques, i.e., under-sampling, over-sampling and SMOTE, to create *balanced* bootstrap samples is a popular approach to classification with unbalanced data [19][42][43][28]. In [42], two new under-sampling methods are developed to create balanced bootstrap samples for a boosting algorithm, and compared to 13 other sampling and boosting approaches in the literature. Similarly, in [18] a new SVM-based under-sampling approach iteratively collects support-vectors found using balanced bootstrap samples which are aggregated in the final classification step. In [4], base classifiers trained on balanced and unbalanced bootstrap samples are compared using a bagging approach for fraud detection in e-Commerce transactions. Using the overall classification accuracy in training, base classifiers trained on balanced samples are found to be more effective.

EMO has also been combined with bagging in ensemble learning [19][44]. In [19], a problem-decomposition approach (e.g. one-vs-rest) is used to evolve a population of binary classifiers for two UCI benchmark tasks with many different minority classes. Using grammatical evolution (GE), two populations are co-evolved for increasing ensemble diversity: classifiers and "points" (balanced bootstrap samples). Three objectives are used: overall error, the level of overlap between correctly learned "points", and a parsimony objective favouring smaller solutions. A winner-takes-all approach of the non-dominated solutions in the evolved populations determines the final prediction. A more thorough description of the algorithm which is also evaluated on more multi-class problems and model complexities is provided in [44].

Theoretical studies in this area also suggest a relationship between ensemble diversity and class performance [21][43]. Increasing ensemble diversity is found to improve minority class accuracies but degrade majority class accuracies in unbalanced data sets. However, the accuracies of both classes improve together when ensemble diversity is increased in balanced data sets. These studies use eight binary and multiple class benchmark tasks from the UCI repository.

Recent work also uses NCL in fitness for class imbalance [20][41]. In [20], NCL is only applied to instances from the minority class (majority class instances are ignored). This adaptation (NCLCost) aims to maximise ensemble accuracy and diversity on the minority class, but only accuracy on the majority class. Comparing NCLCost to a bagging approach and traditional NCL (calculated with all training instances), both NCL-based methods show the best minority class accuracies, while traditional NCL shows the best diversity. The average class accuracies for NCLCost is higher than traditional NCL, suggesting that very high diversity can negatively impact on majority class accuracy (similar to [21]). In [41], NCL is combined with SPEA2 to evolve a diverse set of Pareto front classifiers in a GP approach for ensemble learning for class imbalance. An ensemble selection strategy is used to select only accurate Pareto front classifiers for the ensembles.

D. Contributions of This Paper

While previous ensemble approaches show good results on some unbalanced data sets, there are some limitations which this work tries to address. Much work uses NNs, decision trees and Naive Bayes as the base classifiers [4][20][13][43][28]. GP has shown much success in evolving accurate classifiers for classification with unbalanced data [11][19][23][24][45]; however, there is little work which investigates whether these performances can be improved using a multi-objective GP framework to evolve ensembles of GP classifiers [19][41]. This paper develops a MOGP approach to evolving accurate and diverse ensembles and compares this performance to both canonical (single-predictor) GP and other methods.

Another important difference between our work and other approaches is that many of the existing works rely on sampling techniques to either create balanced bootstrap samples when training bagging approaches [4][19][43], or re-balance the training data when diversity measures (such as NCL) are used in fitness evaluation [20][21]. Our approach uses the original unbalanced training data "as is" in the GP learning process without the need to artificially re-balance the class distributions in the data sets. This allows us to concentrate on the EMO and diversity measures in the MOGP algorithm, and remove any dependence on a sampling algorithm.

There has also been very little work which focuses on adapting the ensemble diversity measures in fitness to account for the skewed class distributions. Some works calculate diversity on all examples irrespective of class when data is balanced [29][15][15], or first re-balance the training data *prior* to the diversity calculation when data is unbalanced [20]. Further, in [20] the diversity with respect to majority class instances is ignored in fitness, utilising only minority class diversity. In contrast, this paper compares two diversity measures (NCL and PFC) in GP, and both are adapted to calculate diversity *separately* for each class using the original unbalanced training data. The diversity on the minority and majority class then contributes equally in fitness evaluation to ensure that the ensembles are equally diverse on both classes.

As discussed, much work in ensemble learning also focuses on how to choose the best base classifiers for the ensembles. Some research provides recommendations on how to choose ensemble sizes *a priori* and suggests that smaller ensembles can be more accurate than large ensembles [8][27][38]. Some others improve ensemble performances using ensemble selection algorithms [37], or optimise the weights specifying each member's contribution to the ensemble [28][30][39]. We try to address this in MOGP by comparing two voting schemes (traditional majority vote and a fitness-weighted vote similar to [39]), and two ensemble selection strategies (an accuracy-based approach used in [41] and off-EEL [37]) in this paper.

III. GP Approaches

In this section we discuss the GP framework for classification, and outline the GP and MOGP approaches.

A. GP Framework for Classification

A tree-based structure is used to represent genetic programs [7]. We use feature terminals (example features) and constant terminals (randomly generated floating point numbers), and a function set consisting of the four standard arithmetic operators, $+, -, \times$, and $\%$, and a conditional operator, `if`. The $+$,$-$ and $\times$ operators have their usual meanings (addition, subtraction and multiplication) while % means *protected* division (usual division except that a divide by zero gives a result of zero). The conditional `if` function takes three arguments. If the first is negative, the second argument is returned; otherwise the third argument is returned. Each genetic program represents a mathematical expression that returns a single output value (floating-point number) for a given input (data example to be classified). This number is mapped onto a set of class labels using zero as the class threshold, i.e., an example is assigned to the *minority* class if the output of the genetic program classifier is zero or positive, or the *majority* class otherwise.

B. Canonical (Single-objective) GP

The standard fitness measure in classification is the overall classification accuracy. This is the number of examples correctly predicted by a classifier as a fraction of the total number of training examples. Using the four outcomes for binary classification shown in Table I and assuming the minority class is the *positive* class, the overall classification accuracy can be defined by Acc (Eq. 1).

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

In classification with unbalanced data, Acc can favour the evolution of solutions *biased* toward the the majority class [2][3][4]. This is because Acc does not take into account the

TABLE I
OUTCOMES OF BINARY CLASSIFICATION.

	Predicted Object	Predicted non-object
Actual Object	True Positive (TP)	False Negative (FN)
Actual non-object	False Positive (FP)	True Negative (TN)

smaller number of examples in the minority class. For example, if a classification task has a minority class represented by only 10% of available learning instances, a trivial solution can score a high fitness (e.g. 90% overall accuracy) by assigning *all* the instances to the majority class.

These two objectives, i.e., minority and majority class accuracy, are usually in conflict where increasing the accuracy on one class results in a trade-off in performance in the other class [5][11]. To capture this trade-off, the fitness function Ave, defined by Eq. (2), uses the weighted average classification accuracy of the minority and majority classes. In Ave, minority accuracy corresponds to the true positive rate (TPR), majority accuracy is the true negative rate (TNR), and weighting coefficient W specifies the relative importance of the minority accuracy to majority accuracy where $0 < W < 1$. When *W* is 0.5, the accuracy of both classes is considered as equally important in fitness. When $W > 0.5$, minority class accuracy will contribute more in the fitness function than majority class accuracy. Similarly, majority class accuracy will contribute more when $W < 0.5$.

$$Ave = W(TPR) + (1 - W)(TNR) \tag{2}$$

where

$$TPR = \frac{TP}{TP + FN} \quad \text{and} \quad TNR = \frac{TN}{TN + FP} \tag{3}$$

C. Multi-objective GP (MOGP)

While Ave has been shown to evolve solutions with better accuracy on both classes in some class imbalance tasks compared to Acc, a major limitation of Ave is that the objective preference must be specified *prior* to the evolutionary search. In real-world classification tasks, determining a good weighting coefficient can be a lengthy trail and error process, requiring multiple optimisation runs with different weighting coefficients. Evolutionary multi-objective optimisation (EMO) offers a useful solution to the problem of optimising multiple conflicting objectives [8][9][10]. The aim of EMO is to simultaneously evolve a *front* of the best trade-off solutions along the objectives in a single optimisation run.

1) Pareto Dominance in Fitness: An important aspect in EMO is the notion of Pareto dominance in fitness [9][10]. This allows solutions to be ranked according to their performance on all the objectives with respect to all solutions in the population. This ranking is important as it affects the way selection is performed if the objectives are to be treated separately in the evolution. In this approach, the two objectives are the classification accuracy of the minority and majority class. Recall that the minority class accuracy is the TPR and majority class accuracy is the TNR, as defined in Eq. (3).

In Pareto dominance, a solution will *dominate* another solution if it is at least as good as the other solution on all the objectives and *better* on at least one. As the two objectives in this approach are to be maximised, this concept is expressed using Eq. (4), where $(S_i)_m$ denotes the performance of solution S_i on the mth objective. Solutions are *non-dominated* if they are not dominated by any solution in the population.

$$S_i \succ S_j \longleftrightarrow \forall m[(S_i)_m \geq (S_j)_m] \wedge \exists k[(S_i)_k > (S_j)_k] \tag{4}$$

2) Two Pareto-based Dominance Measures: Two common Pareto-based dominance measures are the dominance rank [10] and dominance count [9] of a given solution. Dominance rank is the number of other solutions in the population that dominate a given solution (lower is better), whereas dominance count is the number of other solutions that a particular solution dominates (higher is better). Each measure has a different bias towards solutions on the Pareto frontier: dominance rank is known to reward exploration at the edges of the frontier while dominance count tends to reward exploitation in the middle of frontier. Two popular EMO approaches which use these measures include SPEA2 [9] and NSGAII [10]; SPEA2 uses both dominance rank and dominance count, while NSGAII uses only dominance rank.

In NSGAII, the fitness value for the solution S_i is its dominance rank, that is, the number of other solutions in the population that dominate S_i, given by Eq. (5). A non-dominated solution will have the best fitness of 0, while high fitness values indicate poor-performing solutions, i.e., solutions dominated by many individuals. Fitness in NSGAII is to be minimised. This scheme is illustrated in Figure 1(a).

$$NSGAII(S_i) = |\{j | j \in Pop \wedge S_j \succ S_i\}| \tag{5}$$

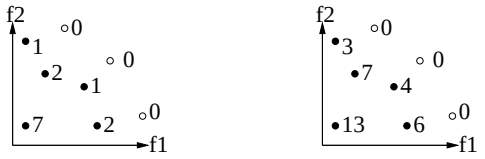
In SPEA2, both dominance rank and dominance count are used in fitness. First, each solution in the population is assigned a *strength* value D; this is the dominance count for solution S_i, i.e., the number of solutions it dominates:

$$D(S_i) = |\{j | j \in Pop \wedge S_i \succ S_j\}|$$

The fitness value for a given solution is determined by the strengths of all its dominators, given by Eq. (6). In this equation, the final fitness value for solution S_i is the sum of all dominance counts of other solutions in the population that are dominated by S_i. Similar to NSGAII, fitness here is to be minimised where non-dominated solutions have the best fitness of 0. This scheme is illustrated in Figure 1(b).

$$SPEA2(S_i) = \sum_{j \in Pop, S_i \succ S_j} D(S_j) \tag{6}$$

In this paper we compare which of these two dominance measures evolves better-performing frontier solutions for our classification tasks (using the same evolutionary search algorithm, outlined in subsequent sections). Previous work has shown that while NSGAII is successful in some tasks, the evolved Pareto front solutions exhibit poor accuracy compared to canonical (single-objective) GP in difficult problems [11]. We attribute this to the presence of large numbers of highly biased solutions along the *edge-regions* of the evolved fronts, i.e., solutions with high accuracy on one class only [11]. We hypothesise that a fitness measure which better rewards



(a) NSGAII Fitness (b) SPEA2 Fitness

Fig. 1. MOGP fitness using (a) NSGAII dominance ranking and (b) SPEA2 "strength" values. Filled circles are dominated whereas non-filled circles are non-dominated. NSGAII fitness has dominated solutions with identical fitness values whereas these solutions have unique fitness values in SPEA2.

exploitation in the middle of the Pareto frontier can evolve better-performing solutions in the middle region of the frontier.

This paper only compares NSGAII and SPEA2. These two popular algorithms are chosen because each uses the two main Pareto-based dominance measures in fitness, i.e., dominance rank and dominance count, in different ways to evolve Pareto fronts. A comparison with other EMO algorithms (such as [46]) is beyond the scope of this work and will be left for future work. However, the experimental results from this investigation can provide useful new directions to further improve the fitness measure.

3) Crowding in Fitness: In addition to Pareto dominance, we also use a secondary "crowding" distance measure to promote a good spread of solutions along the trade-off frontier. Crowding is the Manhattan distance between solutions in *objective-space*, where sparsely populated regions of objective-space are preferred over densely populated regions. Crowding is only used to resolve selection when the primary fitness (Pareto dominance measure) is equal between two or more individuals. This means that if two or more individuals have the same Pareto rank, the individual with the better crowding distance is preferred. The MOGP approaches using the two fitness schemes both use the same crowding measure, that is, average distance to the two neighbouring solutions on either side of the given solution along each of the objectives [10]. Solutions with larger crowding distances indicate that their nearest neighbours are far apart; these are preferred to smaller distance values. Details can be seen in [10].

4) MOGP Search Algorithm: The MOGP evolutionary search here is based on the algorithm used in NSGAII [10]. The parent and offspring populations are merged together at every generation. The fittest individuals in this merged parent-child population are then copied into a new population (called the archive population). The archive population serves as the parent population for the next generation (the archive population is the same size as the original parent population). At every generation the offspring population is generated using traditional crossover and mutation operators. The archive population is used to simulate elitism in the population, that is, to preserve the set of non-dominated solutions over generations.

We use this search algorithm as it is fast and very similar to the SPEA2 algorithm. The only difference between this algorithm and the original SPEA2 algorithm is that SPEA2 uses an additional truncation operator in the archive population to remove non-dominated solutions with very similar performances on the objectives. We ignore this additional operator as it requires one more evolutionary search parameter to configure, and the "crowding" measure can achieve a similar effect in the selection process.

IV. Evaluation of MOGP Fitness Schemes

In this section we discuss the evolutionary parameters and data sets used in the experiments, and evaluate and compare the MOGP approaches using the Pareto-based fitness schemes.

A. Evolutionary Parameters and Data Sets

The ramped half-and-half method is used for generating programs in the initial population and for the mutation operator [7]. For both MOGP and GP approaches, the population size is 500, maximum program depth is 8 (to restrict very large programs in the population), and the evolution is allowed to run for a maximum of 50 generations or terminated when a solution with optimal fitness is found. For the GP approaches, crossover, mutation and elitism rates were 60%, 35% and 5%, respectively, and tournament selection is used with a tournament size of 7. For the MOGP approaches, crossover and mutation rates were 60% and 40%, respectively, and tournament selection is used with a tournament size of 2 (these MOGP settings follow those recommended in [10][9]). Note that due to the nature of the MOGP approaches, further elitism is not required.

Six benchmark binary classification problems, summarised in Table II, are used in the experiments. These are taken from the *UCI Repository of Machine Learning Databases* [47] and the Intelligent Systems Lab at the University of Amsterdam [6]. For each task, half of the examples in each class were randomly chosen for the *training* and the *test* sets. This ensures that both training and test sets preserve the same class imbalance ratio as the original data set.

These benchmark data sets are carefully selected to encompass a varied collection of problem domains to ensure that our evaluation of the different MOGP approaches is not problem-specific. These problems have varying levels of class imbalance (minority class ranges between 7–35% of total examples), and complexity where some tasks are easily-separable (e.g. Yst_2) compared to others. The training/test sets also range from being well-represented (Ped has approximately 12000 instances), to sparsely represented (Spt has 134 instances, only 27 from the minority class). These tasks also range between high and low dimensionality (Ion has 34 features while Bal only has 4), and binary and real-valued feature types. We expect that these data sets can represent class imbalance problems of varying difficulty, dimensionality, size and (feature) types reasonably well.

B. Evaluating Front Hyperarea

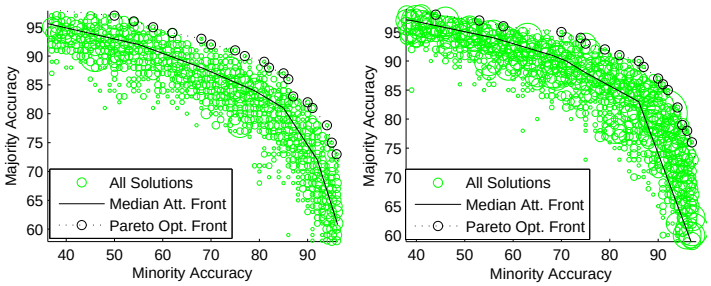
We use the *hyperarea* (also known as the hypervolume) [48] of the evolved Pareto-approximated fronts as a "single figure" to measure which MOGP fitness scheme is better on these tasks. The hyperarea is the area under the Pareto-approximated front in *objective-space* [48], similar to the area under the ROC curve (or AUC). However, while the AUC represents the performance of a single classifier at varying classification thresholds, the hyperarea represents the classification performance of the *set* of classifiers along the front. The hyperarea is calculated by taking the sum of the

TABLE II
UNBALANCED CLASSIFICATION TASKS USED IN THE EXPERIMENTS.

Name	Classes (Minority/Majority)	Number of Examples Total	Minority	Majority	Imb. Ratio	Features No.	Type
Ion	Good/bad (ionosphere radar signal)	351	126 (35.8%)	225 (64.2%)	1:3	34	Real
Spt	Abnormal/normal (cardiac tomography scan)	267	55 (20.6%)	212 (79.4%)	1:4	22	Binary
Ped	Pedestrian/background (image cut-out)	24800	4800 (19.4%)	20000 (80.6%)	1:4	22	Real
Yst$_1$	*mit*/non-target (protein sequence)	1482	244 (16.5%)	1238 (83.5%)	1:6	8	Real
Yst$_2$	*me3*/non-target (protein sequence)	1482	163 (10.9%)	1319 (89.1%)	1:9	8	Real
Bal	Balanced/unbalanced (balance scale)	625	49 (7.8%)	576 (92.2%)	1:12	4	Integer

TABLE III
AVERAGE ($\pm$ STANDARD DEVIATION) HYPERAREA OF EVOLVED PARETO-APPROXIMATED FRONTS, PARETO-OPTIMAL (PO) FRONT, AND TRAINING TIMES (IN SECONDS 'S' OR MINUTES 'M') FOR THE MOGP APPROACHES OVER 50 RUNS.

Task	NSGAII Fitness Hyperarea Average	NSGAII Fitness Hyperarea PO Front	NSGAII Fitness Training Time	SPEA2 Fitness Hyperarea Average	SPEA2 Fitness Hyperarea PO Front	SPEA2 Fitness Training Time
Ion	0.793 $\pm$ 0.041	0.952	8.3s $\pm$ 1.3	**0.848 $\pm$ 0.041**	**0.992**	9.3s $\pm$ 2.4
Spt	0.733 $\pm$ 0.026	0.938	16.9s $\pm$ 2.1	0.732 $\pm$ 0.032	**0.971**	9.7s $\pm$ 2.5
Ped	0.881 $\pm$ 0.013	0.903	3.5m $\pm$ 52.6	**0.902 $\pm$ 0.019**	**0.922**	3.9m $\pm$ 1.1
Yst$_1$	0.793 $\pm$ 0.008	0.917	23.5s $\pm$ 4.5	0.793 $\pm$ 0.009	**0.931**	20.8s $\pm$ 7.1
Yst$_2$	0.942 $\pm$ 0.008	0.986	23.5s $\pm$ 4.4	**0.949 $\pm$ 0.011**	**0.991**	20.1s $\pm$ 8.1
Bal	0.749 $\pm$ 0.049	**0.993**	20.1s $\pm$ 2.6	0.757 $\pm$ 0.063	0.985	15.2s $\pm$ 3.9



(a) NSGAII Fitness (b) SPEA2 fitness

Fig. 2. Accuracy of all evolved solutions (circles), median attainment surface (solid line) and Pareto-optimal front (dotted line) for the MOGP approaches over 50 runs (on the Ped task). Circle size is proportional to frequency.

areas of individual trapezoids fitted under each front solution in objective-space [48]. Hyperarea values range between 0 and 1 where the higher the value, the better the performance.

Table III reports the average (and standard deviation) hyperarea of the evolved Pareto-approximated fronts on the *test* set, as well as the average training times in seconds (s) or minutes (m), for the two MOGP approaches over 50 runs. Table III also includes the hyperarea of the *Pareto-optimal* (PO) front with respect to all MOGP runs. The PO front is the set of non-dominated solutions from the union of all Pareto-approximated fronts evolved from the 50 independent runs. For example, Figure 2 shows the PO front (dotted line) of the 50 different evolved Pareto-approximated fronts for the MOGP approaches (on the Ped task). In Figure 2, each circle represents an evolved solution in objective-space where circle size is proportional to frequency (the larger the circle, the more populated a particular point in objective-space).

In Table III, the MOGP approach with the significantly better hyperarea is highlighted in bold for each task. The statistical significance test (of the average hyperarea) is calculated using the common random numbers technique at a 95% level of significance. This technique computes the 95% confidence interval of the hyperarea differences between the two MOGP approaches, on a run-by-run basis over 50 independent runs.

According to Table III, SPEA2's average hyperarea is statistically better than NSGAII on the three tasks, and not statistically different to NSGAII on the remaining three tasks. The hyperarea of the Pareto-optimal (PO) front is also better in SPEA2 for all tasks except Bal (where NSGAII is better). Table III also shows that the two MOGP approaches show similar average training times.

C. Comparing Pareto Fronts and Canonical GP

To investigate why the MOGP approach using SPEA2 is able to outperform NSGAII on three out of the six tasks, as well as compare the classification performance of the Pareto-approximated fronts to canonical (single-objective) GP, we use *attainment summary surfaces* to approximate an "average" evolved front over 50 independent runs for each MOGP approach on the tasks. Attainment summary surfaces are a useful technique to summarise the outcome of a series of multi-objective experiments, where a potentially different set of non-dominated solutions can be returned from each MOGP run [49]. Each attainment surface comprises of evolved solutions (from all runs) that have identical *attainment* values, where the number of attainment surfaces correspond to the number of MOGP runs (50). A solution's attainment value is the probability that the MOGP system will evolve another solution which *weakly dominates* the given solution on all objectives [49]. The *median* attainment surface, i.e., the set of solutions with attainment values of 0.5, corresponds to those solutions with a 50% probability of attainment with respect to all runs. This set represents an "average" evolved front over 50 independent runs. For example, the solid line in Figure 2 shows the "average" evolved front (median attainment surface) over 50 runs for the MOGP approaches (on the Ped task).

The "average" evolved front and Pareto-optimal front for the two MOGP approaches over independent 50 runs (on the *test* set) is shown in Figure 3 for the six tasks. Figure 3 also includes the average performance of the fittest evolved solution on the two classes (also on the test set) using canonical (single-objective) GP with the two fitness functions, Acc (Eq. 1) and Ave (Eq. 2), over 50 runs. The GP fitness function Ave uses seven different weighting coefficients between 0.2 and 0.8 (at intervals of 0.1). The GP training times using either Acc and

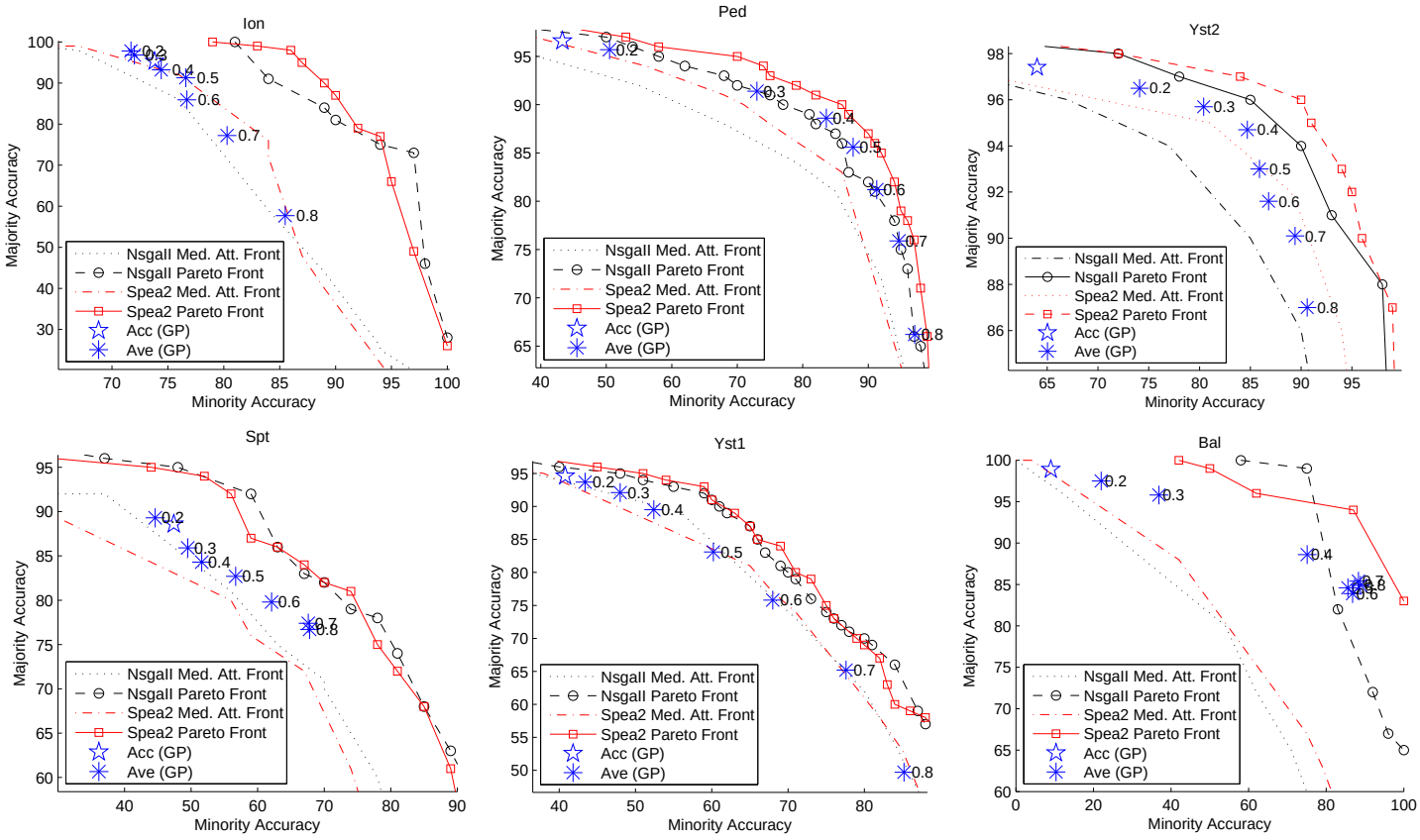


Fig. 3. Classification performance of evolved solutions using two MOGP approaches (NSGAII and SPEA2 fitness), and single-objective GP using two fitness functions (Acc and Ave). The top row shows those tasks where the average hyperarea using SPEA2 fitness is statistically *better than* NSGAII fitness (there is no significant difference in hyperarea for the tasks in the bottom row).

Ave (with a particular weighting coefficient) are very similar to each other on the tasks. On average this is approximately 1.4 seconds for Ion and Spt, 3.2 seconds for Bal, 6.6 seconds for the Yst tasks, and 2.2 minutes for Ped.

Figure 3 shows that the evolved Pareto fronts, particularly the MOGP approach with SPEA2, contain an accurate set of solutions along the minority and majority class trade-off frontier for these tasks. This highlights an important advantage of MOGP over canonical (single-objective) GP: a *single* run of the MOGP algorithm can trace out a good set of trade-off solutions, leaving the final choice for the decision-maker. In contrast, in canonical GP, this trade-off must be determined *a priori* needing multiple GP runs using Ave (one for each weighting coefficient) to generate a frontier. Although a single run of the single-objective GP method uses less time than the MOGP approaches, the single-objective GP method requires a much longer time to get a reasonable Pareto front.

Figure 3 also shows that the standard GP fitness function Acc evolves biased solutions on the tasks with high majority class accuracies but low minority class accuracies.

Further inspection of the results in Figure 3 (particularly the median attainment surfaces for SPEA2 and NSGAII) shows that in some tasks (top row in Figure 3), SPEA2's average front lies along the single-objective GP frontier (using Ave); whereas the average front for NSGAII lies below the Ave frontier in these tasks. The explains why the average hyperarea for SPEA2 (from Table III) is statistically better than NSGAII in these three tasks (Ion, Ped and Yst_2). Similarly, the PO front for SPEA2 also clearly dominates the PO front for NSGAII in two of these tasks (Ped and Yst_2). This suggests that on these tasks, MOGP with SPEA2 can evolve frontier solutions in a single run that perform better than, or at least as well as, multiple runs of canonical GP using Ave. However, MOGP with NSGAII cannot achieve this to a sufficient level of accuracy, as the canonical GP solutions along the Ave frontier clearly dominate the NSGAII average front.

A likely reason for this difference in behaviour is the inherent bias between the two fitness schemes. SPEA2 tends to evolve more solutions in the middle region of the frontier, pushing this front outwards toward the *zenith* point (100% accuracy on both objectives). NSGAII fitness tends to evolve a spread of solutions along the whole of the frontier. For these classification tasks, edge-region solutions are less desirable than middle-region solutions, as these represent biased classifiers. Figure 2 illustrates this difference for the Ped task, and clearly shows that SPEA2 evolves more solutions in the middle region of the frontier compared to NSGAII over all runs.

V. Evolving Ensembles using MOGP

In this section we adapt the MOGP approach to evolve ensembles and discuss two ensemble-diversity measures in the fitness function.

A. Adapting MOGP Fitness

One of the key advantages of this EMO approach is that the evolved Pareto front represents highly accurate classifiers, each with a different performance bias toward either class. Up until now, our goal has been to present these classifiers to the end-user for the final selection. However, as the front of non-dominated solutions has as much information as any single individual, utilising the *combined* classification ability of these solutions in a competitive voting or ensemble-based scenario can be beneficial [8][33][29]. In an ensemble of classifiers, one simple classification strategy uses a *majority vote* approach: each ensemble member votes on what class label to assign to a given data instance, and the class label with the most number of votes determines the class of that particular instance [8]. This strategy has proved successful in previous ensemble learning approaches [8][33][29].

A key condition for an ensemble of classifiers to be more accurate than any of its individual members is that the ensemble members must be accurate and diverse with respect to their outputs [26]. Diverse ensemble members should not make the same errors on the same inputs, otherwise the ensemble will risk misclassifying all the same inputs together. In other words, in a good ensemble, if one individual generates an error for a given input, i.e., votes for the incorrect class label, the other members should not also make the same error.

As previously discussed (Section II), one of the main techniques to construct diverse ensembles involves injecting randomness into the learning algorithm [26]. In this approach, we rely on the stochastic way in which new GP solutions are created (e.g. using the genetic operators) to evolve diverse classifiers, where the non-dominated solutions in the population at each generation constitute the ensemble.

However, without an explicit diversity objective in fitness to encourage the evolved solutions to make different errors on different inputs, the ensemble members are not guaranteed to be diverse with respect to their predictions. For this reason, we adapt the MOGP approach to incorporate a diversity objective into the fitness function, aiming to reward solutions which have better diversity with better fitness values. We investigate two measures to promote the evolution of diverse solutions in the population, negative correlation learning (NCL) and pairwise failure crediting (PFC).

Both measures have proven effective in evolving diverse ensembles of NNs for classification [15][20][29]. In this paper, these measures are adapted to calculate a solution's diversity *separately* for each class to account for the skewed class distributions in these tasks; otherwise, these diversity measures risk being biased toward the majority class. The average across both the minority and majority class is then used as the final diversity value in fitness to encourage the ensemble members to be equally diverse with respect to both classes. This is different to the way some previous approaches (such as [13][15][20][29][35]) have used the diversity measure in fitness. As discussed in Section II-C and Section II-D, only [20] has adapted the NCL measure in a similar manner to account for unbalanced class distributions. However, in [20], the NCL is only applied to minority class instances (diversity on the majority class instances is ignored). The other approaches (mentioned above) apply the diversity measure to all examples irrespective of class (for classification with balanced data sets).

1) Negative Correlation Learning (NCL): The first measure to encourage diversity among the individuals in the population uses NCL as a correlation penalty term in the fitness function [13][15][29]. NCL measures the phenotypic differences between the solutions in the ensemble and the rest of the population. The NCL measure, given by Eq. (7) below, calculates the *average* correlation penalty for each class, for a given solution p in the population.

$$NCL_p = \frac{1}{2}\sum_{c=1}^{K}\left(\frac{1}{MN_c}\sum_{i=1}^{N_c}(G_i^p - E_i)\left[\sum_{j=1,j\neq p}^{M}(G_i^j - E_i)\right]\right) \tag{7}$$

where

$$G_i^p = \frac{1}{1+e^{gp_i^p}}$$

In Eq. (7), K is the number of classes, and N_c is the number of training examples in class c. G_i^p is the processed output and gp_i^p is raw output of genetic program p when evaluated on the i^{th} example in class c. E_i is the output of the ensemble on the i^{th} example in class c, i.e., 1 or 0 to denote a minority or majority class label, respectively. The ensemble output (E_i) is a majority vote of the predicted class labels of each ensemble member. The ensemble size, i.e., the number of non-dominated solutions in the current generation, is given by M. The lower the NCL values, the better the diversity of the solutions.

The NCL penalty is incorporated into MOGP by using Eq. (7) as the *secondary* fitness measure instead of the "crowding" distance. This means that the NCL term is used to resolve selection (e.g. for crossover/mutation and archive selection) when the primary fitness measure (Pareto ranking using SPEA2) is equal between two or more individuals. NCL is used as the secondary fitness measure because Eq. (7) utilises the ensemble output (E) in its calculation. This means that the primary fitness measure must be applied to the population first to determine which solutions are non-dominated in the population, i.e., the current Pareto front, as these solutions then determine the ensemble output.

2) Pairwise Failure Crediting (PFC): The second diversity measure is also used as a penalty function but unlike the NCL, PFC is a population-level diversity measure [15]. This means that PFC measures the errors (on the training set) of each solution with respect to *all* other solutions in the population; whereas NCL compares the outputs of a solution to the ensemble only. Eq. (8) below calculates the PFC penalty for solution p with respect to class c.

$$PFC_{c,p} = \frac{1}{T-1}\sum_{j=1,j\neq p}^{T}\frac{\sum_{i=1}^{N_c} I(gp_i^p, gp_i^j)}{Err_c^p + Err_c^j} \tag{8}$$

where

$$I(gp_i^p, gp_i^j) = \begin{cases} 1 & \text{if pred}(gp_i^p) \neq \text{pred}(gp_i^j) \\ 0 & \text{otherwise} \end{cases}$$

TABLE IV
AVERAGE (± STANDARD DEVIATION) HYPERAREA OF EVOLVED PARETO-APPROXIMATED (PA) FRONTS, PARETO-OPTIMAL (PO) FRONT, AND TRAINING TIMES FOR THE MOGP APPROACHES. STATISTICALLY BETTER HYPERAREA VALUES (5% LEVEL OF SIGNIFICANCE) ARE HIGHLIGHTED IN BOLD.

Task	MOGP Baseline			MOGP NCL			MOGP PFC		
	Hyperarea		Train	Hyperarea		Train	Hyperarea		Train
	PA front	PO front	Time	PA front	PO front	Time	PA front	PO front	Time
Ion	**0.848 ± 0.041**	0.992	9.3s ± 2.4	**0.849 ± 0.039**	0.981	1.4m ± 6.8	0.828 ± 0.032	0.982	30.4s ± 2.6
Spt	0.732 ± 0.032	0.971	9.7s ± 2.5	0.733 ± 0.031	0.964	1.2m ± 5.3	0.719 ± 0.025	0.964	29.5s ± 1.6
Ped	**0.902 ± 0.019**	0.922	3.9m ± 1.1	**0.905 ± 0.011**	0.926	90.2m ± 1.3	0.883 ± 0.010	0.921	17.7m ± 24.8
Yst_1	**0.793 ± 0.009**	0.931	20.8s ± 7.1	**0.795 ± 0.010**	0.922	5.6m ± 14.4	0.774 ± 0.009	0.923	1.2m ± 4.5
Yst_2	**0.949 ± 0.011**	0.991	20.1s ± 8.1	**0.949 ± 0.007**	0.989	5.3m ± 18.9	0.928 ± 0.011	0.989	1.1m ± 5.9
Bal	0.757 ± 0.063	0.985	15.2s ± 3.9	**0.810 ± 0.078**	1.0	2.4m ± 8.5	**0.800 ± 0.065**	1.0	45.1s ± 3.2

and

$$\text{pred}(gp_i^p) = \begin{cases} 1 & \text{if } gp_i^p \geq 0 \text{ (i.e. minority class)} \\ 0 & \text{otherwise (i.e. majority class)} \end{cases}$$

In Eq. (8), N_c is the number of training examples in class c, and gp_i^p is the raw output of genetic program p when evaluated on the i^{th} example in class c (as used in Eq. (7) for NCL); and T is population size. Indicator function $I(\cdot)$ returns 1 if the predicted class label between two solutions is different for a given input, or 0 otherwise; this is used to compute the *Hamming distance* between the predictions of two genetic programs on all inputs in class c. The errors, Err_c^p and Err_c^j, are the number of incorrect predictions in class c for two solutions p and j in the population. An incorrect prediction occurs when the predicted and actual class labels differ for a given input. Eq. (8) will return values between 0 and 1 where the higher the PFC, the better the diversity.

As the outputs of each solution in the population are compared to all others, Eq. (8) aims to make solutions in the population uncorrelated to all others. This is different to NCL which aims to minimise the correlation between solutions and the ensemble. As a result, Eq. (8) does not require the ensemble output in the PFC calculation, allowing the PFC measure to be used at *any* stage in the fitness evaluation. To take advantage of this flexibility, Eq. (8) is incorporated into the objective performance of the evolved solutions (alongside the classification accuracy) *before* the primary fitness measure (Pareto ranking using SPEA2) is applied to the population. This gives equal selection preference to accurate and diverse solutions. This is represented by Eq. (9), where $(S_p)_c$ is the objective performance of solution p on objective c. We use weight factor W where $0 < W < 1$ to specify the trade-off between accuracy and diversity, and set W to 0.5 to treat these two measures as equally important for ensemble membership.

$$(S_p)_c = W\left(\frac{1 - Err_c^p}{N_c}\right) + (1 - W)PFC_{c,p} \qquad (9)$$

The advantage of incorporating the accuracy and diversity of evolved solutions into the objective performance is that the Pareto rankings (according to SPEA2) are not solely based on the accuracy of the solutions on the two classes (as is the case for MOGP using NCL). This means that these ensembles can contain more diverse but potentially less accurate solutions compared to the NCL-based ensembles.

Note that the computational overhead required to compute Eq. (8), where each solution is compared to all others in the population ($T(T-1)$ comparisons), can be minimised by simultaneously accumulating PFC values between any two solutions. For example, if the PFC for solution p is being computed with respect to solution q, then both the Hamming distances and errors will be the same for both solutions. In this case, only $\frac{1}{2}T(T-1)$ comparisons are required to compute the PFC for the entire population.

3) Baseline MOGP Ensemble: To compare the relative effectiveness of the two ensemble-diversity measures in fitness, we use MOGP with SPEA2 fitness (Eq. 6) to represent a *baseline* approach where no explicit ensemble-diversity objective is used in fitness. This baseline approach investigates whether the stochastic way in which new classifiers are created within the GP process is sufficient to evolve diverse ensembles (compared to two ensemble-diversity measures in fitness). A similar approach is used in [27] where only the randomness of the initial weights of a neural network-based ensemble (using bagging) is shown to be sufficiently effective for diversity. This paper extends this idea to genetic program-based ensemble classifiers. Note that MOGP with SPEA2 is chosen as the baseline approach as this Pareto-based Dominance measure in fitness has been shown (in the previous section) to evolve better-performing solutions than NSGAII on these tasks. However, as both SPEA2 and NSGAII have a different bias towards certain regions of the Pareto frontier, the diversity of the evolved Pareto-approximated fronts using both methods can also be different; we leave exploring the difference in diversity between these two methods for future work.

B. MOGP Evaluation using Ensemble-Diversity Measures

Before we compare the different MOGP ensemble performances, we first investigate what effect the diversity objectives have on the hyperarea of the evolved Pareto fronts (on the *test* sets). Table IV reports the average (and standard deviation) hyperarea of the evolved Pareto-approximated fronts, the hyperarea of the Pareto-optimal front, and the average training times for the three MOGP approaches, over 50 independent runs on the tasks. An analysis of variance (ANOVA) *F-test*[1] [50] of the average hyperarea of the Pareto-approximated fronts from the three approaches is used to statistically test the null hypothesis, i.e., no statistical difference in hyperarea values over 50 runs at a 5% level of significance. This test indicates that for all tasks except Spt, there is a statistically significant difference in the hyperarea values for the three approaches over 50 runs, i.e., null hypothesis rejected. As a

[1]We use the F-test in Table IV (and not the common random numbers technique used in Table III) as more than two systems are being compared.

TABLE V
ENSEMBLE ACCURACY ($\pm$ STANDARD DEVIATION) ON THE TEST SET AND AVERAGE ENSEMBLE SIZE USING THE FULL PARETO FRONT AS THE ENSEMBLE (MAJORITY VOTE AND WEIGHTED VOTE) AND WITH ACCURACY-BASED ENSEMBLE SELECTION.

Task	MOGP	Pareto Front Ensemble					Reduced Ensemble		
		Majority Vote (PF-vote)			Weighted Vote (PF-Wvote)		Majority Vote (RPF-vote)		
		Size	Minority Acc	Majority Acc	Minority Acc	Majority Acc	Size	Minority Acc	Majority Acc
Ion	Baseline	8.8	84.5 $\pm$ 6.2	83.5 $\pm$ 9.9	82.5 $\pm$ 5.8	89.1 $\pm$ 8.3	7.8	79.9 $\pm$ 7.2	87.2 $\pm$ 9.1
	NCL	12.7	85.5 $\pm$ 5.2	86.8 $\pm$ 7.3	80.9 $\pm$ 5.9	91.4 $\pm$ 5.9	10.8	82.6 $\pm$ 6.9	91.0 $\pm$ 3.2
	PFC	28.1	84.9 $\pm$ 5.1	92.4 $\pm$ 6.4	79.6 $\pm$ 6.2	96.3 $\pm$ 3.7	22.3	81.7 $\pm$ 5.8	95.8 $\pm$ 3.8
Spt	Baseline	7.8	44.5 $\pm$ 5.5	88.8 $\pm$ 2.7	86.4 $\pm$ 13.7	59.9 $\pm$ 36.4	2.8	69.9 $\pm$ 11.7	70.1 $\pm$ 17.4
	NCL	9.5	48.6 $\pm$ 5.6	86.5 $\pm$ 2.9	84.1 $\pm$ 12.0	63.3 $\pm$ 32.7	4.1	71.1 $\pm$ 9.0	78.4 $\pm$ 8.7
	PFC	27.3	44.6 $\pm$ 5.4	90.8 $\pm$ 2.3	75.9 $\pm$ 10.4	72.6 $\pm$ 22.2	12.1	62.1 $\pm$ 8.0	80.5 $\pm$ 4.8
Ped	Baseline	154.9	12.5 $\pm$ 27.2	87.1 $\pm$ 28.2	89.1 $\pm$ 3.6	83.3 $\pm$ 3.1	87.8	81.4 $\pm$ 14.1	79.3 $\pm$ 28.9
	NCL	52.6	71.8 $\pm$ 8.9	91.7 $\pm$ 2.7	88.0 $\pm$ 3.2	83.5 $\pm$ 3.7	43.2	87.4 $\pm$ 4.5	84.2 $\pm$ 6.1
	PFC	71.6	82.4 $\pm$ 5.6	92.1 $\pm$ 2.4	92.5 $\pm$ 1.6	84.1 $\pm$ 3.1	40.1	91.6 $\pm$ 1.9	85.2 $\pm$ 3.0
Yst1	Baseline	46.7	58.0 $\pm$ 4.0	87.1 $\pm$ 2.4	70.0 $\pm$ 4.2	77.3 $\pm$ 4.3	24.1	68.7 $\pm$ 3.2	77.5 $\pm$ 3.8
	NCL	25.8	63.6 $\pm$ 3.8	83.0 $\pm$ 3.3	70.6 $\pm$ 3.7	76.8 $\pm$ 4.4	17.0	69.0 $\pm$ 2.6	80.7 $\pm$ 1.8
	PFC	39.7	64.6 $\pm$ 4.8	82.5 $\pm$ 4.3	71.8 $\pm$ 5.3	75.4 $\pm$ 6.5	16.5	71.0 $\pm$ 4.4	75.5 $\pm$ 5.4
Yst2	Baseline	18.5	77.1 $\pm$ 4.6	96.2 $\pm$ 1.1	82.8 $\pm$ 3.6	95.1 $\pm$ 1.3	15.2	80.6 $\pm$ 7.8	94.8 $\pm$ 2.2
	NCL	16.1	77.6 $\pm$ 6.0	95.3 $\pm$ 1.7	83.6 $\pm$ 4.7	93.7 $\pm$ 1.7	13.4	89.6 $\pm$ 5.2	91.9 $\pm$ 2.9
	PFC	27.9	81.2 $\pm$ 4.9	95.5 $\pm$ 1.5	89.6 $\pm$ 3.2	92.1 $\pm$ 1.9	20.6	89.2 $\pm$ 3.2	92.3 $\pm$ 1.8
Bal	Baseline	9.8	53.3 $\pm$ 21.4	94.1 $\pm$ 4.4	84.2 $\pm$ 12.5	71.4 $\pm$ 23.6	4.7	82.6 $\pm$ 13.7	59.0 $\pm$ 21.4
	NCL	8.4	59.2 $\pm$ 16.1	87.8 $\pm$ 6.6	86.9 $\pm$ 11.8	66.0 $\pm$ 29.5	4.9	61.8 $\pm$ 4.2	94.1 $\pm$ 5.9
	PFC	20.8	51.7 $\pm$ 18.2	95.4 $\pm$ 3.5	87.3 $\pm$ 9.3	74.1 $\pm$ 17.1	10.1	83.6 $\pm$ 9.4	79.5 $\pm$ 10.3

result, a *post-hoc* multiple comparisons test using the Kruskal-Wallis (KW) method [50] is used to determine the statistically significant differences between group means. The KW test conducts a series of pairwise comparisons[2] using the hyperarea values from the MOGP approaches, and outputs a set of 95% confidence intervals for each comparison based on the *studentized range* distribution q (similar to a Students *t*-test). The KW method is a non-parametric test for when the experimental data is not assumed to be normally distributed.

In all tasks except Bal and Spt, the average hyperarea for both the baseline and NCL approaches are *statistically* better than PFC, but are not statistically different from one another. These two hyperarea values (baseline and NCL) are highlighted in bold in Table IV for these tasks. In one task (Bal), the average hyperarea for both NCL and PFC are statistically better than the baseline approach, but are not statistically different from one another. These two hyperarea values (NCL and PFC) are also highlighted in bold for Bal. In the Spt task, the average hyperarea values for all three approaches are not statistically different from one another.

These results suggest that in most tasks, the PFC approach evolves non-dominated solutions with lower classification accuracy on the two classes, compared to both the baseline and NCL. However, these solutions may be non-dominated because they are highly diverse with respect to their errors (this is explored further in the next section). It is interesting that for the Bal task, both NCL and PFC approaches evolve at least one non-dominated solution with 100% accuracy on both the minority and majority class. This solution represents the Pareto-optimal hyperarea of 1 in Table IV for Bal, as the Pareto-optimal frontier consists of only this one particular point. The baseline MOGP approach is not able to accomplish this in any task.

Table IV also shows that as expected, both NCL and PFC incur longer average training times than the baseline approach. This is due to the additional computational effort required to calculate the corresponding diversity measure in fitness evaluation. However, this increase is not a serious concern in most tasks. PFC also shows faster average training times than NCL in the tasks. For the largest training set, Ped (more than 24000 examples), NCL incurs substantially longer training times than the two other approaches.

[2] $k(k-1)\frac{1}{2}$ total comparisons where k is the number of MOGP systems.

VI. MOGP ENSEMBLE CLASSIFICATION RESULTS

In this section we discuss the MOGP ensemble classification results and compare these approaches to canonical GP and other machine learning approaches on the tasks.

A. Voting Accuracy using Full Pareto Front

Table V reports the average minority and majority class accuracy (with standard deviations) of the evolved ensembles using the three MOGP approaches (Baseline, NCL and PFC), on the *test* sets over 50 runs. In the left-most column of Table V (called PF-vote), the ensembles use a *majority-vote* of the Pareto-approximated front, i.e., the set of non-dominated solutions in an evolved population. The majority vote specifies that the class label with the most votes from the ensemble members is taken as the ensemble output. The average number of Pareto front solutions (ensemble sizes) are also included in Table V alongside the classification accuracy.

Immediately noticeable in Table V are the strong majority class performances for the three MOGP approaches using this voting strategy. In some tasks such as Ion and Yst_2, the corresponding minority class accuracies are still reasonably good, while in the others, particularly Ped and Spt, this is very poor. This shows that in most tasks, the evolved Pareto fronts can contain more solutions biased toward the majority class than the opposite case, i.e., solutions with good minority accuracy or middle-region solutions, as these biased solutions can influence the final ensemble vote, thus biasing the final ensemble prediction.

B. Limiting the Influence of Biased Ensemble Members

Three strategies are evaluated to limit the influence of biased ensemble members on the final ensemble vote. The first strategy uses a weighted majority vote of the Pareto front solutions where the weight or *contribution* of a given individual in the voting process is based on the fitness of that individual on the training set (used in [39]). For NCL and PFC, an individual's fitness is the average accuracy and diversity (Eq. (7) and Eq. (9), respectively) on both classes; while for the baseline MOGP, this is the average accuracy on both classes. This means that biased Pareto front solutions with poor fitness on one class will contribute less in the voting process, while Pareto front solutions with good fitness on both classes will contribute more. The ensemble results using the weighted majority vote are shown in the middle column in Table V. For presentation convenience, the fitness-weighted majority vote of the full Pareto front in the ensemble is called the PF-Wvote strategy in Table V.

The second strategy uses a naive but effective ensemble selection strategy (used in [41][27]) to simply *remove* biased Pareto front solutions from the final ensemble. Pareto front solutions with less than 50% accuracy on either the minority or the majority class (on the training set) are removed from the ensemble, allowing the reduced ensembles to contain only relatively accurate members (with at least 50% accuracy on both classes). In this ensemble selection strategy, shown in the right-most column in Table V (called RPF-vote), each member's vote contributes equally during voting. Ensemble members with at least 50% accuracy on both classes implies that a solution is better than random guessing on the tasks [41][27].

The third strategy uses off-EEL [37] for ensemble selection (these results are discussed in the next section).

Table V shows that all three MOGP approaches with the PF-Wvote and RPF-vote strategies have more balanced class performances with better minority class accuracies in all tasks (except Ion), particularly Spt, Ped (for the baseline MOGP) and Yst_2. This suggests that the PF-Wvote and RPF-vote strategies succeeded in reducing the influence of biased Pareto front solutions in the ensembles in these tasks. These two strategies typically produce similar (or non-dominating) ensemble results for the NCL and PFC approaches in nearly all tasks (except Bal), suggesting that both are similarly effective in keeping the ensemble performances well-balanced on both classes. In Bal, the NCL results for PF-Wvote and RPF-vote vary in their minority and majority class bias, e.g., minority class accuracy is higher using PF-Wvote, while majority class accuracy is higher using RPF-vote. This may be due to noise or the comparatively high level of class imbalance in Bal.

Interestingly, the PF-Wvote results dominate the RPF-vote results for the baseline MOGP approach in all tasks (except Spt where both strategies show non-dominated results). This suggests for the baseline MOGP, the RPF-vote ensembles still contain some individuals that do not positively contribute to the ensemble, as performances improve when the influence of these members on the ensemble are reduced using PF-Wvote. This may be because the baseline MOGP uses no additional ensemble-diversity objective in fitness (unlike NCL and PFC), and its Pareto front solutions are more accurate on the two objectives but less diverse in their outputs (as discussed in Section V-B).

Table V also shows that ensemble sizes for RPF-vote are smaller than PF-vote in all tasks. In exactly 4 tasks (Ion and Yst_2 are the two exceptions), roughly half of the number of Pareto front solutions are excluded from the RPF-vote ensembles (compared to PF-vote), as these solutions have less than 50% accuracy on both classes. This reaffirms the notion that not all Pareto front solutions make useful contributors in the ensemble. For Ion, the ensemble results for the PF-Wvote and RPF-vote strategies have higher majority class accuracies (than minority class accuracies) compared to PF-vote, and the RPF-vote ensemble sizes are not much smaller than the PF-vote ensembles. This suggests that the evolved Pareto fronts for Ion did not contain many biased solutions, as the differences in ensemble sizes between the PF-vote and RPF-vote is larger in other tasks. This may be due to the comparatively low level of class imbalance in Ion.

Interestingly, the baseline MOGP ensembles perform as well as NCL and PFC in some tasks, particularly for the PF-Wvote strategy. For example, in Ion, Ped and Yst_2, the baseline and NCL results are similarly good; while all three MOGPs (baseline, NCL and PFC) show similar results in Yst_1. This is surprising as the baseline MOGP uses no ensemble-diversity objective in fitness. This suggests that in these tasks, the stochastic processes within GP alone are sufficient to evolve diverse solutions when the PF-Wvote strategy is used for ensemble voting (compared to NCL and PFC which use explicit ensemble-diversity measures in fitness). However, further investigation of the differences between the three MOGP approaches is needed and is explored in Section VI-D.

C. Off-EEL Algorithm for Ensemble Section

The RPF-vote strategy for ensemble selection offers a naive yet effective approach to choosing which individuals to use in the final ensemble (from the set of evolved Pareto front classifiers). A more exhaustive and arguably better ensemble selection approach is the off-EEL (offline evolutionary ensemble learning) algorithm [37] (discussed in Section II-B). In this section we evaluate the MOGP ensembles using off-EEL to investigate if this algorithm can improve ensemble performances compared to the RPF-vote strategy.

The off-EEL algorithm uses a greedy search to construct the ensembles from a pool of base classifiers (in this case, an evolved Pareto front). This algorithm sorts the input set of base classifiers according to their fitness values on the training set (similar to the PF-Wvote), from the fittest to the least fit classifiers. Then, each classifier is removed from the (sorted) input set and inserted into the ensemble where, at each step, the ensemble is evaluated using a majority vote of the base classifiers in the current ensemble. Once all the base classifiers from the input set are processed, the ensemble with the best performance is taken as the final ensemble. In this paper, only odd numbered ensembles sizes are considered as these constitute ensembles where *no draws* can occur in the voting process.

TABLE VI
ENSEMBLE ACCURACY (± STANDARD DEVIATION) ON THE TEST SET AND AVERAGE ENSEMBLE SIZE USING OFF-EEL ENSEMBLE SELECTION [37].

Task	MOGP	Size	Minority Acc	Majority Acc
Ion	Baseline	5.6	83.7 ± 5.8	89.2 ± 8.8
	NCL	9.9	82.2 ± 5.5	89.5 ± 8.1
	PFC	21.2	83.6 ± 5.2	96.6 ± 2.8
Spt	Baseline	3.9	56.0 ± 10.1	83.6 ± 4.8
	NCL	5.2	53.9 ± 9.7	83.8 ± 4.8
	PFC	10.7	66.3 ± 8.5	79.9 ± 6.6
Ped	Baseline	65.3	89.5 ± 1.5	84.6 ± 2.3
	NCL	40.4	88.8 ± 3.0	83.4 ± 2.8
	PFC	55.2	90.6 ± 1.5	87.9 ± 1.5
Yst1	Baseline	33.6	68.5 ± 5.5	80.4 ± 5.2
	NCL	13.5	64.1 ± 5.0	83.4 ± 4.0
	PFC	29.2	70.6 ± 5.4	78.8 ± 5.5
Yst2	Baseline	6.5	92.3 ± 2.9	90.7 ± 2.9
	NCL	8.6	80.4 ± 7.3	94.5 ± 2.1
	PFC	17.2	93.1 ± 2.6	90.8 ± 2.4
Bal	Baseline	4.7	71.4 ± 15.9	85.6 ± 9.0
	NCL	5.5	62.1 ± 17.5	85.6 ± 7.3
	PFC	10.9	81.4 ± 12.1	86.2 ± 9.1

Naturally, when the pool of base classifiers is large, this algorithm incurs an additional computation cost as the ensemble must be evaluated on all training examples at each iteration. For the MOGP approaches, the pool of base classifiers (evolved Pareto front) is not sufficiently large to incur a substantial increase in training time. The Pareto fronts contain fewer than 100 classifiers in most tasks except Ped for the baseline MOGP where the average Pareto front size is 154 (as shown in Table V).

Table VI shows the ensemble results for the off-EEL ensemble selection algorithm over 50 MOGP runs. As expected, the off-EEL ensemble performances either dominate, or are non-dominated with respect to, the RPF-vote results (in Table V) in these tasks. The off-EEL algorithm also shows a better balance in minority and majority accuracies (i.e. both are similarly high) than RPF-vote in the two tasks with the highest levels of class imbalance (Yst_2 and Bal) for the baseline and PFC approaches. These results are not unexpected as the off-EEL algorithm uses a greedy search to find good ensembles with high accuracy on both classes compared to the naive RPF-vote strategy.

D. Counting Ensemble "Wins"

For further analysis of the differences between the MOGP approaches and the ensemble voting/selection strategies, we also compare the outcomes of each MOGP approach on a *run-by-run* basis over the 50 independent runs. This enables us to investigate which MOGP fitness approach (Baseline, NCL or PFC) and ensemble selection strategy (PF-Wvote, RPF-vote or off-EEL) produces better overall results across all MOGP experiments and tasks. These questions are difficult to answer using only the average performances reported in Table V and Table VI. For clarity, in our experimental setup, a single run of the three MOGP approaches (baseline, NCL and PFC) all use the same random starting seed and initial population.

A run-by-run analysis of the MOGP approaches has a two-dimensional aspect, as both the majority and minority class accuracies of the ensembles must be taken into account when determining if one approach is better than another (compared to a single-figure measure such as the overall accuracy). We use the Pareto dominance relation between the outcomes of any two MOGP approaches to determine if one approach is better than the other, in terms of a "win", "lose" or "draw" result. For two MOGP approaches, gp_1 and gp_2, these three outcomes for a particular run can be defined as follows.

- Win for gp_1 if gp_1 dominates gp_2 (loss for gp_2).
- Win for gp_2 if gp_1 is dominated by gp_2 (loss for gp_1).
- Draw otherwise.

Table VII shows the pairs of ensemble "wins" between two MOGP approaches, when each approach is compared with every other on a run-by-run basis for 50 independent runs on the tasks. Each win-pair in Table VII corresponds to the three pairwise comparisons between the baseline, NCL and PFC approaches for a particular ensemble combination strategy. The ensemble combination strategies corresponds to PF-Wvote, RPF-vote and off-EEL.

For example, the first win-pair entry in Table VII for Ion with PF-Wvote ("8 / 8") shows the number of wins when the baseline MOGP is compared to the NCL approach. In this case, both the baseline and NCL score 8 wins each (each approach dominates the other exactly 8 times); these approaches are non-dominated with respect to each other in the remaining 34 experiments. Similarly, when the baseline and PFC approaches are compared (also with PF-Wvote) for Ion, "7 / 14" means that PFC wins against (dominates) the baseline in 14 experiments, and the baseline wins against PFC in 7 experiments. These approaches are non-dominated in the remaining 29 experiments.

The last two rows in Table VII reports the total number of "wins" for each pair, and the total number of "draws" (non-dominated performance), over *all* runs and tasks. The total number of wins and draws (each column) in Table VII sum to 300 (50 runs $\times$ 6 tasks).

These three outcomes, i.e., win/lose/draw, represent a *multinomial* distribution over N independent runs. This means that the proportion of wins for one approach (call this p_1), the proportion of wins for the other approach (call this p_2), and the proportion of draws between them (call this p_3), always sums to 1 over N runs. In a multinomial distribution, we can calculate a 95% confidence interval of the *difference* in the proportion of wins between two approaches ($p_1 - p_2$) for a particular task, to determine if one MOGP ensemble *significantly dominates* another over all runs. The 95% confidence interval of this difference between any two MOGP approaches can be calculated using Eq. (10), where $var(p_i)$ is the variance of p_i for the ith approach in N (50) runs.

$$(p_1 - p_2) \pm 1.96\sqrt{var(p_1 - p_2)} \tag{10}$$

where

$$\begin{aligned} var(p_1 - p_2) &= var(p_1) + var(p_2) - \\ &\quad -(var(p_1 + p_2) - var(p_1) - var(p_2)) \\ &= 2var(p_1) + 2var(p_2) - var(p_1 + p_2) \end{aligned}$$

TABLE VII

"WIN" PAIRS BETWEEN ANY TWO MOGP APPROACHES ON A RUN-BY-RUN BASIS FOR THE ENSEMBLE VOTING STRATEGIES. A "WIN" IS WHEN ONE APPROACH DOMINATES THE OTHER ON A GIVEN RUN. THE "WINS" ARE SUMMED OVER 50 RUNS FOR EACH TASK, AND THEN SUMMED OVER ALL TASKS (50 RUNS × 6 TASKS). BOLD RESULTS INDICATE A STATISTICALLY SIGNIFICANTLY BETTER ENSEMBLE PERFORMANCE (95% SIGNIFICANCE LEVEL).

Task	Pareto Front Weighted Vote (PF-Wvote)			Reduced Pareto Front Ensemble (RPF-vote)			off-EEL Ensemble Selection		
	Baseline *vs* NCL	Baseline *vs* PFC	NCL *vs* PFC	Baseline *vs* NCL	Baseline *vs* PFC	NCL *vs* PFC	Baseline *vs* NCL	Baseline *vs* PFC	NCL *vs* PFC
Ion	8 / 8	7 / 14	8 / **19**	8 / 14	5 / **22**	4 / 14	11 / 6	6 / **22**	3 / **20**
Spt	3 / 5	1 / 5	1 / 3	7 / 10	7 / 10	7 / 8	10 / 7	3 / 6	1 / **9**
Ped	11 / 3	1 / **25**	0 / **19**	0 / **20**	1 / **24**	0 / **42**	15 / 7	0 / **20**	0 / **26**
Yst1	7 / 8	4 / 2	5 / 7	8 / 9	8 / 5	6 / 5	5 / 3	6 / 3	1 / 3
Yst2	7 / 5	5 / 2	3 / **12**	4 / **16**	5 / 3	3 / **14**	5 / 1	8 / 7	0 / **4**
Bal	10 / 7	11 / 14	10 / 9	13 / 13	11 / 15	9 / 15	14 / 8	12 / 16	4 / **21**
Total Wins	46 / 36	29 / 62	27 / 69	40 / 82	37 / 79	29 / 98	60 / 32	35 / 74	9 / 83
Total draws	218	209	204	178	184	173	208	191	208

and

$$var(p_i) = \frac{p_i(1-p_i)}{N}$$
$$var(p_1+p_2) = \frac{(p_1+p_2)(1-p_1-p_2)}{N}$$

The results of the 95% confidence intervals are shown in Table VII, where the statistically signficantly better ensemble performance is highlighted in bold for a particular win-pair. As we construct three separate confidence intervals for each pairwise comparison, the statistical relationship only applies to a specific pair.

1) PFC better than NCL over all tasks: Table VII highlights the overall differences in ensemble performances between the three MOGP approaches. The total number of wins (over all tasks) when NCL is compared to PFC is higher in PFC for all three ensemble combination strategies. This means that in all three ensemble combination strategies, the PFC ensembles dominate the NCL ensembles *more often* than the opposite case over all runs and tasks. For the off-EEL strategy in particular, this difference in total wins between PFC and NCL is very large (PFC has 83 total wins but NCL only has 9). This is due to the PFC ensembles achieving statistically significantly better performances than NCL in nearly all tasks. This suggests that the PFC approach, particular with off-EEL, produces better ensemble results than NCL in these tasks.

A similar conclusion can be drawn when the PFC ensembles are compared to the baseline MOGP for the three ensemble combination strategies. PFC always scores more total wins (over all tasks) than the baseline when these two approaches are compared against each other for all three strategies. The better PFC performances may be due to better cooperation between the ensemble members than the baseline MOGP on these tasks, due to better diversity from the PFC measure in the fitness function. In contrast, NCL only scores more total wins (over all tasks) than the baseline MOGP using one strategy (RPF-vote); whereas for the other two strategies, PF-Wvote and off-EEL, the baseline scores *more* total wins (over all tasks) than NCL. Indeed, Table VII shows that for both PF-Wvote and off-EEL strategies, there is no statistically significant difference in the wins between the baseline and NCL in any tasks. This suggest that the NCL ensembles only show an improvement over the baseline MOGP for the RPF-vote ensemble selection strategy; while for the PF-Wvote and off-EEL strategies, both MOGPs perform similarly.

2) Further Discussions: The above results show that the PFC ensembles performed better than NCL on these tasks, particularly for the off-EEL selection algorithm. This may be due to two reasons. The first is the different ways NCL and PFC create "spread" (or diversity) in the population (see [36] for theoretical insights into how NCL creates spread in a population as discussed in Section II-B). The second is the different ways NCL and PFC are used in MOGP: NCL is calculated after the population is ranked (using SPEA2) on the objectives, while PFC is calculated before Pareto ranking is done (see Section V-A for details).

Developing an approach which incorporates NCL into the objective performance before Pareto-ranking is done (similar to PFC) may improve ensemble performances for NCL. Likewise, new diversity measures (such as the root quartic NCL proposed in [35][36] discussed in Section II-B) may also improve ensemble performances for NCL, due to different ways in which these measures create "spread" in a population. However, more investigation is required to confirm these hypothesis. As this is outside the scope of this work, we will investigate them in future work.

E. Comparison with Canonical GP, NB and SVM

We also compare the ensemble performances with canonical (single-predictor) GP using three fitness functions, and two other popular machine learning techniques, namely, Naive Bayes (NB) and Support Vector Machines (SVM), on the tasks. Table VIII shows the average class accuracies of the fittest evolved solutions using canonical GP (on the test sets) over 50 independent runs. The three GP fitness functions correspond to Acc (Eq. 1), Ave (Eq. 2), and the area under the ROC curve (AUC) [51] (denoted as GP Auc in Table VIII). For Ave, only $W = 0.5$ is shown as this configuration treats the accuracy of the two classes as equally important in fitness. To calculate the AUC for each solution (for GP Auc), seven TP/FP rates[3] are used to build an ROC curve, and the trapezoidal technique to estimate the area under this curve (for more details see [51]). Note that the same complexity constraints are placed on both the canonical GP classifiers and the MOGP Pareto front classifiers (base classifiers in the

[3]Each TP/FP rate is generated by evaluating the classifier at seven distinct class thresholds spread uniformly over the range of genetic program outputs; seven is recommended in [51] for a fast and accurate approximation.

TABLE VIII
CLASSIFICATION ACCURACY OF THE MINORITY AND MAJORITY CLASS FOR THE DIFFERENT APPROACHES ON THE TASKS.

Task	GP Acc		GP Ave ($W = 0.5$)		GP Auc		NB		SVM	
	Minority	Majority	Minority	Majority	Minority	Majority	Minority	Majority	Minority	Majority
Ion	73.8 ± 7.7	95.3 ± 3.9	76.6 ± 6.3	91.3 ± 6.1	81.1 ± 5.2	81.3 ± 6.5	63.4	88.9	87.5	99.1
Spt	47.4 ± 4.6	88.6 ± 2.5	56.7 ± 8.3	82.7 ± 3.6	70.2 ± 6.7	70.0 ± 5.8	66.7	83.0	37.0	94.3
Ped	43.3 ± 14.5	96.6 ± 1.6	87.7 ± 2.3	85.6 ± 2.8	86.2 ± 1.5	86.1 ± 1.6	83.7	81.4	53.8	92.4
Yst_1	40.8 ± 4.2	94.6 ± 1.4	60.2 ± 4.6	83.1 ± 3.8	73.0 ± 1.4	72.8 ± 1.5	43.4	96.4	32.8	97.4
Yst_2	64.0 ± 8.1	97.4 ± 0.6	85.9 ± 4.0	93.0 ± 2.1	86.8 ± 2.7	88.2 ± 4.1	66.7	98.0	58.0	97.9
Bal	9.0 ± 17.5	98.9 ± 1.1	85.6 ± 11.4	84.6 ± 11.7	82.8 ± 8.3	87.1 ± 11.0	0.0	100.0	0.0	100.0

ensembles), e.g., a maximum depth limit of 8 (as outlined in Section IV-A).

A single run for NB and SVM is generated using the WEKA package [52]. The SVM uses a sequential minimal optimisation algorithm with an RBF kernel and Gamma value of 10 (this Gamma value generally gave the best performance from experiments using 0.1, 1, 10, and 100).

Tables V and VI show that all three MOGP ensembles using the PF-Wvote and off-EEL strategies achieve much more balanced (and better) results than canonical GP using Acc, NB and SVM in all tasks (except Ion) in Table VIII. In those tasks with high levels of class imbalance (such as Spt, Ped, Yst_1 and Bal), these single-predictor methods show biased results. In Bal in particular, none of these methods achieve more than 10% accuracy on the minority class (Bal has highest level of class imbalance). In Ion, SVM achieves the best results (87% and 99% on the minority and majority class, respectively). The MOGP ensembles cannot, on average, match the SVM results. However, closer examination of the PFC results with off-EEL on a run-by-run basis finds that the three *best* PFC runs score a better accuracy on both classes than SVM. These three runs achieve 88/99%, 88/100%, and 92/100% on the minority/majority class, respectively.

On average, the PFC ensembles with off-EEL dominate canonical GP using Auc in three tasks (Ion, Ped and Yst_2). In Bal, canonical GP (with Auc) and PFC (with off-EEL) achieve very similar results (within 1% accuracy for each class). As the model complexity of the evolved genetic program classifiers are the same in both canonical GP and MOGP, the PFC ensembles are better than canonical GP on some of these tasks for two main reasons. Firstly, this is due to more support for two learning objectives (minority and majority accuracy) in MOGP. In other words, in canonical GP with Auc, each classifier tries to achieve the best trade-off between the two objectives *individually* (by maximising their AUC); whereas in MOGP, each classifier is one point (of many) along the Pareto front. Secondly, combining these Pareto front classifiers into an ensemble where individuals work together (by voting) further improves performances, as the ensemble performs *at least as well* as its individual members.

When a diversity objective such as PFC is introduced in the fitness function during evolution, the ensemble performs better than most of its individual members, as this performance *dominates* the performance of the individual members. This can be seen by comparing the MOGP ensemble results (in Tables V or VI) to Figure 3, particularly the median attainment surface by MOGP with SPEA2. This shows that the ensemble performance (using PF-Wvote, RPF-vote or off-EEL) for the two diversity-based ensembles, in particular PFC, typically lies *above* the average front; whereas the performance of the baseline ensemble usually lies *on* that front.

Even in those tasks where the MOGP ensemble results are similar to, or dominated by, canonical GP using Ave or Auc (such as Yst_1), the ensembles still perform better than most of its individual members. In these tasks a likely reason for the not very good MOGP ensemble performance is the relatively poor performance of the Pareto-approximated fronts compared to the frontier generated by Ave (different W values in Eq. 2). In Figure 3, the average evolved front in both the Ped and Bal tasks, i.e., the median attainment surface by MOGP with SPEA2, is clearly dominated by the Ave frontier. In other words, very high accuracy cannot be expected from the ensemble if the individual ensemble members themselves are not sufficiently accurate when compared to the Ave frontier. This highlights the importance of developing a good underlying multi-objective algorithm to trace out an accurate and diverse set of ensemble members across all the tasks. We leave this improvement for future work.

F. Analysis of GP Trees

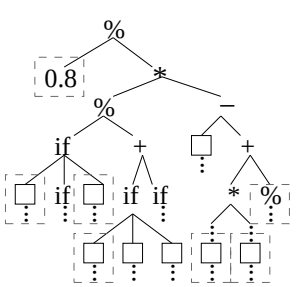
An advantage of GP is the representation of the evolved classifiers. Examining the evolved GP trees can provide useful insights into how GP learns to solve a particular problem. We examine a number of typical evolved MOGP classifiers (using the PFC approach) from the Bal task as the high level of class imbalance in Bal makes this a difficult classification problem to solve (as demonstrated by the highly-biased classification results for canonical GP, NB and SVM in Table VIII). Figures 4 and 6 show these programs where the four input features in Bal correspond to $f0 - f3$ in these programs (both programs[4] have a depth of 8).

The first program we analyse is shown in Figure 4. This solution represents a non-dominated solution in the evolved population which achieves 83% and 91% accuracy on the minority and majority class, respectively, on the test set. The second program we analyse (another non-dominated solution from the same run), scores 90% and 80% accuracy on the minority and majority class, respectively, and is identical to Figure 4 except for seven major differences (underlined in Figure 4). These seven differences, shown in Figure 5(b), are responsible for the variation in performance between the two solutions. The overall tree structure shared by both these non-dominated solutions are shown in Figure 5(a), where the (dashed) squares around a particular sub-tree show where in

[4]For convenience, function nodes in Figures 4 and 6 whose input arguments all correspond to leaf nodes that are randomly generated numbers have been manually replaced by their evaluated ouput (e.g. sub-tree $(+\ 0.5\ 0.1)$ replaced by leaf node 0.6).

```
(% 0.8 (* (% (if<0 (* (- (% 0.6 f0) (% f0 f3)) (if<0 (- 0.5
f2) f2 (- f1 -0.7))) (if<0 (if<0 (% -0.7 f3) (+ f1 f3) (+
f0 f2)) (% (% f2 0.6) (- f1 f2)) (+ (- f2 f1) (- f3 f0)))
(* (- f2 (* f0 -0.5)) (% (+ -0.6 f2) (+ -0.1 f2)))) (+ (if<0
(% (- f2 f1) (% 0.5 f0)) (- (f3 (- f3 f1)) (+ (if<0 0.2 f1
0.2) (if<0 f1 f3 f3))) (if<0 (+ (- -0.6 0.2) (* 0.2 -0.1))
(* (* f1 0.4) f0) (- (% f0 f2) -0.8)))) (- (% (if<0 (* (*
-0.4 f2) (- 0.5 f2)) (* 1.0 (* -0.1 f1)) (if<0 f3 (% f0 0.1)
0.9)) (% (+ (% f0 -1.0) 0.8) (- (+ -0.7 -0.1) (* f3 f2))))
(+ (* (- (+ f1 f0) (if<0 f3 f2 f0)) (% -0.1 (- f2 -0.5)))
(% f1 0.7)))))
```

Fig. 4. A good evolved GP tree (for the Bal task).



(a)

```
1. f0
2. 0.2
3. (* (% 0.3 f1) 1.3)
4. (i<0 f0 0.9 f0)
5. 0.7
6. (if<0 f0 -0.1 f3)
7. (if<0
       (% 0.5 (+ -0.7 f2))
       -0.8
       -1.7)
```

(b)

Fig. 5. (a) Overall structure of two GP trees (for Bal) where □ represents a sub-tree (omitted) and the dashed rectangles (around a given sub-tree) show where in the overall structure the seven differences occur; and (b) sub-trees in the second GP tree that are different from Figure 4.

the tree the seven differences occur. Figure 5(a) shows that the overall structure of the evolved GP programs can be decomposed to examine how the learnt GP classifiers solve a particular problem. These two particular solutions use a series of nested `if` conditions within the tree, in combination with the other functions ($+$,$-$,$\times$ and %).

The third non-dominated solution from the same run that we analyse is shown in Figure 6. This solution achieves lower accuracies than the previous solutions, 72% and 67% on the minority and majority class, respectively, and is also noticeably smaller. This solution does not share the same overall structure as the previous two solutions discussed above (the right side of this tree, relative to the root node, is infact completely different). For example, Figure 6 only has two `if` conditions and these occur deep within the tree near the leaf nodes (unlike in the previous two solutions).

```
(- 0.9 (+ (+ (* (* (if<0 0.4 f0 (- f2 f1)) (- f1 f2) (- f1
(% (* 0.6 f3) (- -0.7 f0)))) f0) (- f3 (* (% (- (if<0 f2
-0.2 f1) (- f2 f0)) (- (- f0 f3) 0.03)) -0.06))))
```

Fig. 6. A smaller evolved GP tree (for the Bal task).

Inspection of the evolved programs for other tasks reveals a similar pattern, that is, the *best* programs evolved by MOGP with PFC share a similar overall structure but this structure is different in other non-dominated solutions. Solutions with similar performances on the objectives have similar overall structures. We expect that when non-dominated programs are grouped together based on their performances on the objectives, the overall structure of programs within each group are relatively similar to each other, but different from programs in other groups. This is because solutions in different groups will have different building blocks. For example, the common `if` conditions in Figure 5(a) may constitute good building blocks as these are common in well-performing solutions in the same run (Figure 4). Likewise, the solution shown by Figure 6 may use different building blocks which allow this particular program (and other similar-performing programs) to specialise on certain parts of the input-space. This diverse nature of evolved programs allow the ensembles to improve system performances.

VII. Further Discussion

This section discusses several important related aspects.

A. Raw Output-Based Ensemble Combination Strategies

To obtain the ensemble output, this paper uses a majority vote of the class decisions of the individual ensemble members. As discussed, other ensemble combination strategies include the average (or weighted average) of the raw outputs of the individual members, or a winner-takes-all approach where the highest individual output is taken as the ensemble output. However, these two combination strategies are not suitable for the MOGP classifiers for two important reasons. Firstly, the raw (real-valued) outputs of the MOGP classifiers have no bounds (can be anything between $-\infty$ and $+\infty$). Secondly, the magnitude of the raw outputs are not an indication of the confidence of the class decision; these can be arbitrarily large or small for different MOGP classifiers. This means that unless the raw outputs are first scaled for consistency in the population, ensemble combination strategies using the raw outputs can be unjustifiably influenced by individuals with large output values. Adapting the MOGP approach (with output scaling that reflects the confidence of an individual's class decision) is beyond the scope of this paper but will be considered for future work.

B. Ensemble Optimisation

Finding the best combination of individuals (from the pool of learnt base classifiers) to form the ensembles can be thought of as a separate combinatorial optimisation problem. After the initial training phase to learn the base classifiers, a secondary optimisation/search process can be invoked to find the best combination of base classifiers which produces the best ensemble results [28][30][39][40]. Previous work has deployed a secondary search to optimise ensemble performances post-training [28][39], or in parallel with training the base classifiers [30][40]. Naturally, this secondary optimisation process incurs an additional computational cost, and in some cases, an extra validation set is required to avoid over-fitting the training set. This area represents an interesting direction for future work but is outside the scope of this paper.

C. Validation Sets

We have tested the MOGP approaches on six benchmark problems with different data and variations. They contain certain levels of noise. The performances reported are on the unseen test tests, i.e., 50% of the original data set, for each

task. Inspection of the training and test results show that no clear overfitting has occurred on these tasks. If overfitting does occur, an additional validation set can be used to address this, e.g., using a validation set to determine the stopping criteria in the training process to avoid overfitting (such as in [39][30]). In this case, classification performances can be examined on both the validation set and test set to determine which is the best learned model (as used in [30]).

VIII. Conclusions

The three goals of this paper were to develop a MOGP framework for classification with unbalanced data using the accuracy of the minority and majority class as learning objectives, compare two Pareto-based fitness strategies in MOGP, and adapt the MOGP approach to evolve accurate and diverse ensembles. These goals have been successfully achieved. The evolved ensembles comprise of non-dominated genetic program classifiers where each member votes on the class of a given instance.

Our experimental results using six binary unbalanced data sets show that the MOGP approaches evolved an accurate set of genetic program classifiers along the minority and majority class trade-off frontier on these tasks, particularly when a combination of dominance rank and dominance count in used (compared to dominance rank alone) in fitness.

The additional ensemble-diversity measures in fitness also encouraged good cooperation among the evolved MOGP ensemble members. The MOGP ensembles typically dominated the performance of its individual members, canonical GP, NB and SVM, particularly on tasks with high levels of class imbalance. The PFC penalty in particular may be more effective in evolving solutions with better diversity compared to NCL on the tasks. We also find that well-performing ensembles can be evolved using only the stochastic processes within the evolution provided that only accurate and diverse solutions are used in the final vote as biased solutions can negatively influence the ensemble vote.

For future work we will evaluate this approach on more unbalanced data sets, and investigate other techniques for ensemble diversity (such as root quartic NCL) and other ways to incorporate these diversity measures into the fitness functions to improve ensemble performances. We will continue to improve the underlying MOGP framework for difficult problems, and investigate other ensemble optimisation strategies.

Acknowledgment

This work is supported in part by the Marsden Fund of New Zealand Government (under contract number VUW0806), administrated by the Royal Society of New Zealand. Xin Yao's research has received funding from the European Union Seventh Framework Programme (grant agreement no. 270428).

References

[1] N. Japcowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent Data Analysis*, vol. 6, no. 5, pp. 429–450, 2002.

[2] R. Barandela, J. S. Sanchez, V. Garcia, and E. Rangel, "Strategies for learning in class imbalance problems," *Pattern Recognition*, vol. 36, no. 3, pp. 849–851, 2003.

[3] G. M. Weiss and F. Provost, "Learning when training data are costly: The effect of class distribution on tree induction," *Journal of Artificial Intelligence Research*, vol. 19, pp. 315–354, 2003.

[4] S. J. Stolfo, D. W. Fan, W. Lee, A. L. Prodromidis, and P. K. Chan, "Credit card fraud detection using meta-learning: Issues and initial results," in *AAAI Workshop on AI Approaches to Fraud Detection and Risk Management*, pp. 83–90, 1997.

[5] J. H. Holmes, "Differential negative reinforcement improves classifier system learning rate in two-class problems with unequal base rates," in *Proceedings of the Third Annual Conference on Genetic Programming*, pp. 635–644, 1998.

[6] S. Munder and D. Gavrila, "An experimental study on pedestrian classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1863–1868, 2006.

[7] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[8] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 38, no. 3, pp. 397–415, 2008.

[9] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," tech. rep., 2001. TIK-Report 103, Department of Electrical Engineering, Swiss Federal Institute of Technology.

[10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast elitist multi-objective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2000.

[11] U. Bhowan, M. Johnston, and M. Zhang, "Multi-objective genetic programming for classification with unbalanced data," in *Proceedings of the 22nd Australasian Joint Conference on Artificial Intelligence*, vol. 5866 of *LNCS*, pp. 370–380, Springer, 2009.

[12] Y. Liu and X. Yao, "Negatively correlated neural networks can produce best ensembles," *Australian Journal of Intelligent Information Processing Systems*, vol. 4, pp. 176–185, 1997.

[13] H. Chen and X. Yao, "Multiobjective neural network ensembles based on regularized negative correlation learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 12, pp. 1738–1751, 2010.

[14] H. Dam, H. Abbass, C. Lokan, and X. Yao, "Neural-based learning classifier systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 1, pp. 26–39, 2008.

[15] A. Chandra and X. Yao, "Ensemble learning using multi-objective evolutionary algorithms," *Journal of Mathematical Modelling and Algorithms*, vol. 5, pp. 417–445, 2006.

[16] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: one-sided selection," in *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 179–186, 1997.

[17] G. Batista, R. C. Prati, and M. C. Monard, "Balancing strategies and class overlapping," in *Advances in Intelligent Data Analysis VI, 6th International Smposium on Intelligent Data Analysis, IDA 2005* (A. F. Famili, J. N. Kok, J. M. Peña, A. Siebes, and A. J. Feelders, eds.), vol. 3646 of *LNCS*, pp. 24–35, Springer, 2005.

[18] Y. Tang, Y.-Q. Zhang, N. Chawla, and S. Krasser, "SVM modeling for highly imbalanced classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 1, pp. 281 –288, 2009.

[19] A. Mclntyre and M. Heywood, "Multi-objective competitive coevolution for efficient GP classifier problem decomposition," in *IEEE International Conference on Systems, Man and Cybernetics*, pp. 1930 –1937, 2007.

[20] S. Wang, K. Tang, and X. Yao, "Diversity exploration and negative correlation learning on imbalanced data sets," in *International Joint Conference on Neural Networks*, pp. 3259–3266, 2009.

[21] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," in *IEEE Symposium on Computational Intelligence and Data Mining*, pp. 324–331, 2009.

[22] E. Alfaro-Cid, K. Sharman, and A. Esparcia-Alcazar, "A genetic programming approach for bankruptcy prediction using a highly unbalanced database," in *Applications of Evolutionary Computing* (M. Giacobini, ed.), vol. 4448 of *LNCS*, pp. 169–178, Springer, 2007.

[23] U. Bhowan, M. Zhang, and M. Johnston, "Genetic programming for classification with unbalanced data," in *Proceedings of the 13th European Conference on Genetic Programming*, vol. 6021 of *LNCS*, pp. 1–13, Springer, 2010.

[24] J. Doucette and M. I. Heywood, "GP classification under imbalanced data sets: Active sub-sampling and AUC approximation," in *Proceedings*

of 11th European Conference in Genetic Programming (EuroGP 08), pp. 266–277, 2008.
[25] D. Song, M. Heywood, and A. Zincir-Heywood, "Training genetic programming on half a million patterns: an example from anomaly detection," *IEEE Transactions on Evolutionary Computation*, vol. 9, pp. 225–239, June 2005.
[26] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems, LNCS*, vol. 1857, pp. 1–15, Springer-Verlag, 2000.
[27] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.
[28] N. Chawla and J. Sylvester, "Exploiting diversity in ensembles: improving the performance on unbalanced datasets," in *Proceedings of the 7th International Conference on Multiple Classifier Systems*, MCS, pp. 397–406, Springer-Verlag, 2007.
[29] A. Chandra and X. Yao, "Divace: Diverse and accurate ensemble learning algorithm," in *Intelligent Data Engineering and Automated Learning*, vol. 3177 of *LNCS*, pp. 619–625, Springer, 2004.
[30] M. Brameier and W. Banzhaf, "Evolving teams of predictors with linear genetic programming," *Genetic Programming and Evolvable Machines*, vol. 2, no. 4, pp. 381–407, 2001.
[31] R. Thomason and T. Soule, "Novel ways of improving cooperation and performance in ensemble classifiers," in *Proceedings of Genetic and Evolutionary Computation Conference*, pp. 1708–1715, ACM, 2007.
[32] H. Abbass, "Pareto-optimal approaches to neuro-ensemble learning," in *Multi-Objective Machine Learning* (Y. Jin, ed.), vol. 16 of *Studies in Computational Intelligence*, pp. 407–427, 2006.
[33] H. A. Abbass, "Pareto neuro-evolution: constructing ensemble of neural networks using multi-objective optimization," in *IEEE Congress on Evolutionary Computation*, vol. 3, pp. 2074–2080, 2003.
[34] H. Abbass, "Pareto neuro-ensembles," in *AI 2003: Advances in Artificial Intelligence*, vol. 2903 of *LNCS*, pp. 554–566, 2003.
[35] R. McKay and H. A. Abbass, "Anticorrelation measures in genetic programming," in *Australasia-Japan Workshop on Intelligent and Evolutionary Systems*, pp. 45–51, 2001.
[36] R. McKay and H. A. Abbass, "Anti-correlation: A diversity promoting mechanisms in ensemble learning," *The Australian Journal of Intelligent Information Processing Systems*, vol. 7, no. 3/4, pp. 139–149, 2001.
[37] C. Gagné, M. Sebag, M. Schoenauer, and M. Tomassini, "Ensemble learning for free with evolutionary algorithms?," *Proceedings of Genetic and Evolutionary Computation Conference*, pp. 1782–1789, 2007.
[38] D. W. Opitz and J. W. Shavlik, "Generating accurate and diverse members of a neural-network ensemble," in *Advances in Neural Information Processing Systems*, pp. 535–541, MIT Press, 1996.
[39] X. Yao and Y. Liu, "Making use of population information in evolutionary artificial neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 28, no. 3, pp. 417–425, 1998.
[40] H. Chen, P. Tino, , and X. Yao, "Predictive ensemble pruning by expectation propagation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, pp. 999–1013, 2009.
[41] U. Bhowan, M. Zhang, and M. Johnston, "Evolving ensembles in multi-objective genetic programming for classification with unbalanced data," in *Proceedings of Genetic and Evolutionary Computation Conference*, pp. 1331–1339, ACM, 2011.
[42] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 2, pp. 539–550, 2009.
[43] S. Wang and X. Yao, "Theoretical study of the relationship between diversity and single-class measures for class imbalance learning," in *Proceedings of the IEEE International Conference on Data Mining Workshops*, ICDMW, pp. 76–81, 2009.
[44] A. Mclntyre and M. Heywood, "Classification as clustering: A pareto cooperative-competitive GP approach," *Evolutionary Computation*, vol. 19, no. 1, pp. 137–166, 2011.
[45] C. Gathercole and P. Ross, "Dynamic training subset selection for supervised learning in genetic programming," in *Parallel Problem Solving from Nature (PPSN III)* (Y. Davidor, H.-P. Schwefel, and R. Manner, eds.), vol. 866 of *LNCS*, pp. 312–321, Springer, 1994.
[46] Z. Wang, K. Tang, and X. Yao, "Multi-objective approaches to optimal testing resource allocation in modular software systems," *IEEE Transactions on Reliability*, vol. 59, no. 3, pp. 563–575, 2010.
[47] A. Asuncion and D. Newman, "UCI Machine Learning Repository," 2007. `http://www.ics.uci.edu/~mlearn/MLRepository.html`. University of California, Irvine, School of Information and Computer Sciences.
[48] C. Coello Coello, G. Lamont, and D. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic & Evolutionary Computation Series)*. Springer, 2007.
[49] J. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," tech. rep., February 2006. No. 214, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich.
[50] D. C. Hoaglin, F. Mosteller, and J. W. Tukey, *Fundamentals of Exploratory Analysis of Variance*. Wiley, 1991.
[51] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve.," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.
[52] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *SIGKDD Explorations*, vol. 11 (1), 2009.