Multi-Objective Competitive Coevolution for Efficient GP Classifier Problem Decomposition*

A. R. McIntyre and M. I. Heywood†

July 26, 2007

Abstract

A novel approach to the classification of large and unbalanced multi-class data sets is presented where the widely acknowledged issues of scalability, solution transparency, and problem decomposition are addressed simultaneously within the context of the Genetic Programming (GP) paradigm. A cooperative coevolutionary training environment that employs multi-objective evaluation provides the basis for problem decomposition and reduced solution complexity, while scalability is achieved through a Pareto competitive coevolutionary framework, allowing the system to be applied to large data sets (tens or hundreds of thousands of exemplars) without recourse to hardware-specific speedups. Moreover, a key departure from the canonical GP approach to classification is utilized in which the output of GP is expressed in terms of a non-binary, local membership function (e.g. a Gaussian), where it is no longer necessary for an expression to represent an entire class. Decomposition is then achieved through reformulating the classification problem as one of cluster consistency, where an appropriate subset of the training patterns can be associated with each individual such that problems are solved by several specialist classifiers rather than by a single 'super' individual.

1 INTRODUCTION

Classification is a central task in machine learning and data mining applications and has received considerable attention in the Genetic Programming (GP) community. While recent results are promising, several problems remain universally acknowledged, specifically with respect to how GP may be readily employed in large, 'real-world' classification problems. In such instances, training sets consist of tens or hundreds of thousands of exemplars and the class distributions

*2007 IEEE Conference on Systems, Man, and Cybernetics, IEEE Press.

†Faculty of Computer Science, Dalhousie University. Halifax, N.S., Canada.

may be unbalanced. Ensuing solutions must provide good generality (performance on unseen exemplars) yet remain computationally tractable. Moreover, many real-world classification problems are multi-class (as opposed to binary) in nature and potentially require several class-specific expressions to collaborate in order to provide adequate coverage over all classes. The conventional approaches to classification within the GP paradigm do not adequately address these questions in general, and the current work is intended to provide a more comprehensive framework for classification under the GP context. Specifically, we employ tools from the evolutionary multi-objective and coevolution literature to achieve scalable, multi-class solutions that decompose the problem from a single population.

Classification problems under the GP paradigm are typically configured to calculate a single value that is used to characterize the quality of an individual in terms of a specific performance metric (e.g. accuracy, sum squared error, weighted combination of sensitivity / specificity etc. . .) [14]. This value is then scaled, providing a fitness value that indicates the individual's probability for selection during evolution. This naturally leads the search to a single individual having the largest fitness value, which is then used to solve the classification problem alone. The implicit assumption in this model however, is that one expression (moreover, a single objective) is sufficient and / or appropriate to solve the classification problem. Moreover in the case of binary problems, classification decisions under GP conventionally take the form of a hard switching function arbitrarily centered at zero. GP outputs greater than zero indicate 'in-class' patterns, while those less than or equal to zero are considered to be 'out-of-class'. This approach has previously been extended to multi-class problems by combining binary classifiers to form hierarchies with one individual assigned to each class. This normally requires a separate set of initializations for each class and while this is computationally costly, a considerable amount of effort has been expended in pursuit of extensions to this approach, where none of these specifically address the large-scale, multi-class nature of the problems directly [11]. Moreover, the hierarchical model precludes their utility when multiple labels are required per exemplar. Scalability, particularly with respect to multi-class problems, thus remains a widely acknowledged issue in the classification context of the GP paradigm where recent approaches in the literature have invoked hardware speedups [7] [17] [1] or sub-sampling of the training sets [8] [21] [3].

The current work considers the evolution of classifiers comprised of multiple, cooperating models with Evolutionary Multi-objective Optimization (EMO) providing a basis for comparing the performance of individuals in the presence of multiple performance objectives using the criterion of Pareto dominance [19]. Unlike previous works, a local membership function (a Gaussian) is employed within an EMO context to encourage collaborations, providing the basis for automatic problem decomposition. Moreover, the scalability of GP with respect to data set size is explicitly addressed through the use of a Pareto competitive coevolution training framework which provides multi-class solutions in a single run of GP. The resulting framework extends the previous cooperative coevolu-

tionary classifier [15] into a competitive-cooperative coevolutionary model, thus significantly extending the versatility of the model as a whole.

2 ALGORITHMS

2.1 Pareto Dominance

The following terminology from the EMO literature will be employed throughout the remainder of this paper. An individual A Pareto dominates individual B if A is no worse than B on any objective and is strictly better than B on at least one [12]. An individual is non-dominated when it is not Pareto dominated by any others, and the Pareto front describes the set of all non-dominated individuals.

2.2 Grammatical Evolution

The following work is undertaken using a specific variant of GP, known as Grammatical Evolution, or GE (although the algorithm is not specific to the type of Genetic Programming employed). GE permits automatic and language independent evolution of programs of arbitrary complexity [18]. There are obvious similarities between GE and GP; however, GE does not operate directly on the expression phenotypes themselves as in traditional GP; rather the programs are stored as a series of Backus-Naur form (BNF) grammar rule selectors, which are in turn indirectly represented by a fixed length binary string genotype. In this sense GE is similar to GA however, the model does benefit from the utilization of context aware variation operators that minimize the disruption of decoded genotypes [9]. Since the algorithm presented here is not specific to GE and because of the numerous similarities, the terms GE and GP are used interchangeably in this paper.

For the classification algorithm presented here, individuals represent simple arithmetic expressions within our GE framework. A sample context free grammar for a problem having i features (or attributes) is provided below:

```
code:  exp
exp:   exp|(exp)|var|exp op exp|preop(exp)
preop: sin|cos|sqrt|log|expn
op:    +|-|*|%
var:   x0| ... | x(i-1)
```

2.3 Classifier Framework

Within the context of this work, classifier individuals take the form of *learners* defined as GE expressions and the *training points* represent indices into the training data. The interaction between a learner and a point is defined by evaluating the learner expression using the point data and producing an output value (*gpOut*) indicating degree of class membership using a local (Gaussian) membership function [15]:

$$y\left(gpOut\right) = exp\left(-\frac{\left(gpOut-\mu\right)^2}{2\sigma^2}\right) \tag{1}$$

The values of μ and σ for a given learner are established by a clustering function (the Potential Function [2]), evaluated on the *gpOut* values corresponding to the point population at the time of the learner's creation. The clustering function identifies:

1. μ: A single value on *gpOut*, defining the centroid of the largest and most dense cluster (the learner is then assumed to identify points having the same class as the *gpOut* point corresponding to μ);
2. M: The set of cluster member points on *gpOut* (i.e. the points corresponding to *gpOut* values near μ).

The set M is then used to estimate the value for σ:

$$\sigma = \sqrt{\frac{1}{|M|-1}\sum_{i=1}^{|M|}(M_i-\mu)^2} \tag{2}$$

For a given point, the individual's classification (i.e. decision output) $\hat{o}$ for a given point mapped on *gpOut* is estimated by:

$$\hat{o} = \begin{cases} 1 & \text{if } y\left(gpOut\right) \in [y\left(\mu-\sigma\right), y\left(\mu\right)] \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

This represents a key departure from the standard GP approach to classification, where the classifier typically invokes a hard global switching function centered at zero to render the decision (i.e. if $gpOut \leq 0$ then return class 0, else return class 1 [11]). Here, the use of a local membership function (LMF) permits expressions to represent a subset of the data such that problem decomposition is facilitated and solutions take the form of several specialist classifiers rather than a single super individual. Moreover, at this stage no attempt is made to incorporate the concept of class membership. Instead the formation of LMFs with consistent class membership will be enforced through the multi-objective fitness evaluation, Section 2.6.

2.4 Training Overview

The approach taken for training in this work employs two main components. The first component provides for a multi-objective evaluation of fitness, Algorithm 1, step 2a. Adopting such a scheme enables the user to encourage, say, simple as well as accurate classifiers. Moreover, as a natural consequence of the EMO paradigm, solutions might take the form of more than one classifier for the same class [15]. The second component addresses the computational overhead of fitness evaluation by using a Pareto (competitive) coevolutionary algorithm

to identify minimal sets of training examples to conduct training, Algorithm 1 step 2b.

The use of a Pareto coevolutionary archiving mechanism in our training algorithm was motivated by the work of Lemczyk et al. [13] (work which was originally based on de Jong's Incremental Pareto-Coevolution Archive (IPCA) algorithm [4]). The archiving framework presented here, however, extends beyond the original binary context to multi-class, where separate learner and point archives are introduced for the purpose of supporting learning on the basis of class-appropriate objectives. Moreover, unlike Lemczyk we also make use of multiple archives with early stopping criterion based on the behavior of the classwise Pareto fronts, Algorithm 1 step 2c, enabling class / problem specific stop points to be identified [12]. The ensuing Pareto competitive archive model therefore reduces the number of exemplars necessary to perform fitness evaluation as well as providing class wise early stopping.

As outlined in Table 1, the algorithm employs $2c+2$ populations and archives, where c is specified by the number of classes in the problem. That is, separate point and learner archives are retained for each class. This ensures that point archives are able to retain the most appropriate tests (training points) for learners associated with each class. The descriptions that follow employ additional data evaluation structures which are summarized in Table 1.

Sections 2.5 to 2.11 provide detailed descriptions of the basic steps of Algorithm 1. All parameter values are specified in Table 2. We note that no attempt was made to optimize these selections. The over-riding interest here is to provide uniformity across all data sets under consideration.

Algorithm 1 *High-level Pareto Coevolution.*

1. *Initializations;*
2. *Training loop:*
   (a) *Multi-objective evaluation and reproduction;*
   (b) *IPCA-based archiving;*
   (c) *Check for early stop criteria.*
3. *Post-training: Assembly of classifiers.*

2.5 Initializations

Initialization proceeds as indicated in Algorithm 2. A single GE-based learner population is initialized by assigning gene values with uniform probability and attempting to map to a legal expression as defined by the grammar (Section 2.2). This process is repeated until a legal mapping is successful such that no degenerate GE individuals are defined in the initial population [18]. There are no requirements on expression length, however the genotype is limited in the number of rule selection values (by the CODONS parameter) and expressions

Table 1: Algorithm Data Structures and Parameters

Populations and archives			
Description	**Abbrev.**	**Num**	**Capacity**
Point population	PP	1	PP_MAX_SIZE
Learner population	LP	1	POP_SIZE
Point archives	PA	c	PA_MAX_SIZE, ea.
Learner archives	LA	c	LA_MAX_SIZE, ea.
Evaluation structures			
Non-converged classes	NC	1	c
Current rank histograms	RH^t	c	$\|LP\|$
Previous rank histograms	RH^{t-1}	c	$\|LP\|$

are constrained to a maximum string length of MAX_EXP_LEN (see Table 2). Archives may only accept individuals satisfying the Pareto dominance criteria, Section 2.8, thus initially contain no individuals. Moreover, the point population is required to maximize diversity, hence is reinitialized after each epoch, Section 2.6. Finally, step 5 of Algorithm 2 represents the initialization of the data structure used to detect early (class-wise) convergence.

Algorithm 2 *Initializations*

1. $LP := initLearners(LP)$
2. $LA_1 \dots LA_c := \{\varnothing\}$
3. $PP := \{\varnothing\}$
4. $PA_1 \dots PA_c := \{\varnothing\}$
5. $NC := \{1 \dots c\}$

2.6 Training

At the outset of each epoch, the point population (PP) is filled using random uniform selection without replacement over the range of all training pattern indices in step 1a of Algorithm 3. A balanced representation from each classes of the training data is enforced where such a bias is known to improve the resilience of decision tree classifiers to the class imbalance problem [22]. The training set (TS) for the epoch is then the union of all point archives and the point population, Algorithm 3 step 1b. Following evaluation of fitness using a multi-objective paradigm (step 1c) in which a Pareto ranking is used to provide a scalar ranking of individuals in the training population. GE variation operators take the form of context aware crossover or mutation [9], steps 1(d)ivA and 1(d)vB respectively. The selection of parents, steps 1(d)i and 1(d)iiiA, is

fitness proportionate.

Algorithm 3 *Evolutionary Training Loop*

1. *For epoch := 1 to MAX_EPOCHS*
   (a) $PP := fillPtPop()$
   (b) *Training set* $TS := PA_1 \cup \ldots PA_c \cup PP$
   (c) $evalLearners(LP, TS)$:
      i. $evalObjectives(LP, TS)$
      ii. $rankLearners(LP)$
      iii. $calcFitness(LP)$
   (d) *For 1 to* $|LP|$
      i. *Select parent* $p1 \in LP$
      ii. $tries := 0$; $p2 := \varnothing$
      iii. *While* $class(p1) \neq class(p2)$ *&* $tries < c$
         A. *Select parent* $p2 \in LP$;
         B. $tries{+}{+}$
      iv. *If* $testXover(PXO) = true$
         A. *Children* $C := applyXover(p1, p2)$
      v. *Else*
         A. $C := \{p1, p2\}$
         B. $applyMutation(C,\ MR)$
      vi. $performReplacement(LP, C, TS)$:
         A. $evalLearners(C, TS)$
         B. $replaceLearners(LP, C)$
         C. $rankLearners(LP)$
   (e) *IPCA-based archiving;*
   (f) *Check for early stop cirteria.*
2. *Post-training: Assembly of classifiers.*

Prior to proceeding to the IPCA-based archiving (step 1e) and evaluation of early stopping criteria (step 1f) we will describe the nature of point and learner populations in more detail.

The point population plays an exploratory sampling role in the coevolution process and maintains a balanced view of the training data from the perspective of the learners. Specifically, the point generation function ($fillPtPop$) step 1a of Algorithm 3 selects exemplar indices (with uniform probability) for each class. The function explicitly represents each class of the problem with equal exemplar counts in the population (assuming that there are enough points in the training

set to do so). The point population thus provides the basis for balancing the training data. Moreover, scalability of the training algorithm is achieved by the sampling property of the point population, as it explicitly limits the maximum number of learner evaluations required at the computationally costly inner loop of GP (step 1(d)viA of Algorithm 3).

GE is employed to induce classifier expressions through the multi-objective evolutionary cycle. To begin each epoch (where new point data has been selected into the point population) the learner population is evaluated against the training set TS, which is the union of the point population and the point archives in steps 1b and 1c of Algorithm 3. The basic procedure for learner generation at each iteration of step 1d first involves selection of two parents, the second parent is a classifier from the same class as the first parent. Genetic variation operators are then applied to the parents to create children. Replacement follows the rule that children always replace the lowest ranked member of the learner population. After the replacement step, the learner population is re-ranked and the next selection iteration begins.

At this stage each learner has an output expressed in terms of a Gaussian LMF, Section 2.3, or $y_i = y(gpOut_i), i \in TS$. The mean and variance of this LMF reflect the region of highest density relative to the mapping

$$gpOut_i = f(x_i); i \in TS, \tag{4}$$

where $f(\cdot)$ is the mapping between multidimensional input x_i and single dimensional output $gpOut$ and i indexes training examples in the training set TS. The basic objective is now to incorporate class consistent properties onto the mapping of (4) and therefore the points associated with the LMF. To this end, a multi-objective model is utilized. Central to establishing these objectives are the concepts of 1) error, relative to exemplar class and degree of LMF membership; 2) in-class membership count, where this is defined with respect to the region defined by the LMF; 3) overlap minimization where in-class exemplars are discouraged from being a member of more than one LMF; 4) solution parsimony. The definitions of the objectives are summarized as follows:

1. **Minimize the sum of squared error (SSE):** This objective explicitly enforces cluster class-consistency by evaluating classification performance over the cluster member points and rewarding true positive classifications while discouraging the occurrence of false positives (an individual mistakenly labeling a pattern of class 0 as class 1). The SSE for an individual is calculated over M, the set of cluster member points (defined in Section 2.3):

$$\text{SSE} = \sum_{i=1}^{|M|} (label_i - y\,(gpOut_i))^2, \tag{5}$$

with y as defined in (1) and $label$ taking on binary values: 1 when the current pattern label is the same as the class as that of the current learner (as determined by the clustering) and 0 otherwise.

2. **Maximize in-class patterns correctly classified:** This objective is designed to encourage survival of individuals that correctly map many patterns densely in *gpOut*, while discouraging the case of single point coverage by classifiers.

3. **Minimize pattern overlap:** This objective is intended to discourage intersection in the sets of patterns that are correctly classified between learner archives and the current learner population. The overlap value for an individual is defined as a count (i.e. sum) of number of times that each exemplar that is correctly classified is also correctly classified by members of the *learner archive*. This makes a significant improvement relative to an earlier version of the MOGE classifier at [15].

4. **Minimize expression length:** Based on results obtained in [19] and [5], this objective imposes parsimony in learner expressions. The expression length is defined as the string length of the unsimplified learner expression.

2.7 Pareto Ranking and Fitness Assignment.

The method of ranking with ties is employed to determine fitness of members of the learner population [12]. When evaluated, the rank of an individual is defined by the number of individuals by which it is Pareto dominated (as defined in Section 2.1) plus one. All non-dominated solutions are given the rank of 1 and in the event of a tie (i.e. two learners having the same value in all objectives), one of the ranks is randomly increased by one. The fitness of an individual is assigned in direct proportion to Pareto rank.

2.8 Point and Learner Archive Entry

Based on the IPCA algorithm, archive insertion (step 1e of Algorithm 3) is driven by the notion of providing distinctions between learners [4]. The concept of distinctions was shown to specifically address the coevolutionary problem of disengagement [6], where the point population dominate the learner population resulting in a loss of training gradient. This can occur when points are rewarded for explicitly defeating the learners rather than distinguishing between them.

The approach employed here differs from IPCA and Lemczyk *et al.* [13] in two respects. First, multiple independent archives are maintained concurrently, each corresponding to a different class of the problem. This is necessary in order to ensure interactions maintain a training gradient that is relevant with respect to each class. Within each point archive, we enforce a 50-50 balance between {in, out}-of-class points on the archive contents (where the notion of an 'in' or 'out-of-class' point is obviously with respect to the class of the archive). This approach demonstrated superior results in the single archive binary system, reported in [13], especially on unbalanced data sets, and follows similar recommendations for other machine learning methods [22].

Secondly, the definition of an outcome has been reformulated to take on real values as opposed to binary. An outcome is the result of an interaction

between a learner and a point. The set of outcomes defined for each learner with respect to its corresponding archive provides the basis for which archive entry is assessed. In this work, outcomes are evaluated with respect the class of the learner and take on real values in the range (0,1] based on the the learner's membership y as defined in (1). A learner's outcome for a given point is defined as:

$$outcome = \begin{cases} y & \text{if point is in-class} \\ 1 - y & \text{otherwise} \end{cases} \tag{6}$$

Consistent with IPCA, a point is said to provide a distinction (i.e. it is considered to be a *useful test*) if the outcomes of a learner that was previously Pareto dominated by the outcomes of at least one member of the learner archive become Pareto equivalent to the learner archive (i.e. it becomes a *useful learner*) with the addition of the point to the point archive. If a newly generated point, p, is useful with respect to a learner, l (of class c) and the class c learner archive, then p is committed to the point archive corresponding to class c. Similarly, if a newly generated learner, l, (of class c) is useful with respect to the class c learner archive when evaluated against the class c point archive, then l enters the class c learner archive. In order to maintain an upper bound on the archives, pruning may be necessary prior to insertion of the new member.

2.9 Archive Pruning

A maximum size is imposed on both learner and point archives in order to sustain computational and resource efficiency in the training algorithm [13]. Efficiency is therefore achieved at the potential expense of accuracy of learner evaluation. In the case of the learner archive, pruning risks correctly identifying a complete set of expressions that are able to decompose the classification problem, while in the case of the point archive pruning may introduce errors in identifying training objectives and cause cycles of forgetting in the learning process.

When a point archive has reached the maximum capacity for its {in, out}-of-class contents, a point is chosen for replacement by selecting the archive point of the same class having the nearest Euclidean distance to the point being inserted [13]. The distance is calculated over the pattern attributes (feature space) associated with each point. While a point is lost from the archive through this process, the assumption is that the point inserted into the archive will provide an alternative test that maintains the previous distinctions while establishing a basis for further learning.

When a learner archive has reached the maximum size, a member must be chosen for replacement. Currently we select for replacement the archive member having the lowest sum of outcomes over the corresponding point archive.

2.10 Early Stopping Criteria

To identify a converged state among learner population members (step 1f of Algorithm 3) we employ the convenient stopping criteria identification method of Pareto-rank histograms, introduced under a GA context by Kumar and Rockett [12]. For this work, rank histograms are generated from the ratio of the number of learners at each rank in the learner population between the current and previous epochs (see Algorithm 4). Separate histograms are generated for each class, such that c histograms are constructed for a c class problem. A match between class rank histograms of two successive epochs indicates that further progress is unlikely on the class.

Algorithm 4 *Evaluation of Early Stop Criteria*

1. $\forall i \in NC$
   (a) $RH_i^t := rankHist(LP, i)$
   (b) *If* $dist(RH_i^t, RH_i^{t-1}) < MIN\_DIFF$
      i. $NC := NC - i$
      ii. $initLearners(\{lp \in LP : class(lp) = i\})$
   (c) *Else*
      i. $RH_i^{t-1} := RH_i^t$
2. *If* $NC = \{\varnothing\}$
   (a) *Exit*

In step 1.b of Algorithm 4, we also require that the number of learners corresponding to the histogram class under consideration must be beyond a minimum threshold (CONV_MIN_POP) and *epoch* (see Algorithm 3) must be beyond a minimum fraction (CONV_FRAC) of MAX_EPOCHS. For parameter specifications, see Table 2.

2.11 Post-training: Assembly of Classifiers

Post-training, the learner archives are merged to form a solution set which is then processed to identify the degree to which each of the individuals will participate in making classifications. In this work, we implement a basic winner-take-all scheme where the individual's y values (as defined in (1)) are weighted with a confidence. The class of the individual having the highest confidence weighted membership value ($w \cdot y$) is predicted by the classifier for a given input pattern. The confidence weighting ($w \in [0, 1]$) for each individual in the solution set is defined in terms of its favorability set (F) as:

$$w = \frac{max\,(F)}{\sum F}. \tag{7}$$

The elements of the favorability set F_i for an individual are defined over the classes ($i = 1 \dots c$) of the entire set of training data TD as [20]:

$$F_i = \frac{\hat{n_i}^2}{t_i \cdot \hat{N}}, \tag{8}$$

where

$$\hat{n_i} = |\{pt \in TD : \hat{o} = 1 \wedge class(pt) = i\}|, \tag{9}$$

$$t_i = |\{pt \in TD : class(pt) = i\}|, \tag{10}$$

and

$$\hat{N} = \sum_{i=1}^{c} \hat{n_i}. \tag{11}$$

3 EXPERIMENTS

All experiments are undertaken using large and unbalanced multi-class classification problems from the University of California at Irvine's (UCI) Machine Learning Repository [16]. Two experiments were designed for the current work and each was run for 50 initializations on the UCI data sets with the parameterizations summarized in Table 2.

1. **RssGE** - A standard binary GE classifier implementation that employs a sigmoid wrapper function. This provides a computationally tractable baseline GE model that employs a class-aware (balanced) version of the random subset selection (RSS) algorithm [8] in place of a point archive. RSS replaces the notion of point population and training set (PP and TS of Algorithm 3) and there is no concept of archives or early stopping in this algorithm (i.e. steps 1e and 1f of Algorithm 3 are removed). Fitness is based on a single metric that employs the error provided by the sigmoid wrapper evaluation. All other details of Algorithm 3 remain in place. As a binary classifier, each class requires a separate classifier expression, and a winner-take-all voting policy is therefore employed for consistency in comparisons.

2. **CMGE** - The Competitve Multi-objective GE described in the current work.

All experiments were conducted with an upper limit on the total number of individual evaluations. By holding the number of evaluations constant we are able to provide comparisons between algorithms such that learning efficiency per evaluation is taken into consideration. Specifically, the evaluation limit for each algorithm is defined as:

$$\text{PP\_MAX\_SIZE} \times \text{POP\_SIZE} \times \text{MAX\_EPOCHS} \tag{12}$$

per class, where these values are supplied in Table 2. Each run of RssGE would therefore be permitted the number of evaluations defined in (12) and the algorithm run for each class of the data set, whereas the equivalent CMGE result would be from a single run that was permitted a maximum number of evaluations defined in (12) multiplied by the number of classes.

3.1 Performance Measures

Evaluation will be performed from the perspective of the quality of classifiers in terms of solution sizes and and predictions made on unseen data. Classification (prediction) performance will be given class-wise in terms median values for detection rate (DR) and false positive (FPR), which are defined in terms of true positive (tp), true negative (tn), false positive (fp) and false negative (fn) as [10]:

$$DR = \frac{tp}{tp + fn} \tag{13}$$

$$FPR = \frac{fp}{fp + tn} \tag{14}$$

Solution sizes are reported in terms of median string length of the unsimplified expression (again the quartiles will indicate variation).

3.2 Data Sets

Evaluation is performed over two well known-data sets, UCI Thyroid (specifically the 'ann-train' and 'ann-test' data sets) and UCI Statlog Shuttle [16]. The general characteristics are provided in Table 3. Both data sets have predefined partitions for train and test which were used here. Each of the data sets was pre-processed to convert nominal data to numeric, remove all duplicates, contradictory patterns and patterns with missing attributes (this is performed over the combined train and test data). The Statlog Shuttle data set is highly unbalanced, with approximately 80% of all data coming from class 1, while as few as 6 patterns are provided for class 6 (in the training data). The Thyroid data set is also highly unbalanced, having approximately 92% of data belonging to class 3.

4 RESULTS

Table 4 presents a summary of Student's T-test results on the overall test accuracies of the two GP classification algorithms for both data sets. The T-test figues are the values returned using the two-tailed, two-sample with unequal variance options. Each T-test value indicates the probability of the two sample means being the same under the assumption that the population means are the same.

Table 2: Parameter Specifications

Grammatical evolution	
Parameter	**Value**
MAX_EXP_LEN	4096
CODONS	256
Archives and populations	
POP_SIZE	50
MAX_EPOCHS	500
LA_MAX_SIZE	30
PP_MAX_SIZE	30
PA_MAX_SIZE	30
Crossover and mutation	
PXO, PCXO	0.50, 0.90
PM, PTSM	0.01, 0.90
Early stopping	
MIN_DIFF	0.1
CONV_MIN_POP	20
CONV_FRAC	$\frac{1}{5}$

Table 3: Data Set Characteristics

Data Sets and Parameters		
Parameter	**UCI Statlog Shuttle**	**UCI Thyroid**
# Training exemplars	45,300	3,709
# Test exemplars	14,500	3,420
# Features	9	21
# Classes	7	3

Table 4: T-Test Results (Overall Test Accuracy)

Test	**Shuttle**	**Thyroid**
RssGE versus CMGE	0.0816	0.0000

Tables 5 and 7 provide a detailed 1st quartile, median, and 3rd quartile characterization of RssGE and CMGE performance on the Shuttle data set. It is immediately apparent that the CMGE algorithm provides much better Detection and False Positive rates, with the RssGE binary classifier only able to improve on the CMGE results for class 3. Moreover, the CMGE results are achieved with a much lower spread than that of the RssGE baseline. In terms of expression lengths, the CMGE paradigm results in simpler solutions, in spite of combining multiple individuals per class. Based on the median number of expressions participating in solutions (final section of Table 7), the CMGE algorithm appears to consistently use the majority of the learner archive capacity to store solutions in order to decompose the problem. In terms of training times, both the RssGE and CMGE algorithms returned final results in median times of under 6 mins per initialization.

Tables 6 and 8 provide the same breakdown of results for the Thyroid data set. The distinction between CMGE and the RssGE baseline is even more apparent, with both Detection and False Positive rates comfortably improving on those established by the baseline. As per the Shuttle data set, the CMGE model is also very consistent, whereas the baseline model results in a wide range of Detection rates (down to 8%) and False Positive rates (up to 54%). Moreover, on classes 1 and 2, the CMGE model does so whilst utilizing more complex solutions than RssGE. This appears to indicate that the parsimony objective does not impede the accurate of classifiers, in effect CMGE is able to design classifiers that are suitably complex when warranted, that is without compromising test performance. The CMGE algorithm again appears to consume the large majority of the learner archive capacity to achieve problem decomposition. In terms of training time, both the RssGE and CMGE returned final results in median times of under one minute per initialization.

5 CONCLUSION

A novel algorithm for multi-class classification on large and unbalanced data sets is presented where the issues of scalability, problem decomposition and solution transparency are addressed simultaneously within a coevolutionary multi-objective GP framework. Scalability with respect to training set size is achieved through a competitive coevolution approach that incorporates multi-class archives and early stopping criteria. Problem decomposition is facilitated using two properties. Firstly, the use of a local wrapper function enables individuals to act as novelty detectors rather than discriminators. Secondly, when designing the EMO component of the model, care is taken to construct the ob-

Table 5: RssGE Results (UCI Statlog Shuttle)

Overall Accuracy (Train / Test)			
c	**Q1**	**Median**	**Q3**
-	0.738 / 0.739	0.79 / 0.787	0.83 / 0.833
Detection Rate (Train / Test)			
1	0.758 / 0.759	0.803 / 0.805	0.923 / 0.920
2	0.000 / 0.000	0.432 / 0.615	0.838 / 0.923
3	0.295 / 0.333	0.750 / 0.731	0.970 / 0.974
4	0.517 / 0.516	0.681 / 0.680	0.975 / 0.973
5	0.000 / 0.000	0.552 / 0.542	0.903 / 0.897
6	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
7	0.000 / 0.000	0.182 / 0.000	1.000 / 1.000
False Positive Rate (Train / Test)			
1	0.003 / 0.004	0.052 / 0.052	0.188 / 0.172
2	0.000 / 0.001	0.002 / 0.002	0.012 / 0.013
3	0.002 / 0.002	0.005 / 0.006	0.035 / 0.035
4	0.063 / 0.065	0.144 / 0.145	0.203 / 0.200
5	0.000 / 0.000	0.000 / 0.000	0.017 / 0.016
6	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
7	0.000 / 0.000	0.000 / 0.000	0.001 / 0.001
Expression Length			
1	17	29.5	55
2	13	22.5	42
3	17	27	42
4	18	40	72
5	13	23	42
6	8	12	23
7	26	42	66

Table 6: RssGE Results (UCI Thyroid)

Overall Accuracy (Train / Test)			
c	**Q1**	**Median**	**Q3**
-	0.405 / 0.417	0.761 / 0.732	0.86 / 0.835
Detection Rate (Train / Test)			
1	0.742 / 0.685	0.887 / 0.842	0.968 / 0.973
2	0.084 / 0.090	0.545 / 0.593	0.775 / 0.785
3	0.409 / 0.419	0.779 / 0.747	0.892 / 0.862
False Positive Rate (Train / Test)			
1	0.014 / 0.019	0.043 / 0.048	0.121 / 0.145
2	0.041 / 0.048	0.127 / 0.160	0.540 / 0.539
3	0.011 / 0.020	0.074 / 0.058	0.148 / 0.164
Expression Length			
1	10	30.5	59
2	10	19	32
3	13	41	83

jectives such that collaborative behavior with respect to other members of the Pareto front is encouraged.

Two large and unbalanced classification problems were investigated in comparison to an active learning approach with results indicating preference for the proposed system over both problems. Solutions were demonstrated to be of smaller size than the baseline RssGE on the Shuttle dataset, despite being a multi-individual model. The proposed system also returned training times on the order of a few minutes for the large data sets under consideration.

Future work will include the comparison of various parameterizations of the current algorithm on a wider range of large and unbalanced multi-class datasets. Moreover, we will investigate the appropriateness of a range of pruning algorithms for both learner and point archives.

6 Acknowledgments

The authors gratefullly acknowledge the support of PRECARN, NSERC Discovery, MITACS, and CFI New Opportunities programs; and industrial funding through the Telecom Applications Research Alliance (Canada), and SwissCom Innovations AG (Switzerland).

References

[1] F. H. Bennett, J. R. Koza, J. Shipman, and O. Stiffelman. Building a parallel computer system for $18,000 that performs a half peta-flop per

Table 7: CMGE Results (UCI Statlog Shuttle)

Overall Accuracy (Train / Test)			
c	**Q1**	**Median**	**Q3**
-	0.785 / 0.784	0.882 / 0.882	0.937 / 0.930
Detection Rate (Train / Test)			
1	0.756 / 0.751	0.868 / 0.866	0.952 / 0.948
2	0.622 / 0.462	0.797 / 0.654	0.919 / 0.846
3	0.462 / 0.436	0.693 / 0.705	0.833 / 0.872
4	0.828 / 0.833	0.922 / 0.923	0.966 / 0.969
5	0.992 / 0.990	0.993 / 0.993	0.994 / 0.994
6	0.833 / 0.750	1.000 / 1.000	1.000 / 1.000
7	0.909 / 0.500	1.000 / 1.000	1.000 / 1.000
False Positive Rate (Train / Test)			
1	0.015 / 0.016	0.048 / 0.046	0.108 / 0.105
2	0.000 / 0.000	0.002 / 0.002	0.007 / 0.006
3	0.005 / 0.006	0.023 / 0.023	0.091 / 0.090
4	0.004 / 0.003	0.011 / 0.012	0.022 / 0.023
5	0.000 / 0.000	0.000 / 0.000	0.000 / 0.001
6	0.000 / 0.000	0.001 / 0.001	0.006 / 0.006
7	0.001 / 0.001	0.013 / 0.014	0.101 / 0.103
Expression (Length / Number of Individuals)			
1	18 / 27	23 / 28	36 / 30
2	21.5 / 26	26 / 27.5	33.5 / 29
3	27 / 27	36.5 / 28.5	56 / 30
4	18 / 27	22 / 28	29 / 29
5	26 / 23	35 / 27	53 / 29
6	22 / 22	30.75 / 24.5	51.5 / 27
7	17 / 23	20.25 / 26	30.5 / 28

Table 8: CMGE Results (UCI Thyroid)

Overall Accuracy (Train / Test)			
c	**Q1**	**Median**	**Q3**
-	0.909 / 0.882	0.931 / 0.915	0.952 / 0.942
Detection Rate (Train / Test)			
1	0.774 / 0.740	0.817 / 0.808	0.871 / 0.863
2	0.618 / 0.599	0.801 / 0.802	0.885 / 0.898
3	0.911 / 0.884	0.943 / 0.928	0.975 / 0.963
False Positive Rate (Train / Test)			
1	0.005 / 0.007	0.007 / 0.008	0.012 / 0.014
2	0.021 / 0.029	0.047 / 0.059	0.073 / 0.089
3	0.028 / 0.020	0.076 / 0.098	0.208 / 0.256
Expression (Length / Number of Individuals)			
1	26 / 29	43.5 / 30	65 / 30
2	35 / 29	49.25 / 30	73 / 30
3	28 / 30	32.5 / 30	49.5 / 30

day. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-1999*, pages 1484–1490, 1999.

[2] S. L. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2:267–278, 1994.

[3] R. Curry and M. I. Heywood. Towards efficient training on large datasets for genetic programming. *17th Conference of the Canadian Society for Computational Studies of Intelligence, {LNAI}*, 3060:161–174, 2004.

[4] E. de Jong. The incremental pareto-coevolution archive. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-04*, pages 525–536, 2004.

[5] E. de Jong, R. A. Watson, and J. B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pages 11–18, 2001.

[6] S. G. Ficici and J. B. Pollack. Pareto optimality in coevolutionary learning. *Sixth European Conference on Artificial Life*, pages 316–325, 2001.

[7] G. Folino, C. Pizzuti, and G. Spezzano. Improving cooperative gp ensemble with clustering and pruning for pattern classification. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-06*, 1:791–798, 2006.

[8] C. Gathercole and P. Ross. Dynamic training subset selection for supervised learning in genetic programming. *Parallel Problem Solving from Nature III*, 866:312–321, 1994.

[9] R. Harper and A. Blair. A structure preserving crossover in grammatical evolution. *IEEE Congress on Evolutionary Computation*, 1:2537–2544, 2005.

[10] N. Japkowicz. Why question machine learning evaluation methods. *Evaluation Methods for Machine Learning (AAAI Workshop)*, pages 6–11, 2006.

[11] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal. Application of genetic programming for multicategory pattern classification. *IEEE Transactions on Evolutionary Computation*, 4(3):242–248, 2000.

[12] R. Kumar and P. Rockett. Improved sampling of the pareto-front in multiobjective genetic optimizations by steady-state evolution: A pareto converging genetic algorithm. *Evolutionary computation*, 10(3):283–314, 2002.

[13] M. Lemczyk and M. I. Heywood. Pareto-coevolutionary genetic programming classifier. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-06*, 1:945–946, 2006.

[14] T. Loveard and V. Ciesielski. Representing classification problems in genetic programming. *Proceedings of the Congress on Evolutionary Computation 2001*, 2:1070–1077, 2001.

[15] A. R. McIntyre and M. I. Heywood. MOGE: GP classification problem decomposition using multi-objective optimization. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-06*, 1:863–870, 2006.

[16] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. Uci machine learning repository.

[17] P. Nordin. *Advances in genetic programming*, pages 311–331. MIT Press, Cambridge, MA, 1994.

[18] M. O'Neill and C. Ryan. *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Springer, May 2003.

[19] D. Parrott, L. Xiaodong, and V. Ciesielski. Multiobjective techniques in genetic programming for evolving classifiers. *IEEE Congress on Evolutionary Computation*, 2:1141–1148, 2005.

[20] S. T. Sarasamma, Q. A. Zhu, and J. Huff. Hierarchical kohonen net for anomaly detection in network security. *IEEE Transactions on Systems, Man, and Cybernetics*, 35(2):302–312, Apr. 2005.

[21] D. Song, M. I. Heywood, and A. N. Zincir-Heywood. Training genetic programming on half a million patterns: An example from anomaly detection. *IEEE Transactions on Evolutionary Computation*, 9(3):225–239, 2005.

[22] 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*, 19:315–345, 2003.