Classification as Clustering: A Pareto Cooperative-Competitive GP Approach

Andrew R. McIntyre armcnty@cs.dal.ca
Faculty of Computer Science, Dalhousie University, Halifax, B3H 1W5, Canada

Malcolm I. Heywood mheywood@cs.dal.ca
Faculty of Computer Science, Dalhousie University, Halifax, B3H 1W5, Canada

Abstract

Intuitively population based algorithms such as genetic programming provide a natural environment for supporting solutions that learn to decompose the overall task between multiple individuals, or a team. This work presents a framework for evolving teams without recourse to prespecifying the number of cooperating individuals. To do so, each individual evolves a mapping to a distribution of outcomes that, following clustering, establishes the parameterization of a (Gaussian) local membership function. This gives individuals the opportunity to represent *subsets* of tasks, where the overall task is that of classification under the supervised learning domain. Thus, rather than each team member representing an entire class, individuals are free to identify unique subsets of the overall classification task. The framework is supported by techniques from evolutionary multiobjective optimization (EMO) and Pareto competitive coevolution. EMO establishes the basis for encouraging individuals to provide accurate yet nonoverlaping behaviors; whereas competitive coevolution provides the mechanism for scaling to potentially large unbalanced datasets. Benchmarking is performed against recent examples of nonlinear SVM classifiers over 12 UCI datasets with between 150 and 200,000 training instances. Solutions from the proposed coevolutionary multiobjective GP framework appear to provide a good balance between classification performance and model complexity, especially as the dataset instance count increases.

Keywords

Genetic programming, Pareto multi-objective optimization, coevolution, problem decomposition, classification.

1 Introduction

Classification represents a widely studied domain, both from the perspective of machine learning in general and genetic programming (GP) in particular. However, irrespective of the machine learning paradigm, solutions still most generally take the form of a single classifier per class.[1] In this work, we are interested in pursuing a framework that is able to partition the problem into subsets of classifiers when it is advantageous to do so. Under a GP setting, the general approach assumed in this paper is to coevolve a set of classifiers that learn to decompose the problem into class consistent subsets of training instances through multiple nonoverlapping classifier behaviors. General underlying motivations for pursuing the problem decomposition approach might include

[1]Hereafter, the terms learners, classifiers, and individuals will be used interchangeably.

scalability to more difficult domains, and support for individuals with a lower complexity than would be apparent under a single classifier per class model.

To date, several mechanisms have been proposed to provide direct support for problem decomposition under GP. Island models evolve *N* populations after which a heuristic is necessary for identifying which individual to utilize from each population (Imamura et al., 2003). Ensemble methods add an additional mechanism for ensuring that each of the *N* populations provide learners that respond to different (if not necessarily unique) subsets of the data set (Iba, 1999). The principle drawback from such an approach is that as many runs are necessary as there are individuals in an ensemble. This naturally does not scale as the number of classes or number of training instances increases.

Conversely, work on teaming from a single population assumes that each individual consists of a fixed (predefined) number of classifiers, the same for each individual. Each classifier making up an individual explicitly contributes to the class label, and a linear combination of the team member responses is generally assumed (Bremeier and Banzhaf, 2001). The error of each classifier contributing to a team is naturally uncorrelated. However, the fitness of individual classifiers appears to follow the weak learner methodology in which all models contribute to all decisions (Soule, 2000; Bremeier and Banzhaf, 2001), thus, although divide and conquer through teaming is supported, each class label is established through a complex interaction between multiple individuals. Naturally, the a priori specification of the number of classifiers represents a similar constraint as observed with the above ensemble approaches.

Recently an orthogonal evolution of teams (OET) framework was proposed (Thomason and Soule, 2007). In this case, the above island and team based approaches are combined. Two formulations are considered. In the first case, *N* independent populations are used to express *N* team members. At each generation, a new team is built by selecting an individual from each population. Fitness evaluation is performed at the team level, whereas selection is performed individualwise. The second model performed fitness evaluation at the level of individuals, whereas selection is performed teamwise. Needless to say, the number of individuals contributing to a team is still fixed a priori, and a multipopulation model is still explicitly required, with the corresponding computational overheads.

One final approach of relevance to the development of the work proposed here is the single population evolutionary ensemble method with a diversity maintenance scheme based on negative correlation (Liu et al., 2000). In this case, a single population is evolved with a fitness function that rewards both error minimization and negatively correlated behavior. When training is complete, the population is clustered using the response of each individual to each training instance. The clusters establish the population species, with individuals representing each cluster considered as representative of the overall population behavior, that is the ensemble/team. A post-training voting heuristic is then imposed in order to provide a single label for instances under test conditions (e.g., winner take all or majority voting). Later work introduced additional refinements, such as supporting the two competing objectives under an explicitly multi-objective context; thus avoiding the need to specify a particular weighting of classifier accuracy versus unique behavior (Chandra et al., 2006). However, natural drawbacks of such a scheme remain, including the cost of evaluation over multiple objectives (further compounding the computational overhead of fitness evaluation), and the post-training imposition of a voting heuristic that may or may not be appropriate.

In this work, we explicitly require a run of a single GP population to support problem decomposition without resorting to an a priori specification of the number of cooperating classifiers. Moreover, the same model should scale to multiclass domains providing classifiers for each class. To do so, a C class problem must be supported by at least C binary classifiers, or one per class (with problem decomposition implying more than C). This is more general that the $C - 1$ scheme typically employed and enables the resulting collection of classifiers to recognize when data not belonging to the training distribution appear (as in the case of an outlier).

In establishing the proposed model we emphasize the following factors:

1. Not all training instances are created equal: At any point during training, only a subset of instances are necessary to distinguish between the performance of classifier(s). Several machine learning paradigms support this hypothesis, including active learning (Cohn et al., 1994), query based learning (Kothari and Jain, 2003), and competitive coevolution (de Jong and Pollack, 2004). The common goal of such methods is to identify a minimal subset of training instances to support the development of learners. Moreover, the training subset identification problem might well be a moving target. Thus, the development of stronger learner(s) provokes the need to identify a more demanding training subset. The potential benefit to GP of pursuing such an approach naturally lies in the decoupling of fitness evaluation from the size of the training partition.

2. Explicitly cooperative behavior: Problem decomposition will be supported through establishing criteria that measure the degree of overlap between the population and the current subset of best learners. Fitness should therefore reward both accurate and diverse behaviors; thus, objectives rewarding accuracy are combined with an explicit requirement to minimize overlap in the subset of instances on which accuracy is optimized.

3. Local membership function (LMF): Program execution under a canonical GP classifier maps training instances from a domain specific attribute vector to a 1-dimensional scalar, or *gpOut* (Figure 1). Most models of GP classification then apply a global membership function (GMF)—such as a sigmoid—to reexpress *gpOut* as a class label by partitioning the *gpOut* number line into one of two regions representing in-class and out-of-class data, respectively (Figure 1). This assumption has the effect of casting the classification problem into that of finding a mapping (i.e., program) that maximizes the discrimination between training instances associated with either side of the GMF. This also means that the model has no fail-safe mode of operation. Specifically, by fail-safe, we imply that the in-class distribution on *gpOut* be delimited from both the left- and right-hand side; thus out-of-class instances encountered under post-training conditions and mapped to the left of a currently established in-class distribution on *gpOut* are not strongly associated with in-class behavior.[2] This is seen as a requirement for a local membership function (LMF) as opposed to a GMF. More generally, an LMF is used to support the evolution of subsets of instances with class consistent behavior. The overall learning problem now takes the form of cooperatively evolving

[2]Explicit support for this fail-safe mode of operation is often denoted novelty detection by the one-class learning literature where training data are limited to in-class data alone (Markou and Singh, 2003).

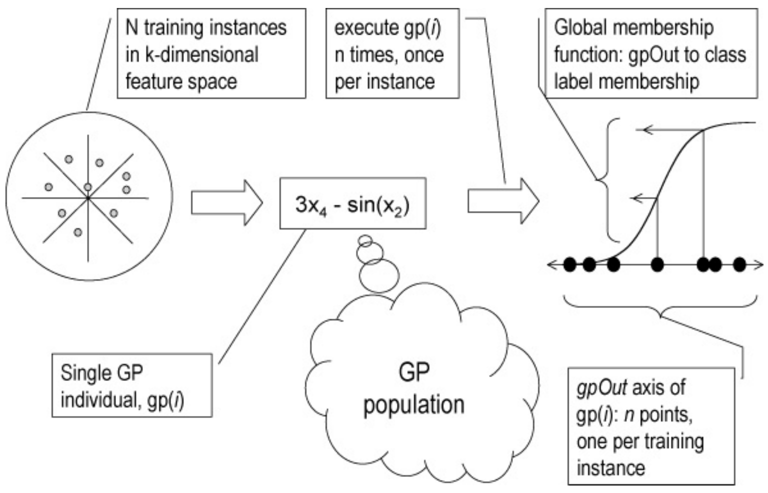
N training instances in k-dimensional feature space
execute gp(*i*) n times, once per instance
Global membership function: gpOut to class label membership
3x4 - sin(x2)
Single GP individual, gp(*i*)
GP population
gpOut axis of gp(*i*): *n* points, one per training instance

Figure 1: Basic mapping process of canonical GP classifier model. Each individual maps instances described in terms of a multidimensional feature space to a 1-dimensional number line *gpOut*. A global membership function is then used to associate points on *gpOut* with in-class and out-of-class partitions. Later comparison against the class labels establishes model error.

individuals to associate nonoverlapping class consistent instance subsets to their respective LMF. Such a methodology represents the converse of that generally pursued by ensemble methods in which a GMF is more appropriate. Specifically, ensemble methods are designed to *maximize the overlap* between classifier behaviors and let the out-of-class instances mapped to the in-class side of the *gpOut* number line be corrected through the post-training voting algorithm.

One natural consequence of the above factors is that a common methodology is required to establish the concept of the best subset of candidate solutions at any given training epoch. Moreover, such a concept should provide mutual support for multi-objective optimization, cooperative coevolution, and training instance subset identification. To do so, Pareto methodologies, as already independently under development for both competitive coevolution and evolutionary multicriteria optimization, will be assumed. Such methodologies, however, have for the most part been developed under the auspices of genetic algorithms (GAs). Combining the two under a cohesive framework for GP classification represents a new development undertaken by this work. Finally, the resulting model is independent of the GP representation employed. Thus, while grammatical evolution (GE) provides the underlying representation and variation operators used here, alternative GP representations are equally appropriate, for example, linear or tree representations. Hereafter, the proposed model will be referred to as coevolutionary multi-objective GP or CMGP.

The CMGP model will be developed by first providing a short review of supporting developments from competitive coevolution and evolutionary multicriteria optimization (Section 2). In particular, we make the case for a combined approach to competitive coevolution, and establish the rationale for favoring one EMO methodology over another. Related work on coevolutionary GP classification is also identified in Section 2.

Section 3 develops the CMGP algorithm, with Section 4 establishing the evaluation methodology adopted for performance evaluation. Section 5 reports results of a benchmark study comparing CMGP against a scalable canonical GP classifier and two recent formulations of support vector machine (SVM) classifiers on 12 data sets. Conclusions and future work are discussed in Section 6.

2 Supporting Techniques

The framework on which CMGP is based assumes a Pareto coevolutionary methodology in order to provide a coherent basis for building the remainder of the model. In the following we provide a short review of this material and establish criteria for the formulation pursued under CMGP. To this end, we divide the Pareto frameworks into two cases, those associated with evolutionary multicriteria optimization (EMO) (Section 2.1), and those associated with competitive coevolution (Section 2.2). Finally, having established this background, methods using EMO or competitive coevolution under a GP classifier context are then reviewed (Section 2.3).

2.1 Evolutionary Multicriteria Optimization

An individual A is said to Pareto dominate individual B over the set of objectives Θ if A is no worse than B on any objective ($\theta \in \Theta$) and is strictly better than B on at least one. Formally for the case of minimization:

$$A \prec B \Leftrightarrow (\forall \theta \in \Theta) \quad A_\theta \leq B_\theta \wedge (\exists \theta \in \Theta : A_\theta < B_\theta). \tag{1}$$

A and B are said to be incomparable when A does not Pareto dominate B and vice versa. An individual is nondominated when it is not Pareto dominated by any others. The set of (incomparable) nondominated individuals constitute the Pareto set or front.

The definition of Pareto dominance can be generalized using the notion of ϵ-dominance, which effectively provides a means to discretize the objective space and constrain the number of elements in the solution set to be finite. The significance of this lies in the guarantee of diversity among solution vectors and practical size of the solution set, known as the ϵ-approximate Pareto set (Laumanns et al., 2002). Multiplicative ϵ-dominance (for minimization $A \prec_\epsilon B$) is defined as:

$$A \prec_\epsilon B \Leftrightarrow (\forall \theta \in \Theta) \quad A_\theta \cdot (1 - \epsilon) \leq B_\theta \mid 0 \leq \epsilon < 1. \tag{2}$$

Moreover, an additive variant can be employed when specification of constant values is desired across objective dimensions,

$$A \prec_\epsilon B \Leftrightarrow (\forall \theta \in \Theta) \quad A_\theta - \epsilon_\theta \leq B_\theta \mid 0 \leq \epsilon < 1. \tag{3}$$

The additive ϵ-dominance definition provides uniform discrete resolution over the objective space as opposed to the multiplicative variant, where larger hyper-rectangular cells are defined as dimension magnitudes increase.

A Pareto ranking mechanism, based on Pareto dominance, provides the basis for the ensuing fitness assignment scheme and guides the evolutionary algorithm by favoring nondominated solutions in the selection and reproduction processes. Critically, the Pareto method provides the mechanism by which the performance vector (multiple objectives) can be summarized in terms of a scalar ranking. Typically, the mapping from rank to fitness is linear or exponential but varies by algorithm. Moreover, in order to avoid dense regions in the objective space consuming the available search resource (potentially resulting in a locally optimal solution), a diversity preservation mechanism may be included where this typically takes the form of a preselection, crowding, or sharing heuristic.

When a Pareto ranking mechanism is combined with an explicit archiving structure (having acceptance criteria based on nondominance) or an elitist replacement scheme, a diverse collection of nondominated solutions result. Generally, specific algorithms are distinguished in terms of the approach taken to their ranking mechanisms, maintenance of diversity, and/or prevention of solution loss. See, for example, the strength Pareto evolutionary algorithm (SPEA/SPEA2; Zitzler and Thiele, 1999), the nondominated sorting genetic algorithm (NSGA-II; Deb et al., 2002), the multi-objective genetic algorithm (MOGA; Fonseca and Fleming, 1993), and the Pareto converging genetic algorithm (PCGA; Kumar and Rockett, 2002). However, of the above schemes, only the PCGA model explicitly provides the following properties of particular importance to the GP classification domain.

- **Diversity Maintenance.** The vast majority of EMO methods assume the utility of a distance based diversity maintenance strategy. Under a GA context, this is entirely plausible, as individuals generally take the form of a point in a multidimensional feature space, thus similarity is readily established through the application of an appropriate distance norm such as the Euclidean distance metric. Under a GP domain, establishing genotypic similarity is much more difficult, and the derivation of appropriate radii parameters is unintuitive. Conversely, the PCGA paradigm, although developed under the GA context, avoids a metric based diversity mechanism by assuming a steady state tournament, and utilizes a replacement strategy relative to the Pareto ranking of the population to maintain diversity.

- **Pareto Stopping.** PCGA provides a simple scheme for determining early convergence of the population based on the content of the Pareto front remaining constant for a fixed number of tournaments. As such, we will later utilize this property for introducing classwise early stopping, with dynamic reassignment of members from the learner population to other classes as need dictates.

2.2 Pareto Competitive Coevolution

Competitive coevolution came to the fore with the Host-Parasite model of Hillis (1990) in which a population of learners (candidate solutions) are rewarded for fitness expressed relative to a population of test points. That is, the test point population samples the set of available states associated with the wider problem domain (i.e., a subset of training instances) and is rewarded for locating states that beat the learners. Later work recognized that it was necessary to discount the reward provided to the point population in order to avoid disengagement between the two populations (Cartlidge and Bullock, 2002). Moreover, it also became increasingly apparent that the competitive coevolutionary model could lead to several additional undesirable properties such as relativism, the red queen effect, and intransitivity (Watson and Pollack, 2001). Conversely, assuming a Pareto competitive coevolutionary model provides a formal basis for preferring the behavior of one learner over another, thus borrowing from the EMO methodology properties such as Pareto ranking (Noble and Watson, 2001). At the same time, it also became apparent that the point population should be rewarded for distinguishing between the performance of learners in the coevolving population as opposed to beating them outright (Ficici and Pollack, 2001). These two observations provided the necessary mechanisms to establish a family of Pareto competitive coevolutionary algorithms based on Pareto archiving (de Jong, 2007). The original objective of such a Pareto archiving model was to establish the grounds for guaranteeing monotonic progress in the behavior of the nondominated individuals associated with test points and learners. The trade-off associated with such an objective is that the archives may grow without bound. Moreover, all of the aforementioned models have been proposed and tested within the context of the GAs, thus potentially making assumptions that do not necessarily carry over directly to GP or the classification domain.

Under the requirements for a GP classification domain, we require fixed size archives, thus implying the utility of appropriate learner/point replacement heuristics, and an appropriate scheme for building the test point population. Specifically, each individual in the test point population represents an index to an instance in the larger training data set, thus there is no structure that could guide the application of variation operators such as crossover. For example, a test point individual representing an index to training instance i that supports the distinction of a learner in the learner archive does not imply that the neighboring training instance indexes $i \pm 1$ are any more or less relevant.

As a consequence, a method is now necessary for generating test point population content at each iteration of the competition. To this end we make use of recent research with decision tree induction under a limited sampling context (Weiss and Provost, 2003). The basic goal was to determine the sampling bias necessary to build decision trees from a subset of the total training dataset such that the ensuing classifier was robust under datasets with unbalanced class distributions. Sampling training subsets with uniform probability resulted in decision trees that were observed to optimize post-training performance under the accuracy metric. Including an additional constraint to enforce equal representation of both in- and out-of-class instances sampled to build the training subset was observed to result in post-training performance evaluation optimizing the area under the curve (AUC) metric. With this observation in mind, the latter class balanced uniform sampling heuristic will be used in this work to establish content for the point population at each generation.

2.3 Related GP Models of Classification

Within the context of GP, it is the cooperative (multi-objective) model that has seen most widespread utility to date. In particular, the multi-objective model was previously used to encourage the evolution of parsimonious yet accurate solutions (Badran and Rockett, 2007). The multi-objective framework also provides a very natural model for evolving multiclass solutions from a single population under classification problems (Parrott et al., 2005; Zhang and Rockett, 2006). However, there is no problem decomposition as such, with the user intervening post-training to identify a single specific solution per class.

Conversely, the Pareto competitive model of coevolution (i.e., training subset selection) has only recently begun to appear under a GP classifier context. Two works report problems with applying generic Pareto competitive coevolution to GP classification. Yo and de Jong (2007) apply a competitive coevolutionary GP model to the (balanced) binary six-parity classification problem and observe that the solutions do not perform as well as the equivalent canonical GP classifier; whereas Lichodzijewski and Heywood (2007) report that a class balanced uniform sampling heuristic for point population generation performed better than a uniform point population generation model of competitive coevolution, albeit in the case of one of three unbalanced real-world datasets. Conversely, the PGPC model (Lemczyk and Heywood, 2007) was never worse than the equivalent canonical GP classifier on two large unbalanced real-world benchmark datasets. The essential difference was the utility of a class balance enforcing uniform sampling heuristic when determining the context of the point population. Without this, the PGPC model would not function nearly as well (Lemczyk, 2006).

Finally, in the case of alternative membership function design, Zhang and Smart (2006) also consider a model in which an LMF is employed. However, *each* individual is forced to associate a single Gaussian with each *class*; thus *all* training instances from the same class are used to parameterize a single LMF per class per individual. Fitness is then expressed in terms of either minimizing the area shared between Gaussian distributions (representing different classes) at the same individual, or maximizing the variance normalized distance between distributions. They also propose to use the population as a whole for classification by way of a post-training voting heuristic. However, there is no explicit mechanism for diversity maintenance or coevolution of behaviors such that different individuals concentrate on different aspects of the problem.

3 CMGP Methodology

The proposed CMGP methodology is composed of two basic components, the evolutionary multi-objective model that establishes the basis for parental selection (McIntyre and Heywood, 2006), and a competitive coevolutionary component that establishes the framework for training subset selection (Lemczyk and Heywood, 2007). The interaction of the two provides the basis for a unique model of cooperative evolution (Figure 2) in which a team of individuals learn to collectively decompose the representation of a class into a series of class consistent subsets or problem decomposition with non-overlapping behaviors. In order to introduce such a model, however, we first review the generic properties of a canonical model of GP classification (Section 3.1) and in so doing establish requirements for evolving a local membership function at each individual. Section 3.2 summarizes the resulting CMGP algorithm.

3.1 Canonical GP Classification

GP program execution maps an instance from a typically multidimensional feature domain to a one dimensional number line hereafter referred to as *gpOut*. An example of this process is provided in Figure 1, where the mapping process is repeated to map n training instances to n points on *gpOut*. Such a mapping alone, however, conveys no class label information.

Following the mapping of instances to *gpOut*, Canonical GP employs a global membership function to impart class labels. Such an approach effectively casts all points on the *gpOut* number line that are greater than (less than) zero as in-class (out-of-class) instances (Figure 1), and counts the number of misclassifications by way of a cost function. Natural variations of this theme include the use of a sum square cost (fitness) function under a sigmoid style membership function. Such a process assumes that partitioning the class labels about the origin of the *gpOut* number line is appropriate

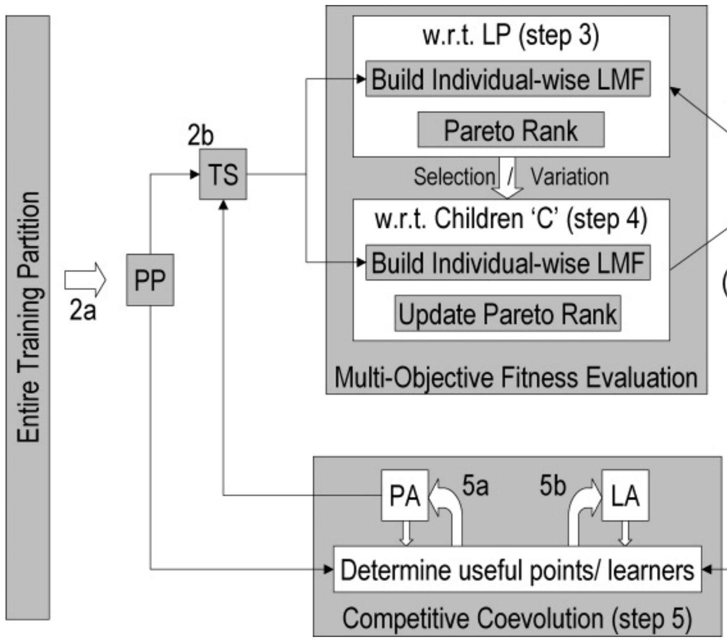
Entire Training Partition
2a
PP
2b
TS
w.r.t. LP (step 3)
Build Individual-wise LMF
Pareto Rank
Selection / Variation
w.r.t. Children 'C' (step 4)
Build Individual-wise LMF
Update Pareto Rank
Multi-Objective Fitness Evaluation
LP(t)
LP
LP (t+1)
PA
5a
5b
LA
Determine useful points/ learners
Competitive Coevolution (step 5)

Figure 2: Overview to CMGP framework. PP is the point population or subset of training instances sampled from the training set under a class balanced sampling heuristic; LP is the current population of learners. PA and LA denote the corresponding point and learner archives identified during competitive coevolution. The steps indicated in parentheses match the high-level pseudocode listing in Figure 3.

for the problem domain in question. However, such an assumption is not likely to be true in practice (Loveard and Ciesielski, 2001; Zhang and Smart, 2006) and leads to a discriminator mode of classification (Markou and Singh, 2003). Under the proposed CMGP model, label assignment is realized by means of a local membership function (LMF) that explicitly assigns class to a specific subregion of *gpOut*; thus fitness should reward the mapping of class consistent instances to the same local neighborhood on *gpOut*. Moreover, by adopting the LMF approach, we are also able to achieve more predictable behavior in the operation of classifiers on unseen (test) instances.

3.2 CMGP Algorithm

From a coevolutionary perspective, there are essentially three basic processes that the proposed competitive multi-objective GP (CMGP) framework uses to address the dual problems of training subset identification and problem decomposition (Figure 2).

1. Sampling of the training partition through the balanced stochastic sampling heuristic, thus establishing the content of the point population (PP).
2. Multi-objective fitness evaluation relative to the learner population (LP), and the associated application of selection/variation operators for diversity maintenance. Diversity appears as a specific objective as measured relative to the learner archive (LA) in order to guarantee uniqueness of training instances solved.

```
                        1. Initialize Learner Population (LP) and data structures
                        2. WHILE ! (Stop Criteria) {
                             a) Point Population (PP) := random (balanced) sample of data
                             b) Training Set (TS) := PP combined with Point Archives (PA)
                             3. FOR i := 1 to sizeof(LP) {
                      /           a) Rank LP
                                  b) Apply operators to Produce Children (C)
                                  4. FOR j := 1 to sizeof(C) {
                                       a) Decode C[j]
Cooperative                            b) Map TS to number line (gpOut) of C[j]
Coevolution through  <                 c) Cluster gpOut of C[j] (find μ,σ )
EMO model                              d) Parameterize Local Membership Function (LMF) of C[j]
                                       e) Evaluate C[j] with respect to:
                                            SSE, Overlap wrt. Learner Archive (LA), Soln. Size, Count
                                       f) Rank C[j] with respect to LP and assign fitness
                                       g) Replacement (insert C[j] into LP)
                      \           }
                             }
                      /      5. Archive PP, LP members based on outcomes (according to IPCA) {
Competitive          <            a) Points in PP enter PA if they provide a distinction
coevolution model                 b) Learners in LP enter LA if they are non-dominated wrt. LA
                      \      }
                             6. Evaluate Stop Criteria (method of Rank Histograms)
                        }
                        7. Deploy final solution (contents of Learner Archives)
```

Figure 3: High-level pseudocode listing of the CMGP algorithm. Variation operators produce C children that are evaluated with respect to the current point population/archive content under a multi-objective cost function.

3. Team identification as an independent process such that coverage of training instances is maximized, that is, competitive coevolutionary identification of learner archive and point archive (PA) content.

Thus, following initialization of PP and LP evaluation of individuals from LP takes place against the content of the training subsample—or $TS = PP \cup PA$—under a multi-objective fitness function. Conversely, the initially empty content of PAs and LAs plays an increasingly important role in (i) retaining training instances explicitly identifying learners as nondominated (and therefore establishing team content), and (ii) establishing the basis for the overlap objective assessed during fitness evaluation. PA and LA content is updated after each epoch of fitness evaluation. The next epoch then begins with the content of the PP (and hence TS) reestablished. Embedded within this process is support for evolution of the local membership function (LMF) and early stopping criteria. With this general architecture in mind, the following detailed points are made relative to the pseudocode listing provided in Figure 3.

3.2.1 Main Loop

The while loop of step 2 encloses the main sections of the algorithm, ensuring that the above three processes of the CMGP architecture are repeated until stopping conditions are met as evaluated at the end of each epoch, or step 6. Steps 2a and 2b set up the training set at each iteration, ensuring a balanced view of data, thus facilitating robustness against problems having unbalanced class distributions (see Section 2.2 for the wider motivation behind this heuristic).

Step 3 establishes the multi-objective fitness function and therefore forms the basis for diversity maintenance in the learner population. Selection takes the form of a steady state tournament (step 3b), after which the transcription of individuals from codon to program is performed (step 4a) and the current subset of instances are mapped to the *gpOut* number line of each individual (step 4b and Figure 1). We now require a mechanism for identifying the neighborhood of the local membership function (LMF) without resorting to inappropriate or arbitrary predefinitions of regions within the

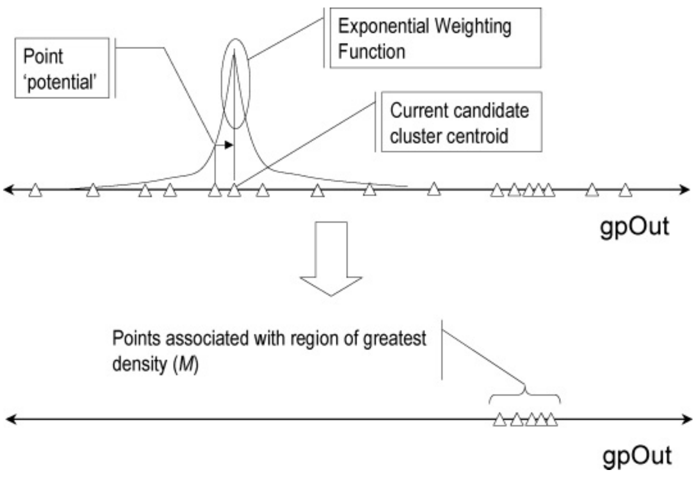
Point 'potential'
Exponential Weighting Function
Current candidate cluster centroid
gpOut
Points associated with region of greatest density (M)
gpOut

Figure 4: Parameters of Gaussian LMF are established through clustering the *gpOut* distribution without reference to label information using the PF clustering algorithm. Nearest neighbor allocation of points to cluster centers establishes the distribution of points assigned to the most dense cluster, M. Parameterization of the Gaussian LMF then follows directly from the variance (σ) and mean (μ) of point subset M.

gpOut number line. To do so, we assume that the neighborhoods of most relevance are those that have the highest density (Figure 4). This requirement is naturally relative to the distribution of points on the *gpOut* number line, and is independent of class label, the latter property being enforced later by way of the multi-objective fitness function.

3.2.2 Evolving the Local Membership Function

At this stage (steps 4c and 4d), we have the basic requirement for a clustering algorithm as applied to the subset of points identified by the competitive coevolutionary model and mapped to each individual's *gpOut* number line (Figure 4). The clustering algorithm returns the location of the midpoint associated with the most dense set of points, μ, and training instances associated with this cluster, M, care of a nearest neighbor allocation of points to cluster centers. Hence, any generic clustering algorithm that does not assume an a priori definition for the number of clusters sought would be appropriate. In this work, the potential function (PF) method is assumed (Chiu, 1994). The PF model iterates over all points as candidate cluster centers, measuring cluster quality in terms of an exponential weighting function. Candidate centers with more neighbors in the vicinity attain higher potential.

3.2.3 Fitness Evaluation, Diversity Maintenance, and Early Stopping

With the properties for the local membership function of the GP mapping determined (per individual), we now introduce training instance labels and apply the multi-objective fitness criterion (steps 4e and 4f) or EMO under the formulation of Kumar and Rockett (2002). The objectives are designed to encourage (1) least ambiguity in cluster membership (sum square error minimization), (2) nonoverlapping behavior of the training instances mapped to different individuals, (3) maximization of the

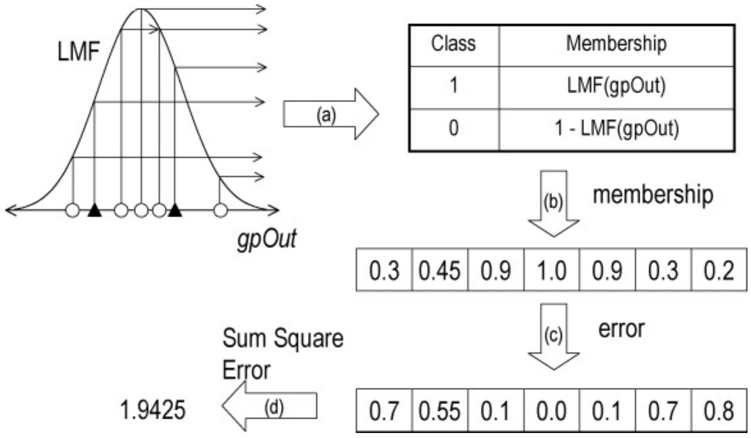

Figure 5: Class labels introduced. The concept of error may now be introduced relative to label and degree of LMF membership (a). Let *M* denote the unique subset of instances over which fitness is evaluated and a circle (triangle) represent instances with an in-class (out-of-class) label. The resulting error (b) and (c) are then weighted as per the sum square error cost function (d).

number of in-class instances mapped to an individual, and (4) simplicity of the GP mapping. The specific case of error estimation is summarized in Figure 5. Each GP individual adopts the class label associated with the training instance with maximum membership, thus the concept of in-class membership is evolved as opposed to being preassigned. Objective 2 encourages diversity over the association of training subset instances to individuals, or problem decomposition. An important trick, however, is to estimate this objective relative to the content of the classwise learner archives defined by competitive coevolution (step 5). Alternative formulations such as estimating overlap relative to the learner population as a whole might result in all individuals contributing overlap, thus confusing the process of credit assignment. Objective 3 provides additional pressure to avoid potentially degenerate solutions that optimize the error minimization objective through minimizing the number of training instances they account for.

The EMO model also introduces the concept of a rank histogram (Kumar and Rockett, 2002), which essentially summarizes the content of the population (in objective space) in terms of the Pareto ranks so that content can be readily compared between training epochs (Figure 6). Specifically, after a minimum number of generations, the content of the best Pareto rank is compared to that at the previous training epoch. If the membership remains, then unchanged convergence (or a fitness plateau) has occurred. When estimated with respect to each class, implying separate learner and point archives for each class, this provides the basis for the evaluation of early stopping (step 6).

3.2.4 Competitive Coevolution

The above inner loop (steps 3 and 4) is performed relative to the content of the PP (step 2b). A competitive coevolutionary mechanism is then used to identify the most discriminatory test points (step 5a) and nondominated learners (step 5b), again using the concept of Pareto dominance. The competitive model thus plays the dual role of

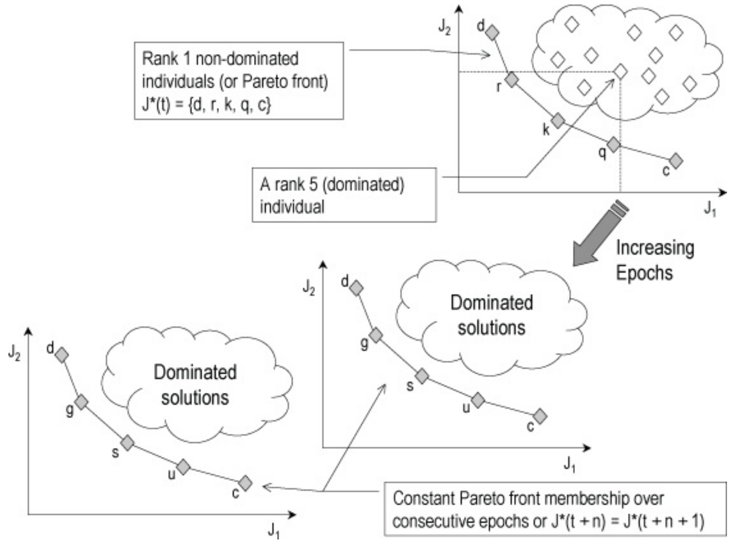
Rank 1 non-dominated individuals (or Pareto front) J*(t) = {d, r, k, q, c}
A rank 5 (dominated) individual
Increasing Epochs
Dominated solutions
Dominated solutions
Constant Pareto front membership over consecutive epochs or J*(t + n) = J*(t + n + 1)
J₁
J₂
d
r
k
q
c
g
s
u

Figure 6: EMO Pareto ranking and early stopping criteria. All solutions on the Pareto front are nondominated (Section 2.2) and retain the same ranking for the purpose of fitness evaluation. When the content of the rank 1 (nondominated) individuals remains unchanged for a minimum number of epochs, convergence is considered to have been reached. The process is performed classwise, with the dominance relation estimated across individuals that have the same class. Note that when the stopping criteria are satisfied, the corresponding classwise team is identified as the LA content, not the EMO archive content.

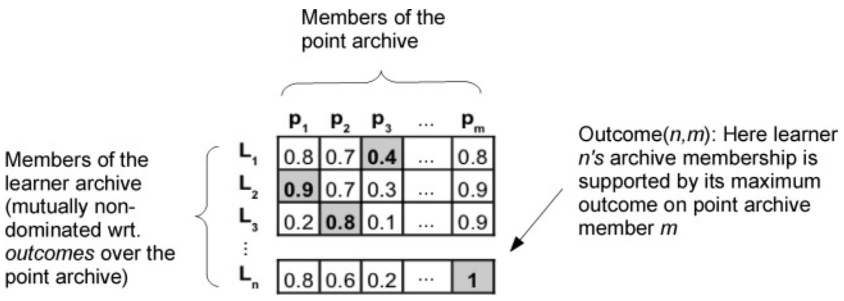
Members of the point archive
Members of the learner archive (mutually non-dominated wrt. outcomes over the point archive)
Outcome(n,m): Here learner n's archive membership is supported by its maximum outcome on point archive member m
p₁ p₂ p₃ ... pₘ
L₁ 0.8 0.7 0.4 ... 0.8
L₂ 0.9 0.7 0.3 ... 0.9
L₃ 0.2 0.8 0.1 ... 0.9
Lₙ 0.8 0.6 0.2 ... 1

Figure 7: Competitive coevolutionary Pareto ranking of classifiers versus points. Nondominated points populate the corresponding classwise learner archives whereas the points (training instances) supporting such distinctions are retained in the corresponding point archive. Archive replacement heuristics are necessary to support finite archive sizes (Section 3.2.5).

establishing instance subsets over which evaluation takes place and an archival role, the latter acting as a memory for the cooperative model against which the above EMO overlap objective is estimated. The archive entry criteria are evaluated in terms of GP classification outcomes (see Figure 7), which are directly related to the LMF definition and its associated performance on the training set. In addition, archive insertion heuristics are necessary for the purpose of enforcing finite archive sizes (Section 3.2.5).

Deployment of the classifier (step 7) takes the form of copying the contents of the learner archives (teams for each class) and assignment of weights to each on the basis of the training data (Sarasamma et al., 2005; McIntyre and Heywood, 2007). A winner-take-all policy with respect to LMF class membership determines the assignment of class labels among the team individuals.

3.2.5 Archive Maintenance Heuristic

In order to maintain an upper bound on the size of the archives, pruning may be necessary prior to insertion of the new learner or point member. A maximum size is therefore imposed on the learner and point archives—LA_{size} and PA_{size} respectively—in order to enforce computational and resource efficiency during training. Efficiency is therefore achieved at the potential expense of accuracy of learner evaluation.

In the case of point archiving, points are added to the archive so long as the archive has not reached 50% in-class or 50% out-of-class capacity ($\text{PA}_{\text{size}} \div 2$), for an in-class or out-of-class point, respectively (Lemczyk and Heywood, 2007). When the capacity limit has been reached, the point (of the same class) having the minimum Euclidean distance to the incoming point is removed (pruned) in favor of the new point. The distance is calculated over the attributes (feature space) associated with each point. While a point is lost from the archive through this process, we assume that favoring newer distinctions over older distinctions is more beneficial from the perspective of maintaining evolutionary pressure and diversity.

When a learner archive has reached capacity (LA_{size}), a current archive member must be chosen for replacement. Two factors are considered, the overall error of an individual and the false positive rate of an individual. Specifically, new learners are added to the archive until the archive limit is reached, after which the current learner with the lowest (sum of) *outcome′* values is replaced; with *outcome′* only being evaluated over the out-of-class points ($gpOut \notin \{inclass\}$ condition, Figure 7). As such, individuals are identified for replacement with the worse false positive behavior and ties broken using the (sum of) *outcome* values over all classes.

In summary, setting the archive limits too low may cause cycles of forgetting in the learning process. Conversely, the potential benefit of maintaining a finite archive lies in maintaining an explicit computational limit on the evaluation complexity, shown to be of the order $\mathcal{O}((archive_{size})^4)$ in the case of the IPCA algorithm on which the competitive model of CMGP is based (McIntyre, 2007). This makes the CMGP model increasingly effective as the size of the overall training set increases (McIntyre and Heywood, 2008b).

4 Benchmarking Methodology

The benchmarking study of Section 5 provides two comparative studies of CMGP performance: CMGP versus a scalable canonical GP classifier under a common parameterization; and CMGP versus two recent examples from the SVM paradigm of classification. With this in mind we first introduce the shared GP representation/search operators (Section 4.1). The benchmarking datasets are then introduced (Section 4.2) and the classifier performance metric is established (Section 4.3).

4.1 Common GP Representation and Variation Operators

The CMGP framework does not place constraints on the choice of representation or variation operators. However, from the perspective of the ensuing benchmarking study, we

Table 1: GE parameters. XO and M denote standard one point crossover and mutation; CXO and CM denote the respective context aware variants. Tests for mutation operators are applied individualwise, with selection of (legitimate) codons performed with uniform probability.

$\text{Prog}_{\text{limit}}$	$\text{Codon}_{\text{limit}}$	EPOCH(limit)	P(XO)	P(CXO)	P(M)	P(CM)
4096	256	500	0.5	0.9	0.01	0.9

Table 2: Population and archive parameters.

LP_{size}	LA_{size}	PP_{size}	PA_{size}
50	30	30	30

assume a common choice of representation and variation operators across GP models. Any performance variation may then be attributed to the differing components of the GP models as opposed to differences in representation/variation operators. To this end, grammatical evaluation (GE) was assumed, thus the representation takes the form of a context free grammar (O'Neill and Ryan, 2003). Variation operators are applied to the codon sequence with the grammar guiding the process of translation from an integer codon sequence to a program. Operators take the form of crossover and mutation, and come in both context aware and noncontext aware forms (Harper and Blair, 2005). With respect to one point crossover and codonwise mutation, the context aware variants are defined as follows.

Context Aware Mutation (CM). Following the process of codon to program transcription, the codons that result in a terminal grammar (i.e., an instruction) are marked. Context aware mutation is limited to the codons at the marked locations, and subject to the set of integer modifications that still result in a terminal symbol at that location.

Context Aware Crossover (CXO). Restricts crossover to codon boundaries where the codon of the second parent expands the same nonterminal type as the codon selected in the first parent. The expectation of context preservation is therefore based on each parent receiving material that is used to interpret the same part of the grammar as the donating parent. Such a crossover operator was demonstrated to perform significantly better than the single point crossover typically employed by GE (Harper and Blair, 2005).

Parameterization of GE is common across all GP classifiers summarized in terms of the generic GE population and variation operator frequencies (Table 1), population and archive parameters (Table 2), and context free grammar (Table 3). The variation operator probabilities are specified for both the standard and context aware variants: P(XO), P(M); and P(CXO), P(CM), respectively. In the case of the generic GE parameterization Table 1, $\text{Prog}_{\text{limit}}$ denotes program length, post codon translation, whereas $\text{Codon}_{\text{limit}}$ defines the maximum number of codons an individual may take.

4.2 Dataset Selection

A set of 12 datasets are sourced from two repositories, as shown in Table 4: 10 appear in the University of California at Irvine's (UCI) Machine Learning Repository (Newman et al., 1998), with two (CENS and KD99) taken from the UCI Knowledge Discovery

Table 3: Common GE context free grammar.

Rule ID	Rule	Option	Option ID	# Options
0	$< start >$	$< exp >$	0	1
1	$< exp >$	$< exp >$	0	5
		$\mid (< exp >)$	1	
		$\mid < exp >< op >< exp >$	2	
		$\mid < preop >< exp >$	3	
		$\mid < var >$	4	
2	$< preop >$	sin	0	5
		| cos	1	
		$\mid sqrt$	2	
		| log	3	
		| exp	4	
3	$< op >$	$\mid +$	0	4
		$\mid -$	1	
		$\mid \times$	2	
		$\mid \div$	3	
4	$< var >$	x_0	0	n
		$\mid \vdots$	$\vdots$	
		$\mid x_{n\text{-}1}$	$n - 1$	

Table 4: Dataset characterization. $|T|$ is the total number of instances in the dataset; Train (test) indicate the instance count for training (test) sets in the case of the dataset having specific partitions, or whether a stratified 10fold is adopted if there is no such partition; F denotes the number of features; and C the number of classes.

Dataset	$\|T\|$	Train	Test	F	C	% Distribution
BOST	506	10fold		13	3	≈ balanced
BUPA	341	10fold		6	2	(42):(58)
CENS	295,173	196,294	98,879	41	2	(94):(6)
CONT	1,425	10fold		9	3	(43):(22):(35)
IMAG	2,310	210	2,086	19	7	≈ balanced
IRIS	147	10fold		4	3	≈ balanced
KD99	222,871	145,584	77,287	41	5	(60.3):(0.7):(37.5):(1.5):(0.04)
PIMA	768	10fold		8	2	65:35
SHUT	58,000	43,500	14,500	9	7	(78):(0.09):(0.3):(15.5):(5.65): (0.01):(0.03)
THYD	7,129	3,709	3,420	21	3	(2):(5):(93)
WINE	178	10fold		13	3	(33):(40):(27)
WISC	675	10fold		10	2	(65):(35)

in Databases (KDD) archive (Hettich and Bay, 1999). In the last column of Table 4 we define the classwise distribution of instances associated with the training partition. The ensuing collection of datasets range from the small, well behaved, but widely used cases (IRIS, WINE, WISC), to the difficult but small and widely used (PIMA, THYD, BUPA), to the difficult, large and unbalanced but less frequently used cases (CENS, KD99, and SHUT).

All datasets were preprocessed by first taking the union of all partitions and mapping nominal attributes to integer values. Any repeating, incomplete, or inconsistent records were then removed from the set and the train/test partitions were restored. Problems that did not predefine training and test partitions were evaluated under 10-fold cross validation (Table 4). Moreover, the same partitions of the data were used for all experiments, irrespective of the algorithm. Naturally, the nature of the mapping assumed can impact the quality of the resulting classifier (Badran and Rockett, 2008), however, this is a constant across all algorithms, in effect treating all algorithms equally. In the case of the later evaluation against nonlinear SVM models of classification, a linear featurewise standardization is employed to limit training data to the unit interval.

4.3 Evaluation Metrics

Accuracy (error) still represents a widely utilized performance metric, despite the well known deficiencies of such a measure. Alternative metrics address the specific disadvantages of the accuracy metric, but not necessarily without introducing other drawbacks of their own (Sokolova et al., 2006). Moreover, performance metrics are generally constructed under the context of binary classification, making their application to multiclass datasets cumbersome at best. With this in mind, the following multiclass metrics are defined for sensitivity and specificity in order to provide a scalar characterization of classifier performance across problem domains consisting of binary or multiclass partitions (eight of the 12 datasets have more than two classes). The metrics assume an equal cost of classification/misclassification with each class contributing equally to the metric, as follows:

$$\text{MC sensitivity} = \frac{\text{DR}_1 + \cdots + \text{DR}_c}{C}; \text{MC specificity} = \frac{\text{SP}_1 + \cdots + \text{SP}_c}{C} \tag{4}$$

where detection rate (sensitivity) per class $\text{DR} = \frac{\text{TP}}{\text{TP+FN}}$ is composed from the count of true positives (TP), and false negatives (FN), and C is the number of classes. Likewise, specificity per class is expressed in terms of true negative (TN) and false positive (FP) counts, or $\text{SP} = \frac{\text{TN}}{\text{FP+TN}}$.

The above multiclass model will therefore summarize these properties under an arbitrary number of classes. Thus, in the case of a C class problem in which a degenerate classifier exists (where all training instances are labeled as one class), the corresponding MC sensitivity would be $\frac{1}{C}$ and the corresponding MC specificity would be $1 - \frac{1}{C}$. Naturally, as the performance of the classifier becomes stronger, both MC sensitivity and specificity will tend to unity; although it is also possible for a model to favor improvements to sensitivity over specificity (or vice versa). We also note that the above MC formulation renders the alternate metric redundant under the binary classification scenario, that is, the standard application of the sensitivity metric requires the utility of specificity to report on what happens to the out-of-class instances, whereas under the multiclass formulation, this is already expressed by explicitly measuring sensitivity under class 2.

Computational efficiency of GP will be evaluated in terms of a common computing platform (Apple X server, 1 GHz G4, 2 GB RAM) using the UNIX time utility as returned by the command user time. In all cases, the computing platform was dedicated to completing tasks as a serial batch process without competition from other users, thus providing an accurate measure of computational requirements for each algorithm. All measurements reflect time to initialize, train, and test the GP classifiers.

5 Classifier Performance Evaluation

Evaluation of the CMGP framework will be conducted in two stages. In the first case, performance is compared relative to a common GP representation trained using a simple class balanced sampling heuristic producing solutions in the form of a single classifier per class (Section 5.1). As such, both GP algorithms scale to large datasets and resist the effect of unbalanced class distributions. The second evaluation compares CMGP to two recent formulations of the support vector machine (SVM) paradigm (Section 5.2) where the SVM paradigm represents an established standard in classification performance.

5.1 CMGP versus Scalable Canonical GP Classifier

5.1.1 Scalable Canonical GP Classifier

The goal of the GP component of the benchmarking study is to establish the relative trade-off in algorithm complexity versus performance on a common GP representation and parameterization. With this in mind, the formulation of the baseline canonical GP classifier is as follows:

> **Balanced Random Subset Selection GP (BRSS).** Various approaches to active learning under GP have been considered including random subset selection, dynamic subset selection (Gathercole and Ross, 1994), and most recently a simple uniform sampling heuristic (Doucette and Heywood, 2008). The latter will be assumed in this work as this duplicates the CMGP heuristic for building the point population. Thus, at each generation, a new training subset is sampled from the original training partition with in-class and out-of-class instances represented equally. In common with canonical GP classifiers, a sigmoid global membership function is utilized for establishing the class label; and fitness evaluation is performed over the sum of the square error metric with respect to all points in the BRSS training subsample.

Relative to a common generation limit—or EPOCH(limit) from Table 1—the corresponding common evaluation limit is summarized by $\text{EVAL}\,(\text{limit}) = C \times \text{PP}_{\text{size}} \times \text{LP}_{\text{size}} \times \text{EPOCH}\,(\text{limit})$. Naturally the binary nature of the canonical GP classifier requires separate runs for each class or $\text{EVAL}\ (\text{limit}) \div C$ per run; whereas CMGP produces all the classifiers for all classes from a single run, thus an evaluation limit of EVAL(limit). In any single experiment, 50 runs are performed, or $C \times 50$ runs per training partition under BRSS.

5.1.2 Classification Performance

Results will be presented in terms of box plots of test set performance expressing the distribution of CMGE solutions as normalized relative to median BRSS performance over the 12 datasets: MC sensitivity (Figure 8) and MC specificity (Figure 9). The relative shift above/below unity provides an intuitive visual summary of how much better/worse the CMGP results are relative to the BRSS baseline. Table 5 details the resulting statistical significance of each pairwise comparison under the F test and Kruskal-Wallis test as well as declaring the BRSS performance normalization.

It is now readily apparent that CMGP provides better classification performance under both sensitivity and specificity in nine of the 12 datasets; whereas BRSS provides better classification results under two of the 12 datasets (CENS and CONT). Neither

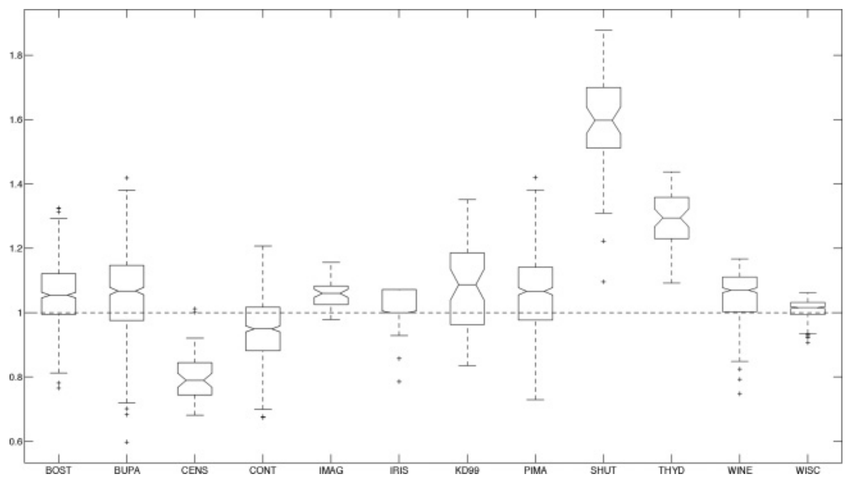

Figure 8: BRSS normalized CMGP MC sensitivity per dataset. Distributions above (below) unity imply a preference for CMGP (BRSS). Table 5 indicates that BRSS is significantly better under CENS and CONT. No statistically significant preference exists under KD99. CMGP provides significant improvements under the remaining nine datasets.

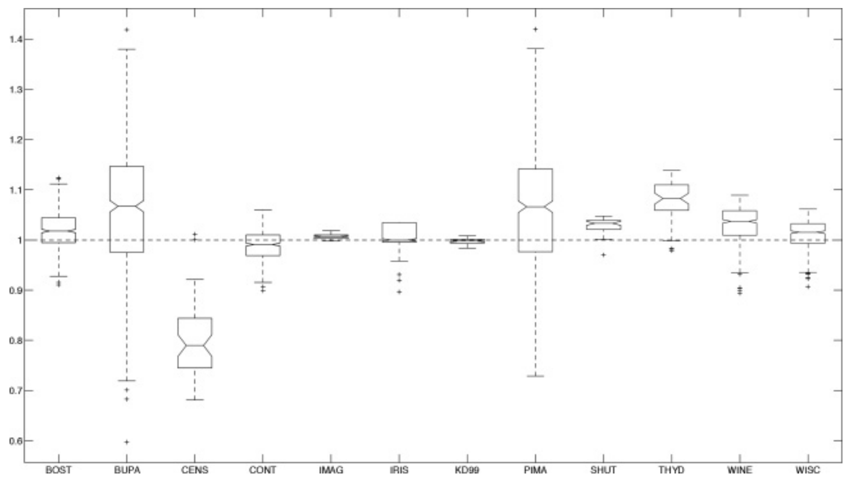

Figure 9: BRSS normalized CMGP MC specificity per dataset. Table 5 indicates that BRSS is significantly better under CENS and CONT. No statistically significant preference exists under KD99. CMGP provides significant improvements under the remaining nine datasets.

algorithm resulted in significantly better results under the KD99 dataset. Naturally, this does not preclude the existence of dataset specific parameter optimizations, however, this falls outside the scope of this study.

5.1.3 Computational Evaluation

The CMGP algorithm requires fitness evaluation to be performed against multiple objectives and deploys a clustering algorithm in the inner loop. Moreover, support for

Table 5: *F* test and Kruskal Wallis on pairwise significance test on CMGP versus BRSS testset distributions.

	Sensitivity			Specificity		
Dataset	CMGP	BRSS	Median	CMGP	BRSS	Median
BOST	*		0.63	*		0.81
BUPA	*		0.59	*		0.59
CENS		*	0.74		*	0.74
CONT		*	0.44		*	0.72
IMAG	*		0.68	*		0.95
IRIS	*		0.93	*		0.97
KD99	NSD		0.44	NSD		0.95
PIMA	*		0.59	*		0.59
SHUT	*		0.52	*		0.95
THYD	*		0.63	*		0.85
WINE	*		0.86	*		0.91
WISC	*		0.94	*		0.94

Note: NSD: no significant differences were deleted between the tests indicated.
Median: the BRSS used to normalize the CMGP MC sensitivity and specificity box plots of Figures 8 and 9, respectively.
* Both tests are true for the method associated with the column.

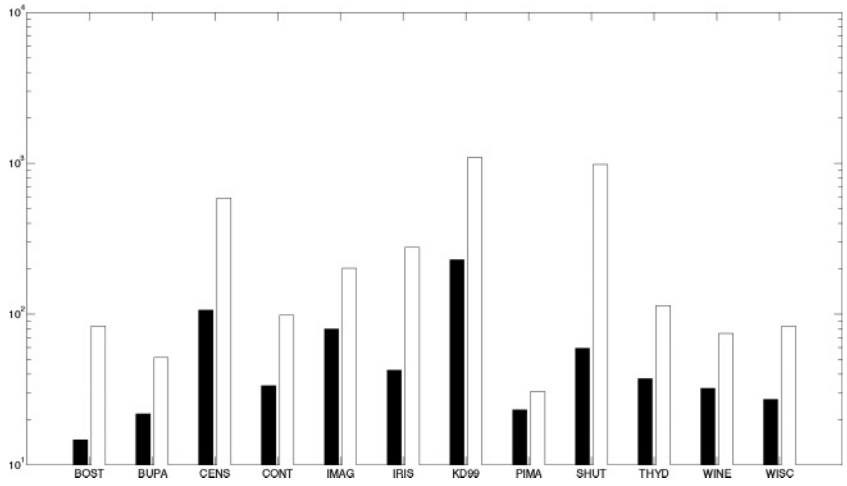

Figure 10: Median BRSS and CMGP model training and test evaluation time on common platform. All times are in log(s). Black (white) bars represent BRSS (CMGP). Only under SHUT is there an order of magnitude difference in training time between BRSS and CMGP.

the archive maintenance heuristics will also add a source of complexity not present in the BRSS model. That said, the early stopping property of CMGP provides the opportunity to reduce the total number of evaluations performed, albeit at the risk of penalizing classification performance. In order to maintain the relative unit of time (i.e., seconds), Figure 10 summarizes performance in terms of median performance of the two algorithms. Needless to say, in all cases the BRSS model is (statistically) significantly faster. However, all CMGP training is completed in between 30 and 1100 s per run, that is, a worst case runtime of 20 min. Moreover, the datasets resulting in the longest runtime are not necessarily those with largest training instance–class count product, but correspond to the cases presenting a more demanding learning problem. Thus, KD99 represents the largest training overhead (total total instance–class product of 727,920), whereas SHUT represents the highest computational cost during training (or an instance–class product of 304,500), resulting in the same computational cost for both KD99 and SHUT. In effect, problem difficulty as opposed to raw instance–class count represents the most significant factor in establishing CMGP computational overhead.

Related work has indicated that CMGP scales more gracefully than the widely utilized LIBSVM implementation of the support vector machine (SVM) model of classification (McIntyre and Heywood, 2008b), in part because the CMGP algorithm's memory requirement is independent of the training partition instance count. The following comparison with recent examples from the SVM paradigm will also demonstrate that this results in a distinct accuracy–complexity trade-off in the resulting solutions, even when the SVM paradigm adopted supports active learning as part of a mechanism for reducing model complexity.

5.2 CMGP versus Support Vector Machine Classifiers

Comparison against an alternative nonlinear classifier is pursued to provide a broader relative baseline for CMGP classification performance. In this study, support vector machines (or SVMs) were assumed. SVMs are renowned for delivering state of the art classification performance (Bottou et al., 2007). However, SVM classifiers are formulated as binary classifiers; where extending the algorithm to the multiclass setting typically employs a one versus one (OVO) or one versus all (OVA) training strategy. The latter OVA scheme represents the more common configuration and implies a requirement for k binary classifiers using, for example, a winner take all (WTA) voting heuristic (i.e., model having the highest output value assigns class) for a k-class problem (Hsu and Lin, 2002). Such a scheme mimics that adopted in the CMGP model where only one of the multiple models is used to establish each class label, again under a WTA voting heuristic. Sections 5.2.1 and 5.2.2 establish the specific properties of the nonlinear SVM paradigms utilized in the following comparative study. Section 5.2.3 introduces the metric for comparing complexity and Section 5.2.4 details results from the benchmarking study.

5.2.1 Baseline SVM Algorithm

Although the SVM paradigm provides a strong formal basis for building classification (and regression) models, the theoretical dependence on large numbers of support vectors can make it a complex and prohibitively expensive model in practice. Specifically, the number of support vectors is known to grow linearly with the number of training instances (Steinwart, 2003). Moreover, the generic SVM algorithm exhibits cubic asymptotic training complexity with the number of support vectors (Chapelle et al., 2006). Indeed, much of the recent SVM research has focused on establishing the means for efficiently implementing the optimization step, where key breakthroughs include the sequential minimization optimization (SMO; Platt, 1999) and SMO-type decomposition algorithms along with variable shrinking and kernel matrix cacheing techniques. The latter have resulted in a number of streamlined SVM kits as well as full implementations for efficiently handling the classification task in an out of the box solution. A widely used and readily available example of the latter is LIBSVM (Chang and Lin, 2001). The LIBSVM suite represents one of the most robust modern implementations of the SVM methodology, implementing all the aforementioned speedups as well as an advanced second order optimization strategy (Fan et al., 2005), as well as support for the multiclass setting and a collection of tools for cross-validation based parameter search (kernel and regularization parameter optimization). Such a background made it a suitable choice as a baseline alternative model for the current work.

5.2.2 Solution Complexity and Sparse Primal SVM

The issue of solution complexity in the SVM context is synonymous with the number of support vectors employed in model building and may be influenced by the number of instances (or subsets thereof) available for training. The sparse primal SVM (sp-SVM) approach of Chapelle et al. (2006) minimizes a primal form of the optimization problem where relaxed constraints and linear design characteristics of this formulation are exploited while controlling for complexity during model building by incrementally adding basis functions (expansions on points from the training set) to maximize accuracy (see also Chapters 2 and 11 in Bottou et al., 2007). There are two implications of the sp-SVM approach. (1) It effectively allows the process to be stopped when the classifier has reached some limiting level of complexity (the solution given approaches the full L2 quadratic penalty function as the number of basis functions increases). (2) Variation in results are inherent in the model building process as the basis element chosen at each step depends on greedy selection from a random subset of the training data. This results in an additional two learning parameters when using the sp-SVM method: the number of basis functions (model complexity) and the random subset size (the empirical recommendation provided by Chappelle et al., 2006, was 10 following an evaluation of datasets with 12,000 to 79,000 training instances).

Clearly, the ability to specify the number of basis functions is advantageous when attempting to provide a comparison of solution complexity. Moreover, the sp-SVM algorithm results in a distribution of models for each complexity limit where this is a function of the stochastic sampling introduced by the active learning component. On this basis, the sp-SVM method was adopted as a second representative model of the SVM paradigm. The principal modifications made to Chapelle's sp-SVM code for use in the present study involved extending the binary algorithm to the multiclass setting, that is, implementing the OVA approach and a WTA voting heuristic.[3]

5.2.3 Complexity Metric and Parameterization

The following benchmarking study is conducted to provide insight into the relative trade-offs between model complexity and classification performance resulting from the three models, CMGP, LIBSVM, and sp-SVM. To do so it is necessary to first establish a common model for quantifying complexity between CMGP and SVM solutions. Specifically, SVM solution complexity takes the form of support vector count, where each support vector applies a kernel to a vector product of attributes. The equivalent CMGP solution complexity is therefore established in terms of the number of vector products that the total instruction count of CMGP models in the learner archive may represent. Plotting support vector count versus MS sensitivity for each model then establishes the relative trade-off in model complexity versus classification performance.

[3]The Sp-SVM code is available online: http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/primal/

Specifically, CMGP solution complexity is defined as

$$\frac{\sum_{i \in \mathrm{LA}} \mathrm{IC}(i)}{F} \tag{5}$$

where the numerator establishes the total instruction count across the programs resident in the learner archive (LA); and F is the attribute count of the corresponding dataset, that is, both LIBSVM and sp-SVM employ all attributes. Naturally, the linear representation implicit in GE results in a large intron count, implying that the numerator is estimated after the structural intron removal.

Parameterization of LIBSVM and sp-SVM generally require optimization relative to a validation set, that is, for the purposes of identifying the relevant kernel function and regularization parameters. In this work, we assume the Gaussian kernel (where previous experience with LIBSVM indicates a strong preference for this kernel under the datasets utilized here; McIntyre and Heywood, 2008b), whereas any regularization parameters are optimized through post-training selection under the MC sensitivity metric with respect to the training partition. As indicated above, sp-SVM introduces two additional learning parameters: the basis function limit and the random subset size. The latter is set as per the earlier study by Chappelle et al.; however, for the purposes of the following study, the basis function limit is subject to incremental variation over independent runs in order to establish a performance curve for sp-SVM complexity versus classification performance.

Parameterization of GP frameworks is potentially expensive on account of the need to assess results over multiple runs. With this in mind, the following approach is adopted: (1) Alternative parameterizations will be limited to considering the impact of doubling the population/archive limits from those in Table 2; (2) Training partition MC sensitivity is used to select the top 50% of CMGP solutions, where this mirrors the utility of training performance under the SVM paradigm to identify the most appropriate learning parameterization. This has the effect of decreasing the negative impact of the stochastic credit assignment process implicit in GP at the potential expense of favoring solutions that result in overlearning.

5.2.4 Performance Comparison

Figures 11 and 12 summarize the resulting test partition performance of each algorithm under the 12 datasets. LIBSVM solutions naturally take the form of a single point with no sources for variation in the model building process (training is performed over the entire training partition in all cases and the best case training performance used to identify a single solution). Solutions from sp-SVM take the form of a curve as the basis function limit is incrementally increased from a single basis function to the point at which the LIBSVM complexity is approximated.[4] Naturally, multiple sp-SVM runs are performed per basis function limit given the utility of training instance sampling during kernel construction (see Section 5.2.2). The resulting sp-SVM curves therefore summarize 1st, 2nd (median), and 3rd quartile classification performance. The degree to which CMGP results fall above (below) the sp-SVM curve illustrate the relative performance improvement (reduction).

The Kruskal Wallis test is again applied to establish the relative significance of classification performance. To do so, a common range of model complexity is first identified

[4]In some cases, numerical instability precluded this limit from being reached.

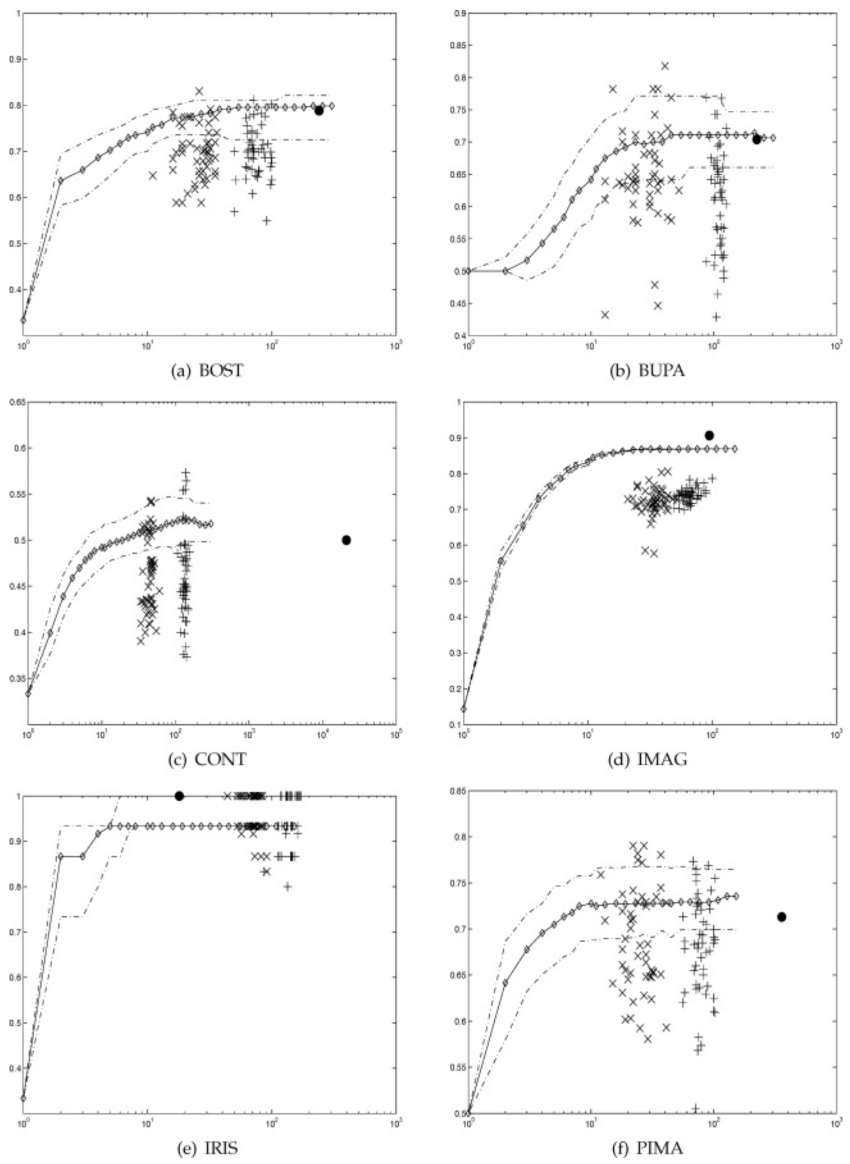

Figure 11: Test of MC sensitivity versus complexity. $\times$ and $+$ denote the CMGP solutions under the original and double the original parameterization; $\bullet$ denotes the LIBSVM solution following learning parameter optimization; whereas the curve denotes the 1st–2nd–3rd quartile variation in sp-SVM solutions as the complexity limit is varied.

between sp-SVM and CMGP solutions and the test performed between all CMGP and the corresponding subset of sp-SVM solutions. A clear preference is then identified for sp-SVM solutions under BOST, BUPA, CONT, IMAG, and PIMA; whereas CMGP solutions are significantly better under CENS, KD99, SHUT, and THYD. No significant difference exists under IRIS or WISC. In effect, algorithm performance preferences correspond to the distinction between small and balanced (sp-SVM) and large and

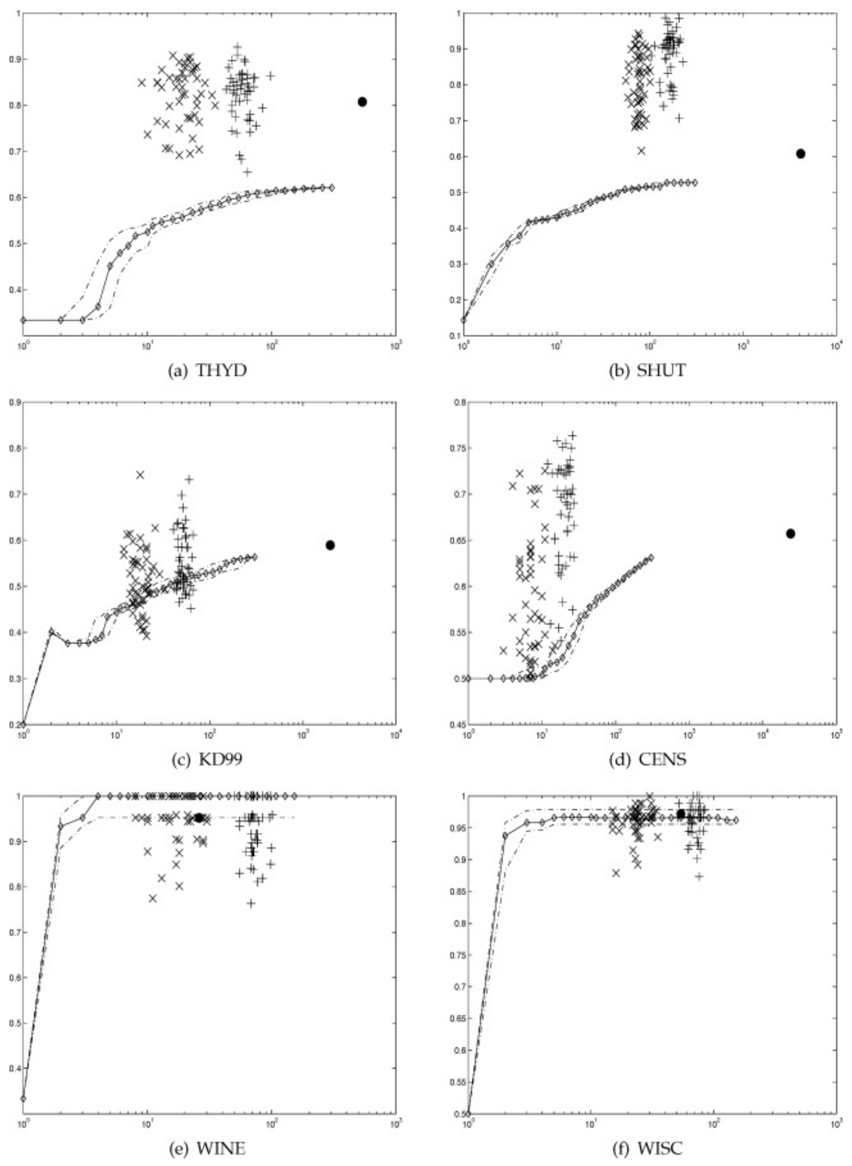

Figure 12: Test MC sensitivity versus complexity. Label legend as per Figure 11.

unbalanced datasets (CMGP). Thus, IMAGE represents the worst case scenario for CMGP or the only dataset in which no CMGP runs bettered the SVM baseline. Given that there are only 30 training instances per class in the training partition, the instance-wise optimization process of credit assignment central to the SVM methodology is clearly able to extract the maximum utility from datasets described in terms of low training instance counts. Conversely, no sp-SVM result approaches the performance of CMGP under THYD, SHUT, or CENS. Moreover, the full SVM (LIBSVM) also failed to approach the classification performance of CMGP under SHUT and was at least an order of magnitude more complex under all four large datasets. Increasing the CMGP population/archive size does not change these trends.

6 Conclusion

A cooperative-competitive model of coevolution was proposed for addressing the goal of scalable problem decomposition under the multiclass classification domain. As such, the ensuing CMGP algorithm makes the following contributions.

- **Scalability.** Without recourse to hardware specific speedups, CMGP provides several mechanisms for addressing the large dataset problem. First, fitness evaluation is conducted over a subset of instances identified dynamically during training through a combination of Pareto competitive coevolution and a balanced sampling heuristic. Second, classwise early stopping criteria allows for dynamic and efficient reallocation of GP resources (population members) as each class converges. Finally, the CMGP model builds classifiers for all classes from the same population, thus avoiding the need to perform as many runs as classes to establish a single multiclass classifier. The resulting scalability of the CMGP framework is supported by empirical results where we confirm that training times are reduced to the order of 20 min on datasets with five to seven classes and up to 150,000 training instances.
- **Robustness to Class Imbalance.** Drawing on the research of Weiss and Provost (2003), the CMGP framework employs a balanced sampling heuristic for building the point population as well as balanced class-specific archives to directly address this shortcoming. This is used in conjunction with the competitive model of coevolution to archive points identifying the nondominated classifiers in a Pareto sense. The effectiveness of the ensuing CMGP classifier performance is reflected in the strong performance across a wide range of unbalanced real-world datasets, both relative to a canonical GP classifier (which shared the class balance sampling heuristic) and recent examples of the SVM paradigm. Indeed, relative to the SVM paradigm, CMGP excelled under the large unbalanced datasets or the scenario for which the CMGP framework was designed. Conversely, the SVM paradigm clearly excels under the small and balanced datasets.
- **Problem Decomposition.** This objective is addressed through two design properties of CMGP. Firstly, the use of a LMF enables individuals to respond to specific subsets of instances. Secondly, when designing the multi-objective component of the model, care is taken to construct the objectives such that collaborative behavior with respect to other members of the corresponding (class-specific) Pareto archive is explicitly encouraged. Thus, individuals are rewarded for classifying points that are not already classified correctly by potential team members (individuals belonging to the associated learner archive). The ensuing comparison of complexity against solutions identified under the SVM paradigm indicates that the nonoverlapping nature associated with CMGP behaviors represents an effective mechanism for establishing a balance between solution complexity and classifier accuracy.

Future work will continue to investigate the utility of CMGP to perform problem decomposition. The nonoverlapping behavioral properties of teams have already been compared to that of evolutionary ensembles (McIntyre and Heywood, 2008a). However, alternative heuristics for pruning the learner archive might be more appropriate than the current formulation. Future work relative to domains with very large attribute spaces is also envisaged. Applications from document categorization have the potential to benefit from associating different subspaces with different within class partitions. Embedded—as opposed to wrapper or filter—classifier paradigms have the capacity to explicitly associate different subspace classifiers with each class partition if problem decomposition is also supported (see, e.g., Doucette et al., 2009). Other developments of the CMGP model might be appropriate for semi-supervised (or one-class) learning in which the underlying objective is to model data distributions, where the combination of problem decomposition and LMF may well provide advantages not present in other GP frameworks.

References

Badran, K. M. S., and Rockett, P. I. (2007). The roles of diversity preservation and mutation in preventing population collapse in multiobjective genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Vol. 2, pp. 1551–1557.

Badran, K. M. S., and Rockett, P. I. (2008). Integrating categorical variables with multiobjective genetic programming for classifier construction. In *Proceedings of the European Conference on Genetic Programming (EuroGP)*, *Lecture Notes in Computer Science*, Vol. 4971 (pp. 301–311). Berlin: Springer.

Bottou, L., Chapelle, O., DeCoste, D., and Weston, J. (Eds.). (2007). *Large-scale kernel machines*. Cambridge, MA: MIT Press.

Bremeier, M., and Banzhaf, W. (2001). Evolving teams of predictors with linear genetic programming. *Genetic Programming and Evolvable Machines*, 2(4):381–408.

Cartlidge, J., and Bullock, S. (2002). Learning lessons from the common cold: How reducing parasite virulence improves coevolutionary optimization. In *IEEE Congress on Evolutionary Computation*, pp. 1420–1425.

Chandra, A., Chen, H., and Yao, X. (2006). Trade-off between diversity and accuracy in ensemble generation (Chap. 19, pp. 429–464). In Y. Jin (Ed.), *Multi-objective machine learning*, Vol. 16 of *Studies in Computational Intelligence*. Berlin: Springer-Verlag.

Chang, C.-C., and Lin, C.-J. (2001). *LIBSVM: A library for support vector machines*. Retrieved from http://www.csie.ntu.edu.tw/~cjlin/libsvm

Chapelle, O., DeCoste, D., and Keerthi, S. (2006). Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 7:1493–1515.

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

Cohn, D., Atlas, L., and Ladner, R. (1994). Improved generalization with active learning. *Machine Learning*, 15:201–221.

de Jong, E. D. (2007). A monotonic archive for Pareto-coevolution. *Evolutionary Computation*, 15(1):61–94.

de Jong, E. D., and Pollack, J. B. (2004). Ideal evaluation from coevolution. *Evolutionary Computation*, 12(2):159–192.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Doucette, J., and Heywood, M. I. (2008). GP classification under imbalanced data sets: Active subsampling and AUC approximation. In *European Conference on Genetic Programming (EuroGP)*. *Lecture Notes in Computer Science*, Vol. 4971 (pp. 266–277). Berlin: Springer.

Doucette, J., Lichodzijewski, P., and Heywood, M. I. (2009). Evolving coevolutionary classifiers under large attribute spaces. In R. Riolo, U.-M. O'Rielly, and T. McConaghy (Eds.), *Genetic programming theory and practice*, Vol. VII (pp. 37–54). Berlin: Springer.

Fan, R., Chen, P., and Lin, C. (2005). Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918.

Ficici, S. G., and Pollack, J. B. (2001). Pareto optimality in coevolutionary learning. In *European Conference on Artificial Life*, pp. 286–297.

Fonseca, C. M., and Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the International Conference on Genetic Algorithms*, pp. 416–423.

Gathercole, C., and Ross, P. (1994). Dynamic training subset selection for supervised learning in genetic programming. In *Parallel problem solving from nature (PPSN III). Lecture Notes in Computer Science*, Vol. 866 (pp. 312–321). Berlin: Springer.

Harper, R., and Blair, A. (2005). A structure preserving crossover in grammatical evolution. In *IEEE Congress on Evolutionary Computation*, Vol. 3, pp. 2537–2544.

Hettich, S., and Bay, S. D. (1999). The UCI KDD archive. Retrieved from http://kdd.ics.uci.edu/

Hillis, W. D. (1990). Coevolving parasites improve simulated evolution as an optimization procedure. *Physica*, D 42:228–234.

Hsu, C., and Lin, C. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425.

Iba, H. (1999). Bagging, boosting, and bloating in genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1053–1060.

Imamura, K., Soule, T., Heckendorn, R. B., and Foster, J. A. (2003). Behavioral diversity and a probabilistically optimal GP ensemble. *Genetic Programming and Evolvable Machines*, 4(3):235–253.

Jin, Y., (Ed.). (2006). *Multi-objective machine learning*, Vol. 16 of *Studies in computational intelligence*. Berlin: Springer-Verlag.

Kothari, R., and Jain, V. (2003). Learning from labeled and unlabeled data using a minimal number of queries. *IEEE Transactions on Neural Networks*, 14(6):1496–1505.

Kumar, R., and Rockett, P. (2002). Improved sampling of the Pareto-front in multiobjective genetic optimizations by steady-state evolution. *Evolutionary Computation*, 10(3):283–314.

Laumanns, M., Thiele, L., Deb, K., and Zitzler, E. (2002). Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3):263–282.

Lemczyk, M. (2006). Pareto-coevolutionary genetic programming classifier. Master's thesis, Dalhousie University, Faculty of Computer Science. Retrieved from http://www.cs.dal.ca/~mheywood/Thesis

Lemczyk, M., and Heywood, M. I. (2007). Training binary GP classifiers efficiently: A Pareto-coevolutionary approach. In *Proceedings of the European Conference on Genetic Programming (EuroGP). Lecture Notes in Computer Science*, Vol. 4445 (pp. 229–240). Berlin: Springer.

Lichodzijewski, P., and Heywood, M. I. (2007). Pareto-coevolutionary genetic programming for problem decomposition in multi-class classification. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Vol. 1, pp. 464–471.

Liu, Y., Yao, X., and Higuchi, T. (2000). Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387.

Loveard, T., and Ciesielski, V. (2001). Representing classification problems in genetic programming. In *Proceedings of the Congress on Evolutionary Computation*, Vol. 2, pp. 1070–1077.

Markou, M., and Singh, S. (2003). Novelty detection: A review. Part 2: Neural network based approaches. *Signal Processing*, 83:2499–2521.

McIntyre, A. R. (2007). *Novelty detection + coevolution = automatic problem decomposition: A framework for scalable genetic programming classifiers*. PhD thesis, Dalhousie University, Faculty of Computer Science. Retrieved from http://www.cs.dal.ca/˜mheywood/Thesis

McIntyre, A. R., and Heywood, M. I. (2006). MOGE: GP classification problem decomposition using multi-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 863–870.

McIntyre, A. R., and Heywood, M. I. (2007). Multi-objective competitive coevolution for efficient GP classifier problem decomposition. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1930–1937.

McIntyre, A. R., and Heywood, M. I. (2008a). Cooperative problem decomposition in Pareto competitive classifier models of coevolution. In *European Conference on Genetic Programming (EuroGP). Lecture Notes in Computer Science*, Vol. 4971 (pp. 289–300). Berlin: Springer.

McIntyre, A. R., and Heywood, M. I. (2008b). Pareto cooperative-competitive genetic programming: A classification benchmarking study. In R. Riolo, T. Soule, and B. Worzel (Eds.), *Genetic programming theory and practice*, Vol. VI, Chap. 4 (pp. 43–60). Berlin: Springer.

Newman, D. J., Hettich, S., Blake, C. L., and Merz, C. J. (1998). UCI machine learning repository. Retrieved from http://mlearn.ics.uci.edu/MLRepository.html

Noble, J., and Watson, R. (2001). Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for Pareto selection. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 493–500.

O'Neill, M., and Ryan, C. (2003). *Grammatical evolution: Evolutionary automatic programming in an arbitrary language*, Vol. 4 of *Genetic programming*. Berlin: Springer-Verlag.

Parrott, D., Xiaodong, L., and Ciesielski, V. (2005). Multi-objective techniques in genetic programming. In *IEEE Congress on Evolutionary Computation*, Vol. 2, pp. 1141–1148.

Platt, J. C. (1999). *Fast training of support vector machines using sequential minimal optimization* (pp. 185–208). Cambridge, MA: MIT Press.

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

Sokolova, M., Japkowicz, N., and Szpakowicz, S. (2006). Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation. In *Evaluation Methods for Machine Learning (AAAI workshop)*, pp. 24–29.

Soule, T. (2000). Heterogeneity and specialization in evolving teams. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 778–785.

Steinwart, I. (2003). Sparseness of support vector machines. *Journal of Machine Learning Research*, 4:1071–1105.

Thomason, R., and Soule, T. (2007). Novel ways of improving cooperation and performance in ensemble classifiers. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1708–1715.

Watson, R., and Pollack, J. (2001). Coevolutionary dynamics in a minimal substrate. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 702–709.

Weiss, G. M., and Provost, F. (2003). Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–345.

Yo, T.-S., and de Jong, E. D. (2007). A comparison of evaluation methods in coevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Vol. 1, pp. 479–486.

Zhang, M., and Smart, W. (2006). Using Gaussian distribution to construct fitness functions in genetic programming for multiclass object classification. *Pattern Recognition Letters*, 27(11):1266–1274.

Zhang, Y., and Rockett, P. I. (2006). *Feature extraction using multi-objective genetic programming*, Chap. 4, pp. 75–99. In Y. Jin (Ed.), *Multi-objective machine learning*, Vol. 16 of *Studies in computational intelligence*. Berlin: Springer-Verlag.

Zitzler, E., and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comprehensive case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.