Managing Team-Based Problem Solving with Symbiotic Bid-Based Genetic Programming

Peter Lichodzijewski
piotr@cs.dal.ca

Malcolm I. Heywood
mheywood@cs.dal.ca

Faculty of Computer Science
Dalhousie University
Halifax, NS, B3H 1W5, Canada

ABSTRACT

Bid-based Genetic Programming (GP) provides an elegant mechanism for facilitating cooperative problem decomposition without an *a priori* specification of the number of team members. This is in contrast to existing teaming approaches where individuals learn a direct input-output map (e.g., from exemplars to class labels), allowing the approach to scale to problems with multiple outcomes (classes), while at the same time providing a mechanism for choosing an outcome from those suggested by team members. This paper proposes a symbiotic relationship that continues to support the cooperative bid-based process for problem decomposition while making the credit assignment process much clearer. Specifically, team membership is defined by a team population indexing combinations of GP individuals in a separate team member population. A Pareto-based competitive coevolutionary component enables the approach to scale to large problems by evolving informative test points in a third population. The ensuing Symbiotic Bid-Based (SBB) model is evaluated on three large classification problems and compared to the XCS learning classifier system (LCS) formulation and to the support vector machine (SVM) implementation LIBSVM. On two of the three problems investigated the overall accuracy of the SBB classifiers was found to be competitive with the XCS and SVM results. At the same time, on all problems, the SBB classifiers were able to detect instances of all classes whereas the XCS and SVM models often ignored exemplars of minor classes. Moreover, this was achieved with a level of model complexity significantly lower than that identified by the SVM and XCS solutions.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning*; I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

General Terms

Algorithms, Experimentation, Performance

Keywords

Genetic Programming, supervised learning, classification, coevolution, active learning, efficiency, problem decomposition, teaming

1. INTRODUCTION

The team-based metaphor for problem decomposition under GP is a cooperative coevolutionary model in which the goal is to evolve a set of individuals that learn to interact without any additional feedback from the fitness function. That is to say, the fitness function is limited to providing an overall measure of solution quality but does not actively guide the process of team building or problem decomposition. The team-based approach naturally inherits the benefits typically associated with a divide-and-conquer approach. Thus, one can expect simpler or more transparent solutions from team-based models than from monolithic models in which solutions take the form of a single super-individual.

Early team-based approaches assumed that the individuals in a given team were specified *a priori* and that a team's composition remained fixed during evolution [1]. The population was therefore comprised of a fixed number of individuals each with the same number of team members. The resulting solutions were found to represent sets of weak learners making post-training evaluation of team member roles difficult [22, 23]. Other approaches evolved ensembles of learners using an island approach [10, 20] in which mechanisms for diversity maintenance (and thus cooperation) were not as direct as they are in an explicitly coevolutionary model. Moreover, these approaches do not scale to producing large teams or training on large datasets since they require one run to generate each individual in the final solution.

More recently, Orthogonal Evolution of Teams (OET) was proposed to combine the island and earlier team approaches by defining appropriate selection and variation operators [23]. OET, however, still pre-specifies the number of team members as a single population-wide property as opposed to letting this characteristic emerge though evolution.

Also of relevance to GP teaming is a negative correlation model that utilizes a multi-objective fitness function (error minimization and negative correlation) to maintain diversity within a single population [3]. Such a framework produces multiple individuals that learn to respond to different parts

of the problem without explicitly specifying team membership. However, natural drawbacks still persist, such as the cost of evaluating the diversity maintenance metric (pairwise correlation) and the need to define a suitable post-training voting heuristic.

One final example of cooperative GP teaming is based on an activation function that is explicitly local (as in a Gaussian) rather than global (as in a Sigmoid) [14]. When used in combination with a multi-objective fitness function and competitive coevolution the model proves effective at problem decomposition from a single population under both binary and multi-class classification domains [15]. The principle drawback in this case is the cost of a clustering step in the inner fitness evaluation loop required to parameterize the local membership function (e.g., the Gaussian).

In this work we build on the bid-based model of coevolution [12, 13] which re-casts the role of GP individuals by using them to evolve a bidding strategy. The scalar actions in this model are selected from a possibly infinite set and assigned at initialization, making the approach most appropriate for domains with discrete action spaces such as classification and reinforcement learning. Earlier examples of bid-based GP considered the influence of different auction models for credit assignment [12] and the effect of using competitive coevolution to scale the approach to large datasets or problem spaces [13]. In both cases, in order to establish the 'winner' on any bidding round all members of the population had to be evaluated. This meant that particular attention had to be placed on the design of appropriate credit assignment mechanisms to avoid degenerate or parasitic behaviors. For example, individuals might concentrate on outbidding others on exemplars already correctly classified instead of focusing on exemplars as yet not correctly classified. It is this problem that is of particular relevance to the SBB model presented in this work.

Symbiotic models of coevolution are explicitly serial in nature as individuals form a meta population defining solutions in terms of individuals indexed from a second population [19]. This is in contrast to parallel models such as island approaches where the populations are isolated. We propose to let the first population define the composition of a team and the second population the GP-based individuals. Individuals in the first population are a variable-length representation and thus are not constrained to specifying the same number of individuals as in OET [23] or the original GP team models [1]. The competitive coevolutionary context is retained resulting in a third point population defining the exemplars over which fitness is competitively evolved under Pareto-coevolution. Moreover, a fitness sharing metric is used to encourage diversity is the point and team populations under finite population limits.

In summary, the SBB model introduced in this work cooperatively coevolves the content of teams under a symbiotic relation. Simultaneously, it utilizes a Pareto-competitive mechanism for coevolving the most useful exemplars supporting the development of the teams as specified by a third population. This model is developed by first reviewing the most pertinent related works in Section 2. Section 3 details the SBB framework for team-based problem decomposition under GP. Evaluation is conducted under large unbalanced binary and multi-class classification problems where the SBB approach is benchmarked with respect to classification performance and model simplicity against the LCS formulation XCS as well as SVM classifiers, Section 4. Finally, recommendations and concluding comments are made in Section 5.

2. BACKGROUND

The proposed SBB model builds on several research themes from evolutionary computation, in particular, Pareto-based competitive coevolution, coevolution through teams, symbiotic coevolution, and bid-based learning. In the following, we summarize how these related works have motivated the model developed here.

2.1 Competitive Coevolution

The interaction between the point and team populations in the SBB training algorithm follows Pareto-based competitive coevolution [7, 18]. A key concept in this framework is the Pareto-dominance relation which is defined as

$$dom(\vec{v}_1, \vec{v}_2) \Leftrightarrow \forall q : \vec{v}_1[q] \geq \vec{v}_2[q] \wedge \exists q : \vec{v}_1[q] > \vec{v}[q] \qquad (1)$$

where $\vec{v}_1$ and $\vec{v}_2$ are two objective vectors, q indexes vector dimensions (objectives), and $dom(\vec{v}_1, \vec{v}_2)$ indicates that $\vec{v}_1$ dominates $\vec{v}_2$. If an individual is not dominated by any other individual, it is said to be part of the Pareto front. Thus, the Pareto front consists of individuals such that favouring one individual over another requires a tradeoff in objective values. This relation, and the idea of the Pareto front, is used by the SBB training algorithm to determine which points and teams are selected into the next generation.

Given the ith team m_i and the kth point p_k the interaction function $G(m_i, p_k)$ returns the outcome of applying team m_i to point p_k. Higher outcomes are assumed to be favoured. The kth objective for a team m_i is then defined as $G(m_i, p_k)$. Given M_{size} teams, objective $M_{size} \cdot i + j$ of point p_k is defined as

$$\begin{cases} 1 & \text{if } G(m_i, p_k) > G(m_j, p_k) \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

for $1 \leq i, j \leq M_{size}$. If this objective is equal to 1, then point p_k is said to distinguish between teams m_i and m_j. Pareto-coevolution therefore searches for teams that achieve high outcomes and points that serve as informative evaluators[1].

Using a Pareto-based competitive coevolutionary framework, guarantees of monotonic progress have been shown to exist [5]. These proofs, however, assume an infinite memory mechanism (e.g., population or archive). In this work, since one of the goals is to train with a reduced computational overhead, a finite memory mechanism is assumed. Furthermore, instead of using a multi-layer approach as is done in LAPCA [5], the SBB algorithm separates the population into two layers only. This places more emphasis on selection within a layer (be it the Pareto front or the set of dominated members) with competitive fitness sharing [21] used to explicitly enforce a bias in favour of unique behaviours.

Such a dual mechanism for rewarding diversity – Pareto dominance and competitive fitness sharing – is utilized on account of the comparatively weak overlapping behaviours of individuals that may legitimately populate the Pareto front [15]. Conversely, introducing an explicitly cooperative mechanism in conjunction with the competitive Pareto method

[1]For consistency with the rest of the paper, the term 'team' is used for what is typically referred to as a 'learner' in Pareto-coevolutionary literature.

is meant to provide a more effective mechanism for evolving teams of individuals with non-overlapping objective values.

2.2 Teaming using Genetic Programming

GP ensembles combine multiple individuals, each designed to fulfill the same role, with the aim of improving overall performance. If evolved independently, as in the case of N-version GP [10], fit individuals that do not cooperate well are generated (i.e., their behaviours overlap). Evolving teams of classifiers explicitly as a unit [1, 22], on the other hand, produces team members that cooperate well but have poor fitness in isolation, raising the question of how the team would improve if the individuals were also fit. OET [23] applies pressure on the team level and on the individual level separately in order to produce fit individuals that cooperate well. Typically, in these cases, performance gains are achieved when errors made by a few team members are 'masked' by the correct decisions of the remaining team members. These approaches therefore do not rely so much on a division of labour as they do on redundancy.

In contrast to traditional voting or weighted combination schemes, bid-based mechanisms have been proposed to coordinate the actions of team members designed to fulfill orthogonal roles [12, 13]. Since only the 'winning' individual acts at any one time, the problem faced by ensemble methods of combining the outputs of team members is a non-issue. These approaches use GP to evolve the bidding behaviour and not to act as the classification model itself and can therefore be naturally applied to multi-class (as opposed to binary) classification problems [13].

2.3 Symbiotic Adaptive Neuroevolution

The interaction between teams and learners in the SBB approach resembles the interaction between blueprints and neurons in the symbiotic adaptive neuroevolution (SANE) architecture [16]. Each SANE blueprint specifies which hidden layer neurons are to be combined into a neural network (the input and output layers are fixed), while the neurons specify the connections that are to be formed and their weights. The neurons are said to form a *symbiotic* relationship because a single neuron specialization cannot form a successful neural network.

Two advantages of this type of symbiotic evolution are suggested. First, since a successful solution requires multiple neuron specializations, diversity is encouraged. Second, a neuron that participates in multiple networks will be evaluated multiple times leading to better estimates of the neuron's fitness. Whereas SANE defines the fitness of a neuron as the aggregate over the top five networks that the neuron participates in, the SBB model does not explicitly specify the teams to be used in learner fitness calculation. Instead, a learner is removed from the population when it is no longer found to be useful, i.e., when no team references it.

2.4 Learning Classifier Systems

LCSs typically fall into one of two categories. In *Michigan*-style classifier systems such as XCS [24], the entire population forms the solution so that each individual represents only a solution sub-component. In *Pittsburgh*-style classifier systems such as GAssist [8], each individual in the population is a complete solution. A team in the SBB approach represents a complete solution, so based on this categorization, the SBB model is more similar to Pittsburgh-style systems. Based on the behaviour of the generated solutions, however, the SBB approach is more like a Michigan-style classifier system.

Specifically, the interaction between the SBB team members when applied to a test case resembles more closely the interaction between Michigan-style individuals. The mechanism for selecting an action in Michigan-style classifiers is analogous to the mechanism for selecting an action in the proposed approach. In the former, the action associated with the highest prediction is selected, and in the latter, the action associated with the highest bid is chosen. Thus, the bid component in the SBB approach plays the same role as the condition and prediction components in Michigan-style classifier systems.

3. MODEL DESCRIPTION

The SBB algorithm coevolves a point population, a team population, and a learner population. The teams represent 'useful' combinations of learners, while the points correspond to 'informative' tests that are used to evaluate the teams (and indirectly, the learners). Each learner associates a GP-based bidding behaviour with an action (e.g., a class prediction).

The output of the training algorithm takes the form of a single team. To classify a test exemplar, each member of the team submits a bid by applying its associated bid program to the input feature vector. The highest bidder is selected as the winner and the action of this learner returned as the predicted label for the test exemplar.

At each generation, P_{gap} new points and M_{gap} new teams are generated. The points are generated by sampling the training data while the teams are generated as offspring of existing teams. In the latter case, learners present in both parent teams are viewed as representing a useful combination and copied into the offspring.

Following evaluation of all the teams on all the points, all but P_{gap} points and M_{gap} teams are selected to appear in the next generation thus making way for new members in the next generation. Pareto-based selection is used to select teams with respect to their outcomes and points with respect to the distinctions they make [7, 18]. The primary criterion for selection is whether a point/team is part of the dominated set or whether it is part of the non-dominated set (the Pareto front). If a ranking of points/teams within these subsets is required, a form of competitive fitness sharing [21] is used. This is done to introduce a bias in favour of the points/teams that exhibit non-overlapping behaviours. Indeed, evolving classifiers under a Pareto-competitive model without an additional requirement for non-overlapping behaviours has been shown to lead to poor performance of the resulting classifier [15].

A detailed model description is given in the following section.

3.1 Training Algorithm: Detailed Description

The system coevolves the following types of individuals in three separate populations:

Learners. Learners associate a bidding behaviour with an action. The bidding behaviour is implemented as a GP program and the action set is assumed to be discrete and finite, e.g., corresponding to the set of class labels. In problems where the actions are naturally con-

Table 1: SBB model algorithm parameters including the values used in the experiments.

	Description	Value
P_{size}	Point population size.	90
M_{size}	Team population size.	90
t_{max}	Number of generations.	30 000
p_d	Probability of learner deletion.	0.1
p_a	Probability of learner addition.	0.1
μ_a	Probability of action mutation.	0.1
ω	Maximum team size.	10
P_{gap}	Point generation gap.	30
M_{gap}	Team generation gap.	60

tinuous, the action set can be generated through discretization. A learner cannot solve the problem alone but must be combined with other learners.

Teams. Teams combine learners to form a complete solution. They represent sets of pointers that reference individuals in the learner population. A team must contain at least 2 references but no more than ω references. Learners of at least two different actions must be present in a team.

Points. Points represent the tests used to evaluate the teams of learners. In classification, these correspond to a subset of the training exemplars.

The SBB training algorithm is summarized in Algorithm 1. Lines 3 and 4 initialize the point, learner, and team populations P^t, L^t, and M^t respectively. In the main loop, line 5, reproduction steps are applied to the populations, lines 6 and 7. Evaluation of the teams on the points in the point population is done in line 10 which then enables selection of points, teams, and learners, lines 13 and 14. The best team is returned in line 17. Specific functions used are detailed below and the algorithm parameters are summarized in Table 1.

InitPoints($\mathbf{P_{size}}$) A set of $P_{size} - P_{gap}$ points is created by generating each point as follows. First, a label is selected with uniform probability from the set of all possible class labels. Once this label is selected, a point is selected with uniform probability from the set of all points matching that class label. If all points of a given class have already been selected into the population, the label is re-selected.

InitTeams($\mathbf{M_{size}}$) This function recognizes that a team is composed of at least two learners whose actions should be different. It generates $M_{size} - M_{gap}$ teams of size two where each team is generated as follows. First, two different actions are selected with uniform probability. For each action, a learner of that action is arbitrarily generated. The two new learners are then used to form a new team. The teams that are created in this way are added to M^t and the learners that are generated are added to L^t.

GenPoints($\mathbf{P^t}$) The model described assumes a classification domain. As the point population corresponds to a set of indexes it contains no implicit structure. This suggests the need for a suitable point generation heuristic. To this end, P_{gap} points are added to the point population P^t where each point is generated as follows. First, a label is selected from the set of all possible labels with uniform probability. A point is then selected with uniform probability from the set of all points matching the chosen label. If all points of a given label are already present in P^t, another label is selected. Furthermore, all points in P^t are required to be unique so that if a duplicate is chosen it is discarded and another point selected. The basic goal of this two stage point generation process – label then point – is to provide a mechanism that is robust to the distribution of class labels. Without such a precaution, degenerate behaviour that labels all points with the major class is likely to result.

GenTeams($\mathbf{M^t}, \mathbf{L^t}$) Two at a time, M_{gap} teams are generated and added to the team population M^t as follows. Two parents m_i and m_j are selected from M_{pop} with uniform probability and are used to form two offspring m_i' and m_j' in two steps. First step, the learners from the parents are reallocated in the offspring but do not change. In the second step, mutation is applied to the learners in the offspring potentially generating new learners. If a new learner does happen to be generated, it will only be referenced in the offspring teams (i.e., the parent teams remain unaltered).

In the first step, references to learners present in both m_i and m_j are included in both m_i' and m_j'. This 'common' genetic material is assumed to represent a useful combination of learners. Each of the references not shared by m_i and m_j is then selected with uniform probability without replacement and added to either m_i' or m_j' respecting the minimum team size two and the maximum team size ω.

The second step is applied to each offspring team separately. Here, the 'original' learners in an offspring refers to the learners present at the beginning of this step, and the order in which they are considered is always shuffled. First, each of the original learners is considered and with probability p_d removed so long as the teams size is more than two. Next, for each original learner l a learner offspring l' is generated with probability p_a so long as the team size is less than ω. Initially, l' is an exact copy of l. The program component of l' is then mutated in an implementation specific manner. With probability μ_a, the action of l' is changed to a value selected with uniform probability. If the program and bid mutation steps result in no changes to l', it is discarded, otherwise, it is included in the offspring team. The second step is repeated if the offspring duplicate a parent.

After the new teams and learners are generated their respective populations are updated accordingly.

Evaluate($\mathbf{m_i}, \mathbf{p_k}$) This function determines the outcome of applying team m_i to point p_k, $G(m_i, p_k)$. Without loss of generality, it is assumed that the outcomes are non-negative and that higher values correspond to better outcomes. In the current context, the outcome is set to 1 if the team correctly labels an instance and 0 otherwise.

SelPoints($\mathbf{P^t}$) The outcomes $G(m_i, p_k)$ calculated in line 10 are used to calculate a distinction vector of dimension $M_{size} \times M_{size}$ for each point as in [7, 18]. These distinction vectors are used to calculate the non-dominated Pareto front members in P^t, denoted $\mathcal{F}(P^t)$, and the set of dominated points in P^t, denoted $\mathcal{D}(P^t)$.

$P_{size} - P_{gap}$ points are then selected into the next generation. If $\mathcal{F}(P^t)$ contains exactly $P_{size} - P_{gap}$ points, then the entire Pareto front it selected into the next generation and all the points in $\mathcal{D}(P^t)$ are discarded.

If $\mathcal{F}(P^t)$ contains more points than are allowed to survive, then points are selected from $\mathcal{F}(P^t)$ only by first ranking

them by their competitive fitness sharing score [21] and then selecting the required number of highest-ranking points. All the points in $\mathcal{D}(P^t)$ are discarded. The sharing score for a point p_k is calculated as

$$\sum_i \frac{d_k[i]}{1+N_i} \tag{3}$$

where i iterates over all entries in the distinction vector, $d_k[i]$ is the ith entry in p_k's distinction vector, and N_i is the sum of the ith entries over the distinction vectors of all the points in $\mathcal{F}(P^t)$ (i.e., the number of points that make this distinction). Points in $\mathcal{D}(P^t)$ are not used in the calculation of N_i since these points are eliminated from consideration as far as point selection goes.

If $\mathcal{F}(P^t)$ contains fewer than $P_{size} - P_{gap}$ points then all points from $\mathcal{F}(P^t)$ are selected as well as some points form $\mathcal{D}(P^t)$. In this case, the points in $\mathcal{D}(P^t)$ are ranked using their competitive fitness sharing score and the necessary number of top ranking points selected. In this case, however, the sum N_i accounts for all the points in P^t.

SelTeams($\mathbf{M^t}, \mathbf{L^t}$) $M_{size} - M_{gap}$ teams are selected into the next generation. Team selection follows the same process as point selection except that here the outcome vectors are used to find $\mathcal{F}(M^t)$ and $\mathcal{D}(M^t)$ and to calculate the sharing score. Outcomes against the point population before selection, P^t, are used in selecting teams because compared to P^{t+1} this is seen as being more informative.

Given the selected population of teams M^{t+1}, any learners that are not referenced by a team are removed from L^t to form L^{t+1}.

Best($\mathbf{M^t}$) Since training produces a population of teams, this step is required to select one of the teams as the final output. In the case of classification problems, a score metric is defined for each team m_i with respect to the training data as

$$score_{m_i} = \frac{1}{|C|} \sum_{c \in C} DR_c(m_i) \tag{4}$$

where C is the set of class labels and $DR_c(m_i)$ returns the detection rate on class c of team m_i. The team with the highest score on the training data is returned.

Algorithm 1 Overview of the SBB training algorithm.

```
1: procedure TRAIN
2:     t = 0                                   ▷ Initialization.
3:     P^t = INITPOINTS(P_size)
4:     (M^t, L^t) = INITTEAMS(M_size)
5:     while t ≤ t_max do                      ▷ Main loop.
6:         P^t = GENPOINTS(P^t)
7:         (M^t, L^t) = GENTEAMS(M^t, L^t)
8:         for all m_i ∈ M^t do
9:             for all p_k ∈ P^t do
10:                EVALUATE(m_i, p_k)
11:            end for
12:        end for
13:        P^{t+1} = SELPOINTS(P^t)
14:        (M^{t+1}, L^{t+1}) = SELTEAMS(M^t, L^t)
15:        t = t + 1
16:    end while
17:    return BEST(M^t)
18: end procedure
```

3.2 Linear Genetic Programming

The learners' bidding procedures were evolved using linear GP [2]. To obtain bids between zero and one and to impose a smooth transition between these two asymptotic values, the Sigmoid $f(y) = (1 + e^{-y})^{-1}$ was applied to each real-valued program output y. Individuals therefore had to focus on one of three regions – zero, transition, and one – without having to implicitly identify these regions.

At initialization (Algorithm 1 line 4), bid program sizes were selected from a predefined range with uniform probability. Four stochastic search operators were then used to obtain program offspring: add, delete, swap, and mutate. Instruction add and delete operators inserted and removed an arbitrary instruction (to allow programs of varying complexity). The swap operator exchanged the location of two arbitrary instructions (in cases instructions were correct but in the wrong order), while the mutate operator flipped an arbitrary bit in the program. These operators were applied with independently specified probabilities, and in all cases, a uniform distribution was used in selecting program elements. This scheme follows the GP implementation of [13].

4. EVALUATION

The SBB model was compared with two alternative models of classification. The first is the Michigan-style learning classifier system XCS [24] which is similar to the SBB approach in that it provides solutions in the form of a cooperating team of production rules evolved through a strength-based bidding methodology. The specific XCS implementation used here, denoted XCSR, was augmented to handle real-valued inputs and has previously been evaluated on several classification problems [11]. The proposed approach was also compared against the SVM implementation LIBSVM (version 2.85) [4] on the basis that SVM models represent an established performance baseline[2]. LIBSVM is an efficient implementation of a second-order model for quadratic optimization, as well as the now widespread Sequential Minimal Optimization decomposition methodology [6].

4.1 Methodology

The datasets used in the evaluation are summarized in Table 2. The Census Income (CEN) dataset was obtained from the UCI KDD Archive [9] while the ANN Thyroid Disease (THY) and Statlog Shuttle (SHU) datasets came from UCI Machine Learning Repository [17]. This choice of problems was made to include large training partitions, multiple classes, and unbalanced class distributions. Preprocessing involved enumerating nominal features, and for XCSR and SVM, performing a linear normalization.

The SBB model parameter values used in all the experiments are detailed in Table 1. Parameters specific to the GP implementation [13] were as follows: minimum program size 1, maximum program size 48, add/delete/swap/mutate probability 0.5, number of program registers 8, function set $\{cos, exp, log, +, \times, -, \div, \%\}$. With the exception of the population sizes and gaps, no fine tuning of parameters was performed. Some fine tuning of the XCSR parameters was performed and the details are available in [11]. Values for the SVM cost parameter C of 1, 10, and 100 were investigated under both the radial basis function (RBF) and Sigmoid

[2]Whereas the SVM experiments were part of this work, the reported XCSR results were obtained form [11].

Table 2: Summary of the datasets used in the evaluation. Shown are the class distributions on the training and test partitions. The value in parentheses following each dataset label indicates the number of features. A '-' denotes the class is not present in a dataset.

		Pattern counts							
		Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	All
THY (21)	Train	93	191	3488	-	-	-	-	3772
	Test	73	177	3178	-	-	-	-	3428
CEN (41)	Train	187141	12382	-	-	-	-	-	199523
	Test	93576	6186	-	-	-	-	-	99762
SHU (9)	Train	34108	37	132	6748	2458	6	11	43500
	Test	11478	13	39	2155	809	4	2	14500

kernels. In all cases, the Sigmoid kernel resulted in inferior classification performance and solutions that were of similar or higher complexity (on SHU, the Sigmoid kernel runs using C values of 10 and 100 did not finish after 210 hours of training due to numerical difficulties), so only the RBF kernel is considered hereafter.

Performance was evaluated from two perspectives: classification on the test partition and model complexity. Classification performance was measured both in terms of the score metric (Def. 4) and the overall accuracy (defined as the fraction of all instances that were labeled correctly). The latter is more widely utilized but sensitive to unbalanced class distributions and is employed here as a means of detecting degenerate classifier behaviours. The score metric weighs each class equally while providing a single scalar summary irrespective of the number of classes involved.

For each dataset, thirty runs of the SBB algorithm were performed using different initializations. This is the same number of runs used in the XCSR evaluation [11]. The SVM results do not depend on the choice of initial conditions therefore only three runs, one for each setting of C, were performed. From these three runs the model with the highest score on the training data was selected for comparison since this was the same criterion used to select the SBB model, line 17 of Algorithm 1. Thus, both SVM and SBB models use the score metric for post-training model selection.

This setup results in a single performance point under the SVM model which is to be compared against thirty points under each of the SBB and XCSR models. We perform this comparison by first normalizing the distribution of the SBB and XCSR points by the single SVM point, i.e., dividing each of the thirty raw points by the single SVM point. The XCSR and SBB results are then characterized in terms of a distribution of results, one for each of the thirty initializations. Relative to the SVM baseline on each dataset, normalized values greater (less) than one representing performance levels above (below) the SVM baseline.

4.2 Results

As the SVM results provide a baseline for comparison these are first detailed in Table 3. A comparison of the SBB, XCSR, and SVM score results is then shown in Figure 1, and a similar comparison of the accuracy results in provided in Figure 2. With respect to score, Figure 1, the SBB approach outperforms both XCSR and SVM. On CEN and SHU, SBB is shown to clearly dominate the other two approaches. On THY, while the distribution of the SBB and XCSR values is similar, both outperform the SVM baseline.

With respect to accuracy, Figure 2, SBB remains competitive under THY and SHU, but its relative performance on CEN decreases. On THY and SHU respectively about half and four-fifths of the SBB runs outperform the SVM baseline. Compared to XCSR, the best SBB accuracies on THY and SHU are about the same and in some cases better. On CEN, the SVM baseline provides the best results while the SBB approach shows the lowest accuracies and the greatest amount of variation.

Table 3: SVM accuracy (acc.) and score (sc.) test results for different settings of the cost parameter C under the RBF kernel. Values in bold, corresponding to the solutions with the highest *score* value on the training data, were used in all comparisons.

C	THY		CEN		SHU	
	acc.	sc.	acc.	sc.	acc.	sc.
1	0.967	0.825	0.948	0.643	0.977	0.564
10	0.966	0.809	**0.949**	**0.658**	**0.975**	**0.572**
100	**0.966**	**0.800**	0.950	0.655	0.993	0.624

Since the SBB and XCSR modes of training return the same number of points, statistical significance tests on the means of the two distributions can be applied to the results, Table 4. These tests support the qualitative observations made on Figure 1 regarding the score values for the two approaches. The test accuracy results are also consistent with the qualitative observations with the exception of THY where the SBB values are shown to be significantly lower than the XCSR values. This may be attributed to the higher variation in the SBB accuracy values with the most extreme outliers corresponding to very poor performance points. Even though the mean/median accuracies using the SBB approach are lower, the maximum accuracy obtained using SBB (raw value of 0.992) is actually higher than the maximum accuracy obtained using XCSR (raw value of 0.984). At the same time, as shown in Figure 2, some of the SBB points fall well below the minimum XCSR points. Thus, on THY, the SBB approach can better XCSR but it is not able to do so on a consistent basis.

On CEN, where the accuracy difference between the SBB approach and XCSR appears to be the greatest, the maximum raw accuracies obtained using the SBB approach and XCSR are 0.868 and 0.938 respectively – based on these two values alone XCSR appears superior. However, the fact that the fraction of class 0 instances in the CEN test partition is 0.938 combined with the low score values of XCSR on CEN (raw values close to 0.500) suggests that the XCSR behaviour on CEN is degenerate. Indeed, the median raw

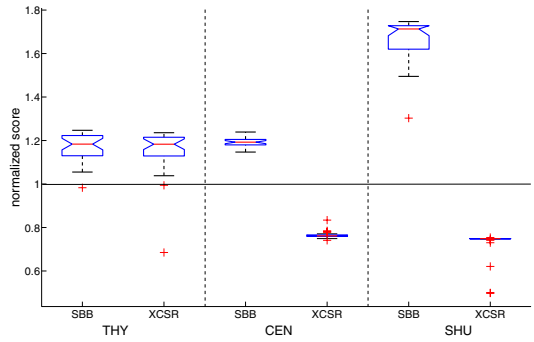

Figure 1: Comparison of the normalized SBB, XCSR, and SVM score values. The bottom/top of each box represents the first/third quartile value across the thirty SBB/XCSR initializations, the horizontal line across each box represents the median. Whiskers extend to the farthest point within 1.5 times the interquartile range from the first/third quartile, and a '+' marks all other outliers. The baseline SVM score is represented by the horizontal line at $y = 1$; an SBB/XCSR point above this threshold represents a values exceeding the SVM baseline.

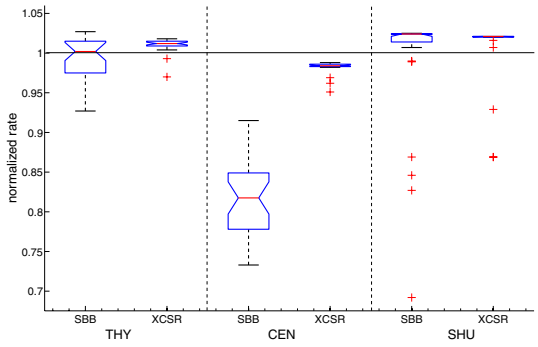

Figure 2: Comparison of the normalized SBB, XCSR, and SVM accuracy values using boxplots as in Figure 1.

Table 4: Results of a one-tailed student t-test applied over thirty initializations, assuming unequal variances, and a significance level of 0.01. Shown are the raw mean accuracy and score values on the test data for XCSR and SBB. For each dataset, if the measure of performance for one approach is significantly better than for the other, the better value is shown in bold.

		accuracy	score
THY	SBB	0.960	0.935
	XCSR	**0.976**	0.924
CEN	SBB	0.773	**0.787**
	XCSR	**0.933**	0.504
SHU	SBB	0.967	**0.953**
	XCSR	0.982	0.416

Table 5: Summary of the solution complexities. Mean (and in parentheses, standard deviation) values over thirty initializations are given for the SBB approach and XCSR while for SVM only a single value is available. The complexities are in terms of the number of learners in a team for the SBB approach, the number of macro-classifiers for XCSR, and the number of support vectors for SVM.

	THY	CEN	SHU
SBB	9.5 (0.9)	5.8 (1.8)	10.0 (0)
XCSR	881.2 (14.3)	965.5 (6.0)	644.8 (39.4)
SVM	479	23857	5318

detection rate on the CEN minority class is 0.006 indicating that XCSR is indiscriminatingly labeling all instances as belonging to class 0. This behaviour is unacceptable given that there are only two classes and that the fraction of class 0 instances (0.062) is not negligible.

On SHU, XCSR achieves high accuracy values despite showing very low score values. This suggests that in this case as with CEN it is not detecting instances of certain classes. In fact, the median raw test detection rates on instances of classes 1, 2, 5, and 6 are all 0.000. These results (high accuracy, low score) are possible because these classes account for less than one percent of the training and test data. The SBB approach, on the other hand, yields both high accuracies and high scores on SHU so it is able to identify instances of even the rare classes.

As with XCSR, compared to the SBB results, the higher SVM accuracies on CEN are not without a cost. As indicated by the relatively low SVM score value (raw score close to 0.650) most of the CEN minority class instances are not detected by the SVM model.

The solution complexities for all three approaches are summarized in Table 5. Direct comparison between the approaches is difficult since the underlying 'units of measurement' (learners, macro-classifiers, support vectors) are different. However, even if learners are viewed as the most complex with their associated bid programs at most 48 instructions long (including introns), the teams generated by the SBB approach intuitively appear to be the most simple of the three models. It is interesting that on CEN the team sizes are well below the upper limit of 10 despite there being no pressure to evolve small teams. On SHU, all models output by the algorithm contain the maximum number of team members and therefore this may be a function of the number of classes in the problem. Incidentally, support vector counts using the Sigmoid kernel were typically higher while returning worse classification performance.

Finally, the median SBB running times over the thirty initializations on THY, CEN, and SHU were 4.0, 2.6, and 4.1 hours respectively. The median XCSR training times were smaller at 0.2 hours on all datasets. However, the advantage in training speed that XCSR holds is not likely to carry over in deployment given the large number of rules that have to be matched. SVM training times are not available but the aggregate duration is expected to be smaller given that evolutionary methods require multiple initializations. In contrast, the benefit of the SBB approach is that it incurs a constant memory footprint and is likely to be much faster when deployed due to the simplicity of its solutions.

5. CONCLUSIONS

This paper introduced the SBB approach, a bid-based model for cooperative problem decomposition using teams in which GP is used to evolve the bidding behaviour of each individual. The key contribution over previous bid-based approaches was the introduction of a symbiotic relationship between a team population and a team member population. Such a symbiotic relation simplifies credit assignment while allowing team memberships, including team sizes, to evolve naturally. A Pareto-competitive component was retained for scalability, while a fitness sharing mechanism was included to help maintain population diversity.

The SBB approach was benchmarked under the classification domain. It was found to outperform the XCSR and SVM models with respect to score and with one exception remained competitive with respect to accuracy. On the dataset where the SBB accuracies were observed to be lower, the XCSR and SVM models were producing degenerate results due to a high class imbalance. The XCSR results were more consistent, however, and we hope that the consistency of the SBB approach can be improved, e.g., through proper parameterization.

One could argue that because the final step of the SBB algorithm selects the best model based on score values, whereas in XCSR no such bias is present, it is only natural that the approach performs well with respect to score. However, a single XCSR run produces a single model thus precluding any such selection in the first place. In addition, the SVM models were selected to optimize score but still resulted in lower test score values.

All SBB teams that were generated contained no more than ten members in contrast to hundreds of classifiers in the XCSR solutions and thousands of support vectors in the SVM solutions. As such, the SBB solutions were found to be much simpler resulting in increased model transparency and efficiency. Although the maximum team size was set at ten by design, the limit did not always have to be enforced (e.g., on CEN) despite the lack of explicit parsimony pressure.

Future work will apply the SBB model to domains with structure in the point population. If, instead of requiring a balanced random sampling heuristic, the current population can be used to guide the search, improvements in efficiency and solution quality are expected. It is also anticipated that the approach will be useful in reinforcement learning domains (where points represent initial environmental conditions and there is a known relationship between neighbouring states) because of the associated large state space combined with the difficulty of establishing a learning gradient.

6. ACKNOWLEDGMENTS

This work was conducted while Peter Lichodzijewski held a Precarn Graduate Scholarship and a Killam Postgraduate Scholarship. Malcolm. I. Heywood would like to thank NSERC, MITACS, and CFI for their financial support.

7. REFERENCES

[1] M. Brameier and W. Banzhaf. Evolving teams of predictors with linear genetic programming. *Genetic Programming and Evolvable Machines*, 2(4):381–407, 2001.

[2] M. Brameier and W. Banzhaf. *Linear Genetic Programming*. Springer, Genetic and Evolutionary Computation Series, 2007.

[3] A. Chandra, H. Chen, and X. Yao. *Trade-off between diversity and accuracy in ensemble generation*, volume 16 of *Studies in Computational Intelligence*, chapter 19 in Multi-Objective Machine Learning, pages 429–464. Springer-Verlag, 2006.

[4] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

[5] E. D. de Jong. A monotonic archive for Pareto-coevolution. *Evolutionary Computation*, 15(1):61–93, 2007.

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

[7] S. G. Ficici and J. B. Pollack. Pareto optimality in coevolutionary learning. In *Proceedings of the 6th European Conference on Advances in Artificial Life*, pages 316–325, 2001.

[8] J. B. J. M. Garrell. Bloat control and generalization pressure using the minimum description length principle for a Pittsburgh approach learning classifier system. *Learning Classifier Systems*, 4399:59–79, 2007.

[9] S. Hettich and S. D. Bay. The UCI KDD Archive [http://kdd/ics/uci/edu]. Irvine, CA: University of California, Dept. of Information and Comp. Science, 1999.

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

[11] P. Lichodzijewski and M. I. Heywood. Coevolutionary bid-based genetic programming for problem decomposition in classification. *Genetic Programming and Evolvable Machines*. Submitted.

[12] P. Lichodzijewski and M. I. Heywood. GP classifier problem decomposition using first-price and second-price auctions. In *Proceedings of the European Conference on Genetic Programming*, pages 137–147, 2007.

[13] P. Lichodzijewski and M. I. Heywood. Pareto-coevolutionary genetic programming for problem decomposition in multi-class classification. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 464–471, 2007.

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

[15] A. R. McIntyre and M. I. Heywood. Cooperative problem decomposition in Pareto competitive classifier models of coevolution. In *Proceedings of the European Conference on Genetic Programming (to appear)*, 2008.

[16] D. E. Moriarty and R. Miikkulainen. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5(4):373–399, 1998.

[17] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI Repository of Machine Learning Databases [http://www.ics.uci.edu/∼mlearn/mlrepository.html]. Irvine, CA: University of California, Dept. of Information and Comp. Science, 1998.

[18] J. Noble and R. A. Watson. Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for Pareto selection. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 493–500, 2001.

[19] J. Paredis. The symbiotic evolution of solutions and their representations. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 359–365, 1995.

[20] G. Paris, D. Robiliard, and C. Ronlupt. Applying boosting techniques to genetic programming. In *Proceedings of the International Conference on Artificial Evolution*, pages 267–278, 2001.

[21] C. D. Rosin and R. K. Belew. New methods for competitive coevolution. *Evolutionary Compuatation*, 5:1–29, 1997.

[22] T. Soule. Cooperative evolution on the intertwined spirals problem. In *Proceedings of the European Conference on Genetic Programming*, pages 434–442, 2003.

[23] R. Thomason and T. Soule. Novel ways of improving cooperation and performance in ensemble classifiers. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1708–1715, 2007.

[24] S. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.